8 Discrete-Time Markov Chains (DTMCs) and Semi-Markov-Processes

Notation

$\mathcal{X}$: Wertebereich der ZV X
$P(X = i) = x[i]$: Wahrscheinlichkeit, dass die ZV X den Wert i annimmt.
$x = (x[0], \dots, x[n-1])$: Wahrscheinlichkeiten für alle möglichen Werte der ZV X als Vektor
p_{ij}: Übergangswahrscheinlichkeit von Zustand i zu Zustand j einer Markovkette
X_i: i-te ZV einer Zeitreihe (in diesem Kapitel durch eine Markovkette erzeugt)
$x_i = (x_i[0], \dots, x_i[n-1])$: Zustandsverteilung der Markovkette nach dem i-ten Übergang

8.1 Basics about DTMCs

8.1.1 Definition

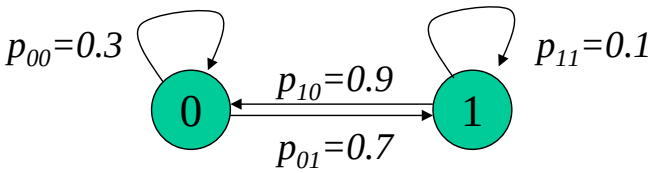


Abbildung 1: State transition graph or state transition diagram.

- Stochastic process
- States (e.g. 0 and 1)
- Start state: X_0
- State transition at certain times $t_i, i = 0, 1, 2, \dots$
- State after transition i: X_i
- Series of X_i forms a Markov chain
- Probability for transition from state i to state j: p_{ij}

8.1.2 Markov Chain Simulation

MCs can be simulated to calculate their average state distribution x.

- The simulation state is X
- Simulation calculates series X_n to find x
- Successor state $X_{n+1} = k$ for $X_n = i$ needs to be simulated
  - Random number $U \in (0; 1)$
  - Given $X_n = i,\ X_{n+1} = min\left(k: \sum_{0 \le j \le k} p_{ij} \ge U\right)$

8.1.3 Sojourn time

- How long does the system remain in state j?
  - $P(k\ steps) = {p_{jj}}^{k-1} \cdot (1 - p_{jj})$
  - Distributed according to the *shifted* geometric distribution

8.1.4 Calculation of consecutive state distributions

- State transition matrix: $P = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix}$
  - Stochastic matrix: sum over row is 1.
- State distribution after transition i: $x_i = \big(P(X_i = 0), P(X_i = 1)\big)$
- $x_{i+1} = x_i \cdot P$
- Example
  - $P = \begin{pmatrix} 0.3 & 0.7 \\ 0.9 & 0.1 \end{pmatrix}$
  - $\mathrm{x}_0 = (1{,}0), \mathrm{x}_1 = (0.3{,}0.7), \mathrm{x}_2 = (0.72{,}0.28), \mathrm{x}_3 = (0.468{,}0.532), \mathrm{x}_4 = (0.6192{,}0.3808), \mathrm{x}_5 = (0.52848{,}0.47152), \ldots, \mathrm{x}_{14} = (0.56284{,}0.43716), \mathrm{x}_{15} = (0.56229{,}0.43771), \mathrm{x}_{16} = (0.56262{,}0.43738), \ldots$

8.1.5 Average state distribution (Cesàro Limit)

- Average state distribution $x_s = \lim_{n\to\infty} \left(\frac{1}{n+1} \cdot \sum_{i=0}^{n} x_i\right)$
- Also called *stationary state distribution*
- Useful to compute performance measures
- x_s is left-hand eigenvector of P: $x_s = x_s \cdot P$
- If series of x_i converges, the resulting limit is equal to the average state distribution.

8.1.6 k-transition probabilities

- Probability that system changes after k steps from state i to state j: p_{ij}^k
- Corresponding transition matrix $P^{(k)}$ satisfies $x_{i+k} = x_i \cdot P^{(k)} \Rightarrow P^{(k)} = P^k$

8.1.7 Matrix powering

- $P = \begin{pmatrix} 0.3 & 0.7 \\ 0.9 & 0.1 \end{pmatrix}$, $P^2 = \begin{pmatrix} 0.72 & 0.28 \\ 0.36 & 0.64 \end{pmatrix}$, $P^4 = \begin{pmatrix} 0.6192 & 0.3808 \\ 0.4896 & 0.5104 \end{pmatrix}$, $P^8 = \begin{pmatrix} 0.56985 & 0.43015 \\ 0.55305 & 0.44695 \end{pmatrix}$, $P^{16} = \begin{pmatrix} 0.56262 & 0.43738 \\ 0.56234 & 0.43766 \end{pmatrix}$, $P^{32} = \begin{pmatrix} 0.56250 & 0.43750 \\ 0.56250 & 0.43750 \end{pmatrix}$
- Matrix powering may quickly approximate $P^\infty = \begin{pmatrix} x_s \\ x_s \end{pmatrix}$
  - Efficient for small matrices

8.2 Examples for Discrete-Time Markov Chains

8.2.1 The Weather in Belfast

- The description in [Stewart] of the daily weather changes in Belfast (Northern Ireland) illustrates nicely the concept of a MC.
- [Stewart] Stewart, W.J.: Introduction to the Numerical Solution of Markov Chains. 1 ed. Princeton University Press, Princeton (1994)
- 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:

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

  On a rainy day, the probability that tomorrow is rainy again is 80%.
- The average state distribution can be computed:

$$P^{\infty} = \begin{pmatrix} 0.7625 & 0.1688 & 0.0688 \\ 0.7625 & 0.1688 & 0.0688 \\ 0.7625 & 0.1688 & 0.0688 \end{pmatrix}$$

  It 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%.

8.2.2 Analysis of Monopoly Using Markov Chains

Question: what is the visiting probability for fields in Monopoly?
Model

- Each field corresponds to a state
- Transition probability to other fields is influenced by two dices and event cards
- Set up state transition matrix
- Stationary state distribution may be taken as visiting probability for fields under the assumption that sufficiently many rounds are played.

See: Jörg Bewersdorff: Glück, Logik und Bluff: Mathematik im Spiel – Methoden, Ergbnisse und Grenzen, Vieweg Verlag, 3. Auflage, 2003, Kapitel 1.16

8.2.3 Analysis of Web Pages: Google's PageRank Algorithm

Question: how important is a web site?
Answer: depends on how often the web page is referenced by other pages and how important the referencing web pages are!
Model

- Each web page w is a state
- Let n_w be the number of links of a web page w; then the state transition probability to their referenced web pages is set to $1/n_w$
- The stationary state distribution yields an estimate of the importance of the web pages.

This idea served in the early days of Google (~ 1999) to return most important web pages first. This idea led to the foundation of Google. Nowadays Google uses more complex, proprietary algorithms. Ideas of PageRank may still be a part of them.

See:

- https://de.wikipedia.org/wiki/PageRank
- Sergei Brin, Lawrence Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In: Computer Networks and ISDN Systems, Band 30, 1998, S. 107–117, http://infolab.stanford.edu/~backrub/google.html

8.3 Where do we find Markov chains?

8.3.1 Discrete-Time Markov Chains (DTMCs)

- Discrete-time stochastic process
- Transitions according to a transition matrix
  - Sequence X_i forms a Markov chain
- Transition points only at integer values, i.e., $t_i = i$
  - This leads to geometrically distributed sojourn times (in terms of time, not only in terms of transition steps) for every state j. More specifically: distributed according to the *shifted* geometric distribution
  - A DTMC has the Markov property, i.e., it is "memoryless".
  - Its evolution depends only on its current state, but not on its past, at any discrete observation point.

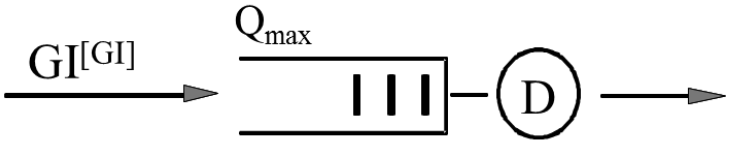
8.3.2 Semi-Markov Processes (SMPs) with Embedded Markov Chains

- Continuous- or discrete-time stochastic process
- Discrete states
- State transitions according a state transition matrix
- Time between state transitions can be general
  - Process is memoryless only shortly before or after transitions, but not between transitions.
  - These are embedded points whose system states form a Markov chain.
  - Transitions from a state to itself are still possible.
- Sojourn times in states do not need to be geometrically distributed because time between transitions may vary.
  - But duration in a state in terms of transition steps follows again a *shifted* geometric distribution.

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

Objective: alternative description of Markov chains and alternative calculation of consecutive state distributions x_i

8.4.1 Example: GI[GI]/D/1 – Q_{max} Queue



Description

- Queue with single server and deterministic service time D
- Finite buffer for at most Q_{max} waiting customers (bytes)
- Inter-arrival time A between customer arrivals
- Batches of B customers arrive
- A and B are iid random variables

Objective of analysis

- Blocking probability for customers
- Waiting time distribution for customers

Analysis

- Time discretized in time units ⇒ discrete-time stochastic process
  - A and D given in multiples of time units
- System state U: unfinished work (time units) in server and queue
  - $U_{max} = D \cdot (1 + Q_{max}),\quad 0 \leq U \leq U_{max}$
- Wanted performance measure
  - Distribution of unfinished work U shortly before arrivals
  - ⇒ Embedding of observation points t_i for U_i shortly before arrivals
- Series of U_i yields an embedded Markov chain, process is a SMP
- When batch of B customers arrives, clients that still fit into the queue are accepted, others are rejected.
- The following is an approximation since fractional users are accepted:
  - New unfinished work: $U' = min(U_i + D \cdot B, U_{max})$
- During an inter-arrival time A, unfinished work is reduced by A time units, but it cannot fall below zero:
  - New unfinished work:
    $U_{i+1} = max(U' - A, 0) = \max(min(U_i + D \cdot B, U_{max}) - A, 0)$
- Full model description requires concrete values
  - System starts with no unfinished work $(U_0 = 0)$
  - $Q_{max} = 5$
  - Distributions for random variables A and B

A=i	**2**	**3**	**4**		**B=i**	**1**	**2**	**3**
P(A=i) (a[i])	**0.2**	**0.6**	**0.2**		**P(B=i) (b[i])**	**0.6**	**0.3**	**0.1**

8.4.2 Formalization of the Functional Description

- Embedded points $\mathcal{T}$ (for the SMP)
  - Example: time instants t_i shortly before customer arrivals
  - Choice determines the model
- State X
  - Example: unfinished work $X = (U)$
  - State space $\mathcal{X} = [0; U_{max}]$
- Initial state distribution x_0
  - Example: start state $X_0 = 0$
  - $x[0] = 1, x[i \neq 0] = 0$
- Factor Y and its distribution
  - Influences system behavior
  - Not part of system state
  - Factor space $\mathcal{Y}$ and distribution y
  - Example: $Y = (B, A)$
    - Size of arriving batches B and inter-arrival times A
    - $\mathcal{Y} = (\mathcal{B} \times \mathcal{A})$
    - B and A are both iid $\Rightarrow P\big(Y = (i,j)\big) = b[i] \cdot a[j]$ determines y
- State transition function f: $(X \times Y) \rightarrow X$
  - Recursive stochastic equation
  - Describes system behavior
  - Example: $U_{i+1} = f\big(U_i, (B, A)\big) = max(min(U_i + B, U_{max}) - A, 0)$
- Full specification of MC given by $\mathcal{D} = (\mathcal{T}, \mathcal{X}, x_0, \mathcal{Y}, y, f)$
  - Both state space $\mathcal{X}$ and factor space $\mathcal{Y}$ may be multi-dimensional.
  - Intuitive description of system behavior at embedded points

See: M. Menth: "Description and Analysis of Markov Chains Based on Recursive Stochastic Equations and Factor Distributions"
http://www.worldacademicunion.com/journal/1746-7233WJMS/wjmsvol07no01paper01.pdf

8.4.3 Algorithmic Calculation of Consecutive State Distributions

8.4.3.1 Backward Algorithm for Computation of Successor Distribution x_{n+1}

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

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

- The conditional probability can be computed using the state transition function:

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

- Together with the preimage $\mathcal{Z}(\mathrm{k}) = \{(\mathrm{i},\mathrm{j}) : \mathrm{f}(\mathrm{i},\mathrm{j}) = \mathrm{k}, \mathrm{i} \in \mathcal{X}, \mathrm{j} \in \mathcal{Y}\}$ of state k with respect to f, this equation simplifies the above expression to

$$x_{n+1}[k] = \sum_{(i,j) \in \mathcal{Z}(k)} x_n[i] \cdot y[j] \qquad \text{(backward equation)}$$

- Backward algorithm requires calculation of preimage
  - Difficult
  - Limits tractability of complex models

8.4.3.2 Forward Algorithm for Computation of Successor Distribution x_{n+1}

Rearrangement of computation steps leads to "forward algorithm"

- Numerically equivalent to "backward approach"

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}

Advantages of "forward approach" vs. "backward approach"

- Does not require calculation of preimage
- Program for calculation of x_{n+1} can be syntactically derived from functional description $\mathcal{D}$.

Advantages of "forward approach" vs. vector-matrix-multiplication $x_{i+1} = x_i \cdot P$

- State transition matrix P not needed
  - Can be very large for large state space in spite of sparse matrix representation, storage problem
- Multiple optimization possibilities speed up computation time
  - E.g. decomposition of state transition function: $f = f_0 \circ f_1$ with additional embedded points, states, and different factors
  - See paper

8.4.4 Derivation of the State Transition Matrix P

8.4.4.1 Backward Algorithm for Computation of State Transition Matrix P

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

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

- Equation corresponds to the backward algorithm
- Rearrangement to forward algorithm possible

8.4.4.2 Forward Algorithm for Computation of State Transition Matrix P

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

One can show that the functional description $\mathcal{D}$ and the state transition matrix P have the same expressiveness (see paper).

8.4.5 MC Simulation Based on the Functional Description $\mathcal{D}$

- Use random number U to realize random variable for factor Y
- $X_{n+1} = f(X_n, Y)$

8.5 Use of $GI^{[GI]}/D/1-Q_{max}$ for Approximation of Waiting Time in $GI/GI/1-\infty$

- Difference between $GI/GI/1-\infty$ and $GI^{[GI]}/D/1-Q_{max}$
  - Arrival process: single vs. batch arrivals
  - Service process: GI process vs. D process
  - Queue: infinite vs. limited
- Why approximation works
  - Service time for batches of customers in $GI^{[GI]}/D/1-Q_{max}$ equals service time of single customer in $GI/GI/1-\infty$

  $\Rightarrow$ Queue length distribution of $GI^{[GI]}/D/1-Q_{max}$ approximates waiting time distribution of $GI/GI/1-\infty$
  - Loss occurs only with limited queue
    - Prerequisite for approximation: large Q_{max} to keep loss small in $GI^{[GI]}/D/1-Q_{max}$
    - Problem: what to do when arriving batch does not fully fit in queue?
      - Fill queue up to Q_{max}; this best approximates the evolution of the queue occupancy of an infinite queue.
      - This is also in line with the state transition function for $GI^{[GI]}/D/1-Q_{max}$ presented in 8.4.1.
- Configuration of $GI^{[GI]}/D/1-Q_{max}$ for approximation
  - Choose discrete time unit u as deterministic service time
  - Discretize distributions for inter-arrival times and service times for batches in multiples of u
    - Possibly truncate and normalize the distributions
    - Make sure that mean and coefficient of variation do not change significantly
  - Choose Q_{max} large enough so that stationary probabilities for states near Q_{max} are sufficiently small.
  - Results of the analysis
    - Stationary distribution of the queue occupation Q which translates into
      - Waiting time of arriving batches in $GI^{[GI]}/D/1-Q_{max}$
      - Waiting time of customers in $GI/GI/1-\infty$
  - Distribution may be used to calculate
    - Distribution function
    - Mean value
    - Variance
    - Standard deviation
    - Coefficient of variation

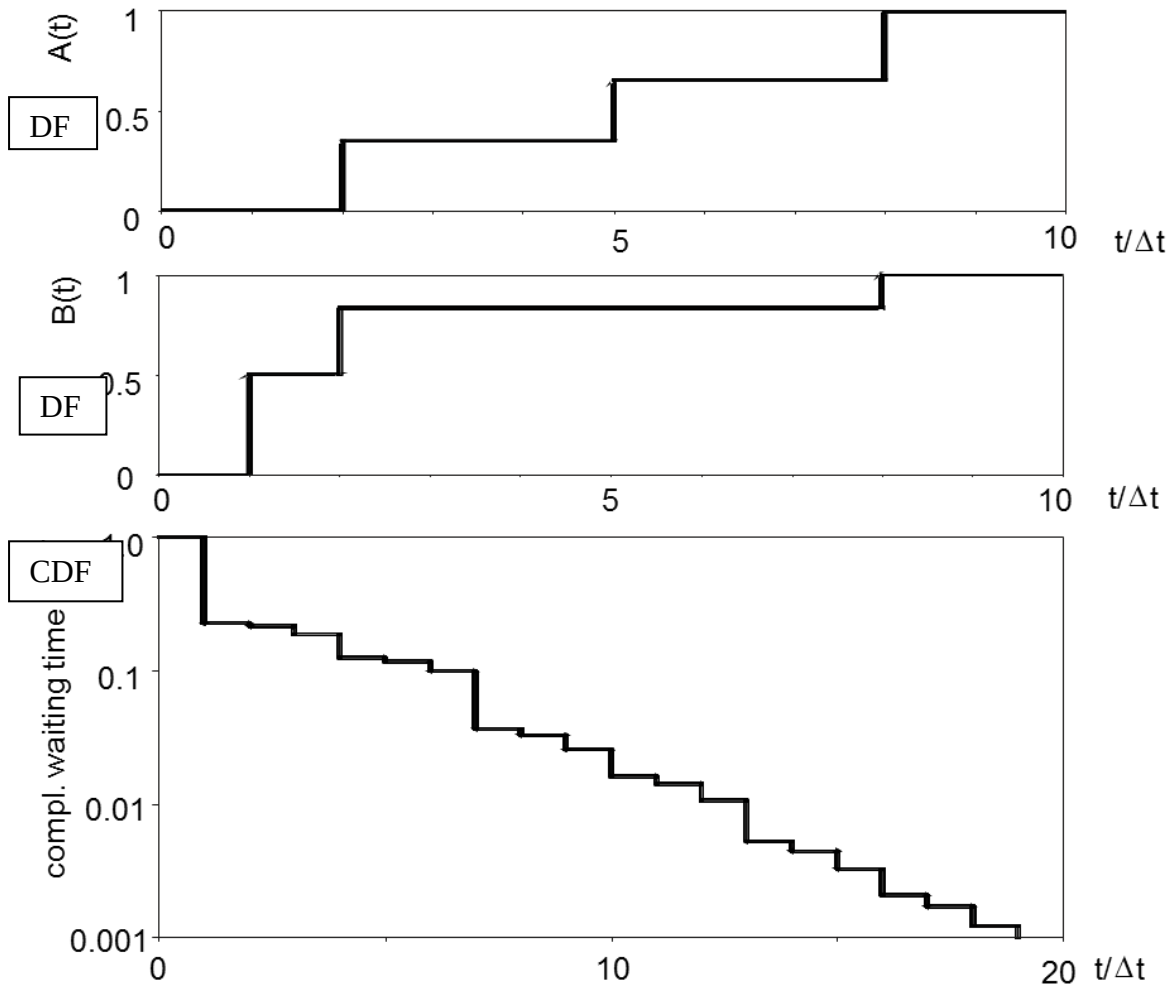


Abbildung 2: Distribution functions A(t) and B(t) are discretized. Therefore, the resulting complementary waiting time distribution function is also a step function. When the discretization is fine enough, steps are no longer visible.
Note: the complementary distribution function is more appropriate for presentation than the distribution function.

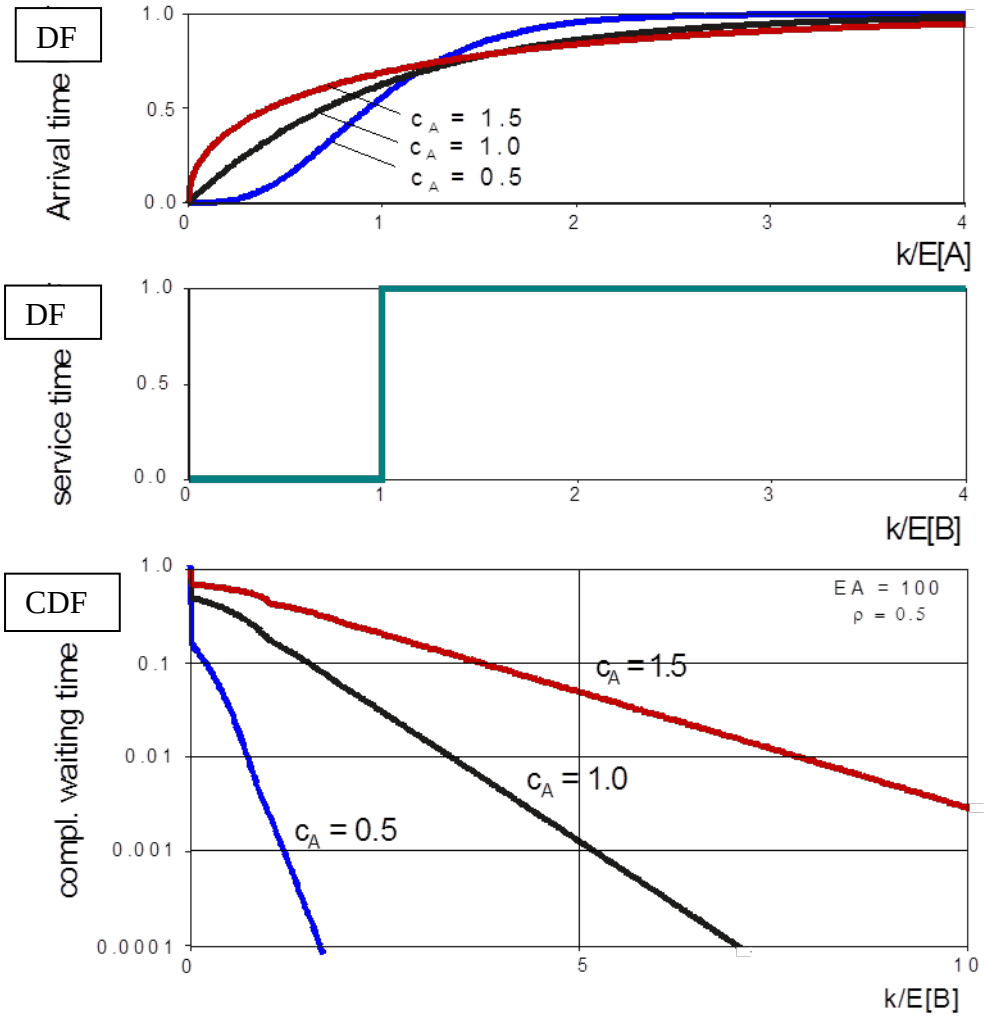


Abbildung 3: Analysis of a G/D/1-∞ waiting system. More variable inter-arrival times lead to larger waiting probabilities and longer waiting times.
Note: the complementary distribution function is more appropriate for presentation than the distribution function.

8.6 Traffic Descriptors, Token Bucket, Markers, Policers, and Shapers

8.6.1 *Traffic Descriptor*

- Describes (upper bound of) intensity and variability of traffic: traffic rates, maximum packet sizes, maximum burst sizes, possibly on different time scales.
- Token bucket is a simple but widespread traffic descriptor.
- Combinations and variations of token buckets and other approaches exist.

8.6.2 *Token Bucket*

- Has a bucket that may hold at most S tokens (e.g., bytes)
- Current fill state of bucket is F.
- Bucket is continuously filled with rate R (e.g. bytes/s).

  UpdateBucket()
  - t=now-lastFill
  - F=min(F+t*R,S)
  - lastFill=now

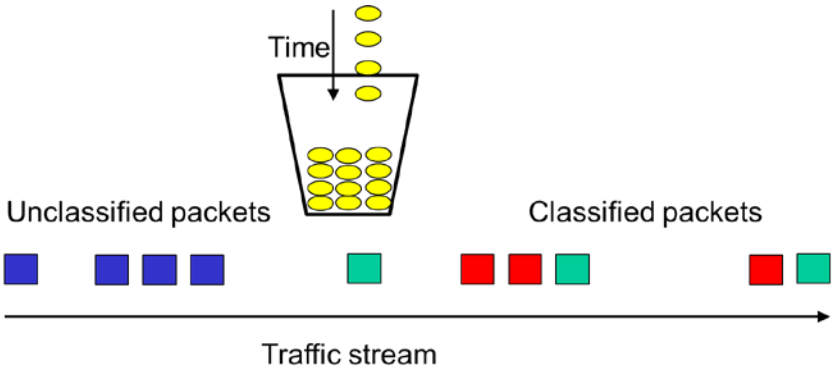


Abbildung 4: Token bucket based traffic meter and classifier.

- Packets of a traffic stream are passed to a token bucket meter that decides whether a packet of size B is in/out-of-profile.

  Check(PacketSize B)
  - UpdateBucket()
  - If (B≤F) then
    - F=F-B
    - Return in-profile
  - Else
    - Return out-of-profile

8.6.3 Traffic Marker

- Some nodes in the Internet mark some packets when overloaded to request rate reduction from source (explicit congestion notification, ECN) [RFC3168].
- Threshold marking marks all packets of a traffic stream as long as the traffic stream exceeds a token bucket descriptor [RFC5670]; marked packets are used to detect “pre-congestion” and to stop admission of new flows (pre-congestion notification, PCN).
- Excess traffic marking marks some packets of a traffic stream, namely those that exceed a token bucket traffic descriptor [RFC5670]; unmarked and marked packets are used to quantify overload and determine the amount of traffic that may be removed from the network.

8.6.4 Policer

- Drops packets of a traffic stream that are out-of-profile with regard to a traffic descriptor.

8.6.5 Spacer

- Delays packets of a traffic stream that are out-of-profile with regard to a traffic descriptor.
- Uses a buffer to delay packets.
- Packets may be lost if buffer overflows.

8.7 Analysis of a Spacer Using $GI^{[GI]}/D/1\text{-}Q_{max}$

We assume a spacer that enforces a maximum *rate of R (bytes/s)* in communication networks by buffering packets between a station and a network. The *buffer is S bytes large*. The spacer makes sure that if a packet of size B (bytes) is sent, the time until the next packet is sent is at least B/R.

- Model of the spacer
  - The bucket of a spacer is modelled by unfinished work U
    - Measured in time
    - Describes time to wait until a newly arrived packet can be sent
    - $U_{max}=S/R$
  - Change of unfinished work at packet arrivals
    - If $(U+B/R \leq U_{max})$
      - U'=U+B/R // accept packet, packet will be sent after U
    - Else
      - U'=U //drop packet
    - Endif
  - Change of unfinished work during a packet inter-arrival time A (time)
    - U''=max(U'-A,0)
- This can be modeled by a modified discrete-time $GI^{[GI]}/D/1-Q_{max}$
  - The queue models the bucket
  - An arriving batch corresponds to an arriving packet of B bytes
  - All bytes of a packet are accepted or dropped
  - Modification of the state transition function presented for $GI^{[GI]}/D/1-Q_{max}$ in 8.4.1 is needed:
    - $U_{max} = S/R$
    - State transition function

      If $U_n + \frac{B}{R} \leq U_{max}$

      $U' = U_n + B/R$

      Else

      $U' = U_n$

      Endif

      $U_{n+1} = max(U' - A, 0)$

- The resulting stationary state distribution indicates the fill state of the spacer at packet arrivals so that the spacing delay and the packet loss due to the spacer can be predicted.

- Example
  - An ATM Virtual Connection (VC) requires a minimum time T between consecutive ATM cells (each of size 53 bytes).
  - The cell stream needs to be spaced before entering the network.
  - The spacer has a buffer for τ_{max} cells (buffer size S) and enforces a maximum rate of 1 cell within T time, i.e., R=1 cell/T.
  - The distribution of inter-arrival times between cells is different before entering and after leaving the spacer
    - The time between cells in the departure process starts only with T.
    - The spacing result depends on the buffer size, analysis for
      - $\tau_{max} = 60$ cells
      - $\tau_{max} = 0$ cells
  - The spacer buffer may overflow ⇒ ATM cells may be lost, especially for small buffers

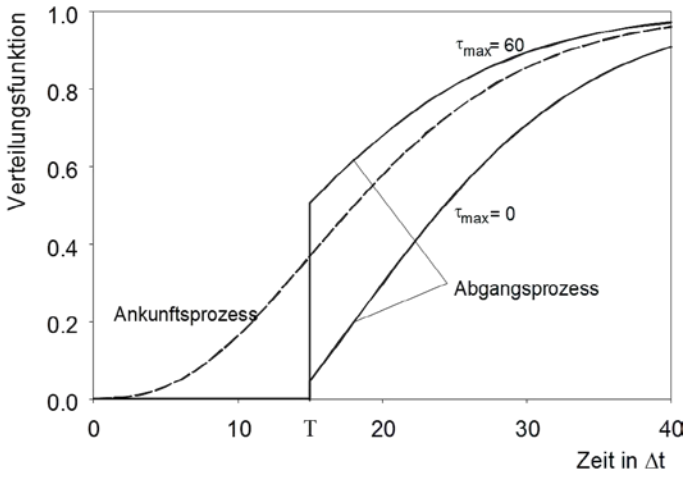


Abbildung 5: Distribution of inter-arrival time between cells before entering and after leaving a spacer.

8.8 Markov Chain Simulation

MCs can be simulated to calculate their average state distribution x.

- The simulation state is X
- Simulation calculates series X_n to find x
- Successor state $X_{n+1} = k$ for $X_n = i$ needs to be simulated

8.8.1 MC Simulation Based on the State Transition Matrix

- Random number $U \in (0; 1)$
- Given $X_n = i,\ X_{n+1} = min\left(k: \sum_{0 \le j \le k} p_{ij} \ge U\right)$

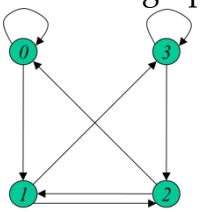
8.8.2 MC Simulation Based on the Functional Description $\mathcal{D}$

- Use random number U to realize random variable for factor Y
- $X_{n+1} = f(X_n, Y)$

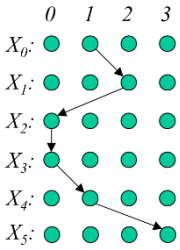
8.9 Comparison of Simulation and Calculation of Subsequent State Distributions for MC Analysis

8.9.1 Comparison of Operation

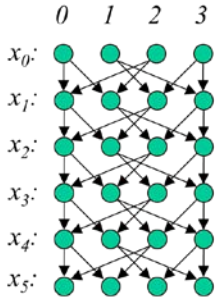
- Consider the following state transition graph



- MC simulation takes random walk through the transition graph and constructs a series $(X_n)_{0 \le n \le n_{max}}$
  - 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 average state probabilities x.



- Mathematical/algorithmic calculation of consecutive state distributions x_n
  - Takes into account all possible transition steps in a single iteration step
  - Probability mass of state distribution x_n is propagated from all states over all possible transitions to potential successor states which leads to the new state distribution x_{n+1}.
  - Here, a single iteration step requires a lot of computation effort, but usually a few tens or hundreds iteration steps are sufficient until x_n have converged with sufficient accuracy.



8.9.2 Comparison of Efficiency

Consider a convergence accuracy of $\|x_n - x\| \leq \varepsilon$. How many samples or iteration steps N are needed for sufficient accuracy?

Efficiency of MC simulation

- An accuracy of $\|x_n - x\| \leq \varepsilon$ requires more than $N > \frac{1}{\varepsilon}$ samples
- If ε decreases exponentially, then N increases exponentially.

Efficiency of calculation of consecutive state distributions x_n

- λ eigenvalue of P if $x \cdot P = \lambda \cdot x$
- P stochastic $\Rightarrow |\lambda| \leq 1$
- Convergence speed of x_n towards x: $\|x_n - x\| \leq (\lambda_s)^n$
    - λ_s largest eigenvalue with $|\lambda_s| < 1$ (subdominant eigenvalue)
- $(\lambda_s)^n \leq \varepsilon \Rightarrow n \geq ln(\varepsilon)/ln(\lambda_s)$
- If ε decreases exponentially, then n increases linearly.

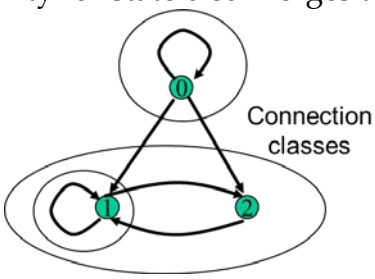
Conclusion

- For small values of ε, MC simulation is less efficient than calculation of consecutive state distributions x_n.
- But may be efficient for large ε.
- Combine simulation and analysis for speedup (optimization method):
    - First simulate x rather quickly using MC simulation for little accuracy
    - Then use x as start vector for calculation of consecutive state distributions x_n

8.10 Markov Chain Classification

Example 1:

- $P = \begin{pmatrix} 0.1 & 0.45 & 0.45 \\ 0 & 0.4 & 0.6 \\ 0 & 1 & 0 \end{pmatrix}$
- $x_0 = (1,0,0)$, $x_1 = (0.1,0.45,0.45)$, $x_2 = (0.01,0.675,0.315)$, $x_3 = (0.001,5895,0.4095)$, $x_4 = (0.0001,64575,0.35415)$, $x_5 = (0.00001,0.6125,0.3875)$, $x_6 = (0.000001,0.6325,0.3675)$, $x_7 = (0.0000001,0.6205,0.3675)$, $x_8 = (0.00000001,0.6277,0.3723)$,…, $x_{19} = (0,0.62499,0.37501)$, $x_{20} = (0,0.62501,0.374993)$, …
- Observation: probability for state 0 converges to zero.

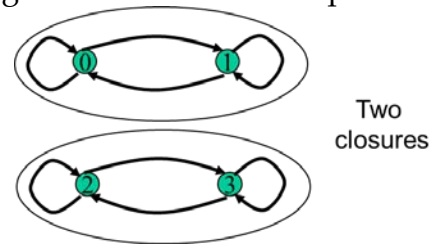


State transition graph:

- *Transient state*: will not be reached again with 100% probability
- *Recurrent state*: will be reached again with 100% probability
- *Connection class*: set of states where each state reaches all other states in that set

Example 2:

- $P = \begin{pmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.1 & 0.9 \end{pmatrix}$
- $x_0 = (1,0,0,0)$, $x_1 = (0.3,0.7,0,0)$, $x_2 = (0.51,0.49,0,0)$, $x_3 = (0.447,0.553,0,0)$, $x_4 = (0.4659,0.5341,0,0)$, $x_5 = (0.4602,0.5398,0,0)$, $x_6 = (0.4619,0.5381,0,0)$, $x_7 = (0.4614,0.5386,0,0)$, $x_8 = (0.4616,0.5384,0,0)$, …
- $x_0 = (0,0,0.5,0.5)$, $x_1 = (0,0,0.3,0.7)$, $x_2 = (0,0,0.22,0.78)$, $x_3 = (0,0,0.188,0.812)$, $x_4 = (0,0,0.1752,0.8248)$, $x_5 = (0,0,0.1701,0.8299)$, $x_6 = (0,0,0.1680,0.8320)$, $x_7 = (0,0,0.1672,0.8328)$, $x_8 = (0,0,0.1669,0.8331)$, …
- Observation: average state distribution depends on start vector x_0



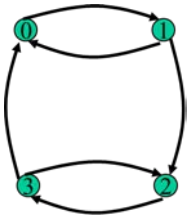
State transition graph:

- A subset of states is *closed* if no other state can be reached from it.
- *Closure*: a minimum closed set.
  - A maximum connection class is either a closure or completely "absorbed".
- A MC is *irreducible* if the set of all states is the only closure, otherwise it is *reducible*.

Example 3:

- $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 1 \\ 0.9 & 0 & 0.1 & 0 \end{pmatrix}$
- $x_0 = (1,0,0,0)$, $x_1 = (0,1,0,0)$, $x_2 = (0.4,0,0.6,0)$, $x_3 = (0,0.4,0,0.6)$, $x_4 = (0.7,0,0.3,0)$, $x_5 = (0,0.7,0,0.3)$, $x_6 = (0.55,0,0.45,0)$, $x_7 = (0,0.55,0,0.45)$, $x_8 = (0.625,0,0.375,0)$, $x_9 = (0,0.625,0,0.375)$, $x_{10} = (0.5875,0,0.4125,0)$, $x_{11} = (0,0.5875,0,0.4125)$, $x_{12} = (0.6063,0,0.3938,0)$, $x_{13} = (0,0.6063,0,0.3938)$, $x_{14} = (0.5969,0,0.4031,0)$, $x_{15} = (0,0.5969,0,0.4031)$,…
- Observation
  - Oscillation with period $p = 2$
  - Consecutive state distributions do not converge!
  - But series $(x_{p \cdot n+k})_{0 \le n < \infty}$ converges

State transition graph:



- A state *i* has *period* p if it can be reached again only after multiples of p transition steps.
- Period p is greatest common divisor of all cycle lengths in state transition graph
- All states of a closure have the same period.
- 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*.

Optimization for calculation of average state distribution for periodic MCs

- Recall average state distribution: $x = \lim_{n \to \infty} \left(\frac{1}{n+1} \cdot \sum_{i=0}^{n} x_i \right)$
  - Slow convergence
- Limit $x = \lim_{n \to \infty} x_n$ converges faster
  - But exists only for aperiodic MCs
  - $x = \lim_{n \to \infty} x_n$ does not exist for periodic MCs
- Use $x = \lim_{n \to \infty} \left(\frac{1}{p} \cdot \sum_{n \le i < n+p} x_i \right)$ for fast calculation of avg. state distribution
  - Period p needs to be computed before

8.11 Analysis of Markov Chains with Stopping Times

Principle

- Markov chain runs until it is stopped by a defined condition, e.g., by the fact that a stop state is reached.

Example: game of dice

- A dice is subsequently thrown until the sum of the pips is at least ten. What is the distribution of the required number of throws and the number of pips at the end of the game?

Model

- State space $\mathcal{X} = \{0, \dots, 15\}$: sum of pips
  - Set of stopping states: $\mathcal{X}_s = \{10, \dots, 15\}$ (end of the game)
  - Set of transient states: $(\mathcal{X} \setminus \mathcal{X}_s) = \{0, \dots, 9\}$ (during the game)
  - Start state $X_0 = 0 \Rightarrow$ start state distribution x_0
- Factor space $\mathcal{Y} = \{1, \dots, 6\}$: pips per throw
  - Factor distribution y: uniform
- State transition function: f(X,Y)=X+Y

Wanted

- Distribution of (n, i)
  - Process stops after n transitions in stopping state $i \in \mathcal{X}_s$.
  - (Game is over after n throws with a sum of i pips.)
- Useful for performance analysis
  - Distribution of number of transitions until Markov chain stops (stopping time)
  - Distribution of pips at the end of the game

Idea for calculation

- Perform all possible state transitions until resulting state is stopping state
- Record probabilities for (n, i)

Algorithm

- Input: x_0
  - $m = 0$ // number of transitions
  - Repeat
    - For all $i \in \mathcal{X}$ // init state probabilities
      - $x_{m+1}(i) = 0$
    - Endfor
    - For all $i \in \mathcal{X} \setminus \mathcal{X}_s$
      - For all $j \in \mathcal{Y}$
        - $k = f(i, j)$
        - $x_{m+1}(k) = x_{m+1}(k) + x_m(i) \cdot y(j)$
      - Endfor
    - Endfor
    - $m = m + 1$
  - Until $\sum_{i \in \mathcal{X} \setminus \mathcal{X}_s} x_m(i) = 0$
- Output: $m, x_n(i), 0 \leq n \leq m, i \in \mathcal{X}_s$

Comments

- $x_n(i), 0 \leq n \leq m, i \in \mathcal{X}_s$ hold probabilities for (n, i) and yield the wanted distribution.
- x_n is not a distribution for $n > 0$ as $\sum_{i \in \mathcal{X}} x_n(i) = 1$ does not hold in general.
- This algorithm terminates only if the underlying Markov chain is stopped within a finite number of transition steps.
- Example calculations
  - Distribution of number of transitions until Markov chain stops (stopping time): $P(N = n) = \sum_{i \in \mathcal{X}_s} x_n(i)$
  - Distribution of pips at end of the game: $P(X = i) = \sum_{0 \leq n \leq m} x_n(i)$
- More on this topic: look for "stopping time" in mathematical literature

Other examples

- M. Menth and F. Lehrieder: "PCN-Based Measured Rate Termination", Computer Networks, vol. 54, no. 13, Sept. 2010, p. 2099 – 2116
- A. Clark and S. Gilmore: "Terminating Passage-Time Calculations on Uniformised Markov Chains", 24th UK Performance Engineering Workshop (UKPEW), London, UK, July 2008
- Chuan Heng Foh and Moshe Zukerman: "Performance Evaluation of IEEE 802.11", IEEE Semiannual Vehicular Technology Conference (VTC, Fall), 2001
- Einführendes Beispiel in Modellierung & Simulation: "Tempo, kleine Fische!" (Übung -1)

8.12 Application Example: Analysis of a GI/GI/1-Q_{max} Queue

Use MCs with stopping times to calculate

- the number of customers C that are served within an interval of size T with an i.i.d. service time B
- the remaining service time D of the following customer that could not be fully served

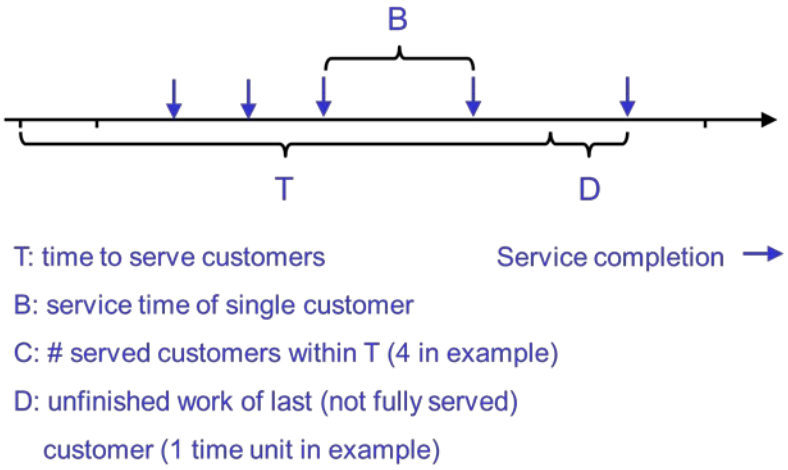


Abbildung 6: C customers are served within time T and a service time D remains from another customer.

- How are (C,D)(T) computed?
  - Use Markov chain with stopping time (similar to game of dice)
  - State: X (cumulated service times), initial state X=0
  - Factor: B (service time)
  - State transition function: f(X,B)=X+B
  - Process stops when X+B > T
  - C = number of transitions when process stops - 1
  - D = X-T > 0

Analyze GI/GI/1-Qmax

- Difference between GI/GI/1-Qmax and GI/GI/1-∞ (previously considered system): queue holds customers, not unfinished work
- State: (Q,U)
  - Q: number of waiting customers in queue (max. size Q_{max})
  - U: unfinished work in server unit (in time)

- State evolution depends on inter-arrival time A and service time B

- Embedded points: shortly *before* customer arrivals
- What happens between embedded points?
  - New customer arrives; if Q= Q_{max}, new customer is dropped
  - Service of customer in service unit possibly completes so that T time is left until next arrival
  - Possibly C additional customers are fully served
  - Possibly another customer is partially served so that D unfinished work remains in server unit
  - Conditional factor (C, D)(T), T≥0
    - C(T): number of fully served customers within time T (C ≥ 0)
    - D(T): unfinished work of (C+1)st customer within time T (D > 0)
- State transition function
  - $Q' = \min(Q_n + 1, Q_{max})$
  - If $(U_n > A)$ // customer in service does not finish
    - $U_{n+1} = U_n - A$
    - $Q_{n+1} = Q'$
  - Else // customer in service finishes, T time remains, within which further C cust may finish and additional unfinished work D may be left
    - $Q'' = max(Q' - 1,0)$
    - $T = \max(\mathrm{A} - \mathrm{U_n}, 0)$
    - If $Q'' \leq C(T)$// all customers in queue can be served
      - $Q_{n+1} = 0$
      - $U_{n+1} = 0$
    - Else
      - $Q_{n+1} = Q'' - C(T)$
      - $U_{n+1} = D(T)$
    - Endif
  - Endif

8.13 Memory Markov Chain (MMC) (see Paper O. Rose, ITC, 1999)

8.13.1 Struktur der MMC

Literatur:

- O. Rose, „A Memory Markov Chain for VBR Traffic With Strong Positive Correlations“, 16th International Teletraffic Congress, Edinburgh
- M. Menth, A. Binzenhöfer, and S. Mühleck: “Source Models for Speech Traffic Revisited”, in IEEE/ACM Transactions on Networking, vol. 17, no. 4, August 2009, IEEE

Problem

- Korrelierte Zufallsvariablen, deren Korrelationen sich nicht oder nur schwer mittels einfacher stochastischer Prozesse beschreiben lassen.
- Beispiel: Folge von Paketgrößen komprimierter Sprachproben
  31, 31, 31, 31, 11, 4, 4, 4, 12, 4, 4, 31, 31, 31, 11, 4, 4, 4, 12, 4, 4, 31, 31, 11, 4, 4, 4, 12, 4, 31, …

Idee: Hidden Markov Model

- Eine Variable S charakterisiert den Zustand eines Prozesses
- Eine bedingte ZV U(S) realisiert eine zufällige aber zustandsabhängige Ausgabe
- Nach jeder Ausgabe kann sich der Zustand S ändern.
- Nur U ist von außen beobachtbar
- S ist von außen nicht beobachtbar, die Markov-Kette ist also versteckt.

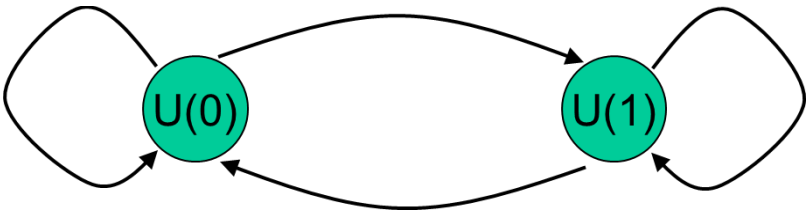


Abbildung 7: Markovkette mit zwei Zuständen und zustandsabhängigen Zufallsvariablen U(S) als Ausgabe.

Simulation

- Realisierung der Ausgabe gemäß der bedingten Zufallsvariable U(S)
  - Benutzt bedingte Verteilungen u(S)
- Berechnung des nächsten Zustandes S der Markov-Kette
  - Benutzt eine Zustandsübergangsmatrix P

Beispiel

- Folge von Paketgrößen komprimierter Sprachproben (z.B. 31, 31, 31, 31, 11, 4, 4, 4, 12, 4, 4, 31, 31, 31, 11, 4, 4, 4, 12, 4, 4, 31, 31, 11, 4, 4, 4, 12, 4, 31, …) soll durch Markov-Kette mit zustandsspezifischer Ausgabe erzeugt werden können.
- Lösung:

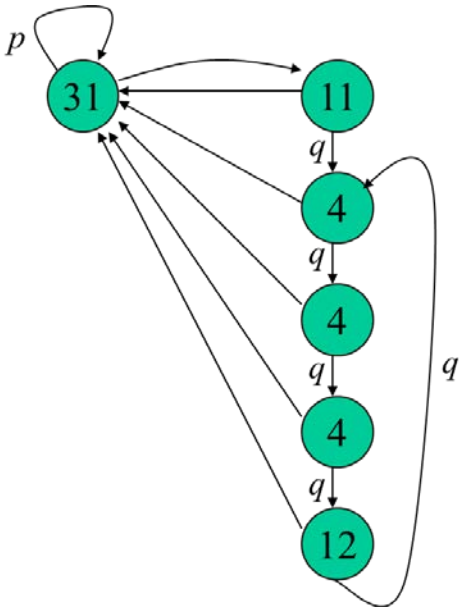


Abbildung 8: Zustandsübergangsdiagramm für die steuernde Markov-Kette einer MMC. Die Aktiv- und Passivzeiten des Sprechers sind geometrisch verteilt mit Parameter p und q. Die Zahl im Kreis entspricht einer Ausgabefunktion für die deterministische ZV in diesem Zustand.

- Zustände: 0, 1, …, 5
- Zustandsspezifische Ausgabe: U(0)=31, U(1)=11, U(k)=4 mit k∈{2,3,4}, U(5)=12
- Übergangsmatrix $P = \begin{bmatrix} p & 1-p & 0 & 0 & 0 & 0 \\ 1-q & 0 & q & 0 & 0 & 0 \\ 1-q & 0 & 0 & q & 0 & 0 \\ 1-q & 0 & 0 & 0 & q & 0 \\ 1-q & 0 & 0 & 0 & 0 & q \\ 1-q & 0 & q & 0 & 0 & 0 \end{bmatrix}$
- Die Verallgemeinerung dieser Struktur ist eine „Memory Markov Chain“ (MMC)

8.13.2 Parametrisierung der MMC

Gegeben ist eine Zeitreihe $x_1, \ldots, x_N$ und gesucht ist eine MMC, die ähnliche Zeitreihen erzeugt.

- MMC(W, M_s, M_a)
- Parameter: Fensterlänge W, Stichprobenzustandsgröße M_s, Durchschnittszustandsgröße M_a
- $\bar{x}_i = \frac{1}{W} \cdot \sum_{i-W<k<i} x_k$ für i=W+1, …, N
- Generierung von Paaren $(\text{sample}, \text{average}) = (x_i, \bar{x}_i)$
- Reduktion der Paare auf einen geeigneten Zustandsraum
  - Zustandsraum $(v, w) \in \{1,\ldots,M_s\} \times \{1,\ldots,M_a\}$; alternativ: M_s bzw. M_a diskrete Werte, die keine natürlichen Zahlen darstellen müssen
  - Zuordnung $(x_i, \bar{x}_i) \sim (y_i, \bar{y}_i) \in \{1,\ldots,M_s\} \times \{1,\ldots,M_a\}$
  - $y_i = \begin{cases} 1 & x_i = \min_j(x_j) \\ \left\lceil \frac{x_i - \min_j(x_j)}{\max_j(x_j) - \min_j(x_j)} \cdot M_s \right\rceil & \textit{otherwise} \end{cases}$
  - $\bar{y}_i = \begin{cases} 1 & \bar{x}_i = \min_j(\bar{x}_j) \\ \left\lceil \frac{\bar{x}_i - \min_j(\bar{x}_j)}{\max_j(\bar{x}_j) - \min_j(\bar{x}_j)} \cdot M_a \right\rceil & \textit{otherwise} \end{cases}$
  - Aufzählung der Zustände (v, w) aufsteigend nach absoluter Ordnung
  - Empirische Bestimmung der Übergangswahrscheinlichkeitsmatrix P anhand der Zeitreihe $(y_i, \bar{y}_i)$
  - Bestimmung der stationären Zustandsverteilung π für die Zustände $(v, w) \in \{1,\ldots,M_s\} \times \{1,\ldots,M_a\}$
  - Assoziation der Zustandskomponenten $v \in \{1,\ldots,M_s\}$ mit einer Ausgabefunktion *u(v)*
    - Wenn $x_i = y_i$ gilt, dann kann für *u(v)=v* gewählt werden. Ansonsten:
    - $\delta(y_i, v) = \begin{cases} 0 & y_i \neq v \\ 1 & y_i = v \end{cases}$
    - $u(v) = \frac{\sum_{W<i\leq N} \delta(y_i, v) \cdot x_i}{\sum_{W<i\leq N} \delta(y_i, v)}$

- Vergleich von empirischer und theoretischer Korrelation
  - Empirische (Auto-)Kovarianz (siehe Kapitel 2)
    - $\overline{\mathrm{COV}}[X,Y] = \frac{1}{n-1} \cdot \sum_{0 \le i < n} E\left[\left[X_i - \overline{X}(n)\right] \cdot \left[Y_i - \overline{Y}(n)\right]\right]$
    - $\hat{C}_j(n) = \frac{1}{n} \sum_{0 \le i < n-j} \left[X_i - \overline{X}\right]\left[X_{i+j} - \overline{X}\right]$
  - Zur Erinnerung: theoretische Kovarianz bzw. Korrelation (siehe Kapitel 2)
    - $\mathrm{Cov}[X,Y] = E[(X - E[X]) \cdot (Y - E[Y])] = E[X \cdot Y] - E[X] \cdot E[Y]$
    - $Cor[X,Y] = \frac{Cov[X,Y]}{\sqrt{VAR[X] \cdot VAR[Y]}}$
  - Analog theoretische Autokovarianz bzw. Autokorrelation für MMC:
    - Vektor *u* enthält die zu *(v,w)* entsprechenden Ausgabewerte *U(v)*.
    - *diag(u)* ist die zu *u* entsprechende Diagonalmatrix.
    - Berechnung der Momente: $E[X^k] = \pi \cdot diag(u)^k \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$
    - Autokovarianz zum Lag k:

      $$Cov(k) = \pi \cdot diag(u) \cdot P^k \cdot diag(u) \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} - E[X]^2$$

    - Autokorrelation zum Lag k: $Cor(k) = \frac{Cov(k)}{E[X^2] - E[X]^2}$
- Falls empirische und theoretische Korrelation nicht gut übereinstimmen, wähle andere Parameter W, M_s, M_a!