Method for Automatic Construction of Queuing System Models

Eimutis Valakevicius*
Department of Mathematical Modeling
Kaunas University of Technology, Lithuania
*Corresponding author

Tomas Lazauskas
University College London, UK

***Abstract*—In the paper is suggested a method for automatic creation of queuing system models. For description of a stochastic system we use the schematic structure of the system based on schematic structure of designed components. The performance of a system is described by Markov chains. A phase-type approximation is used for modelling non-Markovian systems. For calculation steady state probabilities of the system the method of embedded chains is applied. The suggested method and tool is presented through an example.**

Keywords-queuing system; markov chain; numerical model; steady state probabilities

I. INTRODUCTION

Markov processes provide very flexible, powerful, and efficient means for description and analysis of dynamic system properties. When a system is modeled by an irreducible continuous time Markov chain(CTMC), there is often an interest in computing a steady state measure which can be expressed as a linear function of the steady state probability distribution of the whole system.

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. Direct and iterative methods can be used for numerical solution of steady state probabilities [3, 4]. Direct methods [5, 6] operate and modify an intensity matrix, and use a fixed amount of computation time independent of the parameter values, but are subject to accumulation of round-off errors and have difficulties with sparse storage [7].

Iterative methods (Power, Jacobi's, Gauss-Seidel's methods) are based on the property of successive convergence to the desired solution. The main advantage of iterative methods, compared with direct methods is that they preserve the sparsity of the intensity matrix, because efficient sparse storage schemes and efficient sparsity-preserving algorithms can be used. The disadvantage of iterative methods is that convergence is not always guaranteed and depends on the method. The rate convergence is highly sensitive to the value of entries in the generator matrix [8].

Discrete event simulation is widely employed for queuing system modeling using static input data [16]. In the recent years the trend of queuing theory development is toward more precision which requires higher mathematical manipulation [15].

In the paper is suggested a method and software for analysis of underlying multi-server and multi-class priority queuing systems queuing systems which enables to simplify the design of schematic structures of queuing systems. In order to model non-Markovian queuing systems we use phase-type approximation [9].

The creation process of numerical models of systems consists of four major steps:

- Specification of the states the system can be in;
- Specification of the rates at which transitions between states take place;
- Computation of the solutions to the model;
- Computation performance measures of the system under investigation.

II. THE IDEA OF CONSTRUCTING QUEUEING MODELS

The basic idea of suggested method is to enable software user simply design the desired queuing system by connecting suitable components.

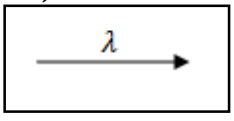
Programmed components are the most common queuing system's elements which have determined set of events and operating principals.

After designing schematic structure, software determines queuing system's state vector and generates all possible states and transition matrix between them depending on architecture of the system. The created software by the method of embedded Markov chains calculates steady states possibilities which are used for estimating system's stochastic characteristics.

The created software enables to define most of the queuing systems in schematic structure which is constructed by connecting and combining specially programmed components. Every component has its parameters and conceptual operating principals.

Let us give detailed information about all the available components.

Customer flow **(Flow).**



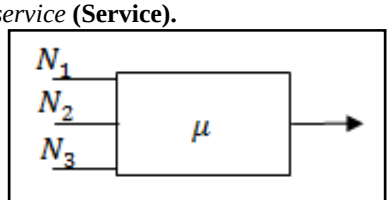
Component's parameters:

- Rate of costumer flow λ (**Rate)**;
- Component number to whom generated costumer is sent to (*To ID*);
- Component's priority row number to whom generated costumer is sent to (*To queue*).

Component's operating principals:

Component is generating costumers by Poisson process with the rate λ to the component's priority row (*To queue*), who's number is put in (*To ID*) parameter.

Costumer service **(Service).**



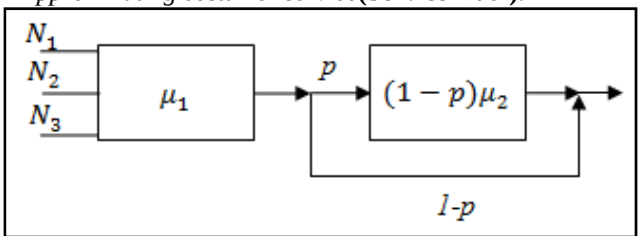
Component's parameters:

- Customer service time rate μ (*Rate*);
- High priority row's length (*L*1);
- Medium priority row's length (*L*2);
- Low priority row's length (*L*3);
- Components' number which customers are received from (From *ID*);
- Component number to whom serviced is sent to (To *ID*);
- Component's priority row number to whom serviced customer is sent to (To *queue*);
- Component's breaking rate (Parameters->*break_int*);
- Component's fixing rate (Parameters->*fix_int);*
- Customers' leaving rate (Parameters->*leave_int)* when service's high priority customers queue's length reaches (Parameters->*leave_n*1) or medium priority customers queue's length reaches (Parameters->*leave_n*2) or low priority customers queue's length reaches (Parameters->*leave*_n3).

Component's operating principals:

Component is serving customers from priority queues. Firstly, all customers from high priority queue are served, then all customers from medium priority and lastly from low priority queue. If medium or low customer is being served at the moment when customer arrives to high priority queue, customer serving stops, returned to its priority queue and customer's from high priority rqueueserving starts. If low customer is served and customer to medium priority arrives, customer's service completes serving low priority customer and begins serving medium priority customer. Customers are arriving from components defined by parameter (From ID), than served with serving time distributed exponentially with rate μ and after serving sent to component's, whose number is defined with parameter (To *ID*) priority queue (To *queue*). Also this component may break down and fixed with the rates (Parameters->*break_int*) and (Parameters->*fix_int*) accordingly distributed by Poisson distribution. When component brakes down costumer which was served at that moment is returned to its priority row if it is not full, otherwise application leaves system. Besides that, costumer may leave its priority row and whole system when priority row's length reaches certain size defined by parameters (Parameters->*leave_n*1, *leave_n*2, *leave_n*3). Service leaving rate is distributed by Poisson process with rate (Parameters->*leave_int*).

*Approximating costumer service***(Service-Dual).**



Component's parameters:

All parameters of costumer's service maybe applied to approximating costumer's service component besides breaking, fixing and leaving parameters. Also two more values must be assigned:

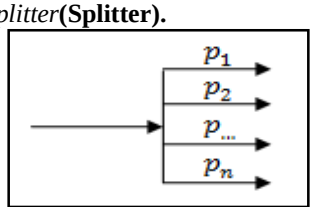
- Second state service time rate μ_2 (Parameters->*intens*_2);
- Probability, that costumer will be sent to second approximating service state is define with parameter (Parameters->*probability*).

Component's operating principals:

This component is design and mostly used for customer services which customer's serving is not distributed by Poisson process, but by other distributions like lognormal, exponential and etc. In this situation useful approximation is obtained by the mixture and convolutions of exponential (phase-type) distributions.

Component's operating principal is almost like costumer's service components, only it cannot be broken, fixed and costumers do not leave their priority rows. Besides that, this component instead of sending served customer to next component may send it to second state with a probability p where it will be serviced by the second costumer service by Poisson process with rate $(1-p)\mu_2$ or with probability $1-p$ customer may leave service without being served in second service.

*Probability splitter***(Splitter).**



Component's parameters:

- Components' number which costumer is received from (From ID);
- Costumers' splitting to components parameter (Parameters->*splits)*.

Component's operating principals:

Customers arrive from components defined by parameter (*From ID*) and according to defined probability splitting (Parameters->*splits*) it directs them to other components. This parameter must follow specific order. Every probable split is separated with semicolon and must have defined probability with which customer is sent to component's priority queue and all these values are separated with commas. For example, let's say we want to send arrived customer to component's, which's ID is 2, low priority queue with probability 0.4 and with probability 0.6 to component's, which's ID is 3, medium priority queue than probability splitter must have defined parameter like *splits*= [0.4, 2, 1; 0.6, 3, 2].

III. Description of a System

Systems state's structure directly depends on a count of customer service and approximating costumer service components that are included in queuing system's schematic structure. Every simple costumer service component may expand state's structure by five additional coordinates. Three of them are used to save priority rows lengths, fourth is for costumer's which is being served priority row's type and a last one is for components health status. As for approximating costumer service component, system's state structure might be expanded by five coordinates. Like in simple costumer service case, first three of them are used to cover priority rows' lengths, fourth for serving costumer priority type in first component's phase and fifth is for costumer priority type in second component's phase.

IV. The Set of Possible Events in the System

Every component described in section III (except probability splitter), has defined (programmed) events that can happen in the system. Let us give detailed information about all the available events.

Customer flow (Flow) events:

- e_1 – costumer flow generates new costumer to referred component.

Costumer service (Service) events:

- e_1 – customer service serves a costumer from priority queues;
- e_2 –customer service breaks down;
- e_3 –customer service is being fixed;
- e_4 –customer leaves high priority queue when it's length reaches size of n1;
- e_5 – customer leaves medium priority queue when it's length reaches size of n2;
- e_6 –customer leaves low priority queue when it's length reaches size of n3;

Approximating costumer service (Service-Dual) events:

- e_1 –approximating customer service serves a customer from priority queue in it's first state and sends it to referred component;
- e_2 –approximating customer service serves a customer from priority queues in it's first state and sends it to its second state.
- e_3 –approximating customer service serves a customer from priority queues in its second state and sends it to referred component.

V. Generation of Possible States and the Infinitesmal Matrix

When we have determined structure of the queuing system and system's set of events, created software generates all the possible system states and transition matrix between them. All the systems analysis of the systems with created software starts with determination of the first state. First state's coordinates are always 0, i.e. no customer in any priority queue of customer services and no customer is being served in any of them.

VI. Example

For better understanding, let us examine an example to show how created software is used for queuing system's analysis. Let us say that customs inspection station has two terminals. First is used for cars' and cars' with priority permits and second is for trucks' inspection. Cars and cars with permits arrival processes are Poisson with rates 2 and 0.01 respectively, and trucks arrival is Poisson process with rate 3. Suppose that maximum 30 cars can be in first terminal's queue and one car with permits in second queue and second terminal queue's length cannot be over 40 trucks. Cars with permits and trucks are checked at the terminals by the exponential processes with the rates 2.5 and 4 respectively. This example as queuing system is represented in Figure 1.

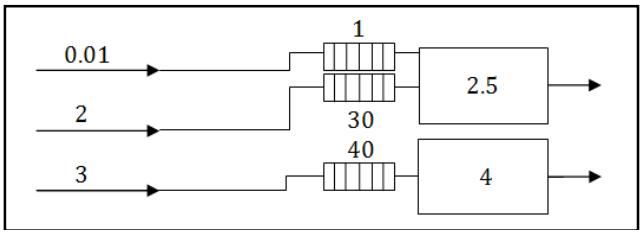


FIGURE I. QUEUING SYSTEM CUSTOMERS CUSTOMERS CONTROL

Let us describe example as a queuing system in stochastic structure using components from section 3. There are three customer flows: cars, cars with permits and trucks, so we need three customer flows who ID numbers respectively are 1, 2 and 3. Also two simple customer services are required for cars (ID = 4) and trucks (ID = 5). Now we can define components' parameters:must be embedded into the text and not supplied separately. Below is an example which authors may find useful.is an example which authors may find useful.

TABLE I. INPUT DATAOFTHE MODEL

ID	Comp.	Rate	From ID	To ID	To Row	Len. 1	Len. 2	Len. 3
1	Flow	0.01	-	4	1	-	-	-
2	Flow	2	-	4	2	-	-	-
3	Flow	3	-	5	2	-	-	-
4	Service	2.5	1,2	-	-	1	30	-
5	Service	4	3	-	-		40	-

In example queuing system's state will be $(n_1, n_2, n_3, n_4, n_5)$ where n_1 - length of high priority queue – cars with permissions and $n_1 = 0,1$, n_2 - length of medium priority queue – cars and $n_2 \in \overline{0,30}$, $n_3 \in \overline{0,2}$ (0 – no car is being inspected, 1 – car with permit is being inspected, 2 – car is being inspected) – priority queue number of a customer who is being served in first terminal, $n_4 \in \overline{0,40}$ - length of medium priority queue – trucks in second terminal and $n_5 = 0,2$ (0 – no truck is being inspected, 2 – truck is being inspected)– identifies if truck is being served in second terminal.

According to section 5 the set of events in example's system: $E = \{e_1, e_2, e_3, e_4, e_5\}$ where:

e_1 – customer arrives to first customer service component's high priorityqueue;

e_2 –customer arrives to first customer service component's medium priority queue;

e_3 –customer arrives to second costumer service component's medium priorityqueue;

e_4 – first customer service component completes servingcustomer;

e_5 – secondcustomer service component completes serving customer.

At the first iteration software tries to carry out all the events from system's events set to the systems first state$(0,0,0,0,0)$.

When system is in $(0,0,0,0,0)$ state and event e_1 occurs (customer arrives to first customer service's high priorityqueue) system's state changes to $(0,0,1,0,0)$, i.e. first customer service's priority queues were empty and no customer was being served in it, so when thiscustomer arrives it will be served immediately. As it was defined in example description, event e_1 happens with rate 0.01. Since generated state $(0,0,1,0,0)$ is new in system, system's states set is appended by it

$$S = \{(0;0;0;0;0), (0;0;1;0;0)\}$$

and transition matrix is expended

$$TRN = \begin{vmatrix} 0 & 0.01 \\ 0 & 0 \end{vmatrix}$$

respectively where $TRN_{12} = 0.01$ defines transition rate between states $(0,0,0,0,0)$ and $(0,0,1,0,0)$.

When event e_2 (customer arrives to first customer service's medium priorityqueue) occurs while system is in

$(0,0,0,0,0)$state, system behaves similar like in previous case, just in this case system's state changes to $(0,0,2,0,0)$with transition rate 2. System states set becomes

$$S = \{(0;0;0;0;0), (0;0;1;0;0), (0;0;2;0;0)\}$$

with transition matrix

$$TRN = \begin{vmatrix} 0 & 0.01 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}.$$

When e_3 (customer arrives to second service's medium priorityqueue) occurs system's state changes to $(0,0,0,0,2)$with transition rate 3, than

$$S = \{(0;0;0;0;0), (0;0;1;0;0), (0;0;2;0;0), (0;0;0;0;2)\}$$

and

$$TRN = \begin{vmatrix} 0 & 0.01 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}.$$

Events e_4 and e_5 can not happen when system is in its first state, because no customer is in system and there is no customer that can be served.

Second iteration starts with the first iteration's generated states and tries to perform all the events from events set to every new state from first iteration. Iteration process is continued till iteration does not generates any new states.

After last iteration created software using embedded Markov chains method calculates system's steady states probabilities and using these calculations, system's computation measures of the system performance.

VII. MODELLING RESULTS

Let us analyze one class one server queuing system with queue limitation. Detailed information about one of the *m* modeling results is displayed in following table 2, where customer flow and customer service intensities are 2 and 3 with medium priority row's length 50.

TABLE II. PROBABILISTIC MEASURES

$\lambda = 2; \mu = 3; m = 50.$					
State	**No**	**Calculated probability**	**Modellated probability**	**Error**	**Relative error**
(0;0)	1	3.33333333565678E-01	3.33333333565678E-01	0.00E+00	0.00E+00
(0;2)	2	2.22222222377119E-01	2.22222222377119E-01	0.00E+00	0.00E+00
(7; 2)	9	1.30061474459402E-02	1.30061474459402E-02	0.00E+00	0.00E+00
(12;2)	14	1.71274369658472E-03	1.71274369658472E-03	0.00E+00	0.00E+00
(13;2)	15	1.14182913105648E-03	0.14182913105648E-03	0.00E+00	0.00E+00
(25;2)	27	8.80047290443781E-06	8.80047290443782E-06	0.00E+00	0.00E+00
(26;2)	28	5.86698193629187E-06	5.86698193629188E-06	-1.02E-20	1.73E-15
(38;2)	40	4.52188634506022E-08	4.52188634506023E-08	-9.93E-23	2.20E-15
(39;2)	41	3.01459089670681E-08	3.01459089670682E-08	-9.93E-23	3.29E-15
(48;2)	50	7.84164273288567E-10	7.84164273288569E-10	-2.07E-24	2.64E-15
(49;2)	51	5.22776182192378E-10	5.2277618219238E-10	-2.07E-24	3.96E-15
(50;2)	52	3.48517454794919E-10	3.4851745479492E-10	-9.82E-25	2.82E-15

VIII. Conclusion

A new method and created software for modelling queuing systems described by Markov chains allows:

- to simply design most of the queuing systems in schematic structure for analysis,
- to analyze non-Markovian queuing systems using phase-type approximation,
- to obtain high accuracy result of systems characteristics.

References

[1] S.P. Meyn, R. Tweedie, Markov Chains and Stochastic Stability.Spinger – Verlag, 1993.

[2] H.C. Tijms, Stochastic Models: An Algoritmic Approach. John Wiley &Sons, 1995.

[3] W. J. Stewart, Numerical Methods for Computing Stationary Distributions of Finite Irreducible Markov Chains. Chapter 3 ofAdvances in Computational Probability. Edited by Winfried Grassmann; Kluwer Academic Publishers, 1999.

[4] Edited Proceedings of The Third International Conference on the Numerical Solution of Markov

[5] Chains, PrensasUniversitarias de Zaragoza, Spain, 344 pages, 1999.

[6] T.J. Sheskin, "A Markov partitioning algorithm for computing stabdy-state probabilities", Oper. Res. 33(1985), pp. 228-535.

[7] W. K. Grassmann, M. I. Taksar, and D. P. Heyman, Regenerative anglysis and steady-state distributions for Markov chains, Oper. Res. 33(1985), pp. 1107-1116.

[8] G. Bolch, S. Greiner, H. Meer, K. Trivedi, Queueing Networks and Markov Chains: Modelling and Performance Evaluation with Computer Science Applications. New York, John Wiley& Sons (1994), 726.

[9] K. Trivedi, Probability and Statistics with Reliability, Queueing, and Computer Science Applications. New York, John Wiley&Sons, 830.

[10] E. Valakevicius, "Numerical Modeling of Queuing Systems Based on Phase Type Distribution", WMSCI 2007, Orlando, Florida, USA, pp.103-107.

[11] W. J. Stewart, Numerical Solution of Markov Chains. Princeton, New Jersey, 1994.

[12] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, Queuing Networks and Markov Chains. Modelling and Performance evaluation with Computer Science applications. New York, Wiley-Interscience, 2006.

[13] Y.P. Zhou, N. Gans, "A Single-Server Queue with Markov Modulated Service Times", Working paper, The Wharton School, University of Pensilvania. [Online]. Available: http:fic.wharton.upenn.edu/fic, 2005, pp. 1-33.

[14] S. R. Mahabhashyam, N. Gautam, "On queues with Markov modulated service rates", Queueing Systems, Vol. 51, 2005, pp. 89-113.

[15] K, Teknomo, "QueuingRuleofThumbbasedon M/M/s QueuingTheorywithApplicationsinConstructionManagement", CivilEngineeringDimension, Vol. 14, No. 3, 2012, pp. 139-146.

[16] R. Akhavian, A. H. Behzadan, "Evaluationofqueuingsystemsforknowledge-basedsimulationofconstructionprocesses, AutomationinConstruction", Vol. 47, 2014, pp. 37-49.