An Algorithm for Creating Markovian Models of Complex Systems

Eimutis VALAKEVICIUS
Department of Mathematical Research in Systems
Kaunas University of Technology
Kaunas, LT - 51368, Lithuania

and

Henrikas PRANEVICIUS
Department of Business Informatics
Kaunas University of Technology
Kaunas, LT - 51368, Lithuania

ABSTRACT

The paper considers a method for automatic construction of numerical models for systems described by Markov processes. The embedded Markov chains are used for computation of stationary probabilities of Markov processes. The performance of the system described in the event language is used for generating a system of Kolmogorov equations. An example of a numerical model for the event – driven local computer network protocol is presented.

Keywords: Stochastic systems, Markov Chain, Stationary distribution, Numerical Model

1. INTRODUCTION

Markov processes are widely used for creating numerical models for such systems as computer networks and telecommunication systems. It is known that creation of analytical models requires large efforts [1,2]. Use of numerical methods permits to create models for a wider class of systems. The process of creating numerical models for systems described by Markov process consists of the following stages: 1) definition of the state of a system; 2) creating equations describing Markov process; 3) computation of stationary probabilities of Markov process; 4) computation characteristics of the system performance.

The questions related to the problem of automatic construction of numerical models for the computation of average characteristics of a system will be presented in the paper. In creating numerical models the most difficult stages are forming of the system of equations and calculating stationary probabilities.

In the first part of the paper the method of embedded Markov chains for computation of stationary probabilities is presented. The algorithm for automatic construction of all the possible states of a system, the matrix of transition rates between the states, the algorithms for computing stationary probabilities and the performance characteristics of the system are presented in the second part. The event language for the description of the functioning of a system is given in the third part. An example of creating a numerical model of event – driven computer network protocol is in the last part.

2. CALCULATION OF THE STEADY STATE PROBABILITIES

Let $P_N = \left( p_{ij}^{(N)} \right)_{N \times N}$ be the matrix of transition probabilities for a Markov chain $\{S^m, m \geq 0\}$ on a finite state space. The stationary probabilities of an embedded Markov chain are determined by the system of linear equations:

$$ p_i = \sum_{j=1}^{N} p_{ij} p_j, \quad i = 1, N, $$
where
\[ p_{ji} = \frac{\lambda_{ji}}{\sum_{i=1}^{N} \lambda_{ji}}, \quad i, j = 1, N, \]

\( \lambda_{ji} \) – the rate of transition and \( p_{ji} \) – the transition probability from state \( S_i \) to state \( S_j \).

Computation of the steady state probabilities \( N_{pi1} \), \( i = 1, N \) of a system states involves two stages: the stage of embedding the Markov chains and the one of computing the steady – state probabilities [3].

The algorithm has the following form:

I. The stage of embedding the Markov chains:

\[ p_{ij}^{(k)} = p_{ij}^{(k+1)} + p_{ij}^{(k+1)} \cdot \frac{p_{ij}^{(k+1)}}{1 - p_{ij}^{(k+1)}}, \quad i, j = 1, k; \quad k = N - 1; \]

II. The stage of computing the stationary probabilities:

\[ p_{i}^{(k)} = 1; \]

\[ p_{i}^{(k+1)} = \frac{\sum_{j=1}^{k} p_{ij}^{(k+1)} p_{ji}^{(k+1)}}{1 - p_{ii}^{(k+1)}}, \quad i = k + 1; \quad k = 1, N - 1; \]

\[ p_{j} = \frac{p_{j}^{(N)}}{\sum_{j=1}^{N} p_{j}^{(N)}}, \quad j = 1, N. \]

The steady – state probabilities \( q_{i} \) of the Markov process are found from the formula:

\[ q_{i} = \frac{p_{i}}{\sum_{j=1}^{N} \lambda_{ij}}, \quad i = 1, N. \]

The suggested formulae are not very convenient in practical problems. As a rule in real systems the transitions on the given set of the possible states are done by means of transition rates and not by transition probabilities. A slightly modified algorithm is proposed. Let \( \Lambda_{N} = \{ \lambda_{ij}^{(N)} \}_{i=1}^{N} \) be the matrix of transition rates for a Markov chain on the set of states \( S = \{ S_{1}, \ldots, S_{N} \} \).

Then

\[ \lambda_{ij}^{(k)} = \lambda_{ij}^{(k+1)} + \frac{\lambda_{i,j,k+1}^{(k+1)} \lambda_{k+1,j}^{(k+1)}}{\sum_{k=1}^{N} \lambda_{k+1,j}^{(k+1)}}, \quad i = 1, k; \quad k = N - 1; \]

\[ r_{1}^{(k)} = 1; \]

\[ r_{i}^{(k+1)} = \frac{\sum_{j=1}^{N} r_{j}^{(N)}}{\sum_{j=1}^{N} r_{j}^{(N)}}, \quad i = k + 1; \quad k = 1, N - 1; \]

\[ q_{i} = \frac{r_{i}^{(N)}}{\sum_{i=1}^{N} r_{i}^{(N)}}, \quad i = 1, N. \]

3. AN ALGORITHM FOR COMPUTATION OF STATIONARY PROBABILITIES

The method for automatic construction of numerical models of systems represented by Markov processes with a countable space of states and continuous time is proposed. The algorithm for automatic generating of all the possible system states as well as the matrix of transition rates between the states and computing of the stationary probabilities and characteristics of system performance under investigation is given below.

To describe the algorithm the following notation will be introduced. Three arrays are needed: \( R \) – the values of the nonzero elements of the matrix of the size \( \tau \); \( NI \) – row indexes and \( NJ \) – indicator of the column indexes of the size \( N \); \( A_{N} \) – a set of states participating in the embedding process; \( B_{k} \) – a set of already eliminated states; \( Z(A_{N}) \) – the rule by which one state is chosen from the set \( A_{N} \); \( \phi(S_{k}) \) – an algorithm determining a set
of adjacent states for every state $S_k$; The algorithm consists of five stages.

1. The initial setting

$$A_0 = \{S_1\}, B_0 = \{\emptyset\}, N I = (0), N J = (0), R = (0);$$

2. The stage of generating a set of states and a transition matrix by means of events description (it will be given latter)

$$A_N = A_{N-1} \cup q(S_k)k = 2, \ldots, N;$$

To store the transition elements in a compact form the arrays $NI, NJ, R$ are filled in the following way. The element $NI(\alpha)$ – the $\alpha$-th element of the array $NI(k)$ contains the index of the row of $\alpha$-th element $R(\alpha)$ in the array $R$. If the first non-zero element of the $\beta$-th column of the given matrix placed in $R(\tau\beta)$, then $\tau\beta$ is stored in the $\beta$-th element $NJ$, i.e. $NJ(\tau\beta) = \tau\beta$.

3. The embedding stage.

After eliminating the next state due, the elements of the matrix are recalculate

$$R_{ij}^{(k+1)} = \begin{cases} R_{ij}^{(k)} + \frac{r_j}{\sum_{j=1}^N r_j} & S_i, S_j \in \partial_k(S_k)k = N-1, \\
R_{ij}^{(k)} & \text{otherwise},
\end{cases}$$

where $R_{ij}^{(k)}$ is the corresponding rate of the transition from state $S_i$ to the state $S_j$ after eliminating $N-1$ states; $\partial_k(S_k) = \{ S_j \in A_N : R_{ij}^{(k)} + R_{ij}^{(k)} \neq 0 \}, S_i \in A_k, k = N-1$ is a set of states adjacent to the state $S_i$ after eliminating $N-1$ states; $n(k)$ is the index of the element being eliminated:

$$S_{n(k)} = \sum_{j=1}^N R_{ij}^{(k)} n(k);$$

4. The stage of computation of stationary probabilities:

$$q_i = \frac{r_i}{\sum_{j=1}^N r_j},$$

where $q_i$ - stationary probabilities of the Markov process.

5. The stage of computation of the system performance characteristics

$$x_i = F_i(q_1, q_2, \ldots, q_N), x_i \in X,$$

where $x_i$ is the system characteristics being computed; $F_i$ – the formula by which $x_i$ is being computed; $X$ – a set of the characteristics being computed.

4. THE DESCRIPTION OF THE EVENT LANGUAGE

The system’s performance is described in the event language. To describe that in the event language the following data are used:

1. The system state vector at any moment of time $t$:

$$S(t) = \{ n_1(t), n_2(t), \ldots, n_p(t) \},$$

where $p$ – the number of the components of the state vector. The components of the state vector are discrete values and define the states of the separate elements of a system at any moment of time.

2. Limitation $n_{i}^{\min}$ and $n_{i}^{\max}$ for every component of a state vector, i.e.

$$n_{i}^{\min} \leq n_{i} \leq n_{i}^{\max}, i = 1, p.$$

3. The set of events $E = \{e_1, \ldots, e_m\}$ which may occur in the system at discrete moments of time.

4. The array of the rates $INTENS = \{\lambda_1, \ldots, \lambda_m\}$ for the definition of transitions from one system state to another.

The algorithm for the description of the system is given below.

1. $i := 1$.

2. The description of the conditions under which the event $e_i$ can occur and the transition of the system from one defined state to the next.

3. If necessary check additional conditions for altering vector co-ordinates of state for the event $e_i$? If not, go to stage 5.

4. The description of additional conditions for altering vectors co-ordinates.

5. The description of altering of system state components for the event $e_i$. 
6. Are all additional conditions for altering vector co-ordinates of state for the event $e_i$ described? If not, go to stage 3.

7. The definition of transition rate \textit{INTENS} for the event $e_i$.

8. $i = m$? If yes, then go to stage 11.

9. $i = i + 1$.

10. Go to stage 2.

11. End of description.

5. **CONCEPTUAL MODEL OF AN EVENT DRIVEN LOCAL COMPUTER NETWORK PROTOCOL**

The configuration of the protocol is a physical bus, where the $N$ stations form a logical ring. The algorithm is based on the noticeable events on the bus (hence the name event-driven bus protocol). The protocol is distributed, except in the initialisation phase. Every station listens to the bus and receives both destination and the source address. A station is also capable of sending the bus and detecting the event "Frame ended". The algorithm for sending and receiving is as follows. We assume that each station receives the message (frame) according Poissonian flow at rates $\lambda_i$, $i = 1, N$ respectively. When a station has a frame to send, it waits until it receives the address of its predecessor. Then it waits for the event "Frame ended". The sending time of frame through the channel is distributed by exponential law at rate $\mu$. After that event the station sends its frame and waits some to hear the next station begin sending. When this happens, the sending phase is ended. If a station has nothing to send when its turn comes, it sends an empty no data frame, a kind of a token, to pass the turn to the next station in sequence. At any point in time every station can be switched-off at rate $\nu_i$, $i = 1, N$ and switched-on at rate $\theta_i$, $i = 1, N$ respectively. Both length of time distributed by exponential laws.

6. **EVENT DRIVEN LOCAL NETWORK PROTOCOL NUMERICAL MODEL**

The vector describes the protocol state at any moment of time:

$$S = \{n_1, ..., n_N, n_{N+1}\},$$

where

$$n_i = \begin{cases} 
0, & \text{if the } i \text{ th station is free;} \\
1, & \text{if customer is waiting for service;} \\
2, & \text{if the } i \text{ th station is turned off;} \\
\end{cases} \quad i = 1, N.$$

$$n_{N+1} = \begin{cases} 
0, & \text{if the chanel is free;} \\
i, & \text{if the chanel is passing a mesage from the } i-\text{th station.} \\
\end{cases}$$

The set of events which may occur in the network:

$$E = \{e_{ii} \mid i = 1, N, e_{2j} \mid j = 1, N, e_{3k} \mid k = 1, N, e_4\},$$

where

$$e_{ii} - \text{a customer arrived to the } i\text{th station;}$$

$$e_{2j} - \text{the } j\text{th station is turned off;}$$

$$e_{3k} - \text{the } k\text{th station is turned on;}$$

$$e_4 - \text{a customer is being served.}$$

The description of the protocol in the event language [3].

$$e_{ii}(i = 1, N):$$

- if $n_i = 0$ and $n_{N+1} \neq i$ then $n_i \leftarrow n_i + 1$ end if

$$e_{2j}(i = N + 1, 2N):$$

- if $n_{j-N} \neq 2$ then $n_{j-N} \leftarrow 2$ end if

$$e_{3k}(i = 1, N):$$

- if $n_{k+1} = i - N$ then $n_{k+1} \leftarrow 0$ end if

$$e_{4}(i = 1, N):$$

- if $n_{N+1} < N$ and $\min_{1 \leq j \leq N} (j : n_j = 1)$ then $n_{N+1} \leftarrow j$ $n_j \leftarrow 0$ end if

$$e_{5j}(i = 1, N):$$

- if $\min_{1 \leq j \leq N} (j : n_j = 1)$ then $n_{j+1} \leftarrow j$ $n_j \leftarrow 0$ end if

End of description.
\[ e_3(i = 2N + 1,3N) : \]
\[
\begin{align*}
\text{if } n_{i-2N} = 2 \\
\quad \text{then } n_{i-2N} \leftarrow 0 \text{ end then} \\
\end{align*}
\]
end if

Intens \leftarrow \theta_{i-2N}
end \ e_3

\[ e_4 : \]
\[
\begin{align*}
\text{if } n_{N+1} \neq 0 \\
\quad \text{then } n_{N+1} \leftarrow 0 \text{ end then} \\
\end{align*}
\]
end if

if \[ n_{N+1} < N \text{ and } \min_{N+1 \leq j \leq N} (j : n_j = 1) \]
\[
\begin{align*}
\text{then } n_{N+1} \leftarrow j \\
n_j \leftarrow 0 \text{ end then} \\
\end{align*}
\]
end if

if \[ \min_{1 \leq j \leq N} (j : n_j = 1) \]
\[
\begin{align*}
\text{then } n_{i+1} \leftarrow j \\
n_j \leftarrow 0 \text{ end then} \\
\end{align*}
\]
end if

Intens \leftarrow \mu
end \ e_4

7. THE RESULTS OF MODELLING

The results of numeric modelling of markovian systems were compared with simulation ones. Data in the Table 1 shows that the agreement between both cases is very good.

8. CONCLUSION

The presented method allows automatically construct the basic stages of numerical models of systems: generating the set of states of a system and the transition rates between the states, calculating steady state probabilities and computation of characteristics of the system being modelled. This method has no limitations for complexity of a system. The method is realised in the form of package of programs in C++.

9. REFERENCES


Table 1. The data and results of the numerical model

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Kind of modelling</th>
<th>The coefficient of full loading of a channel</th>
<th>The average time of servicing a customer</th>
<th>The average number of operating stations</th>
<th>The average time of waiting for a station</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda = 1, \nu = 3, \theta = 3, )</td>
<td>simulation</td>
<td>0.536</td>
<td>0.97</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>( \mu = 1, N = 3, i = 1,3 )</td>
<td>numerical</td>
<td>0.534</td>
<td>0.72</td>
<td>1.5</td>
<td>0.189</td>
</tr>
<tr>
<td>( \lambda = 1, \nu = 0.05, \theta = 0.05, )</td>
<td>simulation</td>
<td>0.71</td>
<td>1.64</td>
<td>2.0</td>
<td>0.68</td>
</tr>
<tr>
<td>( \mu = 1, N = 4, i = 1,4 )</td>
<td>numerical</td>
<td>0.709</td>
<td>1.2288</td>
<td>1.9999</td>
<td>0.519</td>
</tr>
<tr>
<td>( \lambda = 1, \nu = 0.33, \theta = 0.33, )</td>
<td>simulation</td>
<td>0.6439</td>
<td>1.088</td>
<td>1.995</td>
<td>0.329</td>
</tr>
<tr>
<td>( \mu = 1, N = 4, i = 1,4 )</td>
<td>numerical</td>
<td>0.649</td>
<td>1.01</td>
<td>2.0</td>
<td>0.366</td>
</tr>
</tbody>
</table>