Analysis of Algorithms for Generating the Matrix of Coefficients of Non-Stationary Queuing System Models

Kirill Shardakov[a] and Vladimir Bubnov[a]

[a] *Emperor Alexander I St. Petersburg State Transport University, 9 Moskovsky pr., Saint Petersburg, 190031, Russia*

Abstract

The article discusses recursive and recursive with grouping algorithms for generating a list of states of a non-stationary queuing system and generating a matrix of coefficients for the system of homogeneous differential Chapman-Kolmogorov equations which describes the queuing system under consideration. The analysis of the operation of these algorithms has been carried out. Pros and cons considered.

Keywords

Non-stationary queuing system, modeling, numerical-analytical model, system of homogeneous differential equations, matrix of coefficients, list of states

1. Introduction

[1] The current state of the issue of non-stationary queuing systems is considered in more detail in [1]. The beginning of the "non-stationary" queuing systems was laid in [2–4] and continued in [5–6].

In [7-8], a model is proposed that uses a recursive with grouping algorithm for generating a list of system states and a matrix of coefficients of a homogeneous differential equations system without constructing a graph of states and transitions of the non-stationary queuing systems and deriving the general equation of the homogeneous nary differential equations system as in [9-12]. In this paper, the advantages and disadvantages of the proposed recursive grouping algorithm are considered in comparison with the recursive algorithm for generating a list of states and a matrix of coefficients. The result of the work makes it possible to justify the choice of a recursive grouping algorithm for use in a non-stationary model, as the fastest and does not lose accuracy.

2. Comparative analysis

A recursive with grouping algorithm for generating a list of possible states of a non-stationary queuing system was developed as an alternative to the existing recursive [13] and sequential algorithms [14].

One of the key differences between recursive and grouped recursive algorithms is the need to pre-calculate the number of possible system states. Another such difference is making changes to the state storage structure. To store undersubgroups, the recursive grouping algorithm uses dictionaries, that is, "key": "value" pairs to quickly check for the existence of a state and find it. Thus, a structure for storing the list of states is obtained, shown in Figure 1, where the state is described by two vectors.

The logic of the recursive algorithm is implemented in such a way that, first of all, the number of possible states of the non-stationary queuing system is calculated, then a zero matrix of coefficients A is created, and only after that new states are generated and changes are made to the matrix on the fly. This gives rise to the need for each individual model to specially select a formula that makes it possible to calculate the possible number of states of the non-stationary queuing system.

The recursive with grouping algorithm avoids this action, since it first generates all possible states, and only when all possible states of the non-stationary queuing system and their number are known, a zero matrix of coefficients A of the required dimension is created.

[1] Intelligent Transport Systems. Transport Security - 2021, May 14, St. Petersburg, Russia

EMAIL: k.shardakov@gmail.com (K.S. Shardakov); bubnov1950@yandex.ru (V.P. Bubnov);
ORCID: 0000-0001-9645-7301 (K.S. Shardakov); 0000-0002-6742-3011 (V.P.Bubnov)

CEUR Workshop Proceedings (CEUR-WS.org)

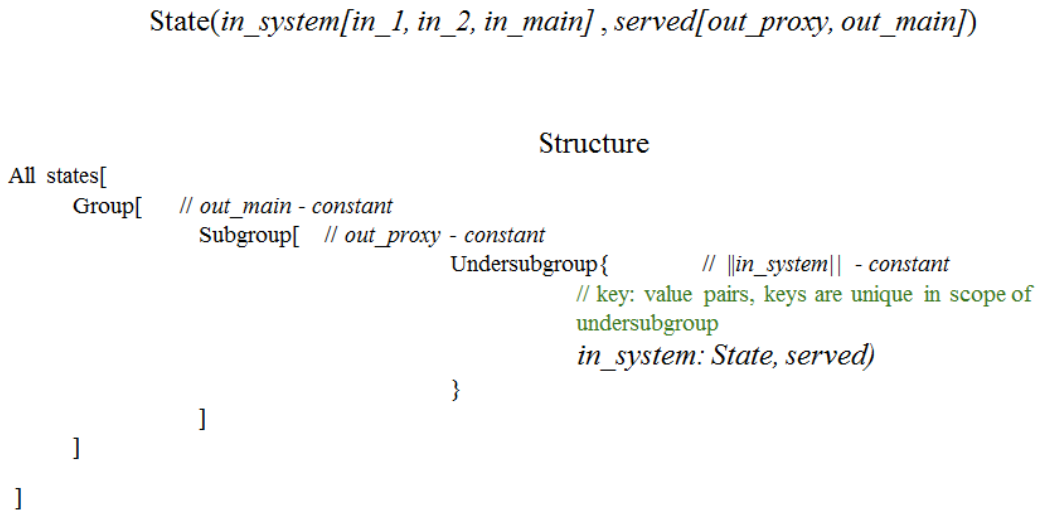

Figure 1: Storage structure of a list of states for a recursive with grouping algorithm

Measurements were taken to compare the performance of these two algorithms. Both algorithms for generating the matrix of coefficients of the homogeneous differential equations system without deriving the general equation of the system were implemented using Python3. All measurements were carried out when running the algorithms on the same device, which allows the obtained values to be considered valid for comparison. The execution time of the algorithms may differ upward or downward depending on the platform performance on which the algorithms are run, but the general trends will remain correct. The recursive algorithm was launched on the simplest single-channel model, and the recursive with grouping on a parallel-sequential model with 2 proxies. However, the initial data were selected in such a way that the total number of possible states of both models for comparable results was as close to each other as possible. The timer starts together with the start of the algorithm and stops after the matrix of coefficients A is completely filled and brought to the lower triangular form. The results of the comparative analysis are presented in table 1 and table 2.

As can be seen from Tables 1 and 2, for both algorithms and models, the number of possible states of the system, and, consequently, the number of equations in the homogeneous differential equations system grows exponentially depending on the number of tasks that can enter to the non-stationary queuing system. The graph for the recursive with grouping algorithm in the parallel-serial model with 2 proxies is shown in Figure 2.

As can be seen from Figure 2, the number of possible states for the parallel-sequential model, as well as for the simplest single-channel model considered earlier, grows exponentially.

Figure 3 shows a comparison of the total execution time of the recursive and recursive with grouping algorithms. The criterion for turning off the timer is a fully filled and triangular form of matrix of coefficients.

Table 1
Recursive algorithm execution time

Tasks	States	Execution time excluding sorting step, seconds	Total execution time, seconds
2	6	0.0010004043579101562	0.0010004043579101562
4	15	0.0010001659393310547	0.0010001659393310547
7	36	0.0009999275207519531	0.008000612258911133
10	66	0.002000093460083008	0.023001432418823242
14	120	0.0060002803802490234	0.11900663375854492
19	210	0.012000799179077148	0.2560145854949951
24	325	0.026001453399658203	0.7240414619445801
30	496	0.06500387191772461	2.1061205863952637
36	703	0.13100767135620117	5.188296794891357
43	990	0.2390139102935791	14.486828565597534
50	1326	0.45102596282958984	37.64615321159363

Table 2
Recursive wuth groupong algorithm execution time

Tasks	States	Full execution time, seconds
1	5	0.0010001659393310547
2	15	0.0009999275207519531
3	35	0.002000093460083008
4	70	0.0060002803802490234
5	126	0.008000612258911133
6	210	0.01500082015991211
7	330	0.029001712799072266
8	495	0.04500269889831543
9	715	0.06800389289855957
10	1001	0.09200549125671387
11	1365	0.11600661277770996

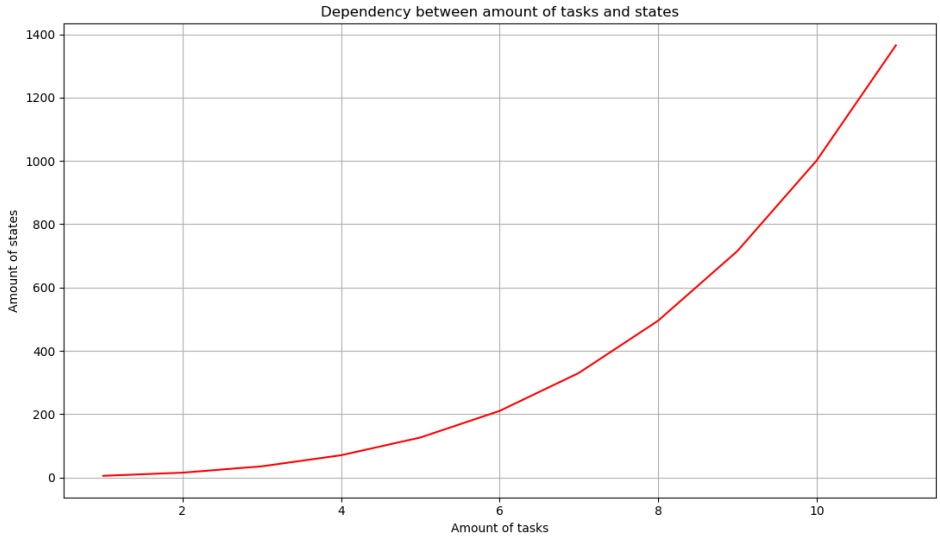

Figure 2: Dependence of the number of states of the non-stationary queueing system on the number of tasks entering to the non-stationary queueing for a parallel-serial model with 2 proxies

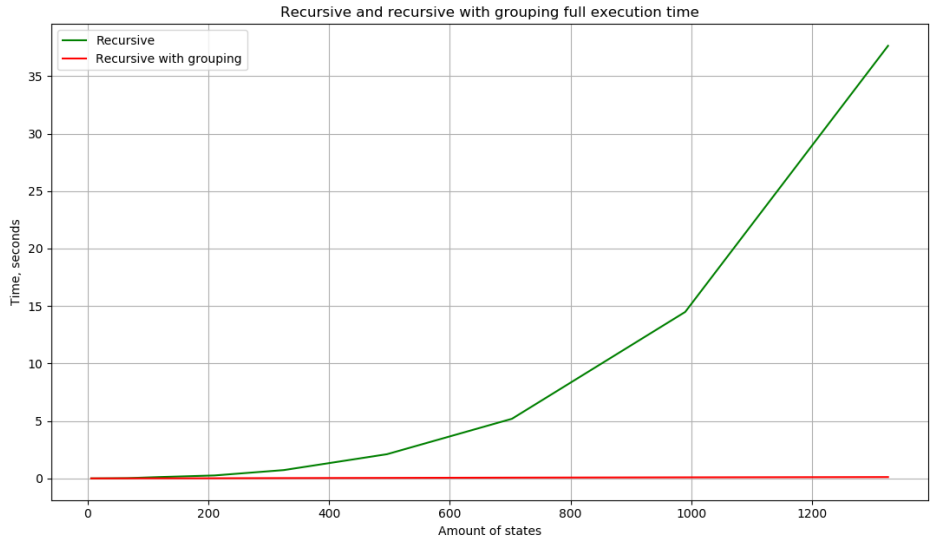

Figure 3: Generation time of triangular matrix of coefficients by recursive and recursive with grouping

As you can see from Figure 3, the behavior is similar to the comparison between sequential and recursive algorithms in [14]. The time taken by the recursive algorithm is significantly greater than the time taken by the recursive grouped algorithm. It should be noted that the more the number of states, the more the gap in the running time of these algorithms increases.

As noted earlier, the recursive algorithm spends most of its time sorting the coefficient matrix when sorting the list of states. Figure 4 shows a graph illustrating a comparison of the total execution time of the recursive with grouping algorithm and the execution time of the recursive algorithm excluding the sorting stage and reducing the matrix of coefficients to a triangular form. This will allow us to compare the pure speed of the algorithms, taking into account only the process of generating the list of states and the initial filling of the coefficient matrix.

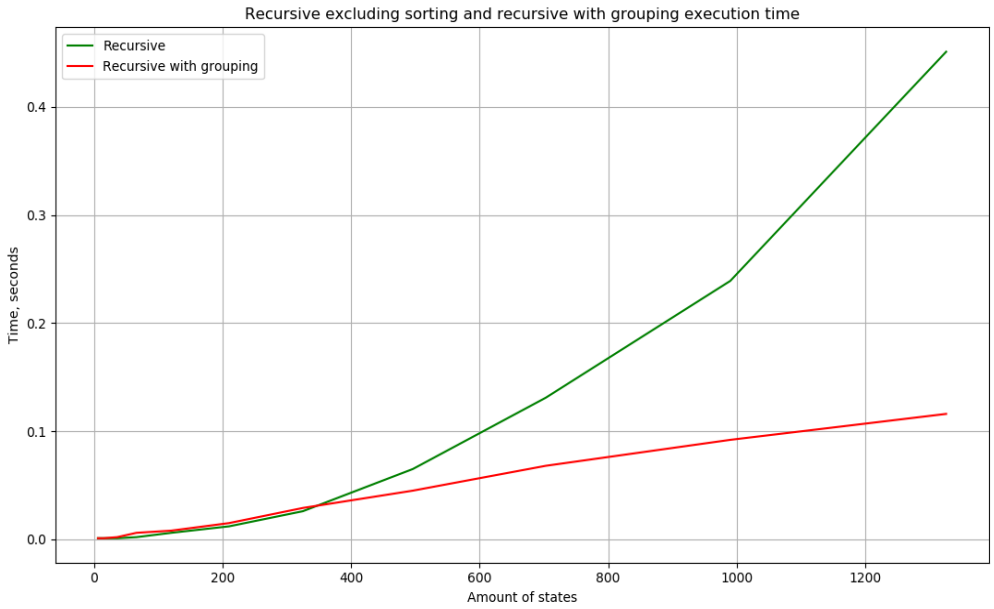

Figure 4: Time of filling the matrix of coefficients with recursive and recursive with grouping algorithms

As you can see from Figure 4, even in the case of skipping the steps of sorting the coefficient matrix A and reducing it to a triangular form in the recursive algorithm, it is still inferior in speed to the proposed recursive with grouping.

Upon closer examination of the recursive algorithm, you can see that most of the time (already keep in mind the exclusion of time for sorting procedure) this algorithm spends on finding a state in the list of already existing states when calling the is_exist() method. A detailed comparison is presented in tables 3 and 4.

According to the values from Tables 3 and 4, several graphs were built, illustrating the division of the running time of the algorithms.

Table 3
State searching time for recursive algorithm

Tasks	States	Execution time excluding sorting step, seconds	State searching time, seconds
20	231	0.016000747680664062	0.012000560760498047
25	351	0.034001827239990234	0.027001380920410156
30	496	0.08300471305847168	0.06400418281555176
35	666	0.11100625991821289	0.10400629043579102
40	861	0.17500996589660645	0.15900921821594238
45	1081	0.27001523971557617	0.24901413917541504
50	1326	0.4470255374908447	0.42302393913269043
55	1596	0.6100349426269531	0.575031042098999
60	1891	0.8840506076812744	0.8310489654541016
65	2211	1.2500712871551514	1.1970674991607666

Table 4
State searching time for recursive with grouping algorithm

Tasks	States	Execution time, seconds	State searching time, seconds
6	210	0.014000892639160156	0.004000663757324219
7	330	0.02800130844116211	0.008000612258911133
8	495	0.052002906799316406	0.011000871658325195
9	715	0.0870048999786377	0.013001203536987305
10	1001	0.10800600051879883	0.0260007381439209
11	1365	0.14000797271728516	0.034002065658569336
12	1820	0.1740100383758545	0.04200029373168945
13	2380	0.29001688957214355	0.057003021240234375
14	3060	0.38802194595336914	0.08100318908691406
15	3876	0.593034029006958	0.09700489044189453

Let us first of all consider the operation of the recursive algorithm for the simplest one-channel model. Figure 5 shows a graph of the ratio of the total execution time of the recursive algorithm (excluding the time for sorting) and the time spent in the process of searching for a state in the list of states. As can be seen from Figure 5, the recursive algorithm spends most of its execution time in the process of searching for a specific state among the existing states in the list. Figure 6 shows a graph of the time spent on the search as a percentage of the total execution time of the recursive algorithm.

As you can see from Figure 6, in percentage terms, the recursive algorithm spends from 75 to 95 percent of its execution time looking for a state in the general list of states.

Consider a recursive with grouping algorithm in a parallel-serial 2-proxy model. Figure 7 shows a graph of the ratio of the total execution time of the recursive with grouping algorithm and the time spent in the process of searching for a state in the structure of states.

As you can see in Figure 7, the recursive with groupong algorithm spends much less of its execution time searching for a specific state among the existing states in the list. Figure 8 shows a graph of the time spent on the search as a percentage of the total execution time of the recursive grouped algorithm.

As shown in Figure 8, in the case of the recursive clustering algorithm, the time spent searching for the desired state in the state structure varies between 16 and 28 percents of the total execution time of the algorithm. Which is much lower compared to the 75-95 percent range for the recursive algorithm.

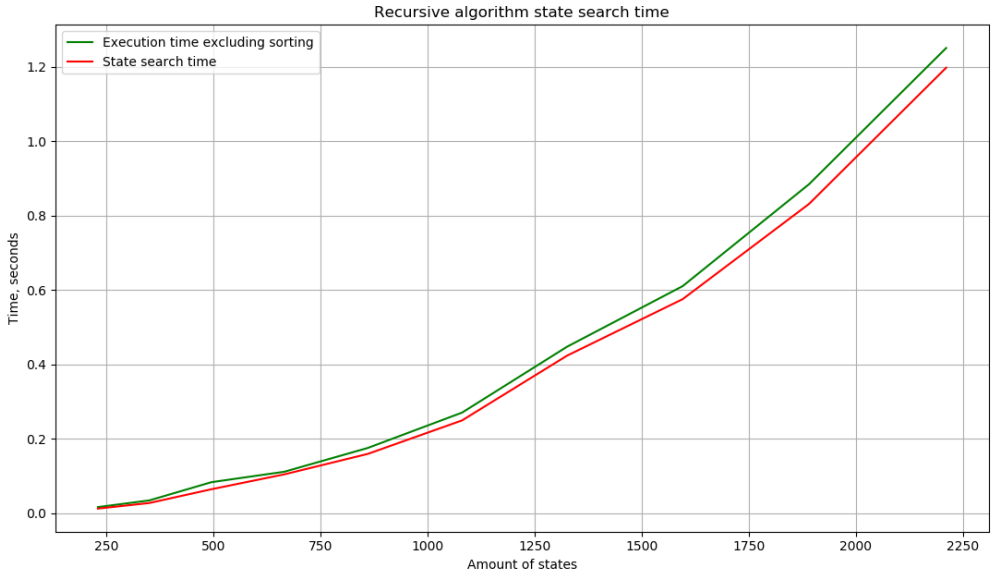

Figure 5: The total time (excluding sorting) to execute the recursive algorithm and the time to search for a state in the list of states

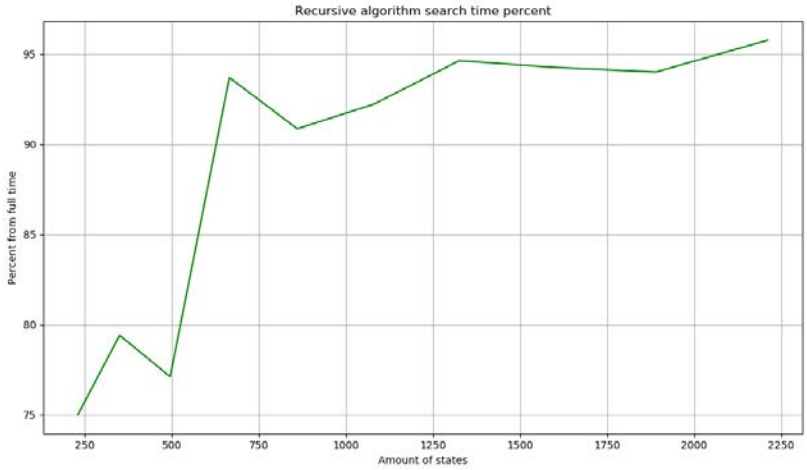

Figure 6: Percentage of time to search for a state in the list of states of the total execution time (excluding sorting) of the recursive algorithm

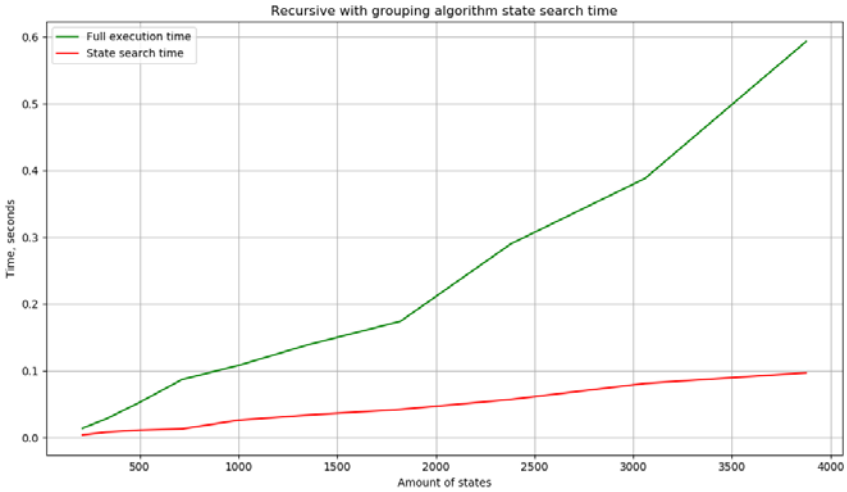

Figure 7: The total time to execute the recursive with grouping algorithm and the time to search for a state in the state structure

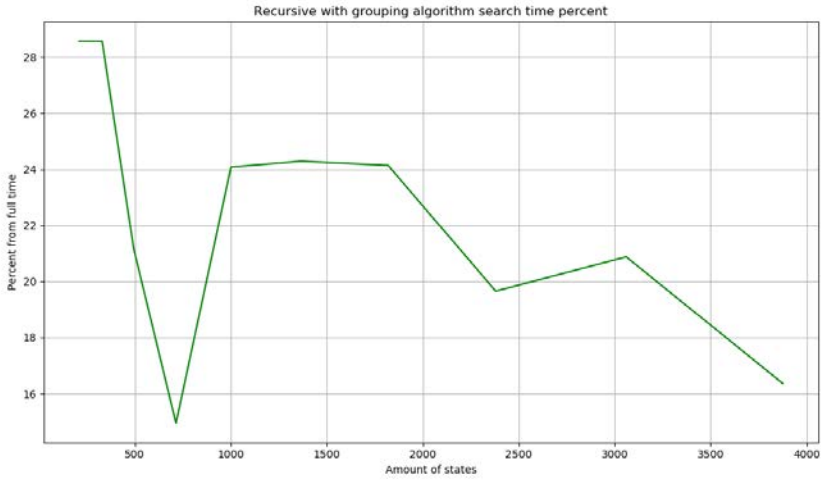

Figure 8: Percentage of time to search for a state in the structure of states of the total execution time of the recursive grouping algorithm

3. Conclusion

The indicators described in the paper confirms the effectiveness of using the grouping of states in the proposed structure for storing states, as well as the use of dictionaries (a data type that stores pairs {"key:" value}) within this structure.

Thus, the recursive with grouping algorithm is much faster than the recursive algorithm due to:

1. there is no need to sort the resulting list of states and the matrix of coefficients to bring it to a triangular form;

2. states grouping addition and a structure for storing them instead of a regular list, which makes it possible to find it immediately when searching for a state, referring to a specific address within the structure instead of going through all the states in the list until the desired one is found.

Further optimization of the recursive with grouping algorithm may include:

1. add parallelization of changes in the coefficient matrix, which will allow to process large models with a large number of possible states more efficiently;

2. add optimization of the structure of storing the list of states by getting rid of empty nested structures, which will save the memory spent on storing the list of states.

4. Acknowledgements

The research described in this paper is partially supported by the Russian Foundation for Basic Research (19–08–00989, 20-08-01046), state research 0073–2019–0004.

References

[1] V. Bubnov, V. Safonov, K. Shardakov, Overview of existing models of non-stationary queuing systems and methods for their calculation. Systems of Control, Communication and Security, 2020, no. 3, pp. 65–121 (in Russian). DOI: 10.24411/2410-9916-2020-10303

[2] I. Kovalenko About the queuing system with the speed of service, depending on the number of requirements in the system, and periodic shutdown of channels. Problems of Information Transmission, 1971, vol. 7, no. 2, pp. 108–114 (in Russian).

[3] I. Ezhov, M. Korneichuk, I. Oliinyk D. Distribution of the number of repair system channels when the flow rate changes in a special way. Kibernetika, 1976, no. 3, pp. 92–97 (in Russian).

[4] L. Nonstationary queuing problem for systems with an infinite number of channels with group receipt of requirements. Problems of Information Transmission, 1968, vol. 4, no. 3, pp. 99–102 (in Russian).

[5] G. Arsenishvili Singleline queuing system with queue-dependent incoming flow rate. Soobshcheniia Akademii nauk Gruzinskoi SSR, 1974, vol. 76, no. 2, pp. 285–288 (in Russian).

[6] B. Conolly Generalized State Dependent Eriangian Queues (speculation about calculating easure of effectiveness). Applied Probability, 1975, no. 2. pp. 358–363.

[7] K. Shardakov, V. Bubnov, Stochastic Model Of A High-Loaded Monitoring System Of Data Transmission Network // Proceedings of Models and Methods of Information Systems Research Workshop. 2019. P. 29–34.

[8] K. Shardakov, V. Bubnov, Non-stationary parallel-serial model of a high-load monitoring system. Informatsiia i kosmos, 2020, No. 3, pp. 56-67 (2020). (in Russian).

[9] V. Bubnov, V. Safonov, V. Smagin About the loading of a computing system with a varying intensity of receipt of tasks, Automatic Control and Computer Sciences, 1987, no. 6, pp. 19–22 (in Russian).

[10] V. Bubnov, V. Safonov Development of dynamic models of non-stationary queuing systems. Saint-Petersburg, 1999. 64 p. (in Russian).

[11] V. Bubnov A. Khomonenko, A. Tyrva Software reliability model with coxian distribution of length of intervals between errors detection and fixing moments// International Computer Software and Applications Conference. 2011.pp. 310-314.

[12] V. Bubnov, A. Tyrva, A. Khomonenko Model of reliability of the software with coxian distribution of length of intervals between the moments of detection of errors // International Computer Software and Applications Conference. 34th Annual IEEE International Computer Software and Applications Conference, COMPSAC 2010. 2010. pp. 238-243

[13] V. Bubnov, A. Khomonenko, S. Sergeev A recursive method for generating a

matrix of coefficients of a system of homogeneous differential equations describing a non-stationary queuing system. Proceedings of 2015 XVIII International Conference on Soft Computing and Measurements (SCM-2015). Saint-Petersburg, 2015, vol. 1, pp. 164–166 (in Russian).

[14] K. Shardakov Sequential algorithm for generating a matrix of a coefficients for a system of homogeneous differential equations in a non-stationary queueing system model. Intellectual technologies on transport, 2018, No 4, p. 20–25 (in Russian).