STATISTICS IN TRANSITION new series, September 2024
Vol. 25, No. 3, pp. 187–196, https://doi.org/10.59170/stattrans-2024-035
Received – 22.10.2023; accepted – 15.05.2024

A finite state Markovian queue to let in impatient customers only during K-vacations

R. Sivasamy[1]

Abstract

We investigate a matrix analysis study for a single-server Markovian queue with finite capacity, i.e. an M/M/1/N queue, where the single server can go for a maximum, i.e. a *K* number of consecutive vacation periods. During these vacation periods of the server, every customer becomes impatient and leaves the queues. If the server detects that the system is idle during service startup, the server rests. If the vacation server finds a customer after the vacation ends, the server immediately returns to serve the customer. Otherwise, the server takes consecutive vacations until the server takes a maximum number of vacation periods, e.g. *K*, after which the server is idle and waits to serve the next arrival. During vacation, customers often lose patience and opt for scheduled deadlines independently. If the customer's service is not terminated before the customer's timer expires, the customer is removed from the queue and will not return. Matrix analysis provides a computational form for a balanced queue length distribution and several other performance metrics. We design a 'no-loss; no-profit cost model' to determine the appropriate value for the maximum value of *K* consecutive vacation periods and provide a solution with a numerical illustration.

Key words: impatient customers, vacation period, queue length, stationary distribution.

1. Introduction

The main objective of this research is to develop a matrix method to obtain the queue length distribution of a single-server service system M/M/1/N, where N represents the maximum capacity of customers waiting. This system allows the server to conditionally take up to K vacations during which the system remains inactive. However, if a server returning from vacation finds a queue of customers, the server starts service according to first-come first-served (FCFS) standards. In addition, we revisit the algorithmic approaches given by (Neuts, 1981) and (Latouche & Ramaswami, 1999) to find a solution for the class of finite two-dimensional continuous-time queue-length processes $Z(t) = \{L(t), J(t); t \geq 0\}$ defined in the space **E**:

$$\mathbf{E} = \mathbf{N}_0 \times \mathbf{K}_0,\ \mathbf{N}_0 = \{0, 1, \ldots, N, (<\infty),\} \text{ and } \mathbf{K}_0 = \{0, 1, \ldots, K\} \qquad (1)$$

[1] Faculty of Social Sciences, Statistics Department, Gaborone, University of Botswana, P.bag 00705, Botswana. E-mail: sivasamyr@ub.ac.bw. ORCID: https://orcid.org/0000-0002-3158-928X.

The basic assumption is that the process **L**(t)= n (≥0) represents the observed queue length at time 't' and **J**(t) = j (j = 0, 1, . . ., (K-1)) indicates the $(j+1)^{th}$ vacation at time 't', where **J**(t) = K indicates the state of the server, whether idle or busy.

1.1. Finite state and finite capacity queue with impatient customers

Let us talk about a single server Markovian (a.k.a. Poisson) queue with a maximum capacity of N, i.e. M/M/1/N. The system services customers and the service time follows an exponential distribution with the rate μ. The customer arrival process follows the Poisson process with an average rate λ. If the server finds the system empty at a departure epoch, the server goes on vacation. If the server finds the customer at the end of the vacation, the server immediately returns to serve the customer. Otherwise, the server will make consecutive vacations until the server takes the maximum number of vacations, say K, then the server will be idle and wait to serve the next arrival.

Each vacation period is assumed to be distributed exponentially with the vacation rate ν. Suppose that during vacation periods of the server, each customer becomes impatient and activates an 'Impatient timer length' T which is exponentially distributed with parameter ψ. If the service period of the impatient client is not terminated before the client's timer expires, the client leaves the queue and does not return. Let us denote the above model by M/M/1/N/(K-vacations) queue with impatient customers.

Section 2 describes the process of limiting the length of the queue process, i.e. Z = {(L, J)} = $\lim_{t\to\infty}\{(L(t), J(t))\}$ of M/M/1/N/(K-vacations) service system with impatient customers. After formulating it as a positive recurrent process, using numerical algorithms, we obtain probability vectors for each level of process Z along with the stationary queue length distribution. Section 3 discusses the calculation of various performance metrics and probability distributions of server's vacation, busy and idle states. In addition, a "no loss; non-profit" cost function is designed to fix the maximum "K" of consecutive vacations, and a solution is also provided using numerical illustration. Section 4 provides a formal concluding report.

2. Performance of M/M/1/N/(K-vacations) queue with impatient customers in the long run

2.1. Citations

As the best members of this class, we select the M/M/1 queue, whose stationary measures were studied by (Zhang, Yue, & Yue, 2005) and impatient behavior with multiple vacations analyzed by (Ammar, 2015), (Sivasamy, 2020) and (Sudhesh & Azhagappan, 2019). It is assured from the contributions of (Kharoufeh, 2011) and (Sivasamy, Thillaigovindan, Paulraj, & Parnjothi, 2019) that the quasi birth-death (QBD) process framework could be used, in a way that is suitable for modelling all types of M/M/1 queues. (Kharoufeh, 2011) discussed both discrete-time and continuous-time versions of the level-dependent quasi-birth-and-death (LDQBD) process that exhibits a block tri-diagonal structure.

Using standard probability arguments, we can state that the bivariate process Z = {(L, J)} forms an aperiodic, regular, and irreducible LDQBD over the state space E. We divide a two-dimensional space into a union. (N+1) levels, say L_i for i = 0, 1, …, N:

$$\boldsymbol{L_i} = ((\boldsymbol{i},\mathbf{0}),(\boldsymbol{i},\mathbf{1}),\ldots(\boldsymbol{i},\boldsymbol{K}));\ \boldsymbol{i} \in \mathbf{N_0},\ \ \mathbf{E} = \bigcup_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{N}} \boldsymbol{L_i} \tag{2}$$

where $L_i \in (2)$ is called the i^{th} level vector of size or order (K+1).

Using the properties of the QBD under FCFS rule, we organize the elements of the transition generator matrix $G = (g(L_i, L_j))$ described in (3):

$$\mathbf{G} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{A_0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{A_2^{(1)}} & \boldsymbol{A_1^{(1)}} & \boldsymbol{A_0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A_2^{(2)}} & \boldsymbol{A_1^{(2)}} & \boldsymbol{A_0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{A_2^{(3)}} & \boldsymbol{A_1^{(3)}} & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \boldsymbol{A_1^{(N-1)}} & \boldsymbol{A_0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A_2^{(N)}} & (\boldsymbol{A_2^{(1)}} + \boldsymbol{A_0}) \end{pmatrix} \tag{3}$$

Each sub-matrix of G is a square matrix of order (K+ 1) indexed by j = 0, 1, . . . , K. The precise structures of B, A_0, A_1, and A_2 are described in (4) to (7), respectively:

$$\boldsymbol{B} = \begin{pmatrix} -(\boldsymbol{\lambda}+\boldsymbol{\gamma}) & \boldsymbol{\gamma} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -(\boldsymbol{\lambda}+\boldsymbol{\gamma}) & \boldsymbol{\gamma} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \cdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \ddots & -(\boldsymbol{\lambda}+\boldsymbol{\gamma}) & \boldsymbol{\gamma} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & -\boldsymbol{\lambda} \end{pmatrix} \tag{4}$$

$$\boldsymbol{A_0} = \begin{pmatrix} \boldsymbol{\lambda} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\lambda} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \vdots & \mathbf{0} \\ \vdots & \cdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \boldsymbol{\lambda} \end{pmatrix} \tag{5}$$

For n = 1,2,…, N,

$$\boldsymbol{A_1^{(n)}} = \begin{pmatrix} -(\boldsymbol{\gamma}+\boldsymbol{\lambda}+\boldsymbol{n\psi}) & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \boldsymbol{\gamma} \\ \mathbf{0} & -(\boldsymbol{\gamma}+\boldsymbol{\lambda}+\boldsymbol{n\psi}) & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \cdots & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \ddots & -(\boldsymbol{\gamma}+\boldsymbol{\lambda}+\boldsymbol{n\psi}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & -(\boldsymbol{\lambda}+\boldsymbol{\mu}) \end{pmatrix} \tag{6}$$

$$\boldsymbol{A_2^{(1)}} = \begin{pmatrix} \boldsymbol{\psi} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\psi} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \cdots & \ddots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \ddots & \boldsymbol{\psi} & \mathbf{0} \\ \mu & \mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \boldsymbol{A_2^{(n)}} = \begin{pmatrix} \boldsymbol{\psi} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\psi} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \cdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{\psi} \end{pmatrix}, \text{ n=2, …, N} \tag{7}$$

Distribution: Let $\Pi = (\pi_0, \pi_1, \pi_2, \ldots \pi_N)$ denote the steady-state distribution of Z process and π_n the row vector $\pi_n = (\pi(n, 0), \ldots, \pi(n, K))$ associated to level L_n, $n \in N_0$.

2.2. Steady-state characteristics of the Z process

The level jumps of the Z process from sate (n, j) ∈ E are restricted only to its adjacent neighbors (n−1, j) and (n+1, j) but not to states of the form (n ± i, j) where i ≥ 2.

For each level, L_n, N ≥n≥0 in G, the diagonal elements of the matrix B and $\mathbf{A_1^{(n)}}$ are completely negative and off diagonal elements are non-negative. Matrices $\mathbf{A_2^{(\boldsymbol{n})}}$ and A_0 are not negative. In each row of G, the sum of the elements is zero (scalar). The structure of the generator matrix G reveals that the process possesses a limiting distribution of the system of equations Π G = 0, Π e =1 (scalar) where 0 denotes the zero vector. We now discuss a simple linear level reduction method in two phases for a positive recurrent case that leads to computation of the limiting distribution:

Phase 1: Iteratively reduce the state space from the level 'N' by removing one level at each step until we reach the level '0' and check if the generator of the level '**0**' corresponds to a positive recurrent Markov process. Compute the U(n) matrices and the rate matrices R_n in terms of the sub matrices of the generator G:

$$\mathbf{U}_{(\boldsymbol{N})} = \mathbf{A_1^{(N)}} + \mathbf{A}_{\mathbf{0}} \tag{8}$$

$$\mathbf{U}_{(\mathbf{n})} = \mathbf{A_1^{(n)}} - \mathbf{A_0}[\,\mathbf{U_{(n+1)}^{-1}}]\,[\,\mathbf{A_2^{(n+1)}}] \text{ for n = (N − 1), (N − 2), . . ., 1} \tag{9}$$

$$\mathbf{U}_{(0)} = \mathbf{B} - \mathbf{A_0}[\,\mathbf{U}_{(1)}^{-1}]\,[\,\mathbf{A}_2^{(1)}] \tag{10}$$

Phase 2: Construct the rate matrix R_n of order (Q+1), for n= 1, 2, …N:

$$\mathbf{R_n} = -\mathbf{A_0}[\,\mathbf{U}_{(n)}^{-1}] \tag{11}$$

Lemma 1: The system of matrix equations which govern all transitions of the of the Z process in terms of its steady-state probability vectors $\boldsymbol{\pi_n}$ and the known sub-matrices of its generator matrix G is derived by solving **Π** G = 0 and reported in (12):

$$\boldsymbol{\pi}_0\,\mathbf{B} + \boldsymbol{\pi}_1\,\mathbf{A}_2^{(1)} = \mathbf{0}$$

$$\boldsymbol{\pi}_{n-1}\,\mathbf{A_0} + \boldsymbol{\pi}_n\,\mathbf{A}_1^{(n)} + \boldsymbol{\pi}_{n+1}\,\mathbf{A}_2^{(n+1)} = \mathbf{0};\ \text{for n=1, 2, …, (N-1)}$$

$$\boldsymbol{\pi}_{N-1}\,\mathbf{A_0} + \boldsymbol{\pi}_N\left[\mathbf{A}_1^{(N)} + \mathbf{A_0}\right] = \mathbf{0} \tag{12}$$

Theorem 1: The unique stationary joint distribution vector **Π** of the queue length plus the inventory level of the Z = (L, J) process is given by $\boldsymbol{\pi_n} = \boldsymbol{\pi_{n-1}}\,\mathbf{R_n}$ for n = 1, 2, …, N and **Π**e = 1.

Proof: The proof of this theorem is organized as an algorithm consisting of four steps:

Step 1: Re-organizing the last equation $\boldsymbol{\pi}_{N-1}\,\mathbf{A_0} + \boldsymbol{\pi}_N\left[\mathbf{A}_1^{(N)} + \mathbf{A_0}\right]$ = of (12), and using the definition $\mathbf{U}^{(N)} = \mathbf{A}_1^{(N)} + \mathbf{A_0}$ of (8), we conclude that $\boldsymbol{\pi}_N = \boldsymbol{\pi}_{N-1}\,\mathbf{R}_N$.

Step 2: Putting n = N−1 in (12), we have

$$\boldsymbol{\pi}_{N-2}\,\mathbf{A_0} + \boldsymbol{\pi}_{N-1}\,\mathbf{A}_1^{(N-1)} + \boldsymbol{\pi}_N\,\mathbf{A}_2^{(N)} = \mathbf{0} \tag{13}$$

Re-organizing the equation (13) with the substitution of $\boldsymbol{\pi}_N = \boldsymbol{\pi}_{N-1}\,\mathbf{R}_N$, we obtain that

$$\boldsymbol{\pi}_{N-2}\,\mathbf{A_0} + \boldsymbol{\pi}_{N-1}\,\mathbf{A}_1^{(N-1)} + \boldsymbol{\pi}_{N-1}\,\mathbf{R}_N\,\mathbf{A}_2^{(N)} = \mathbf{0}$$

$$\Rightarrow \boldsymbol{\pi}_{N-1}\,[\mathbf{A}_1^{(N-1)} + \{-\mathbf{A_0}[\mathbf{U}^{(N)}]^{-1}\}\,\mathbf{A}_2^{(N)}] = \boldsymbol{\pi}_{N-1}\,[\mathbf{U}^{(N-1)}] = -\boldsymbol{\pi}_{N-2}\,\mathbf{A_0}$$

$$\Rightarrow \boldsymbol{\pi}_{N-1} = \boldsymbol{\pi}_{N-2}\,\mathbf{R}_{N-1}\ \textbf{on using the fact}\ \ \mathbf{R}_{N-1} = -A_0\,[\mathbf{U}^{(N-1)}]^{-1} \tag{14}$$

Step 3: Continuing the similar iterative process for n = (N-2), (N-1), … 1 and using the results of step preceding step, we can establish that $\boldsymbol{\pi_n} = \boldsymbol{\pi_{n-1}}\,\mathbf{R_n} = \boldsymbol{\pi}_0 \sum_{k=1}^{n} \mathbf{R_k}$ for n = 1, 2, …, N.

Step 4: To find the vector $\boldsymbol{\pi_0}$, the normalizing condition is $\mathbf{\Pi}$e = 1, we use the following steps.

i) Solve $\boldsymbol{\pi_0}\, \mathbf{U}^{(0)} = \mathbf{0},\ \boldsymbol{\pi_0}\mathbf{e} = 1$.

ii) Compute $\boldsymbol{\pi_n} = \boldsymbol{\pi_{n-1}}\, \mathbf{R_n}$ for n = 1, 2, ..., N.

iii) Calculate $\mathbf{\Pi}$e and renormalize $\mathbf{\Pi}$ using $\mathbf{\Pi} = (\mathbf{\Pi} / \mathbf{\Pi}e)$.

Thus, now the steady-state probability vector $\mathbf{\Pi}$ of the Z process is completely determined. We can now discuss the steady-state probabilities of various events, such as the conditional probability that a server is on vacation, busy, or idle, etc., and measures of system performance. In addition, we can investigate the optimal number of vacations to minimize the average operating cost of the system.

3. Performance Measures: M/M/1/N/(K-vacations) queue with impatient customers

Let us compute the conditional mean system size $\overline{\mathbf{Q}}_{\mathbf{j}}$ given J = j of the (j+1)th vacation for j = 0,1, . . . ,(K-1), K:

$$\overline{\mathbf{Q}}_{\mathbf{j}} = \sum_{\mathbf{n=1}}^{\mathbf{N}} \mathbf{n}\, \boldsymbol{\pi}(\mathbf{n},\mathbf{j}),\ j = 0, 1, 2, \ldots, K \tag{15}$$

The conditional probability P_V that the server is on vacation:

$$\mathbf{P_V} = \sum_{\mathbf{n=0}}^{\mathbf{N}} \sum_{\mathbf{j=0}}^{\mathbf{K-1}} \boldsymbol{\pi}(\mathbf{n},\mathbf{j}) \tag{16}$$

The conditional probability P_B that the server is busy:

$$\mathbf{P_B} = \sum_{\mathbf{n=1}}^{\mathbf{N}} \boldsymbol{\pi}(\mathbf{n},\mathbf{K}) \tag{17}$$

The conditional probability P_I that the server is idle:

$$\mathbf{P_I} = \mathbf{1} - \mathbf{P_B} - \mathbf{P_V} = \sum_{\mathbf{n=1}}^{\mathbf{N}} \boldsymbol{\pi}(\mathbf{0},\mathbf{K}) \tag{18}$$

The expectation is that $P_V + P_B + P_I = 1$. The conditional probability P_{lost} of lost customers is given by:

$$\mathbf{P_{lost}} = \sum_{\boldsymbol{j=0}}^{\boldsymbol{K}} \boldsymbol{\pi}(\boldsymbol{N},\boldsymbol{j}) \tag{19}$$

The effective arrival rate λeff is given by:

$$\boldsymbol{\lambda_{eff}} = \frac{\boldsymbol{\lambda}}{\mathbf{1 - P_{lost}}} \tag{20}$$

The unconditional mean system size $\bar{\mathbf{Q}}$ is given by

$$\bar{\mathbf{Q}} = \sum_{j=0}^{K} \bar{\mathbf{Q}}_j = \sum_{j=0}^{K} \sum_{n=1}^{N} \mathbf{n}\, \boldsymbol{\pi}(\mathbf{n}, \mathbf{j}) \tag{21}$$

The mean waiting time $\bar{\mathbf{W}}$ can be calculated by Little's law. Thus,

$$\bar{\mathbf{W}} = \frac{\bar{\mathbf{Q}}}{\boldsymbol{\lambda_{eff}}} \tag{22}$$

3.1. Numerical illustrations

We now discuss the numerical calculations of some of the measurements discussed so far in the previous discussions. The crux of the calculation lies in the calculation of the stationary probability vectors $\{\pi_n;\ n = 0, 1, \ldots, N\}$ because there is no explicit expression for each probability function π_n. Calculations are based on the algorithm given from the matrix equations reported in Theorem 1.

Any external observer can find the server in one of the three mutually exclusive states i.e. "V: Vacation, B: Busy and I: Idle". Let the probabilities of these three states V, B and I be P_V, P_B, and P_I respectively. Then, P_V, P_B, and P_I ($P_V + P_B + P_I = 1$) values can be easily calculated and checked from the stationary probability vector Π satisfying the conditions $\Pi G = 0$ and $\Pi\, e = 1$.

Using server state distribution {P_V, P_B, and P_I ($P_V + P_B + P_I = 1$)}, we now notice an issue related to fee collection and loss of rental owner. Assume the server is a leased machine. Let us say the server rental is USD (245.25, 80.8, 125.5) per unit time. Let the vacation state of the server cost be 425.25 USD per unit time to manage loss due to impatient customers and non-availability of service which each customer spends on vacation. Suppose that the management collects profit USD 80.8 per unit time from customers when the server is busy and USD 125.45 when server is idle. The objective of the management is to fix the maximum number 'K' of vacations for the server to take consecutively, which ensues 'no loss and no gain' status. For this experiment, random values of input quantities are selected as $\lambda = 2.2$, $\psi = 1.1$, $\nu = 2.25$ and $\mu = 2.5$. For j = V, B, and I, let us calculate the vector of probabilities $V(j) = (P_j^{(K=1)}, P_j^{(K=2)}, \ldots, P_j^{(K=9)}, P_j^{(K=10)})$, respectively for $K = 1, 2, \ldots, 10$.

Let us compute the following costs:

- $T1(K) = 425.25\ V_{(1)}$, called vacation loss cost;
- $T2(K) = 80.6\ V_{(2)}$, called the busy server profit;
- $T3(K) = 125.45\ V_{(3)}$, called the idle server profit.

The objective is now finding a K* value such that the total cost T4(K*) = 0:

$$T4(K) = (T2(K) + T3(K)) - T1(K) > 0 \text{ for all } K < \boldsymbol{K}^*$$

$$T4(K) = (T2(K) + T3(K)) - T1(K) < 0 \text{ for all } K > \boldsymbol{K}^* \tag{23}$$

The general trend is that the function T1(K) increases as the values of K increase. On the other hand, both functions T2(K) and T3(K) decrease as the values of K increase. We call the total costs T4(K) = T2(K) T3(K) – T1(K). If any of the components of T4(K) >0, it gives a profit or a loss for that value of K.

To demonstrate these facts and to get an optimum on K to meet the no-loss no-profit condition, numerical values are computed for the given data set and the corresponding curve is plotted in Figure 1.

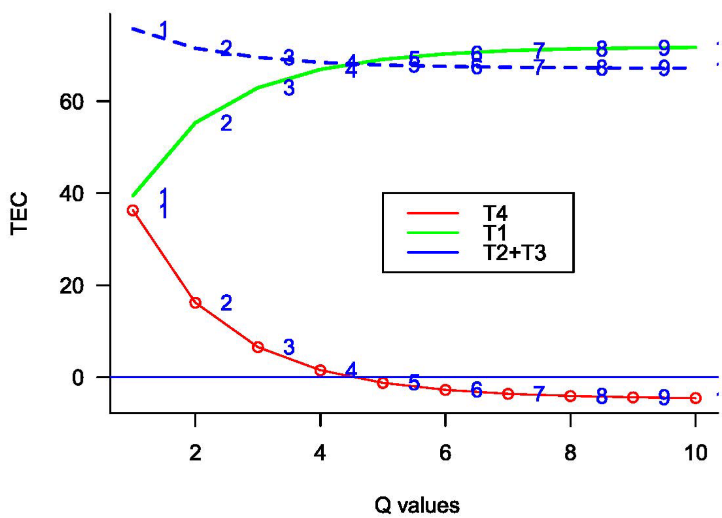

Figure 1: Relationships among Vacation, Busy and Idle states of the server's cost for "no loss and No gain level as K = 4

A simple look at Figure 1 tells us that an optimum of vacations is attained if K* = 4. This means that if management allows K = 1, 2 and 3 consecutive vacations, management earns a positive profit. There is no profit or loss from the maximum number of vacation periods K* = 4. But if K ≥ 5, the lead is only a negative profit or loss. This type of test can also be designed to monitor a typical K-value, which provides "no loss, no gain" efficiency while adding a larger number of cost effects.

4. Conclusions

A single-server M/M/1/N/(K-vacations) queue with impatient customers, which conditionally accepts impatient customers in every vacation period, is well studied by matrix analysis. The special thing is that if the customer's service does not end before the random deadline chosen by the customer, he can leave the queue. Additionally, the server can take multiple vacations in a row, but no more than K vacations.

The steady-state results of this study are supported by numerical algorithms to obtain the necessary probability vectors and the optimal number of K* for consecutive vacations. We obtain the steady-state queue length distribution and the scalar value of the vector expression for multiple events and measurements. A "no Loss; no Profit" cost model is proposed to test the appropriate value for the maximum K-value of consecutive vacations and provide a solution using a numerical representation.

The proposed methodology is implemented by showing an exponential distribution of arrival times, service times, impatient customer deadlines and vacation periods. Our opinion is thatthe proposed theoretical and computational aspects of single-server Poisson queue M/M/1/N are useful not only for academics but also for all practitioners dealing with queues that have many vacations.

Future scope can be expanded by replacing exponential distribution with general distribution in single server or multi-server queues.

Acknowledgement

The author thanks his colleagues for their constructive suggestions and thanks the institutions of the University of Botswana for encouraging this research without financial support.

References

Ammar, S., (2015). Transient analysis of an M/M/1 queue with impatient behavior and multiple vacations. *Applied Mathematics and Computation, 260*, doi: 10.1016/j.amc.2015.03.066, pp. 97–105.

Kharoufeh, J. P., (2011). *Level-Dependent Quasi-Birth-and-Death Processes,* doi: 10.1002/9780470400531.eorms0460: Wiley Encyclopedia of Operations Research and Management Science.

Latouche, G. A., Ramaswami, V., (1999). *Introduction to Matrix Analytic Methods in Stochastic Modeling.* Pennsylvania: Society for Industrial and Applied Mathematics, Philadelphia.

Neuts, M., (1981). *Matrix geometric solutions in stochastic models—an algorithmic approach,* Baltimore: John Hopkins University Press.

Sivasamy, R., (2020). Two Server Poisson Queues with a Slow Service Provider for Impatient Customers. *International Journal of Mathematics and Statistics, 2,* ISSN 0974–7117 (Print); ISSN 0973-8347 (Online), https://www.ej-math.org, pp. 57–73.

Sivasamy, R., Thillaigovindan, N., Paulraj, G. A., and Parnjothi, N., (2019). Quasi birth-death processes of two-server queues with stalling. *OPSEARCH, 56,* https://doi.org/10.1007/s12597-019-00376-1, pp. 739–756.

Sudhesh, R., Azhagappan, A., (2019). Transient solution of an M/M/∞ queue with system's additional tasks and impatient customers. *Int. J. of Mathematics in Operational Research, 16, No. 1,* https://dblp.org/db/journals/ijmor/index, pp. 82–97.

Zhang, Y., Yue, D. A., and Yue, W., (2005). Analysis of an M/M/1/N queue with Balking, Reneging and Server Vacations. *International Symposium on OR and Its Applications, 1,* https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje)), pp. 37–67.