University of Würzburg
Institute of Computer Science
Research Report Series

A Numerical Framework for Solving Discrete and Finite Markov Models

Michael Menth

Report No. 326 February 2004

Department of Distributed Systems
Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-8886644, fax: (+49) 931-8886632
{menth}@informatik.uni-wuerzburg.de

A Numerical Framework for Solving Discrete and Finite Markov Models

Michael Menth
Department of Distributed Systems
Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-8886644, fax: (+49) 931-8886632
{menth}@informatik.uni-wuerzburg.de

Abstract

Discrete time Markov chains (DTMCs) are commonly described by a state transition matrix that contains sufficient information to figure out the stationary state distribution of the Markov model. In this work, we propose recursive stochastic equations as an alternative description that clearly separates the functional behavior of the model from its random influences that determine the state transition matrix. For the computation of the stationary state distribution, several optimization methods are proposed that take partly advantage of the new specification scheme and that can be applied both to aperiodic and periodic Markov chains. We have written a compiler that converts the recursive equations with all optimization steps in a numerical program to evaluate even large and multi-dimensional Markov models in a short time.

Keywords: Discrete-time Analysis, Markov Chains, Power Method, Relaxation

1 introduction

Discrete time Markov chains (DTMCs) are often the basis for performance evaluation in today's telecommunication and manufacturing systems. Simulations are an alternative approach for the investigation of such systems but they can become very expensive, e.g., when small blocking probabilities are required.

Most research in DTMCs has concentrated on analytical solutions of special queuing systems [1]. Matrix analytical methods take advantage of specially structured transition matrices for a fast computation of their stationary distribution [2, 3]. However, most approaches make extensive use of the state transition matrix which can not be handled any more for large systems.

We present a simple algorithmic framework to obtain numerical results for the stationary state distribution of a finite discrete Markov model. The description uses recursive stochastic equations [4] and the evaluation scheme consists of a vector iteration method. It is a generalized formalization of the discrete-time analysis used in [5, 6, 7, 8]. Periodic, aperiodic, reducible, and irreducible Markov chains can be treated. The solving algorithm is similar to a sparse matrix multiplication method and performs well even for large Models. The simplicity and flexibility of the scheme allow to describe highly complex systems that otherwise can not be solved by analytical methods.

In the next section, an introduction to DTMCs is given and our Markov model specification using recursive stochastic equations is explained. Section 3 describes several optimization method. Section 4 illustrates the power of the framework by some modelling examples. Section 5 summarizes the work.

2 Specification and Evaluation of Markov Chains

DTMCs are commonly described by their their state transition matrix. In the following our alternative specification which is closer to the behavior of the Markov model because it clearly separates the functionality of the system from influencing random components. Based on this specification, the stationary distribution of the Markov model can be computed by numerical analysis as well as by simulation. We show that the proposed specification has the same modeling power as the conventional state transition matrix P.

For the sake of clarity, we introduce our notation regarding an arbitrary random variable X:

$\mathcal{X}$	range of X (contains discrete values)
$X_{min}, \quad X_{max}$	minimum and maximum value of X,
$x, \quad x[i]$	distribution of X and probability $Prob(X = i)$,
$X(Y = i)$	random variable X conditioned on $Y = j$,
$x(Y = j), \quad x[i](Y = j)$	distribution and probability conditioned on $Y = j$,
$\overline{X} = \sum_{i=X_{min}}^{X_{max}} x[i] \cdot i$	mean of X,
$p(Y = j)$	probability p conditioned on $Y = j$,
$p = \sum_{j=Y_{min}}^{Y_{max}} p(Y = j) \cdot y[j]$	unconditioning the conditonal probability $p(Y = j)$ by y

2.1 Discrete Time Markov Chains

We intend to give a comprehensive introduction to our model also for practitioners. Therefore, we review the definition of DTMCs, the state transition matrix, and a common classification system of DTMCs.

2.1.1 Markov Processes

A stochastic process is characterized by the time dependent random variable $X(t)$ where $X(t)$ is an element of the finite or infinite state space $\mathcal{X}$. A stochastic process is *Markov* if it shows the *memoryless property* at all time instants. The future behavior of the system at an arbitrary time t depends only on the present system state $X(t)$ and not on its past. For continuous time processes this can be expressed by the following conditioned probability:

$$P(X(t + \Delta t) = i \,|\, X(t) = j, X(t - \delta) = k) \quad = \quad P(X(t + \Delta t) = i \,|\, X(t) = j) \tag{1}$$

This must hold for any time Δt and δ as well as for arbitrary states $i, j, k \in \mathcal{X}$.

We consider a stochastic process with a discrete state space $\mathcal{X}$. As a consequence, the value of $X(t)$ changes only at discrete time instants $\{t_n = n \cdot \Delta t, n \in \mathbb{N}_0\}$. We abbreviate $X(t_n)$ by X_n. If the process is Markov, the time between a transition from q to r is exponentially distributed. This is a strong restriction on the applicability of Markov processes.

An arbitrary continuous or discrete time process has possibly time instants where the system forgets about its past and fulfills the memoryless property (Equation 2). We call them renewal points $\mathcal{R} = \{t_n : n \in \mathbb{N}_0\}$. The time range of original process can be restricted to $\mathcal{R}$ and represents then an *embedded semi-Markov process* (SMP). The series $(X_n)_{n=0}^{\infty}$ forms a *discrete time Markov chain* (DTMC). The Markov condition for a DTMC can be formulated as

$$P(X_{n+1} = i \,|\, X_n = j, X_{n-m} = k) \;=\; P(X_{n+1} = i \,|\, X_n = j). \tag{2}$$

for $0 < m \leq n$, $m \in \mathbb{N}_0$, and for $i, j, k \in \mathcal{X}$. In this paper, we focus only on DTMCs with a finite state space $\mathcal{X}$.

2.1.2 State Transition Matrix

In the following, only *homogeneous* Markov chains are considered where the conditioned probabilities in Equation 2 are constant for all renewal points. This state transition probability from state i to state j is then denoted by $p_{i,j}$ (Equation 3). All state transition probabilities can by summarized in a state transition matrix P (Equation 4) which is stochastic since the sum of a row sums to 1 (Equation 5).

$$p_{i,j} \;=\; P(X_{n+1} = j \,|\, X_n = i), \quad i, j \in \mathcal{X}. \tag{3}$$

$$P \;=\; (p_{i,j})_{i,j \in \mathcal{X}} \tag{4}$$

$$\sum_{j \in \mathcal{X}} p_{i,j} \;=\; 1, \quad i \in \mathcal{X} \tag{5}$$

The observation of a Markov model starts at renewal point t_0. The probability that the system is in state i at this time is given by $Prob(X_0 = i) = x_0[i]$ which induces the state distribution x_0 at time t_0. The initial distribution may be deterministic if the system starts only in a special state. Given the state distribution x_n, the state distribution x_{n+1} is computed by a simple matrix multiplication:

$$x_{n+1} \;=\; x_n \cdot P. \tag{6}$$

The n-step transition probability $p_{i,j}^{(n)}$ is given by the n-step transition matrix $P^{(n)} = P^n$. The average (stationary) state distribution related to all renewal points is often required for the performance evaluation of technical systems and this is also our objective.

2.1.3 Example

The description for the daily weather changes in Belfast, Northern Ireland [2] illustrates nicely the concept of a DTMC. The state is given by the weather: rainy (1), cloudy (2), and sunny (3). The state transition probabilities can be retrieved from empirical data and are given by the following state transition matrix:

$$P \;=\; \begin{pmatrix} 0.8 & 0.15 & 0.05 \\ 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \end{pmatrix} \tag{7}$$

On a rainy day the probability that tomorrow is rainy again is 80%. The average state distribution can be computed and reveals that the probability for a rainy day in Belfast is 76.25%, for a cloudy day it is 16.88%, and the sun shines with a probability of 6.88%.

2.1.4 Markov Chain Classification

Markov chains have different properties that influence the computation of their average state distribution. We take the nomenclature regarding DTMCs from [9, 10] and refer to the graph theoretical concepts in [11].

The state transition graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by the set of vertices $\mathcal{V}$ which are the states $\mathcal{X}$, and the set of edges $\mathcal{E}$ that represents state transitions. There is a state transition from state i to state j if $p_{i,j} > 0$ holds.

A state j can *be reached by or is accessible from* state i if $p_{i,j} > 0$ holds. This is denoted by $i \to j$. State k is also accessible by state i if there is a state j such that $i \to j$ and $j \to k$ are true. Hence, $i \to j$ holds if there is a path from i to j in $\mathcal{G}$. Two states i and j *communicate*, denoted by $i \leftrightarrow j$ if $i \to j$ and $j \to i$ hold. *Connection classes* can be built with respect to this relation. In graph theory, a connection class is a strongly connected component.

A subset of states $\mathcal{C} \subseteq \mathcal{X}$ of a DTMC is *closed* if no other state $j \notin \mathcal{C}$ can be reached from any state $i \in \mathcal{C}$. A minimum closed set of states $\mathcal{C}$ is called a *closure*. A closure $\mathcal{C}$ absorbs state i if there is a state $j \in \mathcal{C}$ such that $i \to j$ is true. Note that a connection class is either a closure or completely absorbed. Figure 1 illustrates the relationship between closures and connection classes.

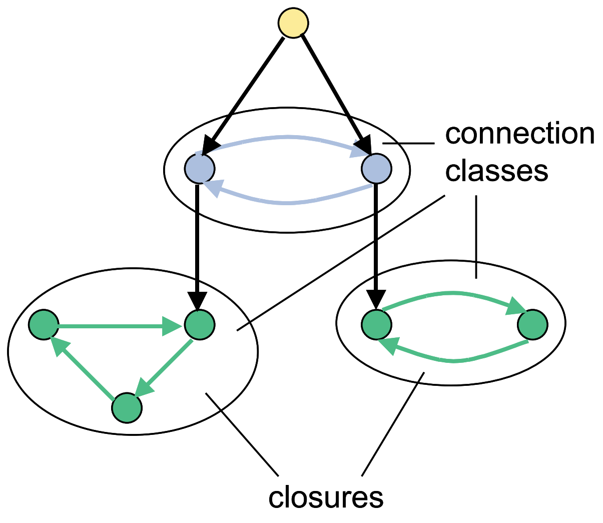

Figure 1: Classification of states in a DTMC

A Markov chain is *irreducible* if there is no closed set other than the entirety of all states, then all states belong to the same connection class and communicate with each other. A DTMC is irreducible if its state transition graph is strongly connected. A DTMC which is not irreducible is called *reducible*.

A state i has *period* p if it can be reached again only after a multiple of p transition steps, i.e

$$p = \min(m : \forall k, n \in \mathbb{N}_0 : n \neq k \cdot m \wedge \wedge p_{i,i}^{n} = 0) \tag{8}$$

All states of a closure $\mathcal{C}$ have the same period. It can be computed as the greatest common denominator of all circle lengths in the subgraph of $\mathcal{G}$ induced by the closure $\mathcal{C}$. An algorithm to find the period of an irreducible DTMC is given in [2]. A closure with period 1 is called *aperiodic* and a closure with period $p > 1$ is called p*-cyclic*. If all closures of a DTMC are aperiodic, the DTMC is also called aperiodic, otherwise it is called periodic.

2.1.5 Markov Chain Analysis

For performance evaluation purposes, the long term behavior of a Markov model is of interest, i.e., the average state distribution

$$x = \lim_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} x_i \tag{9}$$

of the DTMC is required. Since P is a stochastic matrix, the limit x does always exist [12] and is determined by x_0. The computation of the series $(x_n)_{n=0}^{\infty}$ by Equation 6 is called the power method. The average state distribution depends in general on the start vector distribution x_0 unless the DTMC is irreducible. The sum in Equation 9 converges only very slowly and there are faster ways to find the stationary distribution that will be presented later, too. The *stationary state distribution* x_s of the DTMC is defined as the left-hand eigenvector of P with eigenvalue 1:

$$x_s = x_s \cdot P. \tag{10}$$

The stationary distribution is only unique for irreducible DTMCs. If the DTMC is reducible, there is a unique stationary state distribution if a start vector x_0 is provided. For example, the identity matrix is reducible and any vector is a left-hand eigenvector. If the series of $(x_n)_{n=0}^{\infty}$ starts with $x_0 = x_s$, $x_n = x_s$ also holds. Hence, the average state distribution and the stationary distribution are equal.

2.2 A New Specification for DTMCs

The state transition matrix is the commonly used description of a DTMC. It is a close description of the model if the state transition probabilities can be taken directly from empirical data like in the example regarding the weather in Belfast. However, in technical systems, the state transition probabilities are often derived based on the known behavior of the technical model and the probability distribution of external factors.

2.2.1 Example: The $\mathbf{GI^{[GI]}/D/1 - Q_{max}}$ Queue

We consider the $GI^{[GI]}/D/1 - Q_{max}$ queuing system that has been investigated in [6]. Customer batches of size B arrive with an inter-arrival time A, where both A and B are identically and independently distributed (i.i.d.) random variables. The time to serve a customer is deterministic and the queue has a capacity of Q_{max} clients. To keep things simple, we restrict $\mathcal{A}$ to multiples of the service time for a single customer. The objective of the analysis is to compute the average queue occupancy to derive blocking probabilities and waiting time distributions for customer requests.

The system state is modelled by the queue occupation Q, hence, we have $\mathcal{Q} = \{0, 1, 2, 3, 4, 5\}$. Renewal points are embedded shortly before a new customer arrives such that we get an SMP and a DTMC $\left(Q_n\right)_{n=0}^{\infty}$. The system starts with Q_0 customers in the queue. When a batch of customers arrives, the clients fitting into the queue are accepted, the others are rejected, i.e., the new queue occupancy is $Q' = \min(Q_n + B, Q_{max})$. During the inter-arrival time A, the queue occupancy is reduced by A customers but does not fall short of zero. Therefore, the queue occupancy at the next renewal point is determined by $Q_{n+1} = \max(Q' - A, 0) = \max(\min(Q_n + B, Q_{max}) - A, 0)$.

Table 1: Batch size and interarrival time for $GI^{[GI]}/D/1 - Q_{max}$

$A = i$	2	3	4		$B = i$	1	2	3
$a[i]$	0.2	0.6	0.2		$b[i]$	0.6	0.3	0.1

We illustrate this model by a numerical example with $Q_{max} = 5$ the distribution of A and B given in Table 1. Since $\overline{A} = 3$ and $\overline{B} = 1.5$, the offered load of to the system is $\rho = \frac{\overline{B}}{\overline{A}} = 0.5$.

2.2.2 Formalization

The above description of the Markov process is based on a recursive stochastic equation. It is based on the Markov property and very close to the system behavior. It is now formalized to become applicable in a generalized context.

The choice of the renewal points $\mathcal{R}$ determines the further description of the model. The state $X = (Q)$ is given by the queue occupancy and the state space is $\mathcal{X} = [0; Q_{max}]$. The initial state distribution is given by $x_0[0] = 1$. The size of the arriving batches B and their inter-arrival times A influence the system behavior and are called input variables $Y = (B, A)$. Since A and B are both i.i.d. variables, the distribution of Y is given by

$$P(Y = (i, j)) \quad = \quad b[i] \cdot a[j]. \qquad (11)$$

The system behavior is described by a state transition function that can be generally defined as $f : (\mathcal{X} \times \mathcal{Y}) \to \mathcal{X}$. For the above example, f is

$$Q_{n+1} \quad = \quad f(Q_n, (A, B)) = \max(\min(Q_n + B, Q_{max}) - A, 0). \qquad (12)$$

Hence, the specification is given by $\mathcal{S} = (\mathcal{R}, \mathcal{X}, x_0, \mathcal{Y}, y, f)$. Note that in general both the state space $\mathcal{X}$ and the input space $\mathcal{Y}$ may be multi-dimensional.

2.2.3 Derivation of the State Transition Matrix P

The state transition matrix P can be computed by

$$p_{i,j} \quad = \quad \sum_{\{k \in \mathcal{Y} : f(i,k) = j\}} y[k]. \qquad (13)$$

The state transition matrix of our example in Section 2.2.1 is given by

$$P \quad = \quad \begin{pmatrix} 0.98 & 0.02 & 0 & 0 & 0 & 0 \\ 0.86 & 0.12 & 0.02 & 0 & 0 & 0 \\ 0.54 & 0.32 & 0.12 & 0.02 & 0 & 0 \\ 0.12 & 0.42 & 0.32 & 0.12 & 0.02 & 0 \\ 0 & 0.12 & 0.42 & 0.32 & 0.12 & 0.02 \\ 0 & 0 & 0.12 & 0.42 & 0.32 & 0.14 \end{pmatrix}. \qquad (14)$$

Hence, the specification $\mathcal{S}$ provides at least as much information as the state transition matrix P. If the input distribution y changes for a technical model, it is just substituted by another distribution in the specification $\mathcal{S}$ while the state transition function f remains unchanged. Thus, our specification separates clearly the random influence y from the model behavior f. In contrast, the state transition matrix changes may change completely.

2.2.4 Comparison of Expressiveness

Above we have shown that the state transition matrix of the Markov model can be derived from the specification $\mathcal{S}$. Conversely, for every finite state transition matrix P, at least one model ($\mathcal{S} = (\mathcal{R}, \mathcal{X}, x_0, \mathcal{Y}, y, f)$ can be found such that it yields the same P. We describe a constructive algorithm to find such a model.

The renewal points $\mathcal{R} = \{t_n = n \cdot \Delta : n \in \mathbb{N}_0\}$ of the new model can be assumed equally spaced. The state space $\mathcal{X} = [0; r-1]$ is given by the range r of the matrix P. The start state distribution may be arbitrary. The sums $s_{i,j} = \sum_{k=1}^{j} p_{i,k}$ are auxiliary variables to construct $\mathcal{Y}$ and y. We further define the vector s^* that contains $\{s_{i,j} : i,j \in \mathcal{X}\}$ in an ascending order and where duplicated elements occur only once. Let s^* have n^* entries. Since P is a stochastic matrix, we have $s_{i,X_{max}} = 1$ and according to the construction, the last entry of $s*$ is 1. We define the distribution of Y by

$$y[0] = s_0^* \quad \text{and} \quad y[i] = s_i^* - s_{i-1}^* \tag{15}$$

and $\mathcal{Y} = [0; n^* - 1]$. For the definition of the state transition function $f(i,j) = k$, we compute the preimage $\mathcal{Z}_{i,j} = \{k : k \in \mathcal{Y}, f(i,k) = j\}$ in such a way that $\mathcal{Y}$ is partitioned among the set $\{\mathcal{Z}_{i,j} : i \in \mathcal{X}\}_j$ and that $p_{i,j} = \sum_{k \in \mathcal{Z}_{i,j}} y[k]$. This is done as follows. Since $p_{i,0}$ corresponds to $s_{i,0}$, there is a $k_{i,0}$ such that $\sum_{h=0}^{k_{i,0}} y[h] = p_{i,0}$. Thus, we assign the inputs values $[0; k_{i,0}]$ to $\mathcal{Z}_{i,0}$. Similarly, for every $p_{i,j}$, $j > 0$, there is a $k_{i,j}$ such that $\sum_{h=k_{i,j-1}}^{k_{i,j}} y[h] = p_{i,j}$ and we assign the inputs $[k_{i,j-1}, k_{i,j}]$ to $\mathcal{Z}_{i,j}$. Finally, the preimage defines the state transition function, i.e., $f(i,j) = k$ holds if $k \in \mathcal{Z}_{i,j}$ is true.

Example We apply this algorithm to the example about the weather in Belfast and get the input distribution y in Table 2 and the corresponding preimage $\mathcal{Z}$ in Equation 16. The specification $\mathcal{S} = (\mathcal{R}, \mathcal{X}, x_0, \mathcal{Y}, f)$ is derived according to our algorithm and yields the same state transition matrix P as in Equation 7.

Table 2: Input distribution y and y^* for the weather in Belfast

Y=i	0	1	2	3	4	5
$y[i]$	0.5	0.2	0.1	0.1	0.05	0.05
$y^*[i]$	0.5	0.2	0.15	0.1	0.05	

However, we can provide an alternative input distribution y^* and preimage $\mathcal{Z}^*$ with a smaller input space that lead to the same state transition matrix P. Hence, the constructed model is not necessarily the smallest one, i.e., the cardinality of the input space is not minimum. It is obvious that the derived functional specification for f as well as the input distribution y are not definite as well as the renewal points $\mathcal{R}$, the state space $\mathcal{X}$ and the start vector distribution x_0. Therefore, the functional specification $\mathcal{S}$ contains more information and is closer to the model than the state transition matrix P.

$$\mathcal{Z} = \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6\} \\ \{1,2\} & \{3,4\} & \{5,6\} \\ \{1\} & \{2,3\} & \{4,5,6\} \end{pmatrix}, \mathcal{Z}^* = \begin{pmatrix} \{1,2,4\} & \{3\} & \{5\} \\ \{1,2\} & \{3,5\} & \{4\} \\ \{1\} & \{2,4\} & \{3,5\} \end{pmatrix} \quad (16)$$

2.3 Markov Chain Simulation

The average state distribution x of a Markov chain can also be calculated by a simulation. The simulation starts in state X_0. A pseudo random number determines a realization of the input variable Y_n at every renewal point t_n. The successor state X_{n+1} is computed based on the state transition function $f(X_n, Y_n) = X_{n+1}$ and the input variable Y_n. The outcome of the n-th transition in the simulation can be stored by the deterministic distribution x_n. The average state distribution can then be computed like above (Equation 9). To determine the end of the simulation, statistical methods like confidence intervals must be used.

In case of an irreducible Markov chain the average state distribution is independent of the start state. If the Markov chain is reducible, the obtained stationary distribution depends on the stationary distribution of the closure that has been chosen by chance, i.e., the eventual outcome depends on random influences.

2.4 Comparison of Analysis and Simulation

The stationary state distribution x can be found by analysis and by simulations. We point out the difference and the affinity between the analytical and simulative approach and compare the computing costs.

2.4.1 Differences and Affinities

To illustrate the mode of operation of both approaches, we define a step-wise transition graph. The graph is infinite in size: the set of nodes consists of the product of the state space $\mathcal{X}$ and all future renewal points, which is an infinite number. The nodes can be arranged as a matrix where each column corresponds to a specific state and each row to a specific transition point. The start state of the model is expressed by a start node. The edges are the performed single-step transitions.

In the analytical approach, all possible transitions are made at the n-th renewal point which corresponds to a breadth-first search in the step-wise transition graph. As a result, each row in the step-wise transition graph can be mapped to a transient state probability distribution x_n. Figure 2 shows the step-wise transition graph for a random walk with an upper and a lower bound.

In contrast, the simulative approach takes only a single transition per renewal point and performs a depth-first search like traversal through the graph. As mentioned above, the outcome of the n-th simulation step is a specific state which can be viewed as a deterministic distribution x_n. Figure 3 illustrates the step-wise transition graph for the random walk.

The step-wise transition graph suggests to view the analysis as a simulation based on distributions and the state transition matrix P instead on random variables and random numbers.

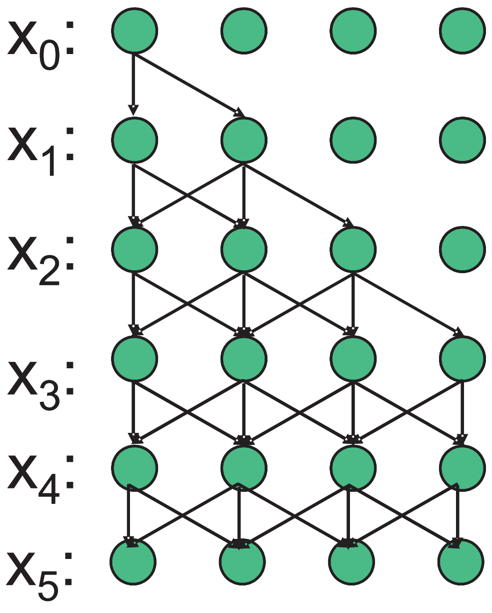

Figure 2: The first 10 transition steps for Markov chain analysis.

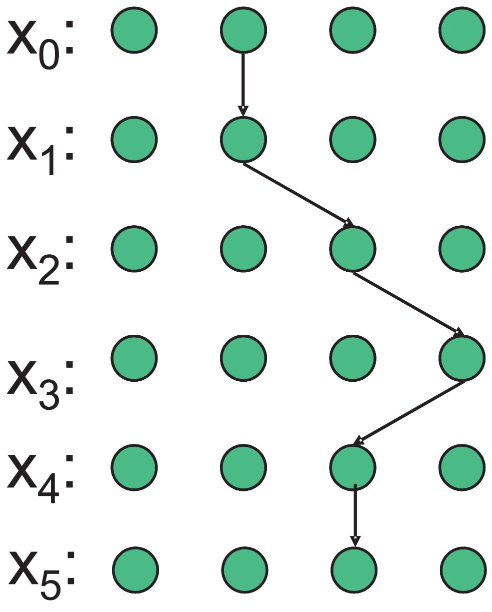

Figure 3: The first 10 transition steps for Markov chain simulation.

2.4.2 Computing Costs

As mentioned above, the proposed analysis is numerically equivalent to the power method using the state transition matrix P. From the power method it is known that the convergence speed is exponential, i.e., $\| x_n - x \| \leq \rho(P)^n$ where $\rho(P)$ is the spectral radius of P:

$$\rho(P) \quad = \quad \max_{\|e\|=1}(e \cdot P). \tag{17}$$

The number of needed iterations can be computed by $\| x - x_n \| \leq \rho(P)^n \leq \epsilon$, i.e., $n \geq \frac{\ln \epsilon}{\ln \rho(P)}$. The computing expenses increase linearly with exponentially decreasing ϵ since the cost for a single iteration is constant.

When simulation results need to have an accuracy of ϵ, the number of samples that are required must be at least in the order of magnitude of $n > \frac{1}{\epsilon}$ since otherwise a probability of $\frac{1}{\epsilon}$ can not be simulated. From that follows $n > \frac{1}{\epsilon}$, i.e., the number of required samples increases exponentially with exponentially decreasing ϵ.

After all, it is clear that analysis yields very accurate results much faster than simulations. However, if the expenses for a single iteration in Equation 6 are extremely costly and the convergence criterion is relatively loose, then simulations might still be faster. In a later section, advantage will be taken from that fact to apply some optimizations to the algorithm for computing the average state distribution.

3 Optimization Methods

In this section, optimizations for the analytical evaluation of DTMCs are suggested. Some of them reduce the required memory, others yield a computation speed up.

Many optimization methods are presented in [2] for a fast computation of the stationary state distribution. However, most of these algorithms take advantage of the structure of the state transition matrix P or they need at least its explicit construction and in many cases even a sparse matrix storing scheme requires too much memory.

Some of the following optimizations aim at accelerating the convergence of the stationary state distribution (Equation 10), some others make a single iteration faster or less memory consuming.

3.1 Limiting Distribution

If a Markov chain is aperiodic, then the limiting distribution

$$x \quad = \quad \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} x_k = \lim_{n\to\infty} x_n \tag{18}$$

does exist [12] which is also called the Cesaro limit. If x_n converges, it is obvious that it converges faster than the series of the averaged distributions $(\frac{1}{n} \cdot \sum_{k=0}^{n-1} x_k)_{n=0}^{\infty}$. The following inequality may be used as a convergence criterion where $\| \ \|_\infty$ denotes the maximum norm:

$$\| x_n - x_{n-1} \|_\infty < \epsilon. \tag{19}$$

In case of a periodic DTMC, the limiting distribution of x_n does not exist. The state space $\mathcal{X}$ of a p-cyclic DTMC may be partitioned into p distinctive numbered *periodic classes* $\mathcal{X}_i$, $0 \leq i < p-1$, such that a single-step transition from a state of class j is only possible to enter a state of class $\mathcal{X}_{(j+1) \mod p)}$. According to the definition of the period p, a path of p steps always leads to a state of the same class. Hence, the p-step transition matrix P^p is reducible and every subset $\mathcal{X}_i$ forms a subchain which is potentially aperiodic. In this case, p cyclo-stationary distributions

$$x^{(i)} = \lim_{n\to\infty} x_{n \cdot p + i} \tag{20}$$

exist. Finally, the average state distribution of a periodic chain can be computed by

$$x \quad = \quad \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} x_k = \lim_{n\to\infty} \frac{1}{p} \sum_{i=0}^{p-1} x^{(i)} = \lim_{n\to\infty} \frac{1}{p} \sum_{k=n}^{n+p-1} x_k \tag{21}$$

The righthand expression can be faster computed than the definition in Equation 9 since the cyclo-stationary distributions converge faster than their averaged distributions. The period p of the model must be considered when testing for convergence and the convergence criterion in Equation 19 must be modified:

$$\| x_n^{(i)} - x_{n-1}^{(i)} \|_\infty < \epsilon \quad \text{, i.e.,} \ \| x_{n+i} - x_{n-p+i} \|_\infty < \epsilon \quad \text{for} \quad 0 \leq i < p. \tag{22}$$

The convergence criterion must be met for all cyclo-stationary distributions $x^{(i)}$. Thus, one needs to store the distributions of p consecutive iteration steps. As soon as the iteration has sufficiently converged, the p stored distributions are averaged to obtain the stationary state distribution.

3.2 Relaxation

Relaxation is a means to modify the eigenvalues of a matrix to achieve faster convergence. Usually, relaxation is done with Gauss-Seidel or Jacobi methods [2, 13]. Our new approach is to transform P into $P(\alpha)$ by

$$P(\alpha) \;=\; \alpha \cdot P + (1-\alpha) \cdot I \quad 0 < \alpha \leq 1 \tag{23}$$

where I is the identity matrix. If α ranges between 0 and 1, $P(\alpha)$ still remains stochastic. $P(\alpha)$ has the property that its eigenvector x to the eigenvalue 1 is the same as the one for P:

$$P(\alpha)x \;=\; \alpha \cdot x \cdot P + (1-\alpha) \cdot x \cdot I = \alpha \cdot x + (1-\alpha) \cdot x = x \tag{24}$$

The effect of the α-relaxation on the state transition matrix is twofold.

Firstly, Figure 4 shows that the α-relaxation with $0 < \alpha < 1$ makes every state reachable from itself by an additional transition and, as a consequence, the resulting Markov chain is aperiodic. As the α-relaxation does not change the stationary state distribution, computing the Cesaro limit of $P(\alpha)$ is an elegant means to find the stationary distribution of a periodic DTMC.

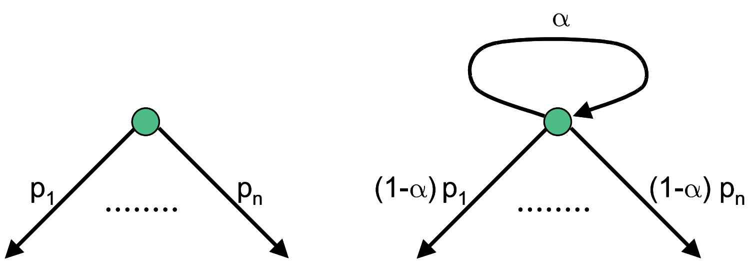

Figure 4: α-relaxation

The α-relaxation is essentially the matrix implementation of a moving average computation: $x_{n+1} = x_n \cdot P(\alpha) = \alpha \cdot x_n \cdot P + (1-\alpha) \cdot x_n$. Therefore, the α-relaxation decreases the difference of consecutive state distributions:

$$\begin{aligned} \| x_{n+1} - x_n \| &= \| \alpha \cdot x_{n+1} + (1-\alpha) \cdot x_n - x_n \| = \alpha \cdot \| x_{n+1} - x_n \| \\ \| x_{n+1} - x_n \| &= \alpha \cdot \epsilon \end{aligned} \tag{25}$$

Thus, the convergence criterion should be increased by $\frac{1}{\alpha}$.

Secondly, the α-relaxation changes the eigenvalues and leads possibly to a faster convergence which accelerates the computation of the stationary state distribution of P. The α-relaxation removes the complex eigenvalues of P with size 1 so that only the eigenvalue 1 is of size 1 remains which is equivalent to removing periods. This denotes that the relaxation alters both the size and the phase of complex eigenvalues. This has also an influence on the subdominant eigenvector whose size is $\rho(P)$ and determines the convergence speed. Applying α-relaxation to P is also called preconditioning to obtain a better convergence behavior.

Nearly decomposable Markov chains are systems that are almost periodic, i.e., the paths in $\mathcal{G}$ that hinder the periodicity have very little probability. As they are almost periodic, the series of $\big(x_n\big)_{n=0}^{\infty}$ converges only very slowly but the acceleration method for periodic Markov chains presented in Section 3.1 does not work. However, α-relaxation achieves usually a substantial calculation speedup. One can think of the α-relaxation in this case as an approach that averages subsequent state distributions x_n over a unknown period p, similarly to Equation 21. We will show the resulting benefit later using an example.

3.3 Sparse Matrix

A trivial optimization is the use of a sparse matrix scheme, i.e., only the non-zero transition probabilities of the transition matrix are stored in a smart data structure. Since many Markov models exhibit the majority of their transition probabilities to be zero, this is a simple and effective trick to save both storage and to avoid redundant floating point multiplications when $x_{n+1} = x_n \cdot P$ is performed.

3.4 Zero-State Omission

Zero-state omission excludes states i with probability zero $(x_n[i] = 0)$ from multiplications. Though this optimization looks simple, it is very effective especially for the first iteration runs. In addition, connection classes that are never entered due to a specific x_0 do not consume any computation time.

3.5 Matrix Powering

The state distribution x_{n+1} can be computed either iteratively by Equation 6 or by $x_{n+1} = x_0 \cdot P^{n+1}$ which requires P^{n+1}. By applying matrix powering,

$$P^{2\cdot n} \quad = \quad P^n \cdot P^n \tag{26}$$

a subset of the series $(P^n)_{n=0}^{\infty}$ can be quickly computed. This can significantly accelerate the calculation of the limiting distribution.

However, matrix-matrix multiplication takes $O(r^3)$ multiplication operations while vector-matrix multiplication takes only $O(r^2)$ multiplication operations where r denotes the range $range(P)$ of P. Hence, as long as the number of required iterations is smaller than r, the vector-matrix iteration approach is computationally cheaper. The range of a matrix depends on the state space and can become very large, e.g. 10^6 states while the number of iteration steps is mostly in the order of 10^3 or smaller. In addition, the vector-matrix multiplication can be also accelerated by zero state omission and sparse matrix representation. Both do not work for matrix-matrix multiplication because there is no vector where zero states can be omitted and matrix P^n is usually a fully occupied even if matrix P is sparsely occupied.

Therefore, at first glance, this methods seems to be quite appealing but on the second glance, it is not beneficial for practical problems.

3.6 Algorithmic Evaluation Using Recursive Equations

A matrix with 10^6 states has 10^{12} possible transitions which require a memory of 8 Terabytes if a single transition is described by an 8 bytes double precision floating point number. In this case, even a sparse matrix representation can not be stored anymore.

The recursive stochastic equations and the distributions of the input variables makes the explicit construction of the state transition matrix obsolete. Basically, the probability of a subsequent state can be computed according to the law of total probability by

$$x_{n+1}[i] \quad = \quad \sum_{j\in\mathcal{X},k\in\mathcal{Y}} Prob(X_{n+1} = i|X_n = j \wedge Y = k) \cdot x_n[j] \cdot y[k]. \tag{27}$$

Since the recursive stochastic equations work on random variables, their outcome is deterministic and so

$$Prob(X_{n+1} = i|X_n = j \wedge Y = k) \quad = \quad \begin{cases} 0 & f(j,k) \neq i \\ 1 & f(j,k) = i \end{cases} \tag{28}$$

simplifies Equation 27 to

$$x_{n+1}[i] \quad = \quad \sum_{\{(j,k)|f(j,k)=i,j\in\mathcal{X},k\in\mathcal{Y}\}} x_n[j] \cdot y[n], \tag{29}$$

i.e., the sum of the compound probability of the preimage of state i with respect to f $\mathcal{Z}_i = \{(j,k)|f(j,k) = i, j \in \mathcal{X}, k \in \mathcal{Y}\}$ is computed. We call Equation 29 a *backward approach* because one has to look back into the past to compute x_{n+1}. It has the advantage that the explicit computation of the transition matrix is not required anymore. Therefore, huge Markov chains can be treated. In [6], the summation over the probability of the elements in the preimage was done by the composition of various operators as, e.g., the discrete convolution operator:

$$x_{n+1}[i] = x_n[k] \star y[k] = \sum_{j=Y_{min}}^{Y_{max}} x_n[i-j] \cdot y[j] \text{ which is essentially} \tag{30}$$

$$x_{n+1}[i] = \sum_{\{(j,k)|f(j,k)=i,j\in\mathcal{X},k\in\mathcal{Y}\}} x_n[j] \cdot y[n], \text{ for } f(i,j) = i + j \tag{31}$$

The benefit of the discrete convolution is that it can be accelerated by fast Fourier transformation. However, as soon as the operator is not the discrete convolution, it is cumbersome and error prone to implement since the preimage computation of an arbitrary function f is a non-trivial task. As soon as the state space becomes multi-dimensional and the state transition function becomes more complex, the corresponding operators become infeasible.

An alternative computation scheme, called *forward approach*, retains the storage savings of the backward approach and does not rely on the preimage which has imposed a limit on the complexity of tractable models. Simply rearranging all computations for x_{n+1} according to Equation 29 yields Algorithm 1.

Algorithm 3.6 derives the state transition matrix P in the same way.

Now, it is obvious that the matrix iteration method using the transition matrix P (Equation 6), the backward approach (Equation 29), and the forward algorithm (Algorithm 1) are numerically identical and have the same convergence behavior. The beauty of the forward approach is that the program for the numerical computation can syntactically derived from the specification $\mathcal{S}$ in contrast to the backward approach.

```
Input:  state distribution $x_n$ and influencing distribution $y$
  Initialize $x_{n+1}$ with zeros
  for all $i \in \mathcal{X}$ do
    for all $j \in \mathcal{Y}$ do
      $x_{n+1}(f(i,j)) := x_{n+1}(f(i,j)) + x_n(i) \cdot y(j)$
    end for
  end for
Output:  $x_{n+1}$
```

Algorithm 1: Forward algorithm

```
Input:  influencing distribution $y$
  Initialize $P$ with zeros
  for all $i \in \mathcal{X}$ do
    for all $j \in \mathcal{Y}$ do
      $p_{i,f(i,j)} := p_{i,f(i,j)} + y(j)$
    end for
  end for
Output:  $P$
```

Algorithm 2: Transition matrix

3.7 Decomposition of the State Transition Function

It can be computationally cheaper to perform several consecutive simple operations on x_n than one very time consuming. This is done by the decomposition of the state transition function. In the example from Section 2.2.1, the state transition function f (Equation 12) can be partitioned into $f = f^1 \circ f^2$:

$$\begin{aligned} f(Q_n,(A,B)) &= \max(\min(Q_n+B,Q_{max})-A,0), \\ f^1(Q_n^1,B) &= \min(Q_n^1+B,Q_{max}) \\ f^2(Q_n^2,A) &= \max(Q_n^2-A,0) \end{aligned} \tag{32}$$

Note that a k-fold decomposition requires the k-fold backing of the state space ($\mathcal{X}^i, \quad i \in [1;k]$) to record the intermediate distributions where $\mathcal{X} = \mathcal{X}^1$. The input space can be written in product form $\mathcal{Y} = \times_{i=1}^k \mathcal{Y}^i$. Algorithm 1 must be modified accordingly.

The achieved speedup by the decomposition depends on the size of the input spaces $\mathcal{Y}^i$. The complexity of the computing algorithm (Algorithm 1) reduces then from $O(|\mathcal{X}| \cdot \prod_{i=0}^{r-1} |\mathcal{Y}_i|)$ to $O(|\mathcal{X}| \cdot \sum_{i=0}^{r-1} |\mathcal{Y}_i|)$. This yields considerable time savings especially for long distributions.

3.8 Conditional Input Variables

So far, the input space was assumed to be in product form, i.e., all dimensions of random variables are independent of each other. Assume again the $GI^{GI}/D/1 - Q_{max}$ queue. We modify it by a conditioned arrival process, i.e., the arrival rate depends on the present queue occupancy Q and we denote that system by $GI^{GI}(Q)/D/1-Q_{max}$. The only way to respect dependencies so far is coding them in the state transition function:

$$f^2(Q_n^2, A_0, ..., A_{Q_{max}}) \quad = \quad \begin{cases} \max(Q_n^2 - A_0, 0) & Q = 0 \\ ... \\ \max(Q_n^2 - A_{Q_{max}}, 0) & Q = Q_{max} \end{cases} \tag{33}$$

This, however, is a poor approach since the Algorithm 1 needs to step through $Q_{max}-1$ loops in vain. A better solution to this problem is to introduce conditional input distributions, denoted by $A(Q)$, which means that the correct distribution is taken depending on the value of Q. Equation 33 can be rewritten as

$$f^2(Q^2, A(Q^2)) \quad = \quad \max(Q^2 - A(Q^2), 0). \tag{34}$$

Note that this feature does not only ease the modelling but it also accelerates the computation tremendously.

3.9 Start Vector Initialization by Simulation

For irreducible Markov chains the average state distribution is equal to the stationary distribution and thus unique. Therefore, given a start state, a quick simulation run can approximate the stationary distribution. The resulting distribution x_0 can be taken as start vector for Algorithm 1. For the accuracy in simulations, the computing expenses rise exponentially, however, for small accuracy less time is required than with the analysis and, therefore, this is a shortcut. Experiences have shown that 10^7 simulation samples provide an accuracy of already about 10^{-5}. Hence, 4 orders of magnitude do not need to be computed by numerical iterations.

3.10 Intelligent Modelling

The major optimization at all lies in intelligent modelling. First of all, effort must be made to keep the state and input space small. In particular, product forms for their description increase the computation time and the storage requirements for $\mathcal{X}$ significantly.

4 Application to Problems in Telecommunication

In this section, we present two different models that illustrate the proposed specification.

4.1 Traffic Spacer

A spacer is a network entity that delays cells or packets until they are conform to a negotiated peak rate. It can be modeled by a byte counter S that has a maximum capacity of S_{max}. If a packet arrives and its size B does not exceed the counter, i.e., $S + B \leq S_{max}$ it is accepted and the counter is increased by that amount. Otherwise, the packet is discarded. The time that the packet has to be delayed is given by the spacer counter S at the arrival time of the packet divided by the peak rate C. Hence, the waiting time is computed by $W = \frac{S}{C}$. The spacer counter decreases linearly over time by the peak rate C and must not fall short of zero. This is depicted in Figure 5.

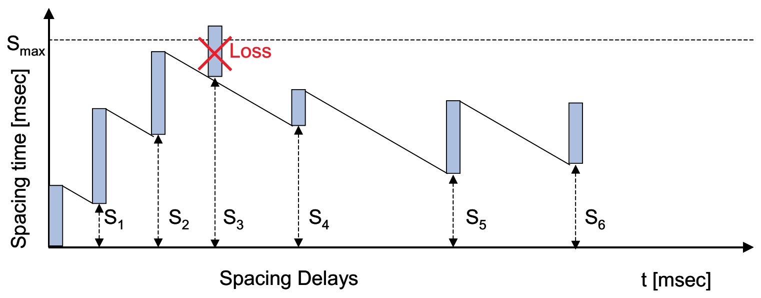

Figure 5: The state evolution of a spacer.

A semi-Markov process can be implemented at the arrival instants of the packets. They are important for computing their loss probability and waiting time distribution. The input variables are the packet size B and the subsequent inter-arrival time A. The resulting state transition function is given by

$$f((S),(B,A)) = \begin{cases} \max(S_n + B - A \cdot C, 0) & S_n + B \le S_{max} \\ \max(S_n - A \cdot C, 0) & S_n + B > S_{max} \end{cases} . \tag{35}$$

4.2 Multiplexing Voice over ATM

The following example is taken from [8]. In the asynchronous transfer mode (ATM), cells of 48 bytes payload are transported. Voice samples are on average 20 bytes large and do not utilize the bandwidth reasonably. Multiplexing voice samples into the payload of an ATM cell overcomes this problem. This is done by the ATM adaptation layer type 2 (AAL2). Figure 6 shows that voice samples are equipped with a multiplexing header of 3 bytes before they are put into the payload of a cell that starts with a 1 byte starter flag. Since voice has real-time requirements, the multiplexing time for a cell must be limited to prevent excessive delays.

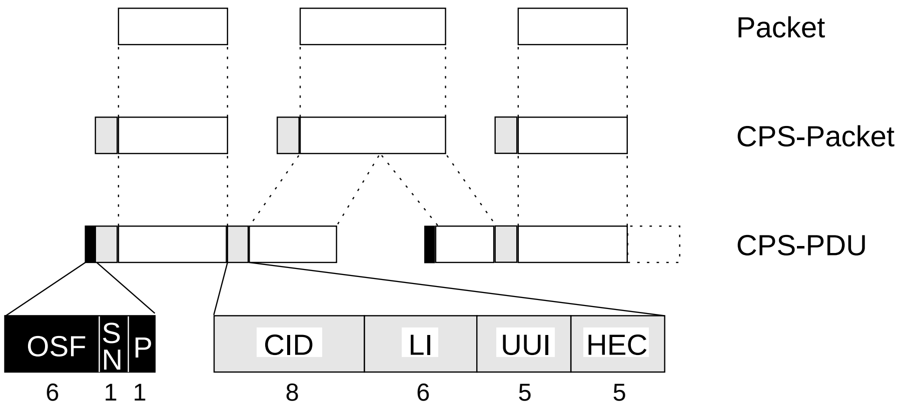

Figure 6: The AAL2 protocol

A model for AAL2 multiplexing works as follows. A cell has a capacity of F_{max} bytes. If packets of size B arrive, the fill state F of the cell is increased. If the packet exceeds the cell's size, a new cell is started. The age of the cell is recorded by the variable T. The first packet in the cell sets the age to zero and the passing time increases it. If T_{max} is exceeded, the cell is sent. The state comprises the fill state F and the age T of a cell. The input is the time since the last packet arrival A and the packet size B.

$$f((F_n, T_n), (A, B)) = \begin{cases} (B, 0) & T + A > T_{max} \\ (F + B, T + A) & T + A \leq T_{max} \wedge F + B \leq F_{max} \\ (F + B - F_{max}, 0) & T \leq T_{max} \wedge F + B > F_{max} \end{cases} \quad (36)$$

The state transition function Equation 36 must be interpreted shortly after the packet arrival time. If the age $T + A$ of the cell at packet arrival time is older than T_{max}, the multiplexing of that has been stopped before and a new cell is started. Therefore, the newly arrived packet of size B is put into a new cell and its age is set to zero. Otherwise, if the packet fits into the cell, the fill state and the age of the cell are updated. If the packet does not fit into the cell, the remaining bytes are put into the new cell and the age of the new cell is set to zero.

ATM's constant bitrate class (CBR) guarantees a fixed PCR and in return the cell stream must not exceed it, i.e., the minimum inter-cell distance is $\frac{1}{PCR}$ which has to be enforced by a spacer. Combining this model with the model of Section 4.1 yields a compound model for AAL2 multiplexing with additional PCR spacing.

For the computation of real world systems, the Markov chains were about 10^6 states large. Periodic systems can be constructed and solved by the presented algorithms. If a periodic system is only sightly changed, the eigenvalues of the resulting state transition matrix yield a bad convergence behavior. Figure 7 shows the benefit of the α-relaxation in terms of required iterations until the convergence criterion is met.

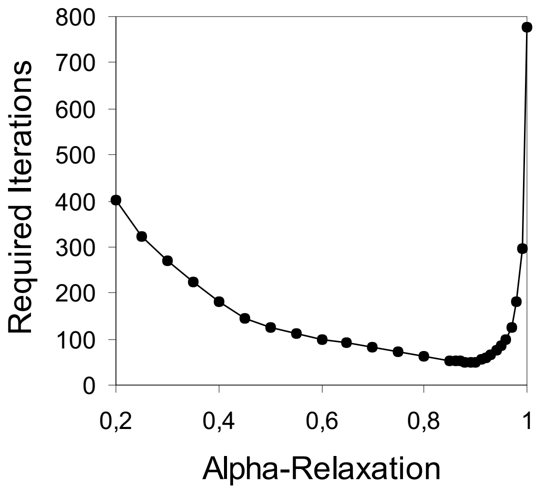

Figure 7: Computation speedup by α-relaxation

4.3 Multiplexing Voice over IP

We take the following example from [14]. Multiplexing voice over IP networks also yields a low bandwidth utilization due to a large header overhead. Sharing this header among several voice packets overcomes this problem, too. Similar to AAL2, a short multiplexing header is prepended to a voice packet (Figure 8).

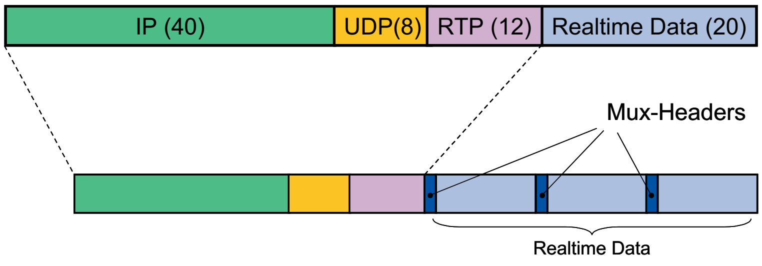

Figure 8: The tunneling and multiplexing protocol header suite

The difference to AAL2 multiplexing is that the maximum size of the IP packet is so large that it is not a limitation of the multiplexing process within a reasonable time. Therefore, the multiplexing time T_{max} is always limited by a timer. For a quality of service IP network like Integrated Services, a peak rate of the carried traffic stream must not be exceeded. Therefore, the resulting IP packet stream needs to undergo a spacing process. This is described in Figure 9.

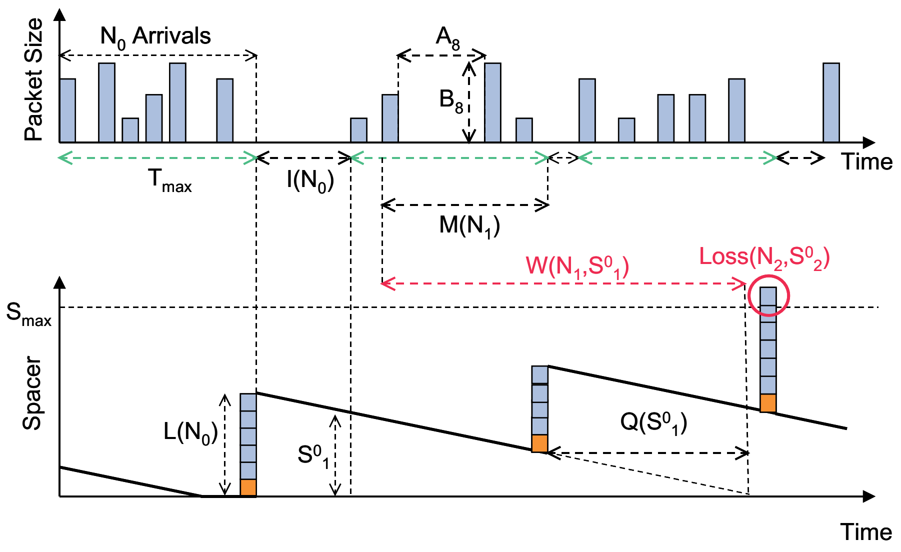

Figure 9: RTP multiplexing with subsequent spacing.

In principle, the model could be formulated like in Section 4.2 where the renewal points are set at packet arrival. A better solution is to embed the SMP at the renewal points of the spacer. Then, the arrivals of several packets can be summarized and the runtime of the program is shorter. Thus, the resulting state transition matrix P has a better convergence behavior.

Essentially, only the inputs for the spacer model (Section 4.1) must be delivered. The number of packets that arrive within the multiplexing time is given by a random variable N that can be computed if their inter-arrival time is given. The length of the resulting IP packet $L(N)$ depends on this number and is computable if the packet size distribution b is given. But also the inter-multiplex time $I(N)$ from the end of the multiplexing process until the arrival of the next packet that starts the next multiplexing process depends on N. This time can be computed if the inter-arrival time is given. Eventually, the compound model depends on the input $Y = (N, L(N), I(N))$. The number of arriving voice packets determines the size of the IP packet $L(N)$ and the inter-multiplex time $I(N)$ which are both the corresponding packet size and inter-arrival time for the above described spacer.

This example clearly shows the advantage of intelligent modelling and the use of conditional random variables.

5 Conclusion

We presented a new specification of discrete and finite Markov models. It is easy to apply as it is very close to the system behavior. We discussed several methods for the calculation of the stationary state distribution of a DTMC that are all based on iterations.

We suggested the α-relaxation $P(\alpha)$ for a state transition probability matrix P which reduces the spectral radius of P and increases the convergence speed by modifying the eigenvalues without changing the corresponding stationary distribution. This method can be combined very generally with other optimization methods, too.

We proposed a new algorithmic approach based on the presented specification for the computation of a single iteration step that does not require the state transition matrix explicitly. In addition, decomposition methods allow to decrease the numerical complexity of a single iteration step. These methods are numerically equivalent to vector-matrix multiplications but run faster and need less memory. With the new description, the complexity of the state transition function of the Markov model no longer limits the computation of the stationary state distribution of a Markov chain. Only the size of the state space and the convergence speed are limiting factors.

The model specification is suitable for syntactical conversion into numerical programs. We developed a prototype for specification converter that generates a numerical program. Many of the presented optimizations can be combined and are already implemented. Future work will elaborate our the prototype whenever new tasks require different features.

References

[1] N. U. Prabhu, M. Miyazawa, and H. Takagi, *Queuing Systems, Theory and Applicatioins - Advances in Discrete Time Queues*, vol. 18. J.C. Baltzer AG, 10 1994.

[2] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. Princeton: Princeton University Press, 1 ed., 1994.

[3] G. Latouche and P. Taylor, *Introduction to Matrix Geometric Methods in Stochastic Modeling*. Philadelphia, PA: SIAM, 1st ed., 1999.

[4] A. Brandt, P. Franken, and L. Bernd, *Stationary Stochastic Models*. Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1 ed., 1990.

[5] M. H. Ackroyd, “Computing the waiting time distribution for the G/G/1 queue by signal processing methods,” *IEEE Transactions on Communications*, vol. 28, pp. 52–58, January 1980.

[6] P. Tran-Gia and H. Ahmadi, “Analysis of a discrete-time $G^X/D/1-S$ queueing system with applications in packet–switching systems,” in *Proc. IEEE Infocom ’88*, (New Orleans), pp. 0861–0870, 1988.

[7] P. Tran-Gia and R. Dittmann, “A discrete-time analysis of the cyclic reservation multiple access protocol,” *Performance Evaluation*, vol. 16, pp. 185–200, 1992.

[8] M. Menth and N. Gerlich, “A Numerical Framework for Solving Discrete Finite Markov Models Applied to the AAL-2 Protocol,” in *MMB ’99, 10th GI/ITG Special Interest Conference*, (Trier), pp. 163–172, Sep. 1999.

[9] W. Feller, *An Introduction to Probability Theory and its Applications*. Wiley, 1957.

[10] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*. New York: Wiley, 2nd ed., 1985.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge: The MIT Press, 5th ed., 1991.

[12] B. Huppert, *Angewandte Lineare Algebra*. de Gruyter, 1990.

[13] U. Krieger, B. Müller-Clostermann, and M. Sczittnick, “Modeling and analysis of communication systems based on computational methods for markov chains,” in *JSAC*, vol. 8(9), pp. 1630 – 1648, Dec. 1990.

[14] M. Menth, “Analytical Performance Evaluation of Low-Bitrate Real-Time Traffic Multiplexing in UMTS over IP Networks,” *Journal of Interconnection Networks*, vol. 2, no. 1, pp. 147–174, 2001.