Informatics and Control Problems 39 Issue 1 (2019) 47-55
journal homepage: www.icp.az

Analysis of perishable queueing-inventory system with MMPP flow and different types of customers

A.Z. Melikov[a*], M.O. Shahmaliyev[b]

[a] *Institute of Control Systems of ANAS, Baku, Azerbaijan*
[b] *National Aviation Academy of Azerbaijan, Baku, Azerbaijan*

ARTICLE INFO

Article history:
Received 10.04.2019
Received in revised form 01.05.2019
Accepted 22.05.2019
Available online 12.06.2019

Keywords:
Perishable queuing-inventory system
Infinite 3D Markov Chain
Stationary distribution
Simulation algorithm
(s, S) replenishment policy
Numerical experiments
MMPP flow
Total run cost optimization

ABSTRACT

Perishable queueing-inventory system with positive service time and different types of customers is considered. Customers are arriving according to the Markov-Modulated Poisson process (MMPP). The inventory size is finite, while the queue length is infinite. Customers after the service completion either purchase the inventory item or leave the system empty-handed. The inventory items perish independently after exponentially distributed random times. The customers in the queue become impatient when the inventory level drops to zero. Impatient customers according to Bernoulli trial leave the system after exponentially distributed random times. If there are no items left in the inventory at the moment of arrival, the customer either enters the queue or leaves the system according to Bernoulli scheme. The (s, S) policy is used for the inventory replenishment. Gillespie's Direct simulation method is used to calculate the stationary distribution and performance measures of the system. The convergence speed of simulation, dependence of performance measures on reorder level and solution of optimization problem are considered and illustrated in the numerical experiment.

1. Introduction

Queueing-Inventory System (QIS) that combines the queueing and inventory control system properties has been studied extensively through last decades by researchers. A variety of models has been proposed and analyzed up to now. The most recent review of QIS models could be found in [1].

In the majority of classical QIS models, the inventory items are considered non-perishable. This assumption is not compliant with the real systems where items has finite lifetimes. The examples are blood banks, food stocks, pharmacy etc. The analysis of the existing literature shows that the perishable QIS models are less studied. The first works in this direction we found are [2-4]. In [2] author considers inventory with items that have deterministic lifetimes. In [3] two joint inventory systems with fresh and older items are analyzed using different replenishment policies. The recent works about perishable QIS models we found are [5-11]. Continuous review perishable inventory system with a finite number of homogeneous sources of demands was studied in [5]. The lifetime of items in inventory is assumed to be exponential. The joint probability distribution of the inventory level and the number of demands is obtained. The perishable QIS system with MAP arrival is considered in [6]. Perishable queueing inventory systems with different types of customers are studied

[*]Corresponding author.
E-mail addresses: agassi.melikov@gmail.com (A.Z. Melikov), mamed.shahmaliyev@gmail.com (M.O. Shahmaliyev)

www.icp.az/2019/1-06.pdf

in [8, 10, 11].

The other assumption made in the most QIS models is that the customer purchases the item after the service completion. This condition is not satisfied in some real systems. For example, the customer may not purchase the goods after the consultation due to the low service level. In that case, the inventory level remains unchanged after the service completion. The non-perishable models with such customers were first studied in [12, 13]. The perishable QIS model with different types of customers was considered in [8].

The analysis of existing literature shows that in the most of the QIS models only customers with Poisson arrival are considered. The reason is the unique mathematical properties of exponential distribution that simplifies the calculation and formula derivation processes. Some examples of QIS models with MAP arrival are studied in [6, 14-16]. The QIS models with variable arrival intensities are more compliant with the real systems.

Taking into consideration the above information, we have been motivated to study the perishable QIS system with MMPP arrival. Markov Modulated Poisson Process (MMPP) is the special case of MAP where the customer arrival alternates between Poisson processes of different intensities [17]. We could not found any works that consider QIS models with MMPP arrival and perishable inventory. The problem is to find the joint stationary distribution of MMPP state, inventory level, queue length of the system, and derive formulas for the performance measures. Total run cost optimization problem is solved and numerical results are presented as well.

2. Model description

Customers are arriving into the single channel and single server system according to MMPP flow with parameters (Σ, Λ), where Σ represent transition matrix of size $K \times K$ between the K possible arrival rates that are described by vector $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_K); K > 1$. The element $\beta_{ij} \epsilon \Sigma$ represents the transition intensity between the arrival rates λ_i and λ_j. Customers are accepted for the service if the server is idle at the moment of arrival. Otherwise, the customer joins the queue of infinite size. Customers are assumed to join the queue with probability ϕ_1 when the inventory level is zero and to leave the system with probability ϕ_2, $\phi_1 + \phi_2 = 1$. When inventory level drops to zero customers that are waiting in the queue become impatient and leave the system independently after some exponentially distributed time with parameter τ.

The customer after service completion either leaves the system empty-handed with probability σ_1 or purchases the inventory item with probability σ_2 according to Bernulli trial, $\sigma_1 + \sigma_2 = 1$. The service intensities for these customers are described by exponential random variables with parameters μ_1 and μ_2 accordingly, where $\mu_1 > \mu_2$. If the customer acquires the item after the service completion the inventory level decreases by unit. Because the queue size is unlimited, we assume that $\frac{\max(\lambda_i)}{\min(\mu_1\sigma_1, \mu_2\sigma_2)} < 1$ in order to guarantee the ergodicity of the system.

The items in the inventory perishes independently after some exponentially distributed time with parameter γ. We assume that the item reserved while serving the customer cannot perish. The inventory level decreases by unit after the item perishing event.

The system has the inventory of limited capacity S that is monitored by the system continuously. The inventory is replenished according to (s,S) policy. The replenishment order of size $S - s$ is placed whenever the inventory level drops to the predefined threshold s, where $s < S/2$. Replenishment order lead time is described by exponential random variable with parameter ν.

We divide the required performance measures into two related categories below:

- Inventory related performance measures:
  - S_{av} – average inventory level;
  - RR – average replenishment rate;

- Γ_{av} – average perishing rate;

- Customer service related performance measures:
  - L_{av} – average number of customers in queue;
  - RL – average customer loss intensity.

The system is modeled using the continuous three-dimensional Markov chain and the current state is denoted by (m, n, k) where the variables m, n, k indicate the inventory level, the number of customers in the system (including the one being served) and the MMPP state correspondingly. The system State Space is defined as the follows:

$$E = \{(m, n, k): m = 0, 1 \dots S;\ n = 0, 1, \dots ;\ k = 0, 1, \dots, K ;\}$$

We consider the following notations before constructing the transition rate matrix:

- $p(m, n, k)$ – the stationary probability of the state (m, n, k);
- $q((m_1, n_1, k_1), (m_2, n_2, k_2))$ – transition rate from the state (m_1, n_1, k_1) to the state (m_2, n_2, k_2);
- $I(A)$ – the indicator function of the event A.

The Q-matrix of the model is described by pseudo-code in Fig. 1. To simplify the transition matrix let us consider the following observations. According to the model description all the state transitions could be divided into three groups:

- MMPP state transition event $(k_i \to k_j)$ occurs with the intensity β_{ij} and other state components $(m_1 = m_1, n_1 = n_2)$ remains unchanged.
- When the inventory level is positive $(m_1 > 0)$ the state transition is possible only after the customer arrival (λ_k), item perishing $((m_1 - I(n_1 > 0))\gamma)$, service completion $(\mu_1\sigma_1 \ or \ \mu_2\sigma_2)$ or inventory replenishment (ν) events. After such transitions MMPP state does not change, $k_1 = k_2 = k$.
- When the inventory level is zero $(m_1 = 0)$ state transitions after service completion is not possible. In that case, state transition occurs only after customer arrival $(\lambda_k\phi_1)$, item perishing $((m_1 - I(n_1 > 0))\gamma)$ or inventory replenishment (ν) events and additionally, when impatient customer leaves the system $(n_1\tau)$.

```
1: function Q-MATRIX(m1, n1, k1, m2, n2, k2)        ▷ q((m1, n1, k1), (m2, n2, k2))
2:     define q := 0
3:     if k1 ≠ k2 and m1 = m2 and n1 = n2  then q := β_{k1 k2}
4:     else if m1 > 0 and k1 = k2 then
5:         if m2 = m1 and n2 = n1 + 1 then q := λ_{k1}
6:         else if m2 = m1 and n2 = n1 − 1 then q := μ1σ1
7:         else if m2 = m1 − 1 and n2 = n1 − 1 then q := μ2σ2
8:         else if m2 = m1 − 1 and n2 = n1 = 0 then q := m1γ
9:         else if m2 = m1 − 1 and n2 = n1 > 0 then q := (m1 − 1)γ
10:        else if m1 <= s and m2 = m1 + S − s and n2 = n1 then q := ν
11:    else if m1 = 0 and k1 = k2 then
12:        if m2 = 0 and n2 = n1 + 1 then q := λ_{k1}φ1
13:        else if m2 = 0 and n2 = n1 − 1 then q := n1τ
14:        else if m2 = m1 + S − s and n2 = n1 then q := ν
15:    return q
```

Fig. 1. Pseudo code of transition matrix

After the stationary distribution is known, the inventory related performance measures could be calculated using the following formulas:

$$S_{av} = \sum_{(m,n,k)\epsilon E} mp(m, n, k) \tag{1}$$

$$\Gamma_{av} = \gamma \sum_{(m,n,k)\epsilon E} mp(m,n,k) \tag{2}$$

$$RR = (\mu_2\sigma_2 I(n>0) + \gamma(s + I(n=0))) \sum_{(s+1,n,k)\epsilon E} p(s+1,n,k) \tag{3}$$

Service related performance measures are calculated as using below formulas:

$$L_{av} = \sum_{(m,n,k)\epsilon E} np(m,n,k) \tag{4}$$

$$RL = \tau \sum_{(0,n,k)\epsilon E} np(0,n,k) + \sum_{(0,n,k)\epsilon E} \lambda_k \phi_2 p(0,n,k) \tag{5}$$

$I(A)$ is the indicator function of the event A. Now let us calculate the stationary distribution in the next section.

3. Simulation algorithm for the calculation of stationary distribution

In order to find the stationary distribution we need to solve the system of linear balance equations. Because we have the infinite transition matrix, there is no universal analytical algorithm to calculate the stationary distribution. In such situation, we may apply simulation approach. There exist different simulation algorithms like Gillespie's Direct method, Gibson Next Reaction method, Explicit Tau-Leap method etc. The more detailed information about simulation of continuous Markov chains is provided in [18].

Let us apply the exact Gillespie's Direct [19] algorithm for the simulation of our model. The implementation of simulation algorithm in Python programming language is illustrated in Fig. 2. The *nextStateSpace* function is used to specify all the possible transitions from the state (m_1, n_1, k_1). The function *Q_elem* on line 7 is calculated according to pseudo-code in Fig. 1. The *reachedStates* variable on line 14 is used to memorize the state path and corresponding sojourn times. The stationary probability of each state is equal to the ratio of its sojourn time to the total simulation time (line 37). We use *random.expovariate* function to draw the random time intervals. The random number r is generated according to the uniform distribution using the function *random.uniform(0,1)* on line 25. After the stationary distribution is found we calculate the performance measures according to formulas (2.1)-(2.5).

The simulation time T is choosen experimentally based on the system parameter values. The provided algorithm is universal and could be used for the simulation of any Markov chain with known transition matrix. In order to apply the given code to any Markov model function *nextStateSpace* must be adjusted according to the transition matrix, while the *simulate* function remains unchanged.

```python
import random
def nextStateSpace(self,m1,n1,k1):
    nextStateSpace = {}
    for m2 in (m1, m1 + 1, m1 - 1, S):
        for n2 in (n1, n1 + 1, n1 - 1):
            for k2 in (k1, k1 + 1, k1 - 1):
                tr = self.Q_elem(m1, n1, k1, m2, n2, k2)
                if tr > 0:
                    nextStateSpace[state] = tr
    return nextStateSpace
def simulate(initialState = 0, simulationTime = 5000):
    currState = initialState
    timePassed = 0
    reachedStates = {} # dict(state: sojournTime)
    nextStatesAll = {} # dict(state: nextStates)
    while timePassed < simulationTime:
        if currState not in reachedStates:
            reachedStates[currState] = 0
        if currState not in nextStatesAll:
            nextStatesAll[currState] = nextStateSpace(currState)
        nextStates = nextStatesAll[currState]
        # transition rate sum
        rateSum = sum(nextStates.values())
        nextInterval = random.expovariate(rateSum)
        r = random.uniform(0,1)
        prevRateSum = 0
        for st in nextStates:
            p_sum = prevRateSum / rateSum
            n_sum = (prevRateSum + nextStates[st]) / rateSum
            if r > p_sum and r < n_sum:
                reachedStates[currState]  += nextInterval
                currState = st
                break
            prevRateSum += nextStates[st]
        timePassed += nextInterval
    totalTime = sum(reachedStates.values())
    return {s: reachedStates[s] / totalTime for s in reachedStates}
```

Fig. 2. The implementation of Gillespie's Direct simulation algorithm in Python

Now that we know how to calculate the stationary distribution and performance measures let us consider the results of numerical experiments in the next section.

4. Numerical experiments

We will evaluate the convergence of simulation algorithm, analyze the dependence of performance measures on the reorder threshold s and solve the optimization problem in this section. The MMPP parameter values are assumed as follows during the experiments:

$$\Lambda = [10,15,20], \Sigma = \begin{pmatrix} -13 & 9 & 4 \\ 15 & -20 & 5 \\ 10 & 8 & -18 \end{pmatrix}$$

Let us evaluate the convergence speed of simulation algorithm provided in Fig. 2. The dependence of stationary distribution convergence from the simulation time is depicted in Fig. 3. We use Euclidean distance between the successive stationary distribution vectors to evaluate the change of stationary distribution. We conclude from Fig. 3 that the system converges to the limiting stationary distribution approximately for the simulation times greater than 2000 units. The system parameters are assumed as following in the numerical experiment:

$$S = 10, s = 3, \mu_1 = 55, \mu_2 = 45, \sigma_1 = 0.4, \phi_1 = 0.8, \gamma = 1, \tau = 1.5, \nu = 1.5.$$

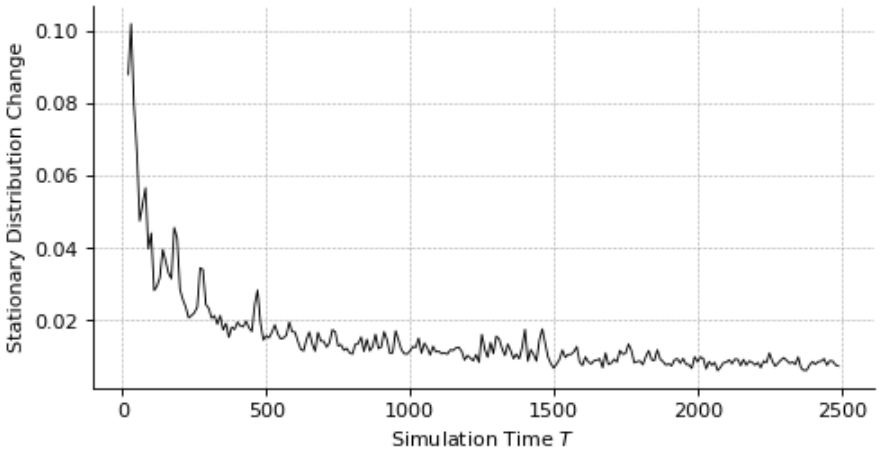

Fig. 3. Dependence of stationary distribution convergence on simulation time

Although convergence time strictly depends on the size of transition matrix we have obtained very fast convergence speeds for different parameter values in our experiments.

Dependence of the inventory related performance measures on the reorder threshold s and order lead-time ν is illustrated in Fig. 4. We used the following system parameter values during analysis of performance measures:

$$S = 15, \mu_1 = 55, \mu_2 = 45, \sigma_1 = 0.4, \phi_1 = 0.8, \gamma = 1, \tau = 1.5.$$

We conclude from Fig. 4 that S_{av} increases linearly proportional to s when order lead-time ν is greater than impatience rate τ and perishing rate γ and decreases otherwise. The reason is that when the inventory is replenished faster than the item perishing the average inventory level increases gradually. Additionally, the intuitive relation $\Gamma_{av} \approx \gamma S_{av}$ holds true. The average reorder rate RR increasse proportional both to s and ν because with the higher values of s and ν the inventory level drops to the threshold faster.

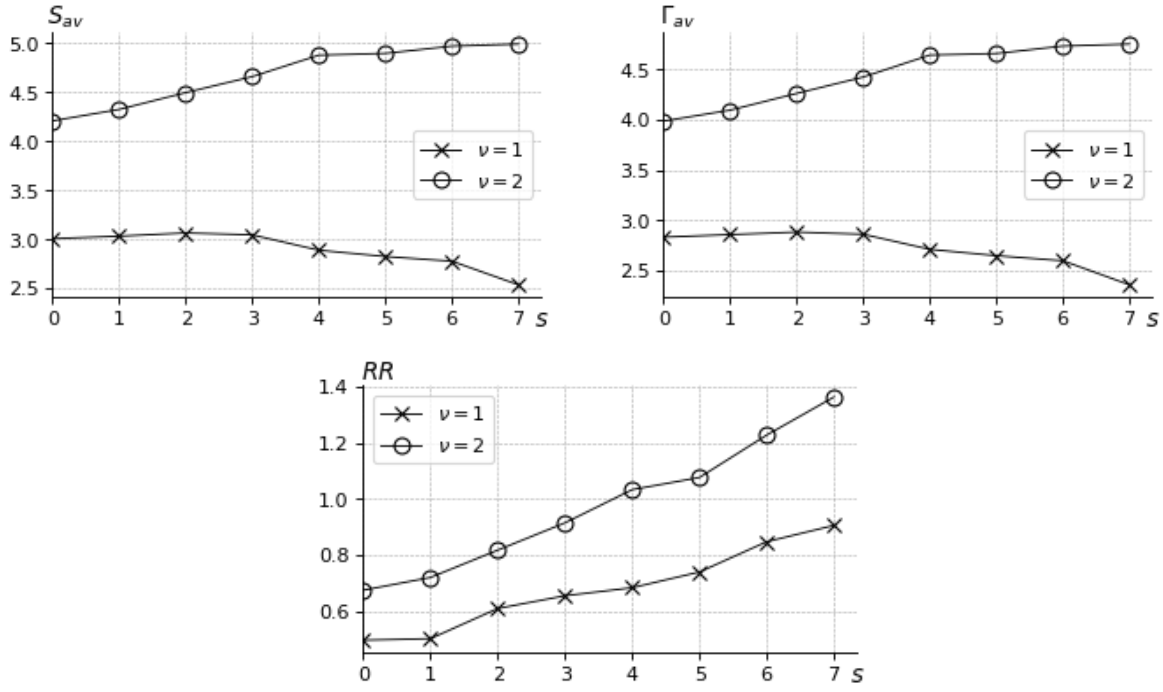

Fig. 4. Dependence of inventory related performance measures (1)-(3) on s.

Dependence graphs of the performance measures related to customer service are provided in Fig. 5. We conclude from the graphs that the average queue length L_{av} and customer loss intensity RL almost do not depend on threshold level s. L_{av} is intuitively lower for the higher value of the order lead time intensity ν. The behavior customer loss rate RL is proportional to L_{av}.

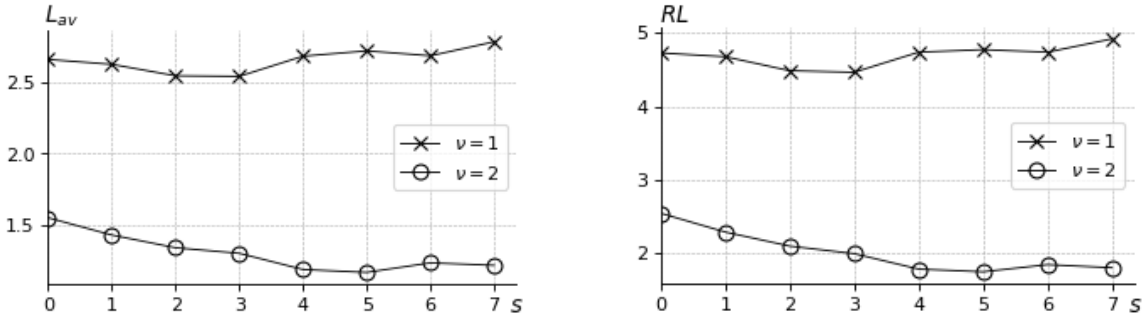

Fig. 5. Dependence of customer service related performance measures (4), (5) on *s*.

5. Optimization problem

Let us consider the following long run total cost function:

$$TC(s) = (K + c_r(S - s))RR + c_s S_{av} + c_p \Gamma_{av} + c_l RL + c_w L_{av}. \quad (6)$$

We assume that all the parameters are fixed except the threshold level s, which is the only controllable parameter. The coefficients in (6) have the following meanings:

- c_s – the inventory maintenance cost per item;
- c_p – per item perishing expence;
- c_l – per customer loss penalty cost;
- c_w – per customer waiting cost in the queue;
- K – fixed replenishment order cost;
- c_r – per item order cost.

The optimization problem is to find the value s^* that minimizes the cost function TC:

$$s^* = \text{argmin}\{TC(s): 0 \le s \le s'\} \quad (7)$$

where $s' = \begin{cases} \left\lceil \frac{S}{2} \right\rceil \ if\ S\ is\ odd \\ \frac{S}{2} - 1 \ if\ S\ is\ even \end{cases}$, $\lceil x \rceil$ – is the integer part of x.

The solution of the problem (7) is provided in Table 1 for different order lead times. Performance measures are calculated using provided simulation algorithm (Fig. 2) and formulas (1)-(5). System parameter values are assumed as follows during the experiment.

$$c_s = 1.5, c_p = 0.2, c_l = 2.0, c_w = 1.0,$$
$$S = 15, \mu_1 = 55, \mu_2 = 45, \sigma_1 = 0.4, \phi_1 = 0.8, \gamma = 1, \tau = 1.5.$$

Table 1
Solution of the optimization problem (7)

ν	(K, c_r)	s^*	TC^*
0.5	(1.5,0.1)	6	23.0311
1	(2.0,0.2)	2	19.2365
1.5	(2.5,0.3)	2	18.792
2	(3.0,0.4)	1	19.6416
2.5	(3.5,0.5)	0	20.4873
3	(4.0,0.6)	0	21.8599

We conclude from Table 1 that the optimal value of threshold s^* decreases for the higher lead-time intensities. We get the minimal cost for the moderate value of $\nu = 1.5$ and $s^* = 2$. Although the solution of optimization problem (7) depends on the given system parameter values, in our case it is recommended to choose middle lead time intensity in order to minimize the long run total cost function TC.

6. Conclusion

Perishable queueing-inventory system with positive service time and different types of customers was considered. Customers arrive into the system according to MMPP flow and join the queue of infinite size when the server is busy. The inventory has finite capacity and is replenished according to (s, S) policy. Customers are assumed to join the queue with some positive probability even when the inventory level is zero while the waiting customers in the queue become impatient and leave the system independently after the exponentially distributed random times. Simulation approach was applied to find the joint stationary distribution of the inventory level, customer count and MMPP state. The high convergence speed of simulation algorithm for the specified system parameter values was demonstrated. The dependence of performance measures on the reorder level and lead time intensity was analyzed and the results of the numerical experiments were illustrated. Finally, the long run total cost optimization problem was solved with respect to threshold level.

References

[1] K. Karthikeyan, R. Sudhesh, Recent review article on queueing inventory systems, Research Journal of Pharmacy and Technology. 9 No.11 (2016) 2056-2066. https://doi.org/10.5958/0974-360X.2016.00421.2

[2] S. Graves, The Application of Queueing Theory to Continuous Perishable Inventory Systems, Management Science. 28 No.4 (1982) 400-406. https://doi.org/10.1287/mnsc.28.4.400

[3] C. Goh, B. Greenberg, H. Matsuo, Two-Stage Perishable Inventory Models, Management Science. 28 No.4 (1993) 633-649. https://doi.org/10.1287/mnsc.39.5.633

[4] C. Goh, B. Greenberg, H. Matsuo, Perishable inventory systems with batch demand and arrivals, Operations Research Letters. 13 No.1 (1993) 1-8. https://doi.org/10.1016/0167-6377(93)90076-S

[5] B. Sivakumar, A perishable inventory system with retrial demands and a finite population, Journal of Computational and Applied Mathematics. 224 No.1 (2009) 29-38. https://doi.org/10.1016/j.cam.2008.03.041

[6] P. Manuel, B. Sivakumar, G. Arivarignan, A perishable inventory system with service facilities and retrial customers, Computers & Industrial Engineering. 54 No.3 (2008) 484-501. https://doi.org/10.1016/j.cie.2007.08.010

[7] Z. Lian, L. Liu, A discrete time model for perishable inventory systems, Annals of Operations Research. 87 No.10 (1999) 103-116. https://doi.org/10.1023/A:1018960314433

[8] A. Melikov, L. Ponomarenko, M. Shahmaliyev, Analysis of Perishable Queuing-Inventory System with Different Types of Requests, Journal of Automation and Information Sciences, Begell House. 49 No.9 (2017) 42-60. https://doi.org/10.1615/JAutomatInfScien.v49.i9.40

[9] S. Nahmias, D. Perry, W. Stadje, Perishable inventory systems with variable input and demand rates, Mathematical Methods of Operations Research. 60 No.1 (2004) 155-162. https://doi.org/10.1007/s001860300335

[10] A. Melikov, L. Ponomarenko, M. Shahmaliyev, Models of Perishable Queueing-Inventory System with Repeated Customers, Journal of Automation and Information Sciences, Begell House. 48 No.6 (2016) 22-38. https://doi.org/10.1615/JAutomatInfScien.v48.i6.30

[11] A. Melikov, M. Shahmaliyev, Markov Models of Inventory Management Systems with a Positive Service Time, Journal of Computer and Systems Sciences International. 57 No.5 (2018) 766-783. https://doi.org/10.1134/S106423071805009X

[12] A. Krishnamoorthy, R. Manikandan, B. Lakshmy, A revisit to queueing-inventory system with positive service time, Annals of Operations Research. 233 No.1 (2015) 221-236. https://doi.org/10.1007/s10479-013-1437-x

[13] A. Krishnamoorthy, R. Manikandan, D. Shajin, Analysis of a Multiserver Queueing-Inventory System, Advances in Operations Research 16 (2015). https://doi.org/10.1155/2015/747328

[14] A. Krishnamoorthy, D. Shajin, MAP/PH/1 Retrial Queueing-Inventory System with Orbital Search and Reneging of Customers. Customers, Analytical and Computational Methods in Probability Theory, Springer International Publishing. (2017) 158-171. https://doi.org/10.1007/978-3-319-71504-9_15

[15] K R K. Rejitha, K.A, Jose Queuing-Inventory System with MAP, Retrials and Different Replenishment Rates, International Journal of Pure and Applied Mathematics. 117 No.11 (2017) 289-297.
[16] A. Dudin, A. Kazimirsky, V. Klimenok, L. Breuer, U. Krieger, The Queueing Model MAP|PH|1|N with Feedback Operating in a Markovian Random Environment, Austrian Journal of Statistics. 34 No.2 (2016) 101-110. https://doi.org/10.17713/ajs.v34i2.403
[17] W. Fischer, K. Meier-Hellstern, The Markov-modulated Poisson process (MMPP) cookbook, Performance Evaluation. 18 No.2 (1993) 149-171. https://doi.org/10.1016/0166-5316(93)90035-S
[18] H. Banks, A. Broido, B. Canter, K. Gayvert, S. Hu, M. Joyner, K. Link, Simulation Algorithms for Continuous Time Markov Chain Models, Studies in Applied Electromagnetics and Mechanics, Simulation and Modeling Related to Computational Science and Robotics Technology. 37 (2012) 3-18. https://doi.org/10.3233/978-1-61499-092-5-3
[19] D. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, Journal of Computational Physics. 22 No.4 (1976) 403-434. https://doi.org/10.1016/0021-9991(76)90041-3