Applied Mathematical Modelling 35 (2011) 2196–2208

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

# Algorithmic analysis of the multi-server system with a modified Bernoulli vacation schedule

Jau-Chuan Ke [a,*], Chia-Huang Wu [b], Wen Lea Pearn [b]

[a] Department of Applied Statistics, National Taichung Institute of Technology, Taichung, Taiwan, ROC
[b] Department of Industrial Engineering and Management, National Chiao Tung University, Taiwan, ROC

ARTICLE INFO

*Article history:*
Received 19 May 2009
Received in revised form 18 October 2010
Accepted 15 November 2010
Available online 23 November 2010

*Keywords:*
Bernoulli vacation schedule
Matrix analytic approach
Quasi-Newton method
Single vacation policy

ABSTRACT

This paper considers an infinite-capacity M/M/c queueing system with modified Bernoulli vacation under a single vacation policy. At each service completion of a server, the server may go for a vacation or may continue to serve the next customer, if any in the queue. The system is analyzed as a quasi-birth-and-death (QBD) process and the necessary and sufficient condition of system equilibrium is obtained. The explicit closed-form of the rate matrix is derived and the useful formula for computing stationary probabilities is developed by using matrix analytic approach. System performance measures are explicitly developed in terms of computable forms. A cost model is derived to determine the optimal values of the number of servers, service rate and vacation rate simultaneously at the minimum total expected cost per unit time. Illustrative numerical examples demonstrate the optimization approach as well as the effect of various parameters on system performance measures.



## 1. Introduction

Most studies on queueing system, servers always serve the waiting customers in the queue until all customers is served exhaustively. In many practical systems, however, it may occur that the service process requires to be temporarily stopped for overhauling at the end of a service. This overhauling can be utilized as a vacation in the presented model. In this paper, the quasi-birth–death (QBD) process and matrix analytic method are used to analyze an infinite capacity multi-server M/M/c queue with a modified Bernoulli vacation schedule. The computational algorithm of stationary probability vectors and optimization of parameters are developed.

Queueing models with server vacations are effective tools for performance analysis of manufacturing systems, local area networks, and data communication systems. Past works on vacation queueing models are either single server or multiple server systems. Excellent surveys on the single server vacation models have been reported by Doshi [1] and Takagi [2]. The variations and extensions of these vacation models were developed by several researchers such as Lee et al. [3,4], Krishna Reddy et al. [5], Choudhury [6,7], Ke [8,9], Ke and Chu [10], Ke et al. [11,12] and many others. A numbers of papers [13–16] have recently appeared in queueing literature in which the server provides to each heterogeneous service with Bernoulli schedule vacation (BSV). The so-called BSV means that when the service of a unit is completed, the server may leave for a vacation of random length with probability $p$ or may continue to serve the next unit, if any with probability $1-p$ (see Choudhury et al. [17]). For the multiple server vacation models, there are only a limited number of studies due

* Corresponding author. Address: Department of Applied Statistics, National Taichung Institute of Technology, No. 129, Sec. 3, Sanmin Rd., Taichung 404, Taiwan, ROC. Tel.: +88 6422196332; fax: +88 6422196331.
*E-mail address:* jauchuan@ntit.edu.tw (J.-C. Ke).

0307-904X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2010.11.019

to the complexity of the systems. The M/M/c queue with exponential vacations was first studied by Levy and Yechiali [18]. Chao and Zhao [19] investigated a GI/M/c vacation system and provided an algorithm to compute the performance measures. Tian et al. [20,21] gave a detailed study of the M/M/c vacation systems in which all servers take multiple vacation policy when the system is empty. Zhang and Xu [22], Zhang and Tian [23,24] and Ke et al. [25] analyzed the M/M/c vacation systems with a "partial server multiple vacation policy" in which some servers (only the idle ones) take single or multiple vacations.

Multi-server vacation models are more flexible and applicable in practice than single server models. Existing research works, including those mentioned above, have not addressed the analytical study and optimization issue in *the infinite buffer multiple-server systems in which the server may take a vacation upon his each service completion.* Besides the lack of research work on this problem, our study is also motivated by a practical production as follows. Consider a production process with a number of machines (or $c$ machines). It may so happen that the production process either needs to be temporarily stopped for overhauling and maintenance of the system after each service completion or continue the service for the next unit/customer in the queue. This overhauling can be utilized as a vacation in the presented model. Another application of such model based on Choudhury et al. [17] may well be found in some transportation systems in which a ferry driver or locomotive driver may have a vacation after every round of trip.

In this paper, we consider an infinite capacity M/M/c queueing system with modified Bernoulli vacation under a single vacation policy. Customers arrive according to a Poisson process with parameter $\lambda$ and their service are provided by $c$ servers, in which the service times are assumed to be exponentially distributed with mean $1/\mu$. It is assumed that customers arrive at the server form a single waiting line and are served in the order of their arrivals; that is, the first-come, first-served discipline. Each server can serve only one customer at a time, and that the service is independent of the arrival of the customers. At each service completion instant of a server, the server inspects the system state and decides whether leave for a vacation or not. If the number of customers in the queue is less than the number of busy/working servers, the server may take a vacation of random length with probability $p$ or continue to serve the next customer, if any with probability $q$ (=$1-p$). The vacation times are exponentially distribution with mean $1/\theta$. If the number of customers in the queue is more than the number of busy servers, the server always keeps working/serving for the next customers waiting in the queue (i.e., $p=0$). From Fig. 1, we easily see the state transition with flow in/out rate. Conveniently, we represent this multi-server system with modified Bernoulli vacation as M/M/c/MBSV queueing system.

The paper is organized as follows; In Section 2, the quasi-birth–death (QBD) model of the M/M/c/MBSV queueing system is set up. The explicit closed-form of the rate matrix is derived and the stable condition is obtained by using the matrix-geometric property. Section 3 we derive an efficient algorithm to compute stationary probabilities by matrix-geometric method. Some system performance measures are derived in Section 4, including some explicit results of special case. In Section 5, a cost model is developed to determine the optimal values of number of servers, service rate and repair rate, simultaneously, in

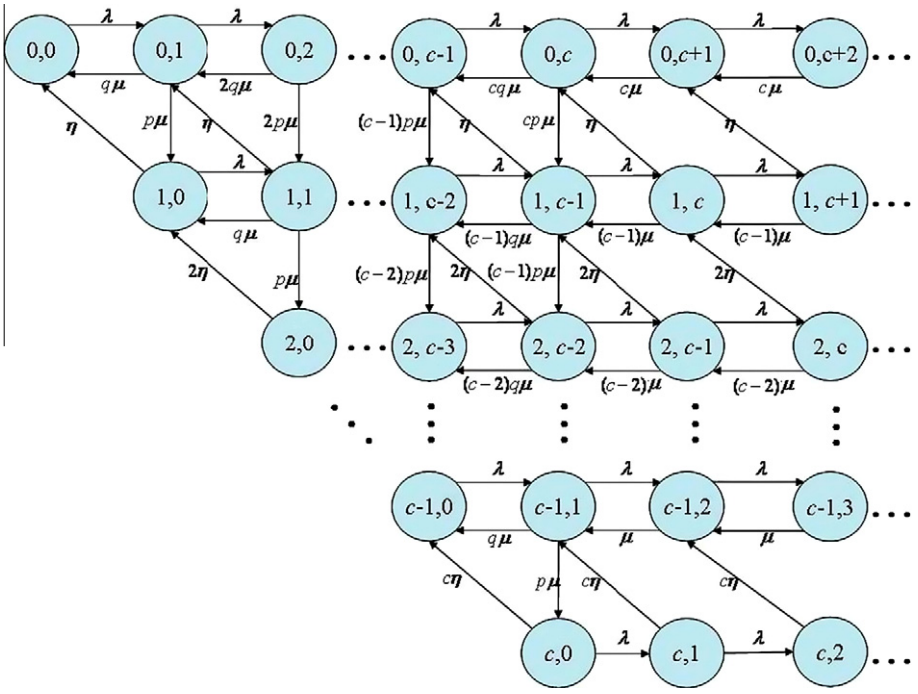


**Fig. 1.** State-transition-rate diagram for an infinite capacity M/M/c queueing system with modified Bernoulli vacation, where $(i,j)$ of circle denotes the state that there are $i$ vacation servers and $j$ customers in the system.

order to minimize the total expected cost per unit time. We use Quasi-Newton method and direct search method to implement the optimization tasks. Some numerical examples are provided to illustrate the optimization methods. Section 6 concludes.

## 2. Mathematical model

We consider the M/M/c/MBSV queueing system. The state of this queueing system can be described by the pair $(i,n)$, $i = 0,1,2,\ldots,c$, $n = 0,1,2,\ldots$, where $i$ denotes the number of vacation servers, and $n$ denotes the number of customers in the system. According to system assumptions, the server keeps busy and serves the next customer waiting in the queue at a service completion instant if the number of customers in the system is greater than the number of total working (no vacation) servers, i.e. $(n \geqslant c - i + 1)$. As $(n \leqslant c - i)$, one server may goes on vacation with probability $p(p > 0)$ or resumes service with probability $q = 1 - p$ at a service completion instant.

In steady-state, we define

$P_i(n) \equiv$ probability that there are $n$ customers in the system when there are $i$ vacation servers, where $i = 0,1,2,\ldots,c$ and $n = 0,1,2,\ldots$

and $\Pi = [\Pi_0, \Pi_1, \ldots, \Pi_c, \Pi_{c+1}, \ldots]$ with $\Pi_n = [P_0(n), P_1(n), \ldots, P_c(n)],\ n = 0,1,2,\ldots$ denotes the steady-state probability vector.

Referring to the state-transition-rate diagram as shown in Fig. 1, the infinitesimal generator $Q$ of the QBD describing the M/M/c queueing system with modified Bernoulli vacation is of the form

$$Q = \begin{bmatrix} A_0 & B & & & & & & & \\ C_1 & A_1 & B & & & & & & \\ & C_2 & A_2 & B & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & C_{c-1} & A_{c-1} & B & & \\ & & & & & C_c & A_c & B & \\ & & & & & & C_{c+1} & A_c & B \\ & & & & & & & \ddots & \ddots & \ddots \end{bmatrix}. \tag{1}$$

The entries $B$, $A_j (0 \leqslant j \leqslant c)$, and $C_j (1 \leqslant j \leqslant c+1)$ are matrices of order $(c+1)$ defined by $B = \lambda I_{(c+1)\times(c+1)}$, where $I_{(c+1)\times(c+1)}$ is the identity matrix of order $c+1$.

$$A_j = \begin{bmatrix} a_{j,1} & & & & & & \\ \eta & a_{j,2} & & & & & \\ & 2\eta & a_{j,3} & & & & \\ & & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ & & & & (c-1)\eta & a_{j,c} & \\ & & & & & c\eta & a_{j,c+1} \end{bmatrix}, \quad j = 0,1,\ldots,c$$

with diagonal elements $a_{j,k} = -[\lambda + \min(j, c+1-k)\mu + (k-1)\eta],\ k = 1,2,\ldots,c+1$.

$$C_j = \begin{bmatrix} & jq\mu & jp\mu & & & & \\ & & jq\mu & jp\mu & & & \\ & & & \ddots & \ddots & & \\ & & & & jq\mu & jp\mu & \\ & & & & & (j-1)\mu & \\ & & & & & & (j-2)\mu \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \} \#=(c+1-j) \\ \\ \\ \\ \\ \#=j \end{matrix}, \quad j = 1,2,\ldots,c+1.$$

It is well-known (Theorem 3.1.1 of Neuts [26]) that the steady-state probability vector exists if and only if

$$xBe < xC_{c+1}e, \tag{2}$$

where $e$ is a column vector with dimension $(c+1)$ and all elements equal to one. $x$ is the invariant probability of the matrix $F = C_{c+1} + A_c + B$. $x$ satisfies $xF = \mathbf{0}$ and $xe = 1$.

Substituting $B$ and $C_{c+1}$ into Eq. (2) and doing some routine manipulations, then we have $x_0 = 1$ and $x_i = 0,\ i = 1, 2, \ldots, c$. The stability condition is

$$\lambda < \sum_{i=0}^{c} (c-i) x_i \mu = c\mu. \tag{3}$$

which is the stability condition of the ordinary M/M/c queueing system.

Let the steady-state probability vector $\Pi$ be the unique solution to $\Pi Q = \mathbf{0}$ and $\sum_{n=0}^{\infty} \Pi_n e = 1$. We note that the vector $\Pi = [\Pi_0, \Pi_1, \ldots, \Pi_c, \Pi_{c+1}, \ldots]$ has the following properties

$$\Pi_{c+k} = \Pi_c R^k, \text{ for } k \geqslant 1. \tag{4}$$

The matrix $R$ is the unique non-negative solution with spectral radius less than one of the equation

$$B + RA_c + R^2 C_{c+1} = \mathbf{0}. \tag{5}$$

It is necessary to solve the rate matrix $R$ of (5), in order to obtain the steady-state solution vector $\Pi = [\Pi_0, \Pi_1, \ldots, \Pi_c, \Pi_{c+1}, \ldots]$. Based on the structures of matrices, $B$, $A_c$ and $C_{c+1}$ which are represented as the lower triangular matrix, thus the matrix solution $R$ is also the lower triangular matrix. (see Latouche and Ramaswami [27])

Doing some arduous algebraic derivations and arrangement, we develop the explicit formula for matrix $R$ as follows:

$$R = \begin{bmatrix} r_{1,1} & & & & & \\ r_{2,1} & r_{2,2} & & & & \\ \vdots & \vdots & \ddots & & & \\ \vdots & \vdots & & \ddots & & \\ r_{c,1} & r_{c,2} & \cdots & \cdots & r_{c,c} & \\ r_{c+1,1} & r_{c+1,2} & \cdots & \cdots & r_{c+1,c} & r_{c+1,c+1} \end{bmatrix}, \tag{6}$$

where $r_{i,i} = \frac{\theta_i - \sqrt{\theta_i^2 - 4(c+1-i)\lambda\mu}}{2(c+1-i)\mu}$, for $1 \leqslant i \leqslant c$, $r_{c+1,c+1} = \frac{\lambda}{\lambda + c\eta}$,

$$r_{i,j} = \frac{(c+1-j)\mu \sum_{k=j+1}^{i-1} r_{i,k} r_{k,j} + j r_{i,j+1} \eta}{\lambda + (c+1-j)(1 - r_{i,i} - r_{j,j})\mu + (j-1)\eta}, \text{ for } 2 \leqslant i \leqslant c+1, \quad i > j,$$

$$r_{i,j} = 0, \text{ for } i < j$$

and $\theta_i = \lambda + (c+1-i)\mu + (i-1)\eta$. Note that $r_{i,i}$ is the corresponding eigen-values of the rate matrix $R$ and the spectral radius of $R$, $sp(R)$ is less than one if $\rho = \lambda(c\mu)^{-1} < 1$.

**Proof.** Firstly, the first diagonal element

$$r_{1,1} = \frac{\lambda + c\mu - \sqrt{(\lambda + c\mu)^2 - 4c\lambda\mu}}{2c\mu} = \frac{\lambda + c\mu - |\lambda - c\mu|}{2c\mu} = \lambda(c\mu)^{-1} = \rho < 1.$$

For $2 \leqslant i \leqslant c$, the diagonal element $r_{i,i}$ is got from the quadratic equation

$$f(x) = (c+1-i)\mu x^2 - \theta_i x + \lambda = 0.$$

It should be noted that there exists exact one real root in (0,1) because

$$f(0) = \lambda > 0,$$
$$f(1) = (c+1-i)\mu - \theta_{i-1} + \lambda = -(i-1)\eta < 0.$$

Finally, $r_{c+1,c+1} = \lambda/(\lambda + \eta) < 1$. Therefore, all diagonal elements (eigen-values) of rate matrix $R$ are less than 1. Therefore, the spectral radius of rate matrix $R$, $sp(R) = \max_{1 \leqslant i \leqslant s+1} \{r_{i,i}\}$ is less than 1. By using the rate matrix $R$, we can solve the steady-state probability more efficiently. □

## 3. Algorithm solution

Under the stability condition, by solving the equation $\Pi Q = \mathbf{0}$ with the normalization condition, we obtain

$$\Pi_0 A_0 + \Pi_1 C_1 = \mathbf{0}, \tag{7a}$$

$$\Pi_{i-1} B + \Pi_i A_i + \Pi_{i+1} C_{i+1} = \mathbf{0}, \quad 1 \leqslant i \leqslant c-1, \tag{7b}$$

$$\Pi_{c-1} B + \Pi_c A_c + \Pi_c R C_{c+1} = \mathbf{0}, \tag{7c}$$

$$\Pi_c R^{i-1-c} B + \Pi_c R^{i-c} A_c + \Pi_c R^{i+1-c} C_{c+1} = \mathbf{0}, \quad c+1 \leqslant i, \tag{7d}$$

$$\sum_{i=0}^{\infty} \Pi_i e = 1. \tag{8}$$

After doing routine substitutions to (7a)–(7c), we have

$$\begin{aligned} &\Pi_0 = \Pi_1 C_1 (-A_0)^{-1} = \Pi_1 \phi_1, \\ &\Pi_{i-1} = \Pi_i C_i [-(\phi_{i-1} B + A_{i-1})]^{-1} = \Pi_i \phi_i, \quad 2 \leqslant i \leqslant c \end{aligned} \tag{9}$$

and

$$\Pi_c \phi_c B + \Pi_c A_c + \Pi_c R C_{c+1} = \mathbf{0}. \tag{10}$$

Consequently, $\Pi_i (0 \leqslant n \leqslant c-1)$ in Eq. (9) can be written in terms of $\Pi_c$ as $\Pi_i = \Pi_c \prod_{i=c}^{i+1} \phi_i$, $i = 0, 1, 2, \ldots, c-1$ where $\phi_1 = C_1 (-A_0)^{-1}$ and $\phi_i = C_i [-(\phi_{i-1} B + A_{i-1})]^{-1}$, $2 \leqslant i \leqslant c$. The rest steady-state vectors $\Pi_c, \Pi_{c+1}, \ldots$ can be calculated recursively as $\Pi_i = \Pi_c R^{i-c}$, for $i \geqslant c$. Once $\Pi_c$ is determined, the steady-state solutions $\Pi = [\Pi_0, \Pi_1, \ldots, \Pi_c, \Pi_{c+1}, \ldots]$ are obtained. The vector $\Pi_c$ is given by solving Eq. (10) with the following normalization condition.

$$\begin{aligned} \sum_{i=0}^{\infty} \Pi_i e &= [\Pi_0 + \Pi_1 + \ldots + \Pi_{c-1} + \Pi_c + \Pi_{c+1} + \Pi_{c+2} + \ldots] e \\ &= \left[ \Pi_c \Pi_{i=c}^{1} \phi_i + \Pi_c \prod_{i=c}^{2} \phi_i + \ldots + \Pi_c \prod_{i=c}^{c} \phi_i + \Pi_c + \Pi_c R + \Pi_c R^2 + \ldots \right] e \\ &= \Pi_c \left[ \sum_{n=1}^{c} \prod_{i=c}^{n} \phi_i + (I-R)^{-1} \right] e = 1. \end{aligned} \tag{11}$$

Solving Eqs. (10) and (11) in accordance with Cramer's rule, we obtain $\Pi_c$. Then the prior state probabilities $[\Pi_0, \Pi_1, \Pi_2, \ldots, \Pi_{c-1}]$ are computed from (9) and $[\Pi_{c+1}, \Pi_{c+2}, \Pi_{c+3}, \ldots]$ are gained by the formula $\Pi_i = \Pi_c R^{i-c}$, $i \geqslant c+1$. The solution procedure of the steady-state probabilities is summarized as below:

**Algorithm 1. Recursive Solver**

**INPUT** $c,\ B,\ A_0, A_1, \ldots, A_c,\ C_1, C_2, \ldots, C_{c+1},\ R,\ e = [1, \ldots, 1]^T$, and $I$ is the identity matrix of order $c+1$
**OUTPUT** $\Pi_0, \Pi_1, \Pi_2, \ldots$
Step 1 set $\phi_1 = C_1 (-A_0)^{-1}$
Step 2 for $i = 2$ to $c$
  Step 3 set $\phi_i = C_i [-(\phi_{i-1} B + A_{i-1})]^{-1}$
Step 4 end
Step 5 for $k = 1$ to $c$
  Step 6 set $\Phi_k = \prod_{i=c}^{k} \phi_i$
Step 7 end
Step 8 Solving $\Pi_c \phi_c B + \Pi_c A_c + \Pi_c R C_{c+1} = \mathbf{0}$, and $\Pi_c \left[ \sum_{i=1}^{c} \Phi_i + (I-R)^{-1} \right] e = 1$
Step 9 for $i = 0$ to $c-1$
  Step 10 set $\Pi_i = \Pi_c \Phi_{i+1}$
Step 11 end
Step 12 for $i = c+1$ to ...
  Step 13 set $\Pi_{i+1} = \Pi_i R$
Step 14 end
Step 15 OUTPUT

## 4. System performance measures

There are several general descriptors (system performance measures) of the M/M/c/MBSV queueing system, such as the expected number of customers in the system (denoted by $L_s$), the expected number of customers in the queue (denoted by $L_q$), the expected number of busy, idle and vacation servers (denoted by $E[B]$, $E[I]$ and $E[V]$, respectively). The expressions for these system performance measures are given by

$$\begin{aligned} L_s &= \sum_{i=1}^{\infty} i\Pi_i e = \sum_{i=1}^{c-1} i\Pi_i e + c\Pi_c e + (c+1)\Pi_c Re + \dots \\ &= \sum_{i=1}^{c-1} i\Pi_c \Phi_{i+1} e + c\Pi_c (I-R)^{-1} e + \Pi_c R(I-R)^{-2} e \\ &= \Pi_c \left[ \sum_{i=1}^{c-1} i\Phi_{i+1} + c(I-R)^{-1} + R(I-R)^{-2} \right] e, \end{aligned} \tag{12}$$

$$\begin{aligned} L_q &= \Pi_1 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} + \Pi_2 \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 2 \end{bmatrix} + \dots + \Pi_c \begin{bmatrix} 0 \\ \vdots \\ c-1 \\ c \end{bmatrix} + \Pi_c R \left( \begin{bmatrix} 0 \\ \vdots \\ c-1 \\ c \end{bmatrix} + e \right) + \dots \\ &= \sum_{i=1}^{c-1} \Pi_c \Phi_{i+1} u_i + \Pi_c (I-R)^{-1} u_c + \Pi_c R(I-R)^{-2} e \\ &= \Pi_c \left[ \sum_{i=1}^{c-1} \Phi_{i+1} u_i + (I-R)^{-1} u_c + R(I-R)^{-2} e \right], \end{aligned} \tag{13}$$

$$E[V] = \sum_{i=0}^{\infty} \Pi_i \begin{bmatrix} 0 \\ 1 \\ \vdots \\ c \end{bmatrix} = \Pi_c \left[ \sum_{i=1}^{c} \Phi_i + (I-R)^{-1} \right] \begin{bmatrix} 0 \\ 1 \\ \vdots \\ c \end{bmatrix}, \tag{14}$$

$$\begin{aligned} E[I] &= \Pi_0 \begin{bmatrix} c \\ c-1 \\ \vdots \\ 0 \end{bmatrix} + \Pi_1 \begin{bmatrix} c-1 \\ c-2 \\ \vdots \\ 0 \end{bmatrix} + \Pi_2 \begin{bmatrix} c-2 \\ c-3 \\ \vdots \\ 0 \end{bmatrix} + \dots + \Pi_{c-1} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \Pi_c \Phi_1 v_1 + \Pi_c \Phi_2 v_2 + \dots + \Pi_c \Phi_c v_c \\ &= \Pi_c \sum_{i=1}^{c} \Phi_i v_i, \end{aligned} \tag{15}$$

$$E[B] = c - E[V] - E[I], \tag{16}$$

where $u_i = \left[ \overbrace{0,\dots,0}^{\#=c+1-i}, \overbrace{1,2,\dots,i}^{\#=i} \right]^T$ and $v_i = \left[ \overbrace{c-i, c-i-1, \dots, 1, 0}^{\#=c-i+1} \overbrace{0,\dots,0}^{\#=i} \right]^T$ are $(c+1)$ dimensional column vector.

To understand how system performance measures (such as $L_s$ and $E[B]$) listed above vary with $\lambda$, $\mu$ and $\eta$, we now perform some numerical investigation to the measures based on changing the value of system parameters. For computation, we let $p = 0.5$. The numerical results of $L_s$ are obtained by considering the following three cases with different values of $c$.

Case 1. $\mu = 5.5,\ \eta = 2.0$, vary $\lambda$ from 2.0 to 5.0.
Case 2. $\lambda = 2.0,\ \eta = 2.0$, vary $\mu$ from 2.5 to 5.5.
Case 3. $\lambda = 2.0,\ \mu = 3.0$, vary $\eta$ from 1.0 to 4.0. Results of $L_s$ are depicted in Figs. 2–4 for Case 1–3, respectively. Fig. 2 reveals that (i) $L_s$ increases quickly as $\lambda$ increases for $c = 1$, and (ii) $L_s$ slightly increases as $\lambda$ increases for $c \geqslant 2$. We observes from Fig. 3 that (i) $L_s$ drastically decreases as $\mu$ increases for $c = 1$, and (ii) $L_s$ slightly decreases as $\mu$ increases for $c \geqslant 2$. One sees from Fig. 4 that $L_s$ slightly decreases as $\eta$ increases. We also interest in the effect of different parameters on the expected number of busy servers ($E[B]$). The following three cases are considered:
Case 4. $E[B]$ versus $\lambda$ from 2.0 to 5.0 when $\mu = 5.5$ and $\eta = 2.0$.
Case 5. $E[B]$ versus $\mu$ from 2.5 to 5.5 when $\lambda = 2.0$ and $\eta = 2.0$.
Case 6. $E[B]$ versus $\eta$ from 1.0 to 4.0 when $\lambda = 2.0$ and $\mu = 3.0$.

The numerical illustrations of the expected number of busy servers are graphically presented in Figs. 5–7 for Case 4–6, respectively. We observe from Figs. 5 and 6 that $E[B]$ increases as $\lambda$ increases or $\mu$ decreases. Fig. 7 reports $E[B]$ is a constant is independent of $\eta$. From the investigation, it is interesting that $E[B]$ nearly equals to $\lambda/\mu$. However, it is very difficult to proof the results. In the next section, we will provide the proof of single server case ($c = 1$).

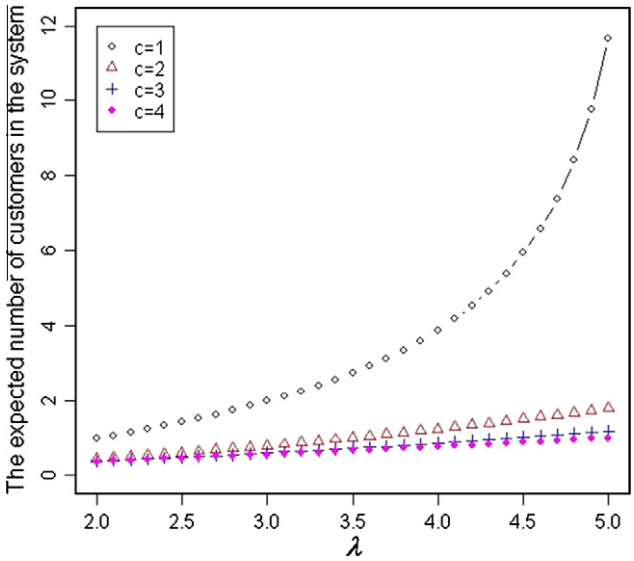


**Fig. 2.** The effect of $\lambda$ on the expected number of customers in the system.

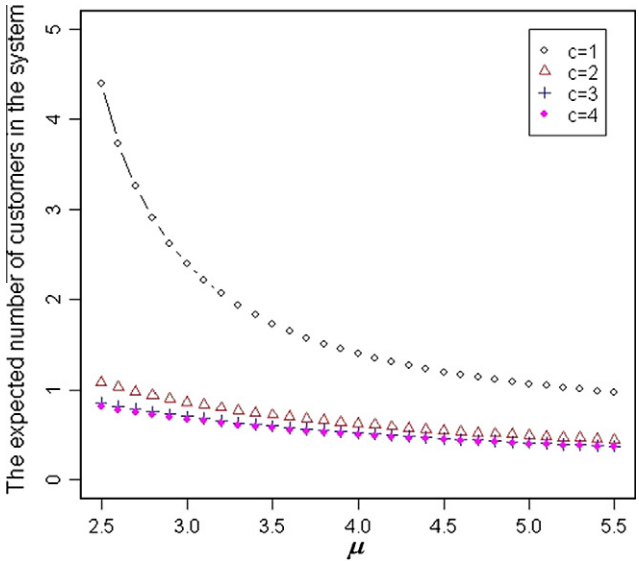


**Fig. 3.** The effect of $\mu$ on the expected number of customers in the system.

### *4.1. Some explicit results of single server*

As a particular case, the M/M/1/MBSV queueing system, in which the server may take a vacation if server is free at service completion instant, steady-state equations in states (0,0), (0,1), and (1,0) are given by:

$$\begin{aligned} &\lambda P_0(0) = q\mu P_0(1) + \eta P_1(0), \\ &(\lambda+\eta)P_1(0) = p\mu P_0(1), \end{aligned} \tag{17}$$

which implies

$$\lambda[P_0(0) + P_1(0)] = \mu P_0(1). \tag{18}$$

Thus $B = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$, $A_c = \begin{bmatrix} -(\lambda+\mu) & 0 \\ \eta & -(\lambda+\eta) \end{bmatrix}$, and $C_{c+1} = \begin{bmatrix} \mu & 0 \\ 0 & 0 \end{bmatrix}$.

Substituting $B,\ A_c,\ C_{c+1}$ into $B + RA_c + R^2C_{c+1} = 0$ and solving the quadratic equation above, we have

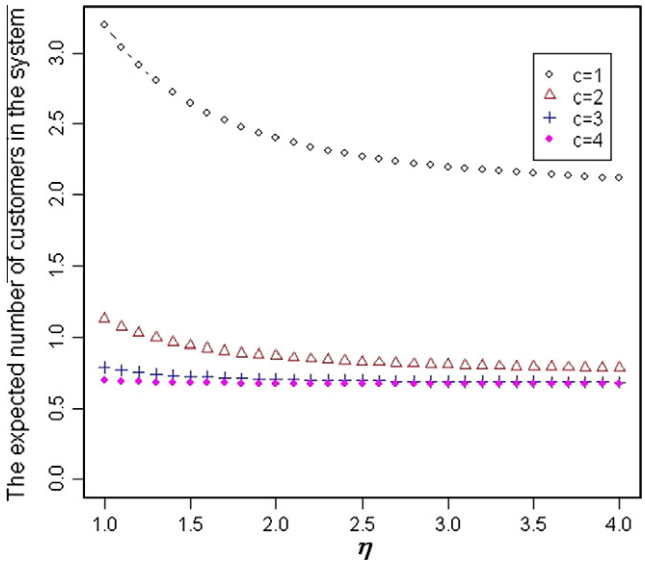


**Fig. 4.** The effect of $\eta$ on the expected number of customers in the system.

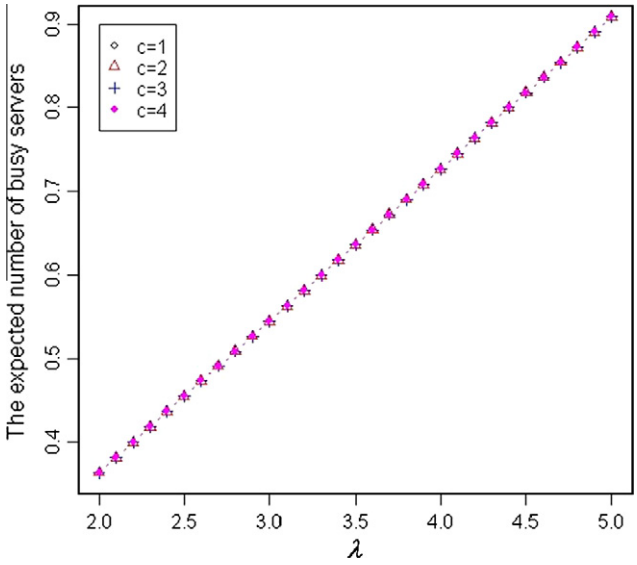


**Fig. 5.** The expected number of busy servers versus $\lambda$.

$$R = \begin{bmatrix} \frac{\lambda}{\mu} & 0 \\ \frac{\lambda}{\mu} & \frac{\lambda}{\lambda+\eta} \end{bmatrix}. \tag{19}$$

Also, Eq. (19) can be obtained from (6).

For the case of single server, the steady-state distribution $\Pi_1 = [P_0(1), P_1(1)]$ satisfies the following

$$[P_0(0), P_1(0)]\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} + [P_0(1), P_1(1)]\begin{bmatrix} -(\lambda+\mu) & 0 \\ \eta & -(\lambda+\eta) \end{bmatrix} + [P_0(1), P_1(1)]\begin{bmatrix} \frac{\lambda}{\mu} & 0 \\ \frac{\lambda}{\mu} & \frac{\lambda}{\lambda+\eta} \end{bmatrix}\begin{bmatrix} \mu & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{20}$$

which leads to

$$\frac{\lambda p \mu}{\lambda+\eta} P_0(1) = (\lambda+\eta) P_1(1). \tag{21}$$

Using the normalization condition (11) to obtain $\Pi_1$

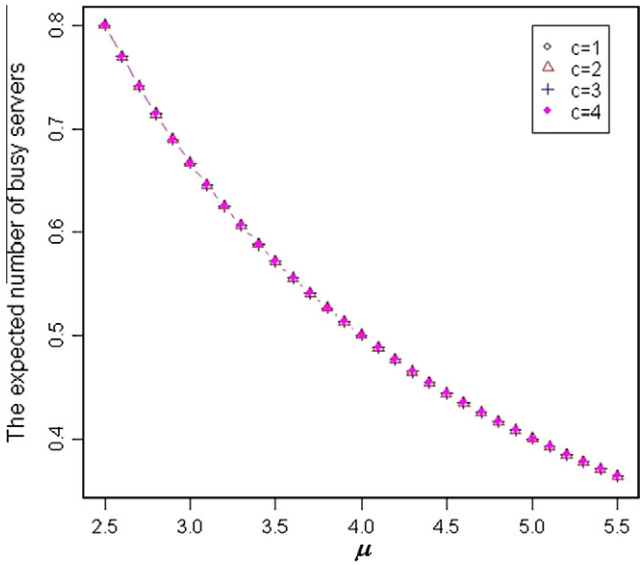


**Fig. 6.** The expected number of busy servers versus $\mu$.

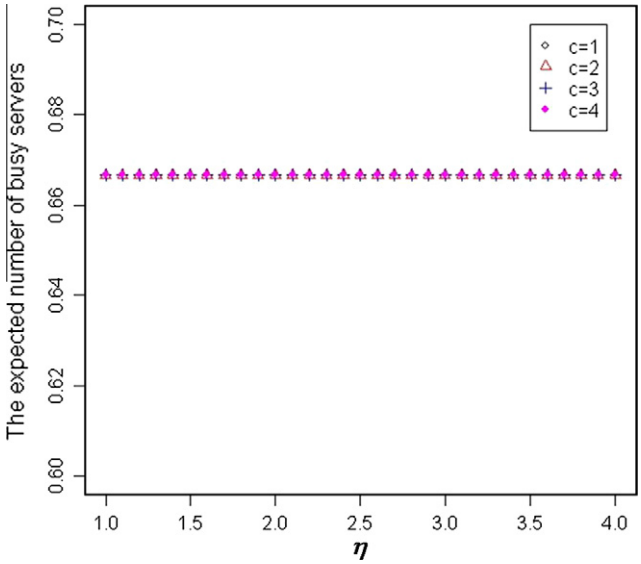


**Fig. 7.** The expected number of busy servers versus $\eta$.

$$[P_0(1), P_1(1)](I-R)^{-1}\begin{bmatrix}1\\1\end{bmatrix} + [P_0(0), P_1(0)] = 1. \tag{22}$$

Substituting (18), (19) and (21) into (22), we get $P_0(1)$ and $P_1(1)$ as following

$$P_0(1) = \frac{\lambda(\lambda+\eta)(\mu-\lambda)\eta}{(p\lambda^2+\eta\lambda+\eta^2)\mu^2} \tag{23}$$

and

$$P_1(1) = \frac{\lambda^2 p\eta(\mu-\lambda)}{(\lambda+\eta)(p\lambda^2+\eta\lambda+\eta^2)\mu}. \tag{24}$$

After the gaining of $\Pi_1$, the rest steady-state probability vectors $\Pi_2, \Pi_3, \Pi_4, \ldots$ can be obtained recursively with $\Pi_2 = \Pi_1 R,\ \Pi_3 = \Pi_2 R, \ldots$, and so on.

The expected number of busy servers is

$$E[B] = [P_0(1), P_1(1)](I-R)^{-1}\begin{bmatrix}1\\0\end{bmatrix} = \left[P_0(1), \frac{\lambda p\mu}{(\lambda+\eta)^2}P_0(1)\right]\begin{bmatrix}\frac{\mu}{\mu-\lambda} & 0\\ \frac{\lambda(\mu+\eta)}{\eta(\mu-\lambda)} & \frac{\lambda+\eta}{\eta}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \left[P_0(1), \frac{\lambda p\mu}{(\lambda+\eta)^2}P_0(1)\right]\begin{bmatrix}\frac{\mu}{\mu-\lambda}\\ \frac{\lambda(\mu+\eta)}{\eta(\mu-\lambda)}\end{bmatrix}$$
$$= \frac{\mu(p\lambda^2+\eta\lambda+\eta^2)}{\eta(\lambda+\eta)(\mu-\lambda)}P_0(1) = \rho. \tag{25}$$

It is interesting that the result of (25) for the M/M/1/MBSV queueing system, in which the server may take a vacation if server is free at service completion instant, is the same as that of the ordinary M/M/1 queue.

## 5. Optimization analysis

In this section, we construct the total expected cost function per unit time based on the system performance measures for the M/M/c/MBSV queueing system, in which the number of servers $(c)$ is a discrete decision variable, and the service rate $(\mu)$ and the vacation rate $(\eta)$ are continuous decision variables. Our main objective is to find the optimum number of servers $c^*$, and the optimum values of service rate and vacation rate $(\mu^*, \eta^*)$ simultaneously to minimum the cost function. Let us define the following cost elements:

$C_h \equiv$ holding cost per unit time per customer present in the system;
$C_s \equiv$ cost per unit time of providing a service rate $\mu$;
$C_v \equiv$ cost per unit time when one server is on vacation;
$C_r \equiv$ cost per unit time of providing a vacation rate $\eta$;
$C_p \equiv$ fixed cost for purchasing one server.

Using the definition of the cost parameters listed above, the total expected cost function per unit time is given by:

$$F(c,\mu,\eta) = C_h L_s + C_s\mu + C_v E[V] + C_r\eta + C_p c, \tag{26}$$

where $L_s$ and $E[V]$ are defined previously.

The analytic study of the optimization behavior of the expected cost function would have been an arduous task to undertake since the decision variables appear in an expression which is a highly nonlinear and complex and non-linear in terms of $(c,\mu,\eta)$.

In the next section, we firstly use the Quasi-Newton method to find the optimal value of continuous variable $(\mu,\eta)$, say $(\mu^*,\eta^*)$, and then use direct search method to search the optimal value of discrete variable $c$, say $c^*$.

### 5.1. Quasi-Newton method

For practice use, the number of servers is bounded by a positive integer $c_U \geqslant 1$. We want to find the joint optimal value $(\mu^*,\eta^*)$ for each given $c$ in the feasible set $\{1,2,\ldots,c_U\}$. The cost minimization problem can be illustrated mathematically as

$$F(c,\mu^*,\eta^*) = \min_{\text{and s.t.(3)}}\{F(c,\mu,\eta)|c\}, \quad c = 1,2,\ldots,c_U. \tag{27}$$

For the problem of (27), we should show the convexity of $F(c,\mu,\eta)$ in $(\mu,\eta)$. However, this work is difficult to implement. We note that the derivative of the cost function $F$ with respect to $(\mu,\eta)$ indicates the direction which cost function increases. It means that, the optimal value $(\mu^*,\eta^*)$ can be found along this opposite direction of the gradient. (see Chong and Zak [28]). That is, for a fixed $c$, Quasi-Newton method is employed to search $(\mu,\eta)$ until the minimum value of $F(c,\mu,\eta)$ is achieved, say $F(c,\mu^*,\eta^*)$. An effective procedure that makes it possible to calculate the optimal value $(c,\mu^*,\eta^*)$ is presented as follows:

**Algorithm 2. Quasi-Newton Method**

**INPUT** Cost function $F(c,\mu,\eta)$, $c$, $R,\lambda,\mu$, initial value $\vec{\theta}^{(0)} = [\mu^{(0)},\eta^{(0)}]^T$, and the tolerance $\varepsilon$.

**OUTPUT** approximation solution $[\mu^*,\eta^*]^T$.

Step 1 Set the initial trial solution for $\vec{\theta}^{(0)}$, and compute $F(c,\mu^{(0)},\eta^{(0)})$.

Step 2 While $|\partial F/\partial\mu| > \varepsilon$ or $|\partial F/\partial\eta| > \varepsilon$ do Step 3–4

Step 3 Compute the cost gradient $\vec{\nabla}F(\vec{\theta}) = [\partial F/\partial\mu, \partial F/\partial\eta]^T$ and the cost Hessian matrix $H(\vec{\theta}) = \begin{bmatrix}\partial^2F/\partial\mu^2 & \partial^2F/\partial\mu\partial\eta\\ \partial^2F/\partial\eta\partial\mu & \partial^2F/\partial\eta^2\end{bmatrix}$ at point $\vec{\theta}^{(i)}$.

Step 4 Find the new trial solution $\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - [H(\vec{\theta}^{(i)})]^{-1}\vec{\nabla}F(\vec{\theta}^{(i)})$.

Step 5 OUTPUT

**Table 1**
The illustration of the implement process of Quasi-Newton method.

| Iterations | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Case (i): $(\lambda, p) = (10, 0.5)$ with initial value $(c, \mu, \eta) = (1, 15, 2.0)$ | | | | | | | |
| $F(c, \mu, \eta)$ | 987.973 | 882.065 | 845.430 | 838.786 | 838.458 | 838.457 | **838.457** |
| $\mu$ | 15 | 16.4035 | 17.3194 | 17.5741 | 17.5901 | 17.5903 | **17.5903** |
| $\eta$ | 2.0 | 2.78381 | 3.59146 | 4.13419 | 4.29150 | 4.30117 | **4.30120** |
| $\frac{\partial F}{\partial \mu}$ | −19.9189 | 16.4035 | −1.11143 | −0.05537 | −0.00008 | $1.57 \times 10^{-8}$ | $-8 \times 10^{-11}$ |
| $\frac{\partial F}{\partial \eta}$ | −176.914 | 2.78381 | −21.1504 | −4.00835 | −0.22011 | −0.00075 | $-8.5 \times 10^{-9}$ |
| $L_s$ | 6.05405 | 4.24438 | 3.28115 | 2.89676 | 2.81312 | 2.80833 | 2.80831 |
| Hessian | $\begin{bmatrix} 14.26 & -0.114 \\ -0.114 & 225.9 \end{bmatrix}$ | $\begin{bmatrix} 6.754 & -0.086 \\ -0.086 & 83.88 \end{bmatrix}$ | $\begin{bmatrix} 4.511 & -0.069 \\ -0.069 & 39.00 \end{bmatrix}$ | $\begin{bmatrix} 4.070 & -0.062 \\ -0.062 & 25.48 \end{bmatrix}$ | $\begin{bmatrix} 4.046 & -0.06 \\ -0.06 & 22.76 \end{bmatrix}$ | $\begin{bmatrix} 4.046 & -0.06 \\ -0.06 & 22.60 \end{bmatrix}$ | $\begin{bmatrix} 4.046 & -0.06 \\ -0.06 & 22.60 \end{bmatrix}$ |
| Case (ii): $(\lambda, p) = (20, 0.2)$ with initial value $(c, \mu, \eta) = (3, 10, 2)$ | | | | | | | |
| $F(c, \mu, \eta)$ | 1052.33 | 971.631 | 942.421 | 936.060 | 935.615 | 935.612 | **935.612** |
| $\mu$ | 10 | 11.5153 | 13.2741 | 14.6682 | 15.1728 | 15.2168 | **15.2171** |
| $\eta$ | 2.0 | 2.46623 | 2.71924 | 2.75081 | 2.74176 | 2.74098 | **2.74098** |
| $\frac{\partial F}{\partial \mu}$ | −59.8568 | −23.5812 | 13.2741 | −1.69159 | −0.12620 | −0.00083 | $-4.8 \times 10^{-8}$ |
| $\frac{\partial F}{\partial \eta}$ | −77.6947 | −24.0490 | 2.71924 | −0.59183 | −0.04460 | −0.00031 | $4.16 \times 10^{-9}$ |
| $L_s$ | 4.82721 | 3.42012 | 2.65292 | 2.31515 | 2.22369 | 2.21614 | 2.21609 |
| Hessian | $\begin{bmatrix} 36.25 & 10.56 \\ 10.56 & 132.3 \end{bmatrix}$ | $\begin{bmatrix} 12.73 & 4.723 \\ 4.723 & 62.22 \end{bmatrix}$ | $\begin{bmatrix} 5.547 & 2.580 \\ 2.580 & 41.83 \end{bmatrix}$ | $\begin{bmatrix} 3.386 & 1.861 \\ 1.861 & 38.37 \end{bmatrix}$ | $\begin{bmatrix} 2.900 & 1.686 \\ 1.686 & 38.22 \end{bmatrix}$ | $\begin{bmatrix} 2.862 & 1.671 \\ 1.671 & 38.21 \end{bmatrix}$ | $\begin{bmatrix} 2.862 & 1.671 \\ 1.671 & 38.21 \end{bmatrix}$ |

To demonstrate the valid and the procedure of optimization solution, we perform some examples shown in Table 1 by considering the following cost parameters as

$C_h$ = \$90/customer/unit time, $C_s$ = \$15/unit time,
$C_v$ = \$30/server, $C_r$ = \$45/unit time, $C_p$ = \$120/server

From Table 1, we can see that the minimum expected cost per unit time of **838.457** is achieved at $(\mu^*, \eta^*) = (\mathbf{17.5903}, \mathbf{4.30120})$ by using 6 iterations, which is $c = 1$ based on Case (i) with initial value $(\mu, \eta) = (15, 2.0)$. Based on Case (ii), the initial value $(c, \mu, \eta) = (3, 10, 2)$ and the minimum expected cost per unit time of **935.612** is achieved at $(\mu^*, \eta^*) = (\mathbf{15.2171}, \mathbf{2.74098})$ by using 6 iterations.

### 5.2. Direct search method

After we obtain the joint optimal value $(\mu^*, \eta^*)$ of the continuous variable $(\mu, \eta)$, we will use direct search method to obtain the optimal $c$ such that the expected cost function $F(c, \mu^*, \eta^*)$ attains a minimum, say $F(c^*, \mu^*, \eta^*)$. Therefore, the cost minimization problem can be illustrated mathematically as:

$$F(c^*, \mu^*, \eta^*) = \min_{c \in \{1,2,\ldots,c_U\}} \{F(c, \mu^*, \eta^*)\}. \tag{28}$$

The procedure to find the optimal solution is described in the following. A numerical example is shown in Table 2 based on (i) $(\lambda, p) = (15, 0.5)$ and (ii) $(\lambda, p) = (20, 0.8)$.

**Table 2**
The optimal value $(\mu^*, \eta^*)$ and the corresponding minimum expected cost.

| $c$ | Initial value | Coverage value$(\mu^*, \eta^*)$ | Iteration | Cost* |
|---|---|---|---|---|
| (i) $(\lambda, p) = (15, 0.5)$ | | | | |
| $c = 1$ | [20,2.0] | [24.32507,5.332980] | 7 | 1052.297 |
| $c = 2$ | [15,2.0] | [15.28433,3.798293] | 6 | 895.4944 |
| $c = 3$ | [10,2.0] | [12.37270,3.088068] | 6 | 920.8427 |
| $c = 4$ | [10,2.0] | [11.00938,2.679454] | 5 | 998.4310 |
| $c = 5$ | [10,2.0] | [10.26962,2.428360] | 5 | 1098.187 |
| (ii) $(\lambda, p) = (20, 0.8)$ | | | | |
| $c = 1$ | [25,5.0] | [30.75986,6.423140] | 6 | 1288.713 |
| $c = 2$ | [20,3.0] | [18.73113,4.824175] | 6 | 1071.252 |
| $c = 3$ | [15,2.0] | [14.85998,4.032956] | 6 | 1073.578 |
| $c = 4$ | [10,2.0] | [13.05122,3.560957] | 6 | 1137.429 |
| $c = 5$ | [10,2.0] | [12.06278,3.260737] | 6 | 1232.625 |

**Table 3**
The optimal value $(c^*, \mu^*, \eta^*)$ and it's minimum expected value for various value of $\lambda$ and $p$.

| $(\lambda, p)$ | (5,0.2) | (10,0.2) | (20,0.2) | (5,0.8) | (10,0.8) | (20,0.8) |
|---|---|---|---|---|---|---|
| $c^*$ | 2 | 2 | 2 | 2 | 2 | 2 |
| $(\mu^*, \eta^*)$ | [7.249477,1.471333] | [11.60659,2.295007] | [19.16225,3.550663] | [7.091449,2.326386] | [11.32231,3.368702] | [18.73113,4.824175] |
| $F(c^*, \mu^*, \eta^*)$ | 532.099 | 685.935 | 932.038 | 610.522 | 792.191 | 1071.252 |
| $L_s$ | 1.154063 | 1.717796 | 2.565803 | 1.481779 | 2.275863 | 3.436747 |
| $E[V]$ | 0.442712 | 0.465296 | 0.463387 | 0.870082 | 0.864552 | 0.796331 |

**Algorithm 3. Direct Search Method**

**INPUT** $c_U$, $F^* = M$ which $M$ is a sufficiently large number
**OUTPUT** approximation solution $S^* = (c^*, \mu^*, \eta^*)$ and $F^* = F(c^*, \mu^*, \eta^*)$.
Step 1 for $c$ =1 to $c_U$
  Step 2 Set a initial trial solution $(\mu, \eta)$
  Step 3 Use Quasi-Newton method to find the optimal value $(\mu^*, \eta^*)$ and the
    cost function $F(c, \mu^*, \eta^*)$
  Step 4 If the algorithm is diverge, back to step 2 end if
  Step 5 If $F(c, \mu^*, \eta^*) < F^*$
    Step 6 $F^* = F(c, \mu^*, \eta^*)$ and $S^* = (c, \mu^*, \eta^*)$
  Step 7 end if
Step 8 end
Step 9 OUTPUT $S^*$ and $F^*$

It is noted that the optimal value $(c^*, \mu^*, \eta^*) = (2, 15.284, 3.7983)$ and the corresponding minimum cost $F^* = 895.4944$ for Case (i). For Case (ii), $(c^*, \mu^*, \eta^*) = (2, 18.731, 4.8242)$ and $F^* = 1071.252$ are optimal.

Finally, we perform a sensitivity investigation to the optimal values $(c^*, \mu^*, \eta^*)$. For various values of $\lambda$ and $p$, the minimum expected cost $F(c^*, \mu^*, \eta^*)$ and the system performance measures $L_s$, and $E[V]$ at the optimum values $(c^*, \mu^*, \eta^*)$ are shown in Table 3.

From Table 3, it is seen that (i) $c^*$ is insensitive to $\lambda$ or $p$; (ii) $\mu^*$ increases as $\lambda$ increases; and (iii) $\eta^*$ increases as $\lambda$ or $p$ increases. Moreover, the minimum expected cost increases $F(c^*, \mu^*, \eta^*)$as $\lambda$ or $p$ increases.

## 6. Concluding remarks

An infinite capacity M/M/c queueing system with modified Bernoulli vacation (M/M/c/MBSV queueing system) was studied used the matrix-geometric method. This system was formulated as a QBD process and the necessary and sufficient condition for the stability of the system was deduced. More important, the explicitly closed-form solution of stable condition and rate matrix of the QBD model was obtained, and then the stationary probability distributions were explicitly developed. We have not only obtained exactly the steady-state probability and the system performance measures using matrix approach but also presented one efficient method to find the optimal number of servers, the optimal service rate and vacation rate, simultaneously, so as to reach the minimum cost. The explicit closed-form of single-server system (M/M/1/MBSV queueing system) was also discussed. We finally have performed a sensitivity analysis between the joint optimal values $(c^*, \mu^*, \eta^*)$ and specific values of $\lambda$ and $p$.

From practice viewpoint in this study, servers can be viewed of machines. Each machine takes a single vacation when it completes a service and finds an empty queue. This corresponds to a planned maintenance progressing for the idle machine. If one machine surely finds someone waiting in the queue at a service completion, it takes a vacation. This corresponds to an unplanned maintenance progressing for the machine. This study is not difficultly extended to the case that server/machine takes multiple vacations when an empty queue is found upon a service completion.

## Acknowledgement

The authors thank editor and the referees for their remarks and comments, which helped me to improve the clarity of the article.

## References

[1] B.T. Doshi, Queueing systems with vacations-a survey, Queueing Systems 1 (1986) 29–66.
[2] H. Takagi, Queueing analysis: A foundation of performance evaluation, vacation and priority systems, Part I, Vol. I, North-Holland, Amsterdam, 1991.

[3] H.W. Lee, S.S. Lee, J.O. Park, K.C. Chae, Analysis of $M^{[x]}/G/1$ queue with N policy and multiple vacations, J. Appl. Prob. 31 (1994) 467–496.
[4] S.S. Lee, H.W. Lee, S. H Yoon, K.C. Chae, Batch arrival queue with N policy and single vacation, Computers Ops Res. 22 (1995) 173–189.
[5] G.V. Krishna Reddy, R. Nadarajan, R. Arumuganathan, Analysis of a bulk queue with N-policy multiple vacations and setup times, Computers Ops Res. 25 (1998) 957–967.
[6] G. Choudhury, An $M^{[x]}/G/1$ queueing system with a setup period and a vacation period, Queueing Systems 36 (2000) 23–38.
[7] G. Choudhury, A batch arrival queue with a vacation time under single vacation policy, Computers Ops Res. 29 (2002) 1941–1955.
[8] J.-C. Ke, Optimal strategy policy in batch arrival queue with server breakdowns and multiple vacations, Mathematical Methods of Operations Research 58 (1) (2003) 41–56.
[9] J.-C. Ke, The optimal control of an M/G/1 queueing system with server startup and two vacation types, Applied Mathematical Modelling 27 (2003) 437–450.
[10] J.-C. Ke, Y.-K. Chu, A modified vacation model $M^{[x]}/G/1$ system, Applied Stochastic Business and Industry 22 (2006) 1–16.
[11] J.-C. Ke, H.-I. Huang, Y.-K. Chu, Batch arrival queue with N-policy and at most J vacations, Applied Mathematical Modelling 34 (2010) 451–466.
[12] J.-C. Ke, K.-B. Huang, W.L. Pearn, The randomized vacation policy for a batch arrival queue, Applied Mathematical Modelling 34 (2010) 1524–1538.
[13] L. Tadj, G. Choudhury, C. Tadj, A quorum queueing system with a random setup time under N-policy and with Bernoulli vacation schedule, Quality Technology & Quantitative Management 3 (2) (2006) 145–160.
[14] G. Choudhury, K.C. Madan, A two phase batch arrival queueing system with a vacation time under Bernoulli schedule, Applied Mathematic and Computation 149 (2004) 337–349.
[15] K.C. Madan, W. Abu-Dayyeh, F. Taiyyan, A two server queue with Bernoulli schedules and a single vacation policy, Applied Mathematics and Computation 145 (2003) 59–71.
[16] G. Choudhury, A two-stage batch arrival queueing system with a modified Bernoulli schedule vacation under N-policy, Mathematical and Computer Modelling 42 (2005) 71–85.
[17] G. Choudhury, L. Tadj, M. Paul, Steady state analysis of an $M^{[x]}/G/1$ queue with two phase service and Bernoulli vacation schedule under multiple vacation policy, Applied Mathematical Modelling 31 (2007) 1079–1091.
[18] Y. Levy, U. Yechiali, An M/M/c queue with servers' vacations, INFOR 14 (1976) 153–163.
[19] X. Chao, Y. Zhao, Analysis of multiple-server queues with station and server vacations, European Journal of Operational Research 110 (1998) 392–406.
[20] N. Tian, Q. Li, J. Cao, Conditional stochastic decompositions in the M/M/c queue with server vacations, Stochast. Models 14 (2) (1999) 367–377.
[21] N. Tian, X. Xu, M/M/c queue with synchronous multiple vacation of partial servers, OR Trans. 5 (3) (2001) 85–94.
[22] X. Xu, Z.G. Zhang, Analysis of multiple-server queue with a single vacation (e,d)-policy, Performance Evaluation 63 (2006) 825–838.
[23] Z.G. Zhang, N. Tian, Analysis of queueing systems with synchronous single vacation for some servers, Queueing Systems 45 (2003) 161–175.
[24] Z.G. Zhang, N. Tian, Analysis on queueing systems with synchronous vacations of partial servers, Performance Evaluation 52 (2003) 269–282.
[25] J.-C. Ke, C.H. Lin, J.Y. Yang, Z.G. Zhang, Optimal (d, c) vacation policy for a finite buffer M/M/c queue with unreliable servers and repairs, Applied Mathematical Modeling 33 (2009) 3949–3962.
[26] M.F. Neuts, Matrix Geometric Solutions in Stochastic Models: an Algorithmic Approach, The John Hopkins University Press, Baltimore, 1981.
[27] G. Latouche and V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling (ASA-SIAM Series on Statistics and Applied Probability). 1999.
[28] E.K.P. Chong, S.H. Zak, An Introduction to Optimization, 2nd ed., John Wiley and Sons, Inc., 2001.