Description and Analysis of Markov Chains Based on Recursive Stochastic Equations and Factor Distributions

Michael Menth

University of Würzburg, Institute of Computer Science

Abstract. In this paper we propose a functional description of Markov chains (MCs) using recursive stochastic equations and factor distributions instead of the state transition matrix P. This new modelling method is very intuitive as it separates the functional behavior of the system under study from probabilistic factors. We present the "forward algorithm" to calculate consecutive state distributions x_n. It is numerically equivalent to the well-known vector-matrix multiplication method $x_{n+1} = x_n \cdot P$, but it can be faster and require less memory. We compare the operation and efficiency of the power method and MC simulation. Then, we propose several optimization techniques to speed up the computation of the stationary state distribution based on consecutive state distributions, to accelerate the forward algorithm, and to save its memory requirements. The presented concept has been implemented in a tool including all optimization methods. To make this paper easily accessible to novices, a tutorial-like introduction to MCs is given.

1 Introduction

Markov chains (MCs) are often used for modelling technical systems to evaluate their performance. Their stationary state distribution x is often computed as it gives insight into the average system behavior. We propose a new approach to compute x and several optimization methods. We briefly explain the contribution of this paper, contrast it with related work, and explain the organization of the following sections.

1.1 Contribution

A MC is usually described by a state transition matrix P and consecutive state distributions x_n can be computed by the vector-matrix multiplication $x_{n+1} = x_n \cdot P$. Applying this equation iteratively until the series of $\left(x_n\right)_{0 \leq n < \infty}$ converges is known as the power method and the converged results is the stationary state distribution x.

The contribution of this paper is a new functional description of MCs using recursive stochastic equations and factor distributions. It is applicable to MCs with a finite state space. We propose the so-called "forward algorithm" to compute consecutive state distributions x_n without using the state transition matrix. Thus, it offers an implementation alternative to the power method. We compare the operation and efficiency of the power method and the MC simulation with regard to the calculation of the stationary state distribution. We propose methods to speed up the convergence of the power method, to accelerate the computation of the forward algorithm, and to reduce its memory requirements. We proved the feasibility of our approach by a tool that turns the functional description into a numerical program to compute the stationary state distribution x including all optimization methods presented in this paper.

Our proposal has multiple benefits. The functional description simplifies the modelling of a technical system by separating the description of its behavior from influencing probabilistic factors. The forward algorithm can be significantly faster and require drastically less memory for MCs with large state spaces than the conventional vector-matrix multiplication. Apart from that, some of the proposed optimization methods can be applied only to the forward algorithm. The simplicity and efficiency of the new method allow the treatment of highly complex systems with multi-dimensional state spaces that otherwise cannot be solved.

1.2 Related Work

Most research regarding MCs has concentrated on analytical solutions of special queuing systems [1]. Matrix-analytical methods take advantage of specially structured transition matrices to achieve a fast computation of stationary state distribution x [2, 3]. Most analytical approaches make extensive use of the state transition matrix P, but they cannot solve large Markov models with 10^6 or more states since then even a sparse matrix representation of P consumes a lot of memory. The proposed functional description can be viewed as a further development, generalization, and formalization of the discrete-time analysis used in [4–6].

In [7] a framework for solving continuous-time Markov chains (CTMCs) is presented. In [8] an efficient strategy is proposed to generate the state space for multi-valued state decision diagrams. The authors of [9] present methods for CTMCs to accelerate the computation of their stationary state distribution and discuss options to avoid the explicit representation of the state transition matrix.

1.3 Organization of the Paper

In the next section, an introduction to the conventional description of MCs is given to make this paper self-contained for novices in the field. We explain the new functional description and the forward algorithm to compute consecutive state distributions in Section 3. Section 4 compares the operation and the efficiency of the power method and MC simulation. Section 5 describes several optimization methods to speed up the convergence of the power method, to accelerate the computation of the forward algorithm, and to minimize its memory requirements. Section 6 summarizes this work.

2 Introduction to Markov Chains

In this section we clarify our notation and give a short introduction to Markov chains (MCs). General MCs are usually described by a state transition matrix P which may have different properties. We introduce the stationary state distribution of a MC which is often needed for performance evaluation purposes. A simple example illustrates this concept.

2.1 Notation

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

- $\mathcal{X}$: range of X (contains discrete values)
- $|\mathcal{X}|$: cardinality of $\mathcal{X}$
- X_{min}, X_{max}: minimum and maximum value of X,
- $x[i]$: probability $P(X = i)$ with $i \in \mathcal{X}$,
- $x = (x[X_{min}], ..., x[X_{max}])$: distribution of X given as a vector of probabilities,
- $\overline{X} = \sum_{X_{min} \leq i \leq X_{max}} x[i] \cdot i$: mean of X.

A random variable X is conditioned on another random variable Y if its distribution $x(Y)$ depends on Y.

- $X(Y = j)$: random variable X conditioned on $Y = j$,
- $x(Y = j)$, $x[i](Y = j)$: distribution and probability of X conditioned on $Y = j$,
- $p(Y = j)$: probability p conditioned on $Y = j$,
- $p = \sum_{Y_{min} \leq i \leq Y_{max}} p(Y = j) \cdot y[j]$: unconditioning of the conditonal probability $p(Y = j)$ by y.

2.2 Markov Processes and Embedded Markov Chains

A stochastic process is characterized by the time-dependent random variable $X(t)$. $X(t)$ can have a finite or infinite state space $\mathcal{X}$. The time parameter is taken from the range of potential state transitions $t \in \mathcal{T}$. This range may be continuous or discrete. If the time indices $t \in \mathcal{T}$ can be numbered by t_n, $n \in \mathbb{N}_0$, consecutive states $X(t_n)$ form a sequence which can be denoted by $(X_n)_{0 \leq n < \infty}$. A set of random variables $\{X_n\}$ forms a Markov chain (MC) if the probability that the next value (state) is $X_{n+1} = j$ depends only upon the current value (state) $X_n = i$ and not upon previous values [10].

A discrete-time Markov chain (DTMC) is a stochastic process with a finite state space and potential transition points only at integer values, i.e. $t_i = i$, and its sequence X_n forms a Markov chain. The Markov property denotes that the evolution of the process depends only on its current state but not on past states, which is also called the memoryless property. This leads to a geometrically distributed sojourn time for every state. In case of continuous-time Markov chains (CTMCs), the sojourn time for all states is exponentially distributed, but we do not consider this other class of Markov chains in this paper.

Apart from DTMCs and CTMCs, there are other processes where the sojourn time in the states is neither geometrically nor exponentially distributed, but the sequence of consecutive states forms a Markov chain. These processes are called semi-Markov processes with embedded Markov chains at the transition points whereby transitions to the same state are possible. It has the memoryless property only at the embedded points.

This work provides an alternative description and analysis for general MCs, DTMCs, and embedded MCs, but not for CTMCs.

2.3 State Transition Matrix

In the following, we consider only *homogeneous* MCs that have the same state transition probabilities for all transition points. The state transition probability from state $i \in \mathcal{X}$ to state $j \in \mathcal{X}$ is then denoted by $p_{i,j}$ (see Equation (1)). All state transition probabilities can be grouped in a state transition matrix P (see Equation (2)) which is stochastic since the entries of each column amount to 1 (see Equation (3)).

$$p_{i,j} = P(X_{n+1} = j \,|\, X_n = i), \quad i, j \in \mathcal{X}, \quad n \in \mathbb{N}_0. \tag{1}$$

$$P = (p_{i,j})_{i,j \in \mathcal{X}} \tag{2}$$

$$\sum_{j \in \mathcal{X}} p_{i,j} = 1, \quad i \in \mathcal{X} \tag{3}$$

The observation of a MC starts at transition point t_0. The probability that the system is in state i at this time is given by $P(X_0 = i) = x_0[i]$ which yields 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} at the next transition point can be computed by a simple matrix multiplication:

$$x_{n+1} = x_n \cdot P. \tag{4}$$

The n-step transition probability $p_{i,j}^{(n)}$ is given by the n-step transition matrix $P^{(n)} = P^n$.

2.4 Markov Chain Classification

Markov chains can be classified according to the structure of their state transition matrix. We take the nomenclature regarding DTMCs from [11, 12] and refer to the graph theoretical concepts in [13].

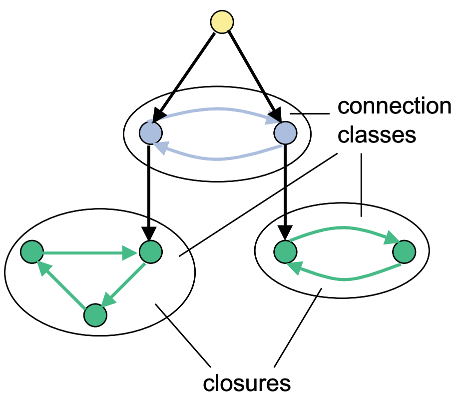

Fig. 1. Classification of states in the state transition graph of a DTMC.

The state transition graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by the set of vertices $\mathcal{V}$ which comprises the states $\mathcal{X}$, and the set of edges $\mathcal{E}$ which represents potential state transitions. A potential state transition from state i to state j exists 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 $i \rightarrow j$. State k is also accessible by state i if there is a state j so that $i \rightarrow j$ and $j \rightarrow k$ are true. Hence, $i \rightarrow j$ 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 both $i \rightarrow j$ and $j \rightarrow i$ 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 MC 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 $i \notin \mathcal{C}$ if there is a state $j \in \mathcal{C}$ so that $i \rightarrow j$ is true. Note that a connection class is either a closure or completely absorbed. Figure 1 illustrates the relationship between closures and connection classes.

A MC 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, and the state transition graph of the MC is strongly connected. A MC 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.

$$p = \min(m : \forall k, n \in \mathbb{N}_0 : (n \neq k \cdot m) \wedge (p_{i,i}^{(n)} = 0)). \tag{5}$$

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 $\mathcal{G}$ which is induced by the closure $\mathcal{C}$. An algorithm to find the period of an irreducible MC 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 MC are aperiodic, the MC is also called aperiodic, otherwise it is called periodic.

2.5 Markov Chain Analysis

For performance evaluation purposes, the long-term behavior of a MC is of interest, i.e., the average of the state distributions at all transition points:

$$x = \lim_{n\to\infty} \frac{1}{n} \sum_{0 \leq i < n} x_i. \tag{6}$$

Since P is a stochastic matrix, the limit x does always exist [14]. The average state distribution depends in general on the start distribution x_0 unless the MC is irreducible. The expression in Equation (6) converges only very slowly with increasing n. For aperiodic MCs, the series $(x_n)_{0 \leq n < \infty}$ also converges. Therefore, the average state distribution in Equation (6) can be computed more quickly by the iterative application of Equation (4) which is called the power method. In Section 5.1 we show how this principle can also be adapted to periodic MCs. The *stationary state distribution* x_s of the MC is defined as the left-hand eigenvector of P with eigenvalue 1:

$$x_s = x_s \cdot P. \tag{7}$$

The stationary distribution is unique for irreducible MCs but not for reducible MCs. For example, the identity matrix is reducible and any vector is a left-hand eigenvector. In any case, the average state distribution x is a stationary state distribution and depends on the start vector x_0.

2.6 Example

The description in [2] of the daily weather changes in Belfast (Northern Ireland) illustrates nicely the concept of a MC. 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:

$$P = \begin{pmatrix} 0.8 & 0.15 & 0.05 \\ 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \end{pmatrix} \tag{8}$$

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%.

3 Specification and Analysis of Embedded Markov Chains Using Recursive Stochastic Equations

In this section, we specify embedded MCs using recursive stochastic equations [15] and factor distributions instead of the state transition matrices. We take the $GI^{[GI]}/D/1-Q^{max}$ queue to illustrate the concept. Then, we generalize and formalize the approach and present an algorithm to calculate consecutive state distributions x_n based on our new description. Finally, we show that our new description has the same expressiveness as a state transition matrix.

3.1 Example: $\mathbf{GI^{[GI]}/D/1-Q^{max}}$ Queue

We consider the $GI^{[GI]}/D/1-Q^{max}$ queuing system that has been investigated in [5]. 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 at the arrival instants to derive blocking probabilities and waiting time distributions for customer requests.

We model the system state by the queue occupancy Q. Hence, we have $\mathcal{Q} = [0; Q^{max}]$. We embed a MC $\left(Q_n\right)_{0\leq n<\infty}$ with embedded points shortly before new customers arrive. The system starts with Q_0 customers in the queue. When a batch of customers arrives, clients that still fit into the queue are accepted, others are rejected. Thus, 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 it cannot fall below zero. Therefore, the queue occupancy at the next embedded point is determined by $Q_{n+1} = \max(Q' - A, 0) = \max(\min(Q_n + B, Q^{max}) - A, 0)$. To fully specify the model, we set $Q^{max} = 5$ and assume distributions for A and B that are given in Table 1.

Table 1. Batch size and inter-arrival time distributions 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

3.2 Formalization of the Functional Description

The model specification presented above is based on recursive stochastic equations and factor distributions. It describes the system behavior at embedded points in a very natural way. We generalize and formalize this approach and call it functional description.

The choice of the embedded points $\mathcal{T}$ determines the 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 we call them factors $Y = (B, A)$. Since A and B are both i.i.d. variables, the joint distribution of Y is given by $P(Y = (i, j)) = b[i]{\cdot}a[j]$. The system behavior is described by a recursive stochastic equation. In this context, we call it state transition function that can be generally defined as $f : (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathcal{X}$. In the above example, f is

$$Q_{n+1} = f(Q_n, (A, B)) = \max(\min(Q_n + B, Q^{max}) - A, 0). \tag{9}$$

Hence, the specification of the MC is given by $\mathcal{D} = (\mathcal{T}, \mathcal{X}, x_0, \mathcal{Y}, y, f)$. In general, both the state space $\mathcal{X}$ and the input space $\mathcal{Y}$ may be multi-dimensional.

3.3 Algorithmic Calculation of Consecutive State Distributions

The recursive stochastic equation and the distribution of the factors allow to calculate the state probability at consecutive embedded points by applying the law of total probability:

$$x_{n+1}[k] = \sum_{i\in\mathcal{X}, j\in\mathcal{Y}} P(X_{n+1} = k | X_n = i \wedge Y = j) \cdot x_n[i] \cdot y[j]. \tag{10}$$

The conditional probability can be computed by means of the state transition function:

$$P(X_{n+1} = k | X_n = i \wedge Y = j) = \begin{cases} 0 & \text{if } f(i, j) \neq k \\ 1 & \text{if } f(i, j) = k \end{cases} \tag{11}$$

which simplifies Equation (10) to

$$x_{n+1}[k] = \sum_{\{(i,j):f(i,j)=k, i\in\mathcal{X}, j\in\mathcal{Y}\}} x_n[i] \cdot y[j]. \tag{12}$$

Thus, we calculate the consecutive state probability $x_{n+1}[k] = \sum_{(i,j)\in\mathcal{Z}_k} x_n[i] \cdot y[i]$ by the sum of the compound probabilities of the tuples in the preimage $\mathcal{Z}_k = \{(i,j) : f(i,j) = k, i \in \mathcal{X}, j \in \mathcal{Y}\}$ of state k with respect to f. We call Equation (12) the *backward algorithm* because it requires the preimage of f. The preimage computation is a difficult task and limits the tractability of complex models. A simple rearrangement of the computation steps for all consecutive states $x_{n+1}[k]$ in Equation (12) leads to the *forward algorithm* which is given in Algorithm 1. It is numerically equivalent to Equation (12). The forward algorithm is simpler than the backward algorithm since it does not require the computation of the preimage because it uses the state transition function directly. Moreover, a program for the numerical computation of the stationary state distribution can be syntactically derived from the functional description $\mathcal{D}$. This is not feasible in an efficient way for the backward algorithm.

Input: state distribution x_n and factor 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: calculation of the consecutive state distribution based on the state transition function.

3.4 Derivation of the State Transition Matrix P

The equivalent state transition matrix P for a Markov model can be computed by

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

This equation corresponds to the backward algorithm. Rearranging the computation according to the forward algorithm simplifies Equation (13) to Algorithm 2 which does not require the calculation of the preimage of f. The state transition matrix of our example in Section 3.1 is

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

As the state transition matrix P can be derived from the functional description $\mathcal{D}$, $\mathcal{D}$ contains at least as much information as P. The functional description separates clearly factors Y from the model behavior f. If the factor distribution y changes in a technical model, it is just substituted by another distribution in $\mathcal{D}$ while the state transition function f remains unchanged. In contrast, the state transition matrix can change completely.

We have presented three different options for the calculation of a consecutive state distribution. The state transition matrix P may be used (Equation (4)), the backward algorithm (Equation (12)), or the forward algorithm (Algorithm 1). They are numerically identical. The state transition matrix for a system with 10^6 states has 10^{12} possible transitions which require a memory of 8 Terabytes when a single transition is described by an 8 bytes double precision floating point number. In this case, even a sparse matrix representation cannot be stored. Thus, the calculation of the consecutive state distribution using the transition matrix becomes infeasible. The forward algorithm requires memory only for x_n, x_{n+1}, and y, but not for the state transition matrix. Therefore, it does not run into this memory problem. In [16], we have calculated the stationary state distribution for a MC with about 10^6 states.

Input: factor 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: Calculation of the state transition matrix based on the state transition function according to the forward algorithm.

3.5 Comparison of Expressiveness

We show that both the functional description $\mathcal{D}$ and the state transition P have the same expressiveness. In the previous section we have shown that the state transition matrix can be derived from a functional description. Now, we construct a functional description $\mathcal{D} = (\mathcal{T}, \mathcal{X}, x_0, \mathcal{Y}, y, f)$ based on a state transition matrix P.

- The embedded points $\mathcal{T} = \{t_n = n \cdot \Delta : n \in \mathbb{N}_0\}$ of $\mathcal{D}$ are arbitrary.
- Without loss of generality we assume the state space $\mathcal{X} = [0; X_{max}]$ for the sake of a simple notation in the following. Then, the state transition matrix P has $X_{max}+1$ columns and rows. The state space for the equivalent functional description is the same.
- The start state distribution x_0 is arbitrary.
- We define auxiliary variables $s_{i,j} = \sum_{0 \leq k \leq j} p_{i,k}$ to construct the factor space $\mathcal{Y}$ and the factor distribution y. We further define the vector s^* that contains $\{s_{i,j} : i, j \in \mathcal{X}\}$ in an ascending order by value, double elements are suppressed. The length of s^* is denoted by $len(s^*)$. 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 factor distribution by $y[0] = s^*[0]$, $y[i] = s^*[i] - s^*[i-1]$, and $\mathcal{Y} = [0; len(s^*) - 1]$.
- We define the state transition function $f(i,k) = j$ by its preimage $\mathcal{Z}_{i,j} = \{k : k \in \mathcal{Y}, f(i,k) = j\}$. This is done as follows. Since $p_{i,0}$ corresponds to $s_{i,0}$, there is a $k_{i,0}$ such that $\sum_{0 \leq h \leq k_{i,0}} y[h] = p_{i,0}$. Thus, we assign the factor values $[0; k_{i,0}]$ to $\mathcal{Z}_{i,0}$. Similarly, there is a $k_{i,j}$ for every $p_{i,j}$ with $j > 0$ and $\sum_{k_{i,j-1} \leq h \leq k_{i,j}} y[h] = p_{i,j}$; thus, we assign the factors $[k_{i,j-1}, k_{i,j}]$ to $\mathcal{Z}_{i,j}$.

To illustrate this small but rather complex construction, we apply it to the state transition matrix Equation (8) for our example in Section 2.6. The most complex parts of the functional description are the factor distribution y, which is given explicitly in Table 2, and the state transition function f which is given by its preimage $\mathcal{Z}$ in Equation (15). The other parts of $\mathcal{D} = (\mathcal{T}, \mathcal{X}, x_0, \mathcal{Y}, f)$ are trivial.

Table 2. Factor distribution y for the functional descriptions of the state transition matrix in Equation (8).

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

$$\mathcal{Z} = \begin{pmatrix} \{1,2,3\} & \{4,5\} & \{6\} \\ \{1,2\} & \{3,4\} & \{5,6\} \\ \{1\} & \{2,3\} & \{4,5,6\} \end{pmatrix} \tag{15}$$

After all, we have shown that the functional description and the state transition matrix are equivalent approaches for the specification of discrete and finite MCs. The state transition matrix is the traditionally used description. If the state transition probabilities can be taken directly from empirical data like in the example regarding the weather in Belfast, then the transition matrix is a simple approach to model the system. However, in technical systems the system behavior is known and provides the state transition function f and the factor distribution is also available. Therefore, the functional description seem to be the natural way of modelling.

4 Markov Chain Simulation

MCs can be simulated to evaluate their average state distribution x. We briefly explain MC simulation based on the state transition matrix P and on the new functional description $\mathcal{D}$. Then, we compare the efficiency of MC simulation and the power method for MC analysis.

4.1 Markov Chain Simulation Based on the State Transition Matrix and the Functional Description

For the calculation of the successor state in a MC simulation, the current state $X_n = i$ and an equally distributed pseudo random number $U \in (0;1)$ are required. If the state transition matrix $P = (p_{i,j})_{i,j\in\mathcal{X}}$ is given, the successor state can be calculated by $X_{n+1} = max(j : \sum_{0\leq k<j} p_{i,k} < U)$.

If a functional description $\mathcal{D}$ is given, the pseudo random number is used to realize a random variable for the factor Y and the successor state is then calculated by the state transition function $X_{n+1} = f(X_n, Y)$. Both alternatives lead one item of the series $(X_n)_{0\leq n<\infty}$.

The simulation records the relative frequencies of the simulated states and uses them as estimates of state probabilities. Appropriate statistical methods provide confidence intervals for these values [17]. In general, these probabilities depend on the start state X_0. If the underlying MC is irreducible, the simulation is expected to yield the same state probabilities for all start vectors (see Section 2.5).

4.2 Comparison of Simulation and the Power Method for MC Analysis

We compare the operation and efficiency of MC simulation and the power method for the calculation of the average state distribution x.

Comparison of Operation To illustrate the operation of the power method and the simulative MC analysis, we consider the transition graph of a MC in Figure 2.

MC simulation takes a random walk through the transition graph according to the state transition probabilities $p_{i,j}$ and constructs a series $(X_n)_{0\leq n\leq n_{max}}$. Thereby, the state space is explored and the relative frequency of the state visitations provides an estimate of the state probabilities. This is visualized in Figure 3. A single iteration step is quite cheap, but it reflects only a single transition. Millions or billions of such simulation steps are usually needed to calculate sufficiently accurate state probabilities.

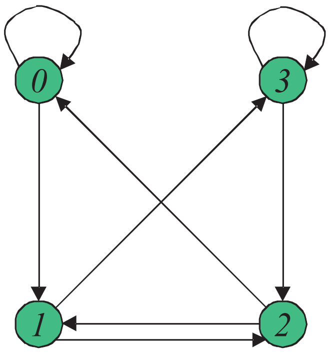

Fig. 2. Transition graph of a small MC example.

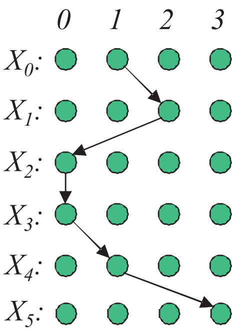

Fig. 3. The first 5 transition steps for MC simulation.

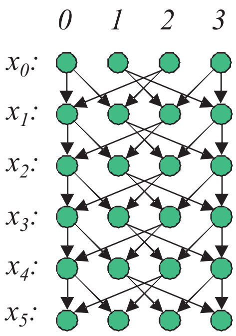

Fig. 4. The first 5 transition steps for the power method.

The power method takes into account all possible transition steps in a single vector-matrix multiplication in the sense that probability mass of state distribution x_n is propagated from all states over all possible transitions to potential successor states which leads the new state distribution x_{n+1}. This is illustrated in Figure 4. Here, a single iteration step requires a lot of computation effort, but usually a few tens or hundreds iteration steps are sufficient for accurate results.

Comparison of Efficiency We study the convergence speed of the power method. For aperiodic MCs, the series $(x_n)_{0 \leq n < \infty}$ converges to the average state distribution x. The convergence speed of the matrix multiplication method is exponential, i.e., $\| x - x_n \| \leq |\lambda|^n$. The parameter λ is the eigenvalue of the state transition matrix P with the largest absolute value smaller than 1, which is also called the subdominant eigenvalue. For an accuracy of $\| x - x_n \| \leq \epsilon$, $n \geq \frac{\ln \epsilon}{\ln \lambda}$ iteration steps are required. Thus, the computation effort increases linearly with an exponentially decreasing accuracy ϵ.

In contrast, the simulative approach needs more than $n > \frac{1}{\epsilon}$ samples for an accuracy of ϵ; otherwise, a probability of ϵ cannot be simulated. Thus, simulations require an exponentially increasing number of samples if the accuracy ϵ decreases exponentially.

Hence, the power method yields results for the average state distribution faster than MC simulation when high accuracy is required. However, MC simulations may be useful to quickly find a suitable start vector x_0 for the power method. The optimization method in Section 5.9 takes advantage of this idea.

5 Optimization Methods

The following optimizations aim at accelerating the convergence of the consecutive state distribution and some others make the computation of a single iteration step faster or less memory-consuming. Some of them are well-known and some others are new, in particular those that rely on the functional description of MCs.

5.1 Convergence Speedup by the Use of the Limiting Distribution for Periodic MCs

If a MC is aperiodic, the limiting distribution $x = \lim_{n\to\infty} x_n$ exists [14], which is also called the Cesaro limit. If $\left(x_n\right)_{0\leq n<\infty}$ converges, it is obvious that it converges faster than the series of the averaged distributions (cf. Equation (6)). The inequality

$$\| x_n - x_{n-1} \|_\infty < \epsilon \tag{16}$$

may be used as a convergence criterion where $\| \dot{\|}_\infty$ denotes the maximum norm. This criterion works well in practice, but it does not assure the accuracy $\| x_n - x \|_\infty < \epsilon$ of the result.

In case of a periodic MC, the limiting distribution of x_n does not exist. The state space $\mathcal{X}$ of a p-cyclic MC is partitioned into p *periodic classes* $\mathcal{X}_i$, $0 \leq i < p-1$ and a single-step transition is only possible from a state of class j to a state of class $\mathcal{X}_{((j+1) \mod p)}$. However, p transition steps lead 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 potentially aperiodic subchain. In this case, p cyclo-stationary distributions $x^{(i)} = \lim_{n\to\infty} x_{n\cdot p+i}$ exist. Finally, the average state distribution of a periodic chain can be computed by

$$x = \lim_{n\to\infty} \frac{1}{n} \sum_{0\leq k<n} x_k = \frac{1}{p} \sum_{0\leq i<p} x^{(i)} = \lim_{n\to\infty} \frac{1}{p} \sum_{n\leq k<n+p} x_k. \tag{17}$$

The righthand expression converges faster than the definition in Equation (6). The convergence criterion in Equation (16) can be adapted to cyclo-stationarity by

$$\| x_n^{(i)} - x_{n-1}^{(i)} \|_\infty < \epsilon \quad \text{, i.e.,} \; \| x_{n+i} - x_{n-p+i} \|_\infty < \epsilon \text{ for } \quad 0 \leq i < p. \tag{18}$$

5.2 Convergence Speedup by the Use of Relaxation

Relaxation is a means to modify the eigenvalues of a matrix to make its convergence radius smaller and to achieve thereby faster convergence. Usually, relaxation is done with Gauss-Seidel or Jacobi methods [2, 18]. Our new approach uses a modification of P to $P(\alpha)$ by $P(\alpha) = \alpha \cdot P + (1 - \alpha) \cdot I$ with $0 \leq \alpha \leq 1$ where I is the identity matrix. This is also called preconditioning. $P(\alpha)$ remains stochastic if $\alpha \in (0; 1)$ holds and its eigenvector x to the eigenvalue 1 is the same as the one for P:

$$x \cdot P(\alpha) = \alpha \cdot x \cdot P+(1-\alpha) \cdot x \cdot I=\alpha \cdot x+(1-\alpha) \cdot x=x. \tag{19}$$

The α-relaxation essentially corresponds to the computation of the consecutive state distribution by a moving average: $x_{n+1} = x_n \cdot P(\alpha) = \alpha \cdot x_n \cdot P + (1 - \alpha) \cdot x_n$. We use this observation to implement this optimization method in our software tool.

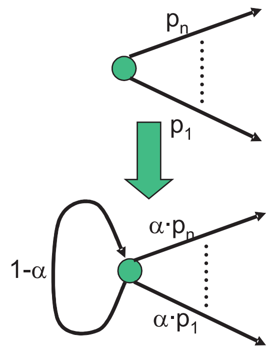

Fig. 5. α-Relaxation induces loops at every node in the state transition graph of the MC and destroys thereby periodicity.

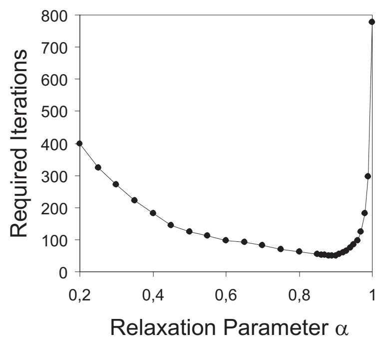

Fig. 6. Convergence speedup by α-relaxation for the AAL2/ATM multiplexing model with subsequent cell spacing.

The effect of the α-relaxation on the state transition matrix is twofold: it turns periodic MCs into aperiodic MCs and it increases the convergence speed. Figure 5 shows that α-relaxation introduces an additional transition in the state transition graph and destroys thereby periodicity. As the α-relaxation does not change the stationary state distribution, the calculation of the Cesaro limit of $P(\alpha)$ is an elegant means to find the stationary distribution of a periodic MC. Periodic MCs have several complex eigenvalues of size 1. The α-relaxation removes the periods and changes the complex eigenvalues to 1 or to other values smaller than 1. Thus, both the size and the phase of the eigenvalues are changed by this kind of preconditioning. As $P(\alpha)$ is stochastic, all eigenvalues remain smaller than or are equal to 1. The largest eigenvector smaller than

1 determines the convergence radius and thereby the convergence speed of the series. Thus, α is required to minimize the resulting subdominant eigenvector. It should not be chosen too small as this also limits the convergence speed:

$$\| x_n \cdot P(\alpha) - x_n \|_\infty = \| \alpha \cdot x_n \cdot (P - I) \|_\infty = \alpha \cdot \| x_n \cdot P - x_n \|_\infty \quad (20)$$

For many applications $\alpha = 0.8$ is a good value. Nearly decomposable MCs are almost periodic, i.e., the state transitions avoiding the periodicity have very little probability. Their series of $\left(x_n\right)_{0 \leq n < \infty}$ converges only very slowly. In this case, α-relaxation achieves usually a substantial calculation speedup.

The multiplexing of compressed voice into AAL2/ATM with subsequent cell spacing has been modelled in [19, 20] by a three-dimensional state space. For some system setting, the resulting MC turned out to be almost periodic. Figure 6 shows the number of required iterations until the convergence criterion is met depending on the α-value for the α-relaxation. With $\alpha = 0.9$ the number of required iteration steps can be reduced from 800 to 100.

5.3 Computation Speedup by Zero-State Omission

Zero-state omission excludes states i with probability zero $(x_n[i] = 0)$ from multiplication when the consecutive state distribution is calculated. This can be easily done in Algorithm 1 by checking $x_n[i] > 0$ before entering the second for-loop. Though this optimization looks quite simple, it is very effective. In addition, connection classes that are never entered due to a specific x_0 do not consume any computation time.

5.4 Computation Speedup and Memory Savings by the Use of a Sparse Matrix

A sparse matrix is a smart data structure storing only its non-zero entries. As most state transitions in many Markov models have probability zero, a sparse matrix representation of the state transition matrix leads to considerable memory savings. In addition, the missing entries save CPU cycles for the matrix multiplication when the consecutive state distribution is calculated by Equation (4). As our new approach does not use the state transition matrix, it cannot take advantage of this optimization method.

5.5 Computation Speedup by Matrix Powering

The state distribution x_{n+1} can be computed either iteratively by Equation (4) or by $x_{n+1} = x_0 \cdot P^{n+1}$ which requires P^{n+1}. The series of $\left(P^{2^n}\right)_{0 \leq n < \infty}$ can be quickly calculated by matrix powering $P^{2^n} = P^{2^{n-1}} \cdot P^{2^{n-1}}$. It presents a subseries of the series $\left(P^n\right)_{0 \leq n < \infty}$ and accelerates the calculation of the limiting distribution in Section 5.1 significantly.

At first glance, this method is quite appealing, but at second glance, it is often not beneficial. Matrix-matrix multiplication requires $|\mathcal{X}|^3$ scalar multiplications while vector-matrix multiplication requires only $|\mathcal{X}|^2$ scalar multiplications. Hence, the power method based on vector-matrix multiplication is computationally cheaper as long as the number of required iterations n^* is smaller than $|\mathcal{X}| \cdot \frac{log(n^*)}{log(2)}$. The state space can become very large, e.g., 10^6 states [16] while the number of iteration steps is mostly in the order of 10^3 or smaller. In addition, the vector-matrix multiplication can be 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 the matrix P^n is usually fully occupied even if matrix P is sparsely occupied so that a sparse matrix representation does not work.

5.6 Memory Savings and Computation Speedup by the Functional Description and the Forward Algorithm

A matrix with 10^6 states has 10^{12} possible transitions which require a memory of 8 Terabytes if each probability is described by a double precision floating point number. This state transition matrix is too large to be constructed explicitly, even as a sparse matrix. The forward algorithm in Section 3.3 relies on the new functional description of the MC and does not need the state transition matrix to calculate the consecutive state distributions. As a consequence, large MCs can be solved without extensive memory [16]. In addition, a vector matrix multiplication like in Equation (4) of Section 2.3 requires $|\mathcal{X}|^2$ multiplication and addition steps whereas Algorithm 1 requires $|\mathcal{X}| \cdot |\mathcal{Y}|$ multiplication and addition steps. This is faster if $|\mathcal{X}| > |\mathcal{Y}|$ holds.

5.7 Computation Speedup by Decomposition of the State Transition Function

The factor space can often be written in product form $\mathcal{Y} = \times_{0 \leq i < k} \mathcal{Y}^i$. Then, the state transition function f may be decomposed into $f = f^0 \circ ... \circ f^{k-1}$. The forward algorithm in Section 3.3 is then applied k times, namely for each state transition function separately. This requires the k-fold replication of the state space ($\mathcal{X}^i$, $0 \leq i < k$, $\mathcal{X} = \mathcal{X}^0$) to record the intermediate distributions. We apply this to the example in Section 3.1 by

$$Q_n^1 = f^0(Q_n^0, B) = \min(Q_n^0 + B, Q_{max}) \tag{21}$$

$$Q_{n+1}^0 = f^1(Q_n^1, A) = \max(Q_n^1 - A, 0) \tag{22}$$

With this decomposition method, the state transition function is now calculated once for each realization of the factor $\mathcal{Y}^0 \cup ... \cup \mathcal{Y}^{k-1}$ and leads to a computation complexity of $O(|\mathcal{X}| \cdot \sum\limits_{0 \leq i < k} |\mathcal{Y}^i|)$. With the conventional approach, the state transition function is calculated for each combination of the different factor realizations $\mathcal{Y}^0 \times ... \times \mathcal{Y}^{k-1}$ and leads to a computation complexity of $O(|\mathcal{X}| \cdot \prod\limits_{0 \leq i < k} |\mathcal{Y}^i|)$. Thus, decomposition yields considerable time savings for large multi-dimensional factor distributions. A practical application example of this optimization technique can be found in [21, 22].

5.8 Computation Speedup by Conditional Factors

We consider again the $GI^{GI}/D/1 - Q_{max}$ queue in Section 3.1. We modify it by a conditional arrival process, i.e., the arrival rate depends on the present queue occupancy

Q and we denote that system by $GI^{GI}(Q)/D/1 - Q_{max}$. This dependency can be modelled by different random variables A^i for the interarrival process and an if-clause in the state transition function:

$$f^1(Q_n^1, A^0, ..., A^{Q_{max}}) = \begin{cases} \max(Q_n^1 - A^0, 0) & \text{if } Q = 0 \\ ... \\ \max(Q_n^1 - A^{Q_{max}}, 0) & \text{if } Q = Q_{max} \end{cases} \tag{23}$$

However, this is very time-consuming since Algorithm 1 needs to step through a loop for the distribution of each A^i, thus Q_{max} loops are stepped through in vain. A conditional factor $A(Q^1)$ solves that problem and Equation (23) can be rewritten as

$$f^1(Q^1, A(Q^1)) = \max(Q^1 - A(Q^1), 0). \tag{24}$$

Depending on the value of state Q^1, the corresponding distribution for $A(Q^1)$ is chosen before entering the (next) factor loop in Algorithm 1. Conditional factors both accelerate the computation tremendously and allow to model problems in more detail. This method has been applied in [22].

5.9 Convergence Speedup with Start Vector Initialization by Simulation

In Section 4 we have shown that the power method leads faster to accurate estimates of the average state distribution than MC simulation. While MC simulation computes an average state distribution with an accuracy of 10^{-5} quite quickly, it takes much longer to achieve an accuracy of 10^{-10}. In contrast, the power method takes only twice the time to achieve an accuracy of 10^{-10} as compared to an accuracy of 10^{-5}. Therefore, MC simulation can be used to quickly get a rough estimate of the average state distribution. The resulting state distribution may be used as start vector x_0 for the power method because it is more efficient in ranges of high accuracy. Experience has shown that 10^7 steps can be simulated quite quickly and provide an accuracy of already about $\| x_1 - x_0 \|_\infty = 10^{-5}$.

5.10 Memory Savings, Convergence and Computation Speedup by Intelligent Modelling

The major optimization potential at all is intelligent modelling. The choice of the embedded points is especially important as it influences the state space, the factor space, and the convergence speed of the power method. Two consecutive embedded points of the same embedded MC should cover a sufficiently large time interval so that sufficiently many changes happen in the real-world system in the corresponding time. This assures a fast convergence speed. Furthermore, the state space and the factor space must be kept small. In particular, multi-dimensional state spaces should be avoided.

6 Conclusion

In this paper, we gave a short tutorial of Markov chains (MCs) which are usually described by a state transition matrix P. We proposed a new functional description $\mathcal{D}$ to model general MCs or also embedded MCs using recursive stochastic equations and factor distributions. It is easy to apply because it captures the behavior of the system under study in a very intuitive way. We proposed the “forward algorithm” to calculate consecutive state distributions x_n. It is usually faster and requires less memory than the equivalent vector-matrix multiplication $x_{n+1} = x_n \cdot P$. This is part of the power method to compute the average or stationary state distribution. We compared the operation and efficiency of MC simulation and the power method for that purpose and showed that the power method is more efficient in ranges of high accuracy. Then, we presented various methods to speed up the convergence of the power method, to accelerate the computation of the forward algorithm, and to reduce its memory requirements.

We have built a converter that takes the functional description $\mathcal{D}$ and composes a numerical program implementing the forward algorithm so that the average state distribution of a MC can be easily and quickly computed. Moreover, we have implemented all presented optimization methods in the tool unless they are applicable only to the state transition matrix. The study in [16] was performed with that tool and a MC with more than 10^6 states was analyzed.

References

1. Prabhu, N.U., Miyazawa, M., Takagi, H.: Queuing Systems, Theory and Applicatioins - Advances in Discrete Time Queues. Volume 18. J.C. Baltzer AG (1994)
2. Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. 1 edn. Princeton University Press, Princeton (1994)
3. Latouche, G., Taylor, P.: Introduction to Matrix Geometric Methods in Stochastic Modeling. 1st edn. SIAM, Philadelphia, PA (1999)
4. Ackroyd, M.H.: Computing the Waiting Time Distribution for the G/G/1 Queue by Signal Processing Methods. IEEE Transactions on Communications **28** (1980) 52–58
5. Tran-Gia, P., Ahmadi, H.: Analysis of a Discrete-Time $G^X/D/1 - S$ Queueing System With Applications in Packet-Switching Systems. In: IEEE Infocom, New Orleans (1988) 861–870
6. Tran-Gia, P., Dittmann, R.: A Discrete-Time Analysis of the Cyclic Reservation Multiple Access Protocol. Performance Evaluation **16** (1992) 185–200
7. Pranevicius, H., Germanavicius, V., Tumelis, G.: Automatic Creation of Numerical Models of Systems Specified by PLA Method. In: 12^{th}International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA), Riga, Latvia (2005)
8. Ciardo, G., Lüttgen, G., Siminiceanu, R.: Saturation: An Efficient Iteration Strategy for Symbolic State-Space Generation. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Genova, Italy (2001)
9. Deavours, D.D., Sanders, W.H.: “On-the-Fly” Solution Techniques for Stochastic Petri Netws and Extensions. IEEE Transactions on Software Engineering **24** (1998) 889–902
10. Kleinrock, L.: Queueing Systems. 1^{st} edn. Volume 1: Theory. John Wiley & Sons, New York (1975)
11. Feller, W.: An Introduction to Probability Theory and its Applications. Wiley (1957)
12. Gross, D., Harris, C.M.: Fundamentals of Queueing Theory. 2nd edn. Wiley, New York (1985)
13. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. 5th edn. The MIT Press, Cambridge (1991)
14. Huppert, B.: Angewandte Lineare Algebra. de Gruyter (1990)

15. Brandt, A., Franken, P., Bernd, L.: Stationary Stochastic Models. 1 edn. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons (1990)
16. Köhler, S., Menth, M., Vicari, N.: Analytic Performance Evaluation of the RED Algorithm for QoS in TCP/IP Networks. In: 9^{th} IFIP Working Conference on Performance Modelling and Evaluation of ATM & IP Networks, Budapest, Hungary (2001) 178–190
17. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. 3^{rd} edn. McGraw-Hill (2000)
18. Krieger, U., Müller-Clostermann, B., Sczittnick, M.: Modeling and Analysis of Communication Systems Based on Computational Methods for Markov Chains. IEEE Journal on Selected Areas in Communications **8(9)** (1990) 1630 – 1648
19. Menth, M., Gerlich, N.: A Numerical Framework for Solving Discrete Finite Markov Models Applied to the AAL-2 Protocol. In: 10^{th} GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), Trier (1999) 163–172
20. Menth, M.: Carrying Wireless Mobile Traffic over ATM – A Performance Study. Master's thesis, University of Würzburg, Faculty of Computer Science (1998)
21. Menth, M.: The Performance of Multiplexing Voice and Circuit Switched Data in UMTS over IP Networks. In: Protocols for Multimedia Systems (PROMS), Cracow, Poland (2000) 312 – 321
22. Menth, M.: Analytical Performance Evaluation of Low-Bitrate Real-Time Traffic Multiplexing in UMTS over IP Networks. Journal of Interconnection Networks **2** (2001) 147–174