Optimal two-threshold policies in an M/G/1 queue with two vacation types
Zhang, Z.G.; Vickson, R.G.; Eenige, van, M.J.A.

Published: 01/01/1995

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
- A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):
Optimal two-threshold policies in an $M/G/1$ queue with two vacation types

Z.G. Zhang
R.G. Vickson
M.J.A. van Eenige

Eindhoven, October 1995
The Netherlands
OPTIMAL TWO-THRESHOLD POLICIES IN AN M/G/1 QUEUE WITH TWO VACATION TYPES

Zhe G. Zhang
Department of Business Administration
University College of the Fraser Valley
Abbotsford, British Columbia, Canada
V2S 7M9

Raymond G. Vickson
Department of Management Sciences, Faculty of Engineering
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Michel J.A. van Eenige
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands

revised, September 1995

1Supported by Grant A9304 from the Natural Sciences and Engineering Research Council of Canada and by a MRCO Grant from the Manufacturing Research Corporation of Ontario.
Abstract

This paper treats two-threshold policies for an M/G/1 queue with two types of generally distributed random vacations: type 1 (long) and type 2 (short) vacations. Upon returning from a vacation, the server observes the queue length. If this is less than the lower threshold, the server takes a type 1 vacation; if it is between the two thresholds, the server takes a type 2 vacation; and if it is at or above the upper threshold, the server resumes serving the queue exhaustively. There is a shutdown cost for starting a series of vacations, a linear customer waiting cost, and type-dependent vacation reward rates. Renewal theory or the PASTA property is used to develop expressions for the average queue length and the average system cost for generally distributed vacations. A search procedure, which is provably finite for exponentially distributed vacations, is developed for determining the optimal threshold values.

Keywords: Vacation Models, Optimal Threshold Policies, Renewal Reward Processes, PASTA.
1 Introduction

Consider a queueing system with a single server who leaves for a vacation period of random length whenever the system becomes empty. There are two types of vacations available upon returning from a vacation. The server inspects the system and decides whether to take a type 1 (long) vacation, a type 2 (short) vacation, or to resume serving the customers exhaustively. This paper focuses on a special class of service policies characterized by two thresholds $0 \leq n \leq N$. If the number of customers, $i$, waiting in the system at the instant of a vacation completion is less than $n$, the server will go on a type 1 vacation. If $n \leq i < N$, the server will go on a type 2 vacation. Finally, if $i \geq N$, the server will resume serving the customers until the system is again empty. This service discipline is called a two-threshold policy and is denoted as $(n, N)$ in the sequel. Many existing queueing systems with vacations are special cases of this model. These cases are summarized in Table 1.

Table 1: Some Special Cases of the M/G/1 Vacation Queue with Two-Threshold Policies

<table>
<thead>
<tr>
<th>Case</th>
<th>$n$, $N$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n = 0$, $N = 0$</td>
<td>M/G/1 queue with a single vacation.</td>
</tr>
<tr>
<td>2</td>
<td>$n = 1$, $N = 1$</td>
<td>M/G/1 queue with multiple vacations.</td>
</tr>
<tr>
<td>3</td>
<td>$1 &lt; n = N$</td>
<td>M/G/1 queue with a single-threshold policy.</td>
</tr>
<tr>
<td>4</td>
<td>$n = 0$, $N &gt; 0$</td>
<td>M/G/1 queue with setup times and multiple vacations.</td>
</tr>
<tr>
<td>5</td>
<td>$0 &lt; n &lt; N$</td>
<td>M/G/1 queue with a general two-threshold policy.</td>
</tr>
</tbody>
</table>

Cases 1–4 in Table 1 have been studied in the past two decades, while case 5 is new. Skinner[11] and Cooper [1] investigated the M/G/1 model with multiple vacations in which the server continues to take vacations until he finds at least one customer in the system. Levy and Yechiali [9] studied systems with multiple vacations and also introduced the single vacation model, where the server takes just one vacation whenever he empties the system. The literature on vacation models is extensive and growing. Doshi ([2], [3]) surveys many of the significant achievements in this area. Hofri [6], Teghem [13] and Kella [7] emphasize
that, for a given cost structure, the decision on when to resume work should be dependent on the state of the system, i.e., the number of customers waiting in the queue. For a single threshold policy in an $M/G/1$ queue with a single type of vacation, Kella [7] introduced a simple algorithm for determining the optimal threshold value for the case of linear holding costs. Federgruen and So [4] showed the optimality of a single-threshold policy for the same system with a more general holding cost function. They also discussed vacation models with setup times. The cost structure in our model consists of a linear customer waiting cost, a vacation startup cost, and a reward (negative cost) during vacations. The vacation reward rates are constant and may depend on the vacation type. In this study, we concentrate on vacation models with two-threshold $(n, N)$ policies and develop a provably finite search procedure to find the optimal values of $n$ and $N$.

At this time we do not have a proof of optimality of two-threshold policies. Such policies are intuitively appealing, and their optimality can be proved if some "reasonable" statements can be shown to hold. For example, we need to know that if it is better to take a type 1 than a type 2 vacation when there are $i$ customers present, it is also better to do this if the number of customers is more than $i$. In a dynamic programming formulation, such statements will hold if the value function of the two-threshold policy is convex in the state (for the policy then satisfies the optimality equation), but such convexity has not been established at this time. However, it has been found to hold in all numerical examples treated so far. Regardless of whether the two-threshold policy is theoretically optimal, it is a policy that a queue manager would certainly find to be of interest. As indicated above, it also generalizes several types of policies extant in the vacation queueing literature. Note that the case in which the server takes type 2 vacations only may be obtained by setting $V_1 = V_2$ in the analysis, but is not a special case of the $(n, N)$ policy. Similarly, the case in which the server takes no vacations at all cannot be obtained by specializing the two-threshold policy, but may be considered separately and compared with the two-threshold case in arriving at an overall optimum.

One example of an application fitting our general model is the following production
control problem. A factory produces two types of items which occasionally are defective. It takes more time to produce a type 1 item than a type 2 item. The good quality items are sold, while the defective ones are kept in storage until they can be reworked to meet specifications. Suppose that one of the machines in the factory may be switched as needed from a production mode to a repair mode to perform this rework, and the rework times are i.i.d. random variables that are the same for both item types. Because of the cost involved in switching modes, this machine will rework all defective items exhaustively, and then switch back to the production mode. To fit this quality control problem to our general model, we can take the defective items as the customers and the special machine as the server, where the server is available for serving waiting customers only when the machine is in repair mode. The service time is the time required to repair a defective item. The vacation times correspond to the production times for both types of items.

Another application of this model is to the scheduling of jobs in an automobile repair garage which specializes in two types of major services, such as brake and exhaust repairs. Assume that customers must leave their vehicles in the garage and are notified when their jobs are done. The garage also provides a third type of service (quick service, usually less than 15 minutes, such as oil change, safety inspection, etc.) on a first come, first served basis. The customers requesting quick service will form a queue. If we consider the two types of major services as two types of vacations and allocate the work station time to the three different types of jobs according to the two-threshold policy, our model will apply to this situation.

The rest of this paper is organized as follows. Section 2 gives the model description and notation. Section 3 develops the average cost function for the system using two different approaches. For the exponential vacation case we develop an exact closed-form expression to compute the average cost. For the general vacation case, we develop a system of recursive equations to determine quantities needed in computing the average cost. The complexity of the recursive equations prevents analysis by a closed-form expression in this case, but numerical computation of the average cost is not difficult. Section 4 justifies, in the expo-
nential vacation case, a finite search procedure to locate optimal threshold values for the model. Since the objective function is not currently known to be convex, or even unimodal in the threshold values, it is necessary to develop results that guarantee the finiteness of a direct search over the values of \( n \) and \( N \), so that in any given problem we can know that the global optimum has been attained. Section 5 presents some numerical examples. Finally, Section 6 gives some directions for future research.

2 Model Description and Notation

The arrivals to the system occur according to a Poisson process with rate \( \lambda \). The service times provided by a single server are general i.i.d. random variables denoted by \( S \). There are two types of vacations for the server to take. Type 1 (Type 2) vacation times are i.i.d. random variables denoted by \( V_1 \) (\( V_2 \)). The server follows a two-threshold policy and serves the queue exhaustively. As usual, the arrival process, the service times, and the vacation times are independent of each other. Type 1 vacations are stochastically larger than type 2 vacations.

For a random variable \( X \), \( F_X(x) = P(X \leq x) \) denotes its probability distribution, \( \tilde{X}(s) = E(e^{-sX}) \) denotes its Laplace-Stieltjes transform (LST), \( \tilde{X}(z) = E(z^X) \) denotes its \( z \) transform, and \( \tilde{X} = E(X) \) and \( X^{(2)} = E(X^2) \) denote its first and second moments.

For \( i = 0, 1, \ldots \), let \( a_i = \int_0^{\infty} p_i(t) \, dF_{V_1}(t) \) and \( b_i = \int_0^{\infty} p_i(t) \, dF_{V_2}(t) \) be the probabilities that there are \( i \) arrivals during a long vacation and a short vacation, respectively. Here \( p_i(t) = e^{-\lambda t}(\lambda t)^i/i! \) is the probability that \( i \) arrivals occur during \([0, t]\). Finally, let \( \rho = \lambda \bar{S} \) and assume \( \rho < 1 \) for stability of the system.

Define a \((n, N)\) cycle period, denoted by \( \theta_{n,N} \), as the random time interval between successive service completion instants at which the system becomes empty. The analysis uses the fact that the \((n, N)\) cycle period \( \theta_{n,N} \) can be divided into three parts. These are the accumulation period \( (T_N) \) during which the \( N \) customers arrive, the forward recurrence time \( (FRT, \text{or residual life}) \) of the last vacation \( (R) \), and the attend period \( (A) \) during which the server processes the customers; see Figure 1. In the sequel we also use the result that
the standard M/G/1 queue busy period $\theta$ has LST determined by $\bar{\theta}(s) = \tilde{S}(s + \lambda - \lambda\bar{\theta}(s))$ and mean $\bar{\theta} = \tilde{S}/(1 - \rho)$ (see [8]).

3 The Average Cost Function

Let $R \equiv R_n, N$ denote the FRT of the last vacation in the $(n, N)$ policy. The mean cycle length is given by

$$\bar{\theta}_{nN} = T_N + \bar{R} + \bar{A} = \frac{N}{\lambda} + \bar{R} + (N + \lambda\bar{R})\bar{\theta} = \frac{1}{1 - \rho} \left( \frac{N}{\lambda} + \bar{R} \right), \text{ for } N \geq n > 0. \quad (1)$$

Below we need also the mean FRT ($\bar{R}_1$) of the last type 1 vacation. This is the same as $\bar{R}_{nN}$, so is related to $\bar{\theta}_{nN}$:

$$\bar{\theta}_{nN} = \frac{1}{1 - \rho} \left( \frac{n}{\lambda} + \bar{R}_1 \right). \quad (2)$$

Let $\overline{TC}_{nN}$ denote the expected total cost incurred during an $(n, N)$ cycle under the cost structure imposed in Section 1. Obviously, $\overline{TC}_{nN}$ consists of the net cost incurred during $T_N$ (denoted as $\bar{C}_{T_N}$), the cost incurred during $R$ (denoted as $\bar{C}_R$), the cost incurred during $A$ (denoted as $\bar{C}_A$) and the rewards earned in $T_N$ and $R$. Notice that we are considering a queueing system with Poisson arrivals and a linear holding cost rate function $H(l) = hl$.

First we determine $\bar{C}_{T_N}$ and $\bar{C}_R$. We have

$$\bar{C}_{T_N} = \frac{h}{\lambda} + \frac{2h}{\lambda} + \cdots + \frac{(N - 1)h}{\lambda} = \frac{N(N - 1)h}{2\lambda}. \quad (3)$$

By conditioning on $R$ and the number of arrivals $k$ during $R$, and noting that conditional upon $k$ Poisson arrivals occurring in an interval of length $t$, the interarrival times have mean
\[ t/(k + 1) \text{ (see [10])}, \quad \bar{C}_R \text{ can be obtained as} \]
\[
\bar{C}_R = hN\ddot{R} + \sum_{k=0}^{\infty} \int_{0}^{\infty} e^{-\lambda t} \left( \frac{\lambda t^k}{k!} \left[ \sum_{i=0}^{k} \frac{hit}{(k + 1)} \right] dF_R(t) \right)
\]
\[
= hN\ddot{R} + \frac{1}{2}\lambda hR^{(2)}.\]  

(4)

Next we determine \( \bar{C}_A \). Define a 1-busy period of the M/G/1 queue which starts with \( l \geq 1 \) customers present to be the time needed to reduce the queue size by 1. Let \( C_l^1 \) be the expected waiting cost during such a 1-busy period:

\[
C_1^1 = \frac{h}{1 - \rho} \left( \frac{\lambda S^{(2)}}{2(1 - \rho)} + \ddot{S} \right), \quad \text{and}
\]
\[
C_l^1 = h(l - 1)\ddot{\theta} + C_1^1, \quad \text{for } l > 1.
\]

Let \( D \) be the number of customers present in the system when \( A \) begins, and let \( d_k \) be the probability that \( D = k \). Then \( \bar{C}_A \) can be computed as

\[
\bar{C}_A = \sum_{k=N}^{\infty} \left( \sum_{l=1}^{k} C_l^1 \right) d_k
\]
\[
= \frac{h\ddot{\theta}}{2} D^{(2)} + \left( C_1^1 - \frac{h\ddot{\theta}}{2} \right) \dot{D}.
\]  

(5)

Since

\[
\dot{D}(x) = z\dddot{R}(\lambda - \lambda x),
\]  

the first two moments of \( D \) are

\[
\bar{D} = N + \lambda \ddot{R},
\]  

(7)
\[
D^{(2)} = N^2 + (2N + 1)\lambda \ddot{R} + \lambda^2 R^{(2)}.
\]  

(8)

Finally we determine the expected total reward \( \pi \) earned during the vacation periods. Let \( VT_i \) be the average time of type \( i \) vacations during a cycle for \( i = 1, 2 \). Note that \( VT_1 = n/\lambda + \dddot{R}_{nn} \) and \( VT_2 = N/\lambda + \dddot{R}_{nn} - VT_1 \). Using (1) and (2), these become \( VT_1 = (1 - \rho)\dddot{\theta}_{nn} \) and \( VT_2 = (1 - \rho)[\dddot{\theta}_{nn} - \dddot{\theta}_{nn}] \). Since there are constant reward rates \( r_1 \) and \( r_2 \) when the server is on vacation, we have

\[
\pi = r_1 VT_1 + r_2 VT_2 = (1 - \rho)[(r_1 - r_2)\dddot{\theta}_{nn} + r_2 \dddot{\theta}_{nn}].
\]  

(9)
The instant of starting the first vacation is a regeneration point of the process. Combining the renewal reward process theorem with (3), (4), (5) and (9) yields the average cost \( g_{nN} \) of the system:

\[
g_{nN} = (\tau_0 + \bar{C}_{T_N} + \bar{C}_R + \bar{C}_A - \pi)/\bar{\theta}_{nN}.
\]  

Here \( \tau_0 \) is the shut-down cost for switching the server from the service mode to the vacation mode; it may include a fixed setup cost incurred whenever the server resumes work.

Another way to derive the average cost is based on the Poisson-arrivals-see-time average (PASTA) property (see Wolff [14]). Let \( \bar{T}_s \) denote the average sojourn time of an arbitrary customer and \( \bar{L} \) the average number of customers in the system. Then, using Figure 1 and the PASTA property, we obtain

\[
\bar{T}_s = \bar{L}\bar{S} + \rho \left( \frac{S^{(2)}}{2S} - \bar{S} \right) + \sum_{k=1}^{N} q_k \left( \frac{N - k}{\lambda} + \bar{R} \right) + q_R \bar{R}_R + \bar{S}.
\]  

Here \( q_k \) is the probability that the arbitrary customer is the \( k \)th arrival during \( T_N \), \( q_R \) is the probability that the arbitrary customer arrives during \( R \), and \( \bar{R}_R \) the expected residual life of the FRT of the last vacation, as measured from the arrival instant of such a customer.

The first two terms of equation (11) represent the mean time the arbitrary arriving customer has to wait for the customers in front of him to be served (including a possible customer in service). The third and fourth terms represent the mean time the arbitrary arriving customer has to wait before the server returns from a vacation and starts serving customers. The final term denotes the mean service time of the arbitrary arriving customer. Rewriting equation (11) yields

\[
\bar{T}_s = \bar{L}\bar{S} + \rho \left( \frac{S^{(2)}}{2S} - \bar{S} \right) + \bar{S} + (1 - \rho) \frac{N/\lambda((N - 1)/(2\lambda) + \bar{R}) + \bar{R} \bar{R}_R}{T_N + \bar{R}}.
\]

Using Little's Law, i.e., \( \bar{L} = \lambda \bar{T}_s \), we finally obtain

\[
\bar{L} = \frac{\lambda \rho S^{(2)}}{1 - \rho \frac{2S}{S}} + \rho + \frac{N((N - 1)/(2\lambda) + \bar{R}) + \lambda \bar{R} \bar{R}_R}{T_N + \bar{R}}.
\]
Note that the first two terms correspond to the average number of customers in the system without vacations, i.e., the standard $M/G/1$ queue (see the well-known stochastic decomposition property of Fuhrmann and Cooper [5]). Knowing $\bar{L}$, we can obtain

$$g_{nN} = \bar{L}h + \frac{1}{\bar{\theta}_{nN}}(r_0 - r_1\bar{V}_1 - r_2\bar{V}_2).$$ \hfill (14)

### 3.1 The Exponential Vacations Case

When both types of vacations are exponentially distributed, we can compute $\bar{R}$, $\bar{R}_1$ and $R^{(2)}$ explicitly for use in (1), (2), (4) and (7)-(9). Note that $R$ can be either the FRT of a long vacation or the FRT of a short vacation. Because the vacations are exponentially distributed random variables and $T_N$ is independent of the server’s return process, the FRT of a vacation has the same probability distribution as the full vacation. Let $P_{nN}^1$ be the probability that $R$ is a type 1 vacation. In terms of the distribution functions, we have

$$F_R(t) = P_{nN}^1F_{V_1}(t) + (1 - P_{nN}^1)F_{V_2}(t).$$ \hfill (15)

The memoryless property of the exponential distribution implies

$$P_{nN}^1 = p^{N-n},$$ \hfill (16)

where $p = \lambda\bar{V}_1/(1 + \lambda\bar{V}_1)$. Thus

$$\bar{R}_1 = \bar{V}_1, \quad \text{and} \quad E(R^j) = p^{N-n}E(V^j_1) + (1 - p^{N-n})E(V^j_2), \quad j = 1, 2.$$

Alternatively, we may use (13) and (14). For exponential vacations the quantity $\bar{R}_R$ is given as

$$\bar{R}_R = \frac{P_{nN}^1\bar{V}_1}{P_{nN}^1\bar{V}_1 + (1 - P_{nN}^1)\bar{V}_2}\bar{V}_1 + \frac{(1 - P_{nN}^1)\bar{V}_2}{P_{nN}^1\bar{V}_1 + (1 - P_{nN}^1)\bar{V}_2}\bar{V}_2.$$ \hfill (17)

After some algebraic manipulation, we can verify that equation (14) is in agreement with equation (10) for the exponential vacations case.

### 3.2 The General Vacations Case

If the vacations are not exponentially distributed, the distribution of $R$ will not satisfy (15). This means that we cannot get explicit expressions for $\bar{R}$, $\bar{R}_1$, $R^{(2)}$ and $\bar{\theta}_{nN}$. However, we
can develop recursive relations to calculate these quantities numerically. In the sequel we compute \( \tilde{\theta}_{n,N} \) and \( \tilde{R}_{n}^{(2)} \) recursively, and obtain \( \tilde{R}_{n,N} \) from (1): \( \tilde{R}_{n,N} = (1 - \rho)\tilde{\theta}_{n,N} - N/\lambda \).

Let \( U \) be the number of customers present in the system at the instant that the server returns from the first vacation. If \( U \geq N \), the rest of the \((n, N)\) cycle consists of \( U \) independent standard \( M/G/1 \) busy periods \( \theta^1, \ldots, \theta^U \) which are distributed like \( \theta \). If \( n \leq U < N \), the rest of the \((n, N)\) cycle has the same distribution as the independent sum of a single-threshold \( \tilde{N} - U \) cycle for type 2 vacations, denoted by \( \tilde{\theta}_{n-U}^{(2)} \), and \( U \) independent standard \( M/G/1 \) busy periods. Finally, if \( U < n \), the rest of the \((n, N)\) cycle will be the sum of an \((n - U, N - U)\) cycle and \( U \) independent standard \( M/G/1 \) busy periods. These observations and a conditional probability argument yield an equation for the LST of \( \tilde{\theta}_{n,N} \) for \( 1 \leq n \leq N \):

\[
\tilde{\theta}_{n,N}(s) = \sum_{k=0}^{n-1} \int_0^\infty E \exp \left[ -s \left( t + \tilde{\theta}_{n-k,N-k} + \sum_{i=0}^{k} \theta^i \right) \right] p_k(t) dF_{V_1}(t)
+ \sum_{k=n}^{N-1} \int_0^\infty E \exp \left[ -s \left( t + \tilde{\theta}_{n-k}^{(2)} + \sum_{i=1}^{k} \theta^i \right) \right] p_k(t) dF_{V_1}(t)
+ \int_0^\infty E \exp \left[ -s \left( t + \sum_{i=1}^{k} \theta^i \right) \right] p_k(t) dF_{V_1}(t).
\]

(18)

This implies

\[
\tilde{\theta}_{n,N} = \tilde{V}_1 + \sum_{k=0}^{n-1} a_k (k\tilde{\theta} + \tilde{\theta}_{n-k,N-k}) + \sum_{k=n}^{N-1} a_k (k\tilde{\theta} + \tilde{\theta}_{N-k}^{(2)})
+ \sum_{k=N}^{\infty} a_k k\tilde{\theta}
= \tilde{V}_1 + \lambda \tilde{V}_1 \tilde{\theta} + \sum_{k=0}^{n-1} a_k \tilde{\theta}_{n-k,N-k} + \sum_{k=n}^{N-1} a_k \tilde{\theta}_{N-k}^{(2)}
= \frac{\tilde{V}_1}{1 - \rho} + \sum_{k=1}^{n} a_{n-k} \tilde{\theta}_{k,N-n+k} + \sum_{k=n}^{N-1} a_k \tilde{\theta}_{N-k}^{(2)}.
\]

(19)

The second equality follows from the fact that \( \sum_{i=0}^{\infty} ia_i = \lambda \tilde{V}_1 \) and the third equality follows because \( \tilde{\theta} = \tilde{S}/(1 - \rho) \). Rewriting (19) in a recursive fashion gives

\[
\tilde{\theta}_{n,N} = \frac{1}{1 - a_0} \left( \frac{\tilde{V}_1}{1 - \rho} + \sum_{k=1}^{N-n} a_{n-k} \tilde{\theta}_{k} + \sum_{k=1}^{n-1} a_{n-k} \tilde{\theta}_{k,N-n+k} \right), \quad \text{for } N \geq n \geq 1,
\]

(20)
where an empty sum is equal to 0.

To complete the computations we may obtain \( \tilde{\theta}_k \) using the following recursive relation derived by Kella [7]:

\[
\tilde{\theta}_k = \frac{1}{1 - \bar{b}_0} \left( \frac{\bar{V}_2}{1 - \rho} + \sum_{j=1}^{k-1} \bar{b}_{k-j} \tilde{\theta}_j \right), \quad \text{for } k \geq 1, \tag{21}
\]

and \( \tilde{\theta}_0 = 0 \). Similarly, for \( N \geq n = 0 \), we have

\[
\tilde{\theta}_{0N} = \frac{\bar{V}_1}{1 - \rho} + \sum_{k=0}^{N-1} a_k \tilde{\theta}_{N-k}, \tag{22}
\]

and for \( N = n = 0 \), we have

\[
\tilde{\theta}_{00} = \bar{V}_1 + a_0 \left( \frac{1}{\lambda} + \bar{\theta} \right) + \sum_{k=1}^{\infty} a_k k \bar{\theta} = \frac{1}{1 - \rho} \left( \bar{V}_1 + \frac{a_0}{\lambda} \right). \tag{23}
\]

Let \( R_{(2),n} \) denote the FRT of the last vacation in a single-threshold policy with threshold \( n \geq 1 \) and type 2 vacations only. Define \( U \) as before. For \( 0 \leq U < n \), \( R_{n,N} \) has the same distribution as \( R_{n-U,N-U} \); for \( n \leq U < N \), \( R_{n,N} \) has the same distribution as \( R_{(2),N-U} \); and for \( U \geq N \), \( R_{n,N} \) equals \( V_1 - T_N \). We thus have

\[
\bar{R}_{0N}(s) = \sum_{k=0}^{N-1} a_k \bar{R}_{(2),N-k}(s) + \mathcal{A}_N(s), \tag{24}
\]

\[
\bar{R}_{nN}(s) = \frac{1}{1 - a_0} \left[ \sum_{k=1}^{n-1} a_k \bar{R}_{n-k,N-k}(s) + \sum_{k=n}^{N-1} a_k \bar{R}_{(2),N-k}(s) + \mathcal{A}_N(s) \right], \tag{25}
\]

where

\[
\mathcal{A}_N(s) = E[1\{U \geq N\}e^{-s(V_1 - T_N)}] = E[1\{T_N \leq V_1\}e^{-s(V_1 - T_N)}] = \int_0^\infty I_N(s,t) \, dF_{V_1}(t),
\]

with

\[
I_N(s,t) = \int_0^t \frac{\lambda^{N-1}}{(N-1)!} e^{-\lambda \tau} e^{-s(t-\tau)} \, d\tau = \left( \frac{\lambda}{\lambda - s} \right)^N e^{-st} - \sum_{k=0}^{N-1} \left( \frac{\lambda}{\lambda - s} \right)^{N-k} \frac{(\lambda t)^k}{k!} e^{-\lambda t}.
\]

10
This gives

\[ A_N(s) = \left( \frac{\lambda}{\lambda - s} \right)^N \tilde{V}_1(s) - \sum_{k=0}^{N-1} a_k \left( \frac{\lambda}{\lambda - s} \right)^{N-k}. \]  

(26)

Similarly,

\[ \tilde{R}_{(2),n}(s) = \frac{1}{1 - b_0} \left[ \sum_{k=1}^{n-1} b_k \tilde{R}_{(2),n-k}(s) + B_n(s) \right]. \]

(27)

where

\[ B_n(s) = \left( \frac{\lambda}{\lambda - s} \right)^n \tilde{V}_2(s) - \sum_{k=0}^{n-1} b_k \left( \frac{\lambda}{\lambda - s} \right)^{n-k}. \]

(28)

Equations (25)-(28) yield recursive expressions for the moments of \( R_{nN} \). In the first moment case the results are not new, as equation (1) implies their equivalence to (20). For the second moment case we have

\[ R_{0N}^{(2)} = \sum_{k=0}^{N-1} a_k R_{(2),N-k}^{(2)} + A_N, \]

(29)

\[ R_{nN}^{(2)} = \frac{1}{1 - b_0} \left[ \sum_{k=1}^{n-1} a_k R_{(2),n-k,N-k}^{(2)} + \sum_{k=n}^{N-1} a_k R_{(2),N-k}^{(2)} + A_N \right], \]

(30)

where

\[ A_N = V_1^{(2)} - \frac{2N}{\lambda} \tilde{V}_1 + \frac{1}{\lambda^2} N(N+1) - \frac{1}{\lambda^2} \sum_{k=0}^{N-1} (N-k)(N+1-k)a_k \]

\[ = A_{N-1} - \frac{2}{\lambda} \tilde{V}_1 + \frac{2N}{\lambda^2} - \frac{2}{\lambda^2} \sum_{k=0}^{N-1} a_k(N-k), \]

(31)

and

\[ R_{(2),n}^{(2)} = \frac{1}{1 - b_0} \left[ \sum_{k=1}^{n-1} b_k R_{(2),n-k,n-k}^{(2)} + B_n \right], \]

(32)

where

\[ B_n = V_2^{(2)} - \frac{2n}{\lambda} \tilde{V}_2 + \frac{1}{\lambda^2} n(n+1) - \frac{1}{\lambda^2} \sum_{k=0}^{n-1} (n-k)(n+1-k)b_k \]

\[ = B_{n-1} - \frac{2}{\lambda} \tilde{V}_2 + \frac{2n}{\lambda^2} - \frac{2}{\lambda^2} \sum_{k=0}^{n-1} b_k(n-k). \]

(33)

Remarks:

1. For the special case of exponentially distributed vacations, (25)-(28) imply

\[ \tilde{R}_{nN}(s) = p^{N-n}/(1 + \tilde{V}_1 s) + (1 - p^{N-n})/(1 + \tilde{V}_2 s), \]

in agreement with (15).
2. For generally distributed vacations and $0 < n < N,$

$$\lim_{\lambda \to 0} \bar{R}_{n,N}(s) = \frac{[1 - \bar{V}(s)]}{s\bar{V}_1}, \quad \text{and} \quad \lim_{\lambda \to 0} \bar{R}_{n,n}(s) = \frac{[1 - \bar{V}_2(s)]}{s\bar{V}_2}. $$

These are intuitive, for $[1 - \bar{V}(s)]/s\bar{V}_1$ is the LST of the equilibrium FRT of the $V_i$ renewal process, or limiting FRT distribution if this limit exists. When $\lambda \to 0,$ arrivals are rare, so that by the time the $N$th arrival occurs many type 2 vacations will have been completed. Note, however, that the result applies even if the $V_i$ renewal process does not have a limiting FRT distribution, as in the case of deterministic vacation times, for example.

3. For numerical stability, $R_{n,N}^{(2)}$ and $R_{(2),n}$ should be computed using the following simpler but equivalent recursions. Define

$$u_{n,N} = R_{n,N}^{(2)} - N(N+1)/\lambda^2 \quad \text{and} \quad u_n = R_{(2),n}^{(2)} - n(n+1)/\lambda^2$$

for $1 \leq n \leq N.$ Then (30) and (32) may be rewritten as

$$u_{n,N} = \frac{1}{1 - a_0} \left( \sum_{k=1}^{n-1} a_k u_{n-k,N-k} + \sum_{k=n}^{N-1} a_k u_{N-k} + V_1^{(2)} - 2N\bar{V}_1/\lambda \right),$$

$$u_n = \frac{1}{1 - b_0} \left( \sum_{k=1}^{n-1} b_k u_{n-k} + V_2^{(2)} - 2n\bar{V}_2/\lambda \right).$$

Using the equations developed so far, we can compute performance measures such as the average queue length and the average operating cost of a given two-threshold policy. However, we cannot determine the optimal $n$ and $N$ values explicitly. The next section justifies a finite search procedure to find these optimal values in the case of exponentially distributed vacation times.

4 Determination of Optimal Threshold Values

A Search Algorithm: the Exponential Vacations Case

Suppose that the vacation time distributions are exponential. We consider three types of stationary policies for the system. For the no-vacation policy, the system is a standard
M/G/1 queue and its average operating cost is simply $g_0 = C_1/(\lambda^{-1} + \bar{\theta})$. The average operating cost $g_N^{(2)}$ for the single-threshold policy with only type 2 vacations can be obtained by using (10) with $n = N$ and $V_1$ taken equal to $V_2$. For the two-threshold policy, the feasible $(n, N)$ combinations must satisfy $0 \leq n \leq N$. The case $n = N$ corresponds to a single-threshold policy with only type 1 vacations and has average cost denoted by $g_N^{(1)}$. For the exponential vacation case, finiteness of the search for the optimal $(n, N)$ is guaranteed in the following theorem.

**Theorem 4.1** Suppose $V_1$ and $V_2$ are exponentially distributed. Then (1) there is a finite computable $n_0$ such that when $n \geq n_0$, the optimal $N$ will equal $n$ and (2) given $n < n_0$, the global minimum of $g_{nN}$ occurs for $N = n + x^*(n)$, where $x^*(n)$ is the computable optimal $x$ in a $g_{0e}$ problem with $n$-dependent cost data.

**Proof:** Proof of (1): For $N = n \geq 0$, a possible sample path of the work process for a cycle period is shown in Figure 2. In this system, the server resumes serving the queue when he completes a type 1 vacation and first finds that the queue length $(Q)$ is greater than or equal to $n$. We select a special service order in which the first $n$ arrivals will not be served until all other subsequent arrivals have been served. Since this order is independent of service times and the service is nonpreemptive, it will lead to the same distribution for the queue length as that in the FCFS order. The average cost is also the same as that in FCFS order for the linear holding cost situation. Note that the cycle period can be decomposed into three parts by using the memoryless property of the exponential vacations. These are (a) the time $\tau_n$ needed to accumulate $n$ customers; (b) the period $\eta_1$ of time from the instant of the $n$th arrival to the instant of either a service or a type 1 vacation completion when the queue length is $n$; and (c) the busy period $\theta(n)$ of an M/G/1 queue which starts with $n$ customers. Actually, this system can be considered as a special case of the system in which the server takes $i$ type 2 vacations after he completes a type 1 vacation and first finds that $Q \geq n$. Our system corresponds to the case of $i = 0$.

Denote the long-run average cost of the system with $i$ type 2 vacations by $g_n(i) = TC_n(i)/T_n(i)$. We first prove the existence of a finite computable $n_0$ such that $n \geq n_0$. 

13
implies
\[ g_n(0) \leq g_n(i). \quad \text{for } i = 1, 2, \ldots. \]  \hspace{1cm} (34)

By the reward renewal theorem, we have
\[
g_{nn} = g_n(0) = \frac{r_0 + \bar{C}(\tau_n) + \bar{C}(\eta_1) + n h \bar{\eta}_1 + \sum_{i=1}^{n} C_i}{\bar{r}_n + \bar{\eta}_1 + n \bar{\theta}}
\]
\[= \frac{r'_0(n) + \bar{C}(\eta_1) + n h \bar{\eta}_1}{\bar{\eta}_1 + \alpha(n)}, \quad (35)
\]
where
\[
r'_0(n) = r_0 + \bar{C}(\tau_n) + \sum_{i=1}^{n} C_i,
\]
\[
\alpha(n) = \bar{r}_n + n \bar{\theta}.
\]

Here $\bar{C}(\tau_n)$ is the expected net total cost during $\tau_n$ and $\bar{C}(\eta_1)$ is the expected net total cost during $\eta_1$, excluding the expected holding cost of the customers present at the beginning of $\eta_1$; these are computed below. Now consider another system in which the server takes just one type 2 vacation after he completes a type 1 vacation and first finds that $Q \geq n$. Comparing the sample path of the work process of this system in Figure 3 with that in Figure 2, we see that the cycle period increases by a sub-cycle period $\eta_2$ starting with a type 2 vacation. The long-run average cost, $g_n(1)$ is given by
\[
g_n(1) = \frac{r'_0(n) + \bar{C}(\eta_1) + n h \bar{\eta}_1 + C(\eta_2) + (n + \lambda \bar{V}_1) h \bar{\eta}_2}{\bar{\eta}_1 + \alpha(n) + \bar{\eta}_2}.
\]  \hspace{1cm} (36)

We thus have $g_n(0) \leq g_n(1)$ if
\[ (\bar{C}(\eta_2) + (n + \lambda \bar{V}_1) h \bar{\eta}_2) (\bar{\eta}_1 + \alpha(n)) \geq (r'_0(n) + \bar{C}(\eta_1) + n h \bar{\eta}_1) \bar{\eta}_2. \]  \hspace{1cm} (37)

Note that
\[
\bar{C}(\tau_n) = \frac{1}{2 \lambda} h n (n - 1) - \frac{n}{\lambda} r_1,
\]
\[
\sum_{i=1}^{n} C_i = \frac{h \bar{\delta}}{2(1 - \rho)} n^2 + \left( \frac{\lambda \bar{S}(2)}{2(1 - \rho)^2} + \frac{\bar{\delta}}{2(1 - \rho)} \right) h n
\]
\[
\bar{\eta}_i = \frac{\bar{V}_i}{1 - \rho}, \quad \text{for } i = 1, 2.
\]
\[ C(\eta) = \frac{h\lambda}{1-\rho} \hat{V}_i^2 + \left( \frac{\lambda S(2)}{2(1-\rho)^2} \right) + \frac{\bar{S}}{1-\rho} \right) \lambda h \hat{V}_i - r_i \hat{V}_i, \quad \text{for } i = 1, 2. \]

\[ \alpha(n) = \frac{n}{\lambda(1-\rho)}. \]

Using the equations above to simplify (37), we thus have \( g_n(0) \leq g_n(1) \) if \( n \geq n_0 \). Here, \( n_0 \) is the smallest nonnegative integer such that

\[ an^2 + bn + c \geq 0 \]  

holds for all \( n \geq n_0 \), and

\[ a = \frac{h\lambda}{2\lambda(1-\rho)} > 0, \]

\[ b = \frac{h}{1-\rho} \hat{V}_i^2 + \left( \frac{h}{2\lambda(1-\rho)} + \frac{r_1 - r_2}{\lambda} \right) \hat{V}_i, \]

\[ c = \frac{h\lambda}{1-\rho} \hat{V}_i^2 + (r_1 - r_2)\hat{V}_i - r_0 \hat{V}_i. \]

We consider now a third system in which the server takes two type 2 vacations. Eq.(34) for \( i = 1 \) reads as

\[ \frac{TC_n(0)}{T_n(0)} \leq \frac{TC_n(0) + C(\eta_2) + (n + \lambda \hat{V}_1)h\eta_2}{T_n(0) + \eta_2}, \]  

or

\[ C(\eta_2) + (n + \lambda \hat{V}_1)h\eta_2 \geq g_n(0)\eta_2 \quad \text{for } n \geq n_0. \]

From the sample path in Figure 4, we have

\[ TC_n(2) = TC_n(0) + 2 \left( C(\eta_2) + (n + \lambda \hat{V}_1)h\eta_2 \right) + \lambda \hat{V}_2 h\eta_2 \]

\[ \geq TC_n(0) + 2g_n(0)\eta_2 + \lambda \hat{V}_2 h\eta_2 \]

\[ \geq g_n(0)(\bar{T}_n(0) + 2\bar{\eta}_2) = g_n(0)\bar{T}_n(2), \]  

where the second inequality holds because of (40). Eq. (41) gives

\[ g_n(0) \leq \frac{TC_n(2)}{\bar{T}_n(2)} = g_n(2) \quad \text{for } n \geq n_0. \]  

In the same way, we can prove that (34) holds for all \( i \).
Now consider a system with a two-threshold policy where $N > n \geq n_0$. Let $Y = \{0, 1, 2, \cdots\}$ be the number of type two vacations the server takes during the leave period and let $P_{nN}(Y = i)$ be the probability that $Y = i$. Then conditioning on $Y$, we obtain

$$
TC_{nN} = \sum_{i=0}^{\infty} E(TC_{nN}|Y = i)P_{nN}(Y = i)
$$

$$
\geq g_n(0) + \sum_{i=0}^{\infty} T_n(i)P_{nN}(Y = i)
$$

$$
= g_n(0)\bar{\theta}_{nN}, \quad (43)
$$

where the inequality follows because of (34). Note that (43) implies $g_{nN} = \bar{TC}_{nN}/\bar{\theta}_{nN} \geq g_n(0) = g_{nn}$. This completes the first part of the theorem.

Proof of (2): For $n < n_0$, a two-threshold policy with $N > n$ can be optimal. Again we decompose the cycle period illustrated in Figure 5 into three parts which are (a) $\tau_n$; (b) the sub-cycle period $\theta_{0,N-n}$ from the instant of the nth arrival to the instant of a service completion when the number of customers in the system becomes n again; and (c) $\theta(n)$. For a given $n$, the average costs during $\tau_n$ and $\theta(n)$ are constant and independent of $N$. The sub-cycle period is actually the cycle period of an M/G/1 queue with a single-threshold ($N-n$) policy and an exceptional (type 1) first vacation and subsequent type 2 vacations.

It is easy to see that the average cost during $\theta_{nN}$ can be written as

$$
g_{nN} = \frac{\tau_0 + \bar{C}(\tau_n) + \bar{C}(\theta_{0x}) + \sum_{i=1}^{n} C_i}{\tau_n + \bar{\theta}_{0x} + n\bar{\theta}}
$$

$$
= \frac{\tau_0(n) + \bar{C}(\theta_{0x})}{\theta_{0x} + \alpha(n)}
$$

$$
= \frac{g'_{0x}\bar{\theta}_{0x}}{\theta_{0x} + \alpha(n)} \quad (44)
$$

where $x = N - n > 0$, $\bar{C}(\theta_{0x})$ is the expected net total customer holding cost during $\theta_{0x}$ and $g'_{0x} = (\tau_0(n) + \bar{C}_{e1})/\bar{\theta}_{e1}$. Note that for a given $n$, $g_{0x}$ can be considered as the average cost for a system with a single-threshold policy and an exceptional type 1 first vacation.
We call this imaginary system the related single-threshold system (RSTS) with shutdown cost \( r_i'(n) \) and customer holding cost function \( H'(i, n) = (n + i)h \). Since Federgruen and So [4] have shown that there exists a finite optimal threshold \( x^*(n) \) for the RSTS, we have:

\[
g_{0x^*(n)}' - g_{0x}' \leq 0 \quad \text{for} \quad x \geq x^*(n) \tag{45}
\]

Then (44) implies that if \( N \geq N_0(n) = n + x^*(n) \) we have

\[
g_{nN_0(n)} - g_{nN} = \frac{g_{0x^*(n)}}{\theta_{0x^*(n)} + \alpha(n)} - \frac{g_{0x}'}{\theta_{0x} + \alpha(n)} \tag{46}
\]

Bring the terms in (46) over a common denominator, then use (45) and \( \theta_{0x^*(n)} \leq \theta_{0x} \) for \( x > x^*(n) \), to conclude that the right-hand-side of (46) is non-positive. Thus

\[
g_{nN_0(n)} - g_{nN} \leq 0 \quad \text{for} \quad N \geq N_0(n) = n + x^*(n). \tag{47}
\]

This implies that the optimal \( N \geq n \) does not exceed \( n + x^*(n) \).

**Remarks:**

1. Since the average cost in the single-threshold case is unimodal in the threshold value (see [7]), the optimal value of \( n = N \) for \( n \geq n_0 \) is easily found by a finite search.

2. Zhang [15] has found a computable upper bound \( x_0 \) for \( x^*(n) \) in the RSTS. In other words, \( x^*(n) \) can be obtained by finite search and we can test in any given problem whether the global optimum has been attained.

3. In extensive numerical computations, we always find that \( g_{0x}' \) is unimodal in \( x \geq 0 \). However, we cannot at present establish this result theoretically. Kella [7] has shown that the average cost for the system with a single-threshold policy and a single type vacation is a unimodal function of the threshold \( n \), and we might expect this property to hold also when the first vacation is of exceptional type. If this property could be established, we would know that the first local minimum is the global minimum and could more quickly locate \( x^*(n) \) for the RSTS.
In the case of exponentially distributed vacations, Theorem 4.1 justifies the following search procedure for finding the optimal policy. (For generally distributed vacations, we merely compute enough $g_{n,N}$ values to be “almost” sure that the minimum has not been missed.)

**Step 1** Find $n_0$ from eq. (38)

**Step 2** For each $n \leq n_0$, compute $N_0(n) = n + x^*(n)$ and find $N^*(n)$ as the value of $N$ that minimizes $g_{n,N}$ for $N \in \{n, \ldots, N_0(n)\}$. Let $g_{\alpha \beta}$ be the resulting minimum average cost, with $n = \alpha \leq n_0$ and $N = \beta \leq N_0(\alpha)$.

**Step 3** Compute $g_0, g_{N^*}^{(1)} = \min_N g_{N}^{(1)}$, and $g_{N^*}^{(2)} = \min_N g_{N}^{(2)}$, where $g_{N}^{(i)}$ is the average cost of a single-threshold $N$-policy with type $i$ vacations only. Find the overall minimum value

$$\gamma = \min\{ g_0, g_{N^*}^{(1)} , g_{N^*}^{(2)} , g_{\alpha \beta} \} \quad (48)$$

and the corresponding optimal policy.

**Note.** The optimal $g_{n,N}$ with $N = n$ and $n > n_0$ is found during the course of determining $g_{N^*}^{(1)}$.

In the next Section we present some numerical examples which illustrate the practicality of the approach.

### 5 Numerical Results

The search algorithm was coded in Borland C, and run on an IBM AT 80486 computer. To test the efficiency of our search procedure and to check the unimodality of $g_{n,N}$, we have made extensive numerical tests. Some of these examples (both exponential and general vacation cases) are presented in the following Tables.
Table 2: Data for Exponential Vacation Examples. Exp means exponential, \(E_2\) means Erlang-2, and Unif. means the uniform distribution on \([0.8,1.2]\) for service times.

<table>
<thead>
<tr>
<th>Example</th>
<th>(\lambda)</th>
<th>(S)</th>
<th>(F_s(.))</th>
<th>(V_1)</th>
<th>(V_2)</th>
<th>(h)</th>
<th>(r_0)</th>
<th>(r_1)</th>
<th>(r_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
<td>1.0</td>
<td>(E_2)</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>1.0</td>
<td>Unif.</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>4</td>
<td>0.7</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>5</td>
<td>0.9</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>1.5</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>7</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>1.5</td>
<td>1.0</td>
<td>2.0</td>
<td>20.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>8</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>25.0</td>
<td>12.0</td>
<td>10.0</td>
</tr>
<tr>
<td>9</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
<td>25.0</td>
<td>11.0</td>
<td>10.0</td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td>1.0</td>
<td>Exp</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>25.0</td>
<td>11.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>

Table 3: Results for Exponential Vacation Examples

<table>
<thead>
<tr>
<th>Example</th>
<th>(g_0)</th>
<th>(g_N^{(1)})</th>
<th>(N^*)</th>
<th>(g_N^{(2)})</th>
<th>(N^*)</th>
<th>(g_{\alpha,\beta})</th>
<th>((\alpha, \beta))</th>
<th>Opt. Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.000</td>
<td>2.725</td>
<td>2</td>
<td>2.815</td>
<td>2</td>
<td>2.637</td>
<td>(1, 2)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>2</td>
<td>2.550</td>
<td>2.275</td>
<td>2</td>
<td>2.365</td>
<td>2</td>
<td>2.187</td>
<td>(1, 2)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>3</td>
<td>2.175</td>
<td>1.900</td>
<td>3</td>
<td>1.990</td>
<td>2</td>
<td>1.812</td>
<td>(1, 2)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>4</td>
<td>4.670</td>
<td>5.617</td>
<td>1</td>
<td>5.363</td>
<td>2</td>
<td>5.442</td>
<td>(0, 2)</td>
<td>No Vacation</td>
</tr>
<tr>
<td>5</td>
<td>18.000</td>
<td>21.043</td>
<td>1</td>
<td>20.110</td>
<td>2</td>
<td>20.688</td>
<td>(0, 1)</td>
<td>No Vacation</td>
</tr>
<tr>
<td>6</td>
<td>2.250</td>
<td>1.219</td>
<td>2</td>
<td>1.573</td>
<td>2</td>
<td>1.238</td>
<td>(1, 2)</td>
<td>Type 1 ((N = 2))</td>
</tr>
<tr>
<td>7</td>
<td>3.000</td>
<td>2.345</td>
<td>2</td>
<td>2.815</td>
<td>2</td>
<td>2.426</td>
<td>(1, 2)</td>
<td>Type 1 ((N = 2))</td>
</tr>
<tr>
<td>8</td>
<td>3.000</td>
<td>3.100</td>
<td>2</td>
<td>3.277</td>
<td>2</td>
<td>3.047</td>
<td>(1, 2)</td>
<td>No Vacation</td>
</tr>
<tr>
<td>9</td>
<td>3.000</td>
<td>2.725</td>
<td>2</td>
<td>2.415</td>
<td>2</td>
<td>2.486</td>
<td>(1, 2)</td>
<td>Type 2 ((N = 2))</td>
</tr>
<tr>
<td>10</td>
<td>1.500</td>
<td>0.600</td>
<td>3</td>
<td>0.443</td>
<td>3</td>
<td>0.416</td>
<td>(2, 3)</td>
<td>(2, 3)</td>
</tr>
</tbody>
</table>

Table 4: Average Costs for Some Policies in Example 1.

<table>
<thead>
<tr>
<th>(N)</th>
<th>(g_{N}^{(2)})</th>
<th>(n)</th>
<th>(N)</th>
<th>(g_{nN})</th>
<th>(n)</th>
<th>(N)</th>
<th>(g_{nN})</th>
<th>(n)</th>
<th>(N)</th>
<th>(g_{nN})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8.200</td>
<td>0</td>
<td>0</td>
<td>4.160</td>
<td>1</td>
<td>1</td>
<td>2.782</td>
<td>2</td>
<td>2</td>
<td>2.725</td>
</tr>
<tr>
<td>1</td>
<td>3.200</td>
<td>0</td>
<td>1</td>
<td>2.940</td>
<td>1</td>
<td>2</td>
<td>2.637</td>
<td>2</td>
<td>3</td>
<td>2.998</td>
</tr>
<tr>
<td>2</td>
<td>2.815</td>
<td>0</td>
<td>2</td>
<td>2.713</td>
<td>1</td>
<td>3</td>
<td>2.989</td>
<td>2</td>
<td>4</td>
<td>3.569</td>
</tr>
<tr>
<td>3</td>
<td>3.200</td>
<td>0</td>
<td>3</td>
<td>3.083</td>
<td>1</td>
<td>4</td>
<td>3.617</td>
<td>2</td>
<td>5</td>
<td>4.315</td>
</tr>
<tr>
<td>4</td>
<td>3.852</td>
<td>0</td>
<td>4</td>
<td>3.723</td>
<td>1</td>
<td>5</td>
<td>4.392</td>
<td>2</td>
<td>6</td>
<td>5.161</td>
</tr>
</tbody>
</table>
Table 5: Optimal Two-Threshold Policies for Non-exponential Vacation Examples. \( \lambda = 0.6, \hat{S} = 1, F_S \sim \text{Exp}, \bar{V}_1 = 2.0, \bar{V}_2 = 1.0, h = 2.0, r_0 = 20, r_1 = 12, r_2 = 11. \) Det means deterministic and \( E_k \) means Erlang-\( k \) distributions for vacations.

<table>
<thead>
<tr>
<th>Example</th>
<th>( F_{V_1} )</th>
<th>( F_{V_2} )</th>
<th>( g_2, g ) ( \alpha, \beta )</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Det</td>
<td>Det</td>
<td>1.878 1 2</td>
<td>2.310</td>
</tr>
<tr>
<td>2</td>
<td>( E_2 )</td>
<td>( E_2 )</td>
<td>2.210 1 2</td>
<td>2.550</td>
</tr>
<tr>
<td>3</td>
<td>( E_4 )</td>
<td>( E_4 )</td>
<td>2.074 1 2</td>
<td>2.431</td>
</tr>
<tr>
<td>4</td>
<td>( E_4 )</td>
<td>( E_2 )</td>
<td>2.102 2 2</td>
<td>2.570</td>
</tr>
<tr>
<td>5</td>
<td>( E_2 )</td>
<td>( E_4 )</td>
<td>2.180 1 2</td>
<td>2.526</td>
</tr>
</tbody>
</table>

Remarks:

1. The problem data and numerical results for the exponential vacation cases are presented in Tables 2-3. There are three classes of policies in Table 4. The \( g^{(2)}_N \) column shows the average cost of the single-threshold policy for type 2 vacations alone, while \( g_{nn} \) and \( g_{nN} \) are, respectively, the average costs for the single-threshold policy with type 1 vacations only and the average cost of the genuine two-threshold policy when \( n < N \). Note that \( g_{0N} \) is the average cost in a multiple-vacation model with a “setup”, or exceptional first vacation.

2. Different service distributions were investigated in the first three examples. Although this difference results in different average operating costs, the location of the optimal threshold values is relatively insensitive to the service distribution change.

3. For most cases, the minimum average cost deviates from the next best value by between 2% and 22%. This indicates that some other, sub-optimal policies, can be the candidates for practical implementation.

4. It is interesting to note that the average operating costs in each column of the Table 4 have a single minimum point. Although at present we cannot prove theoretically the unimodality or convexity of the average cost function, extensive numerical tests have demonstrated these properties. If we could establish these properties, the search algorithm would be further simplified.
5. We have also examined some general vacation cases. Some of the examples (Erlang-k distributed and deterministic vacations) are presented in Table 5. The optimum is obtained by computing the expressions in Section 3 for numerous values of \( n \leq N \) and choosing the best result. Since a formal optimality test is not available for the case of generally distributed vacations, we simply calculate \( g_{nN} \) over a large range of \( n \) and \( N \), until the numerical values of \( g \) become consistently large and are consistently increasing with \( N \), thus making it apparent that the optimum occurs at smaller values of \( N \). (The average costs for the optimal single-threshold policies and no-vacation policy are not included. They can be obtained exactly by using the recursive scheme in [7].

6. A large number of cases with different parameters were studied using the problem solver based on our search procedure. These studies have shown that the search procedure is quite efficient. For most problems, the computation times are less than 5 seconds. Note that for exponentially distributed vacations an alternative approach for computing \( g_{nN} \) would be to use (44) instead of (10) or (14).

6 Summary and Suggestions for Further Research

We have discussed the two-threshold vacation policy in this paper and have developed the formulas for the performance measures such as the average queue length, the average waiting time, and the average cost. Although we did not find an explicit formula for the optimal values of \( n \) and \( N \), we found a practical computational approach to locate optimal policies. For the case of exponentially distributed vacation times the procedure incorporates explicit tests that permit us to know that the global minimum has been obtained. Our approach is appropriate for finding performance measures and optimal policies for all the cases in Table 1.

Additional theoretical work needs to be done as well. We have looked at two-threshold policies as plausible generalizations of the single-threshold policies used with single-vacation-type models. The question of their optimality remains open at this time. Another question
concerns unimodality properties of $g_nN$, which, if true, would simplify and accelerate the search for an optimum. It would also be desirable to extend the global optimality tests of Section 4 to the case of generally distributed vacation times.

Another direction for further research would be to extend the current analysis to the multiple-vacation, multiple-threshold case.

Acknowledgements

The authors would like to thank NSERC and MRCO support of this research. We are grateful to anonymous referees for valuable remarks and suggestions. The third author thanks Dr. I.J.B.F. Adan for stimulating discussions.

References


Figure 1: A sample path of the work process in an $M/G/1$ queue with two vacation types and a two-threshold policy ($n = 2, N = 4$).
Figure 2. A sample path of the work process in a system with $n = N = 3$ ($n = 3, i = 0$).
Figure 3: A sample path of the work process in a system with $n = N = 3$
($n = 3$, $i = 1$).
Figure 4: A sample path of the work process in a system with \( n = 3 \) and \( i = 2 \).
Figure 5 : Another way of decomposing of the sample path of the work process in an $M/G/1$ queue with two vacation types and a two threshold policy ($n = 2$, $N = 4$).