KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

The optimal allocation of server time slots over different classes of patients

Stefan Creemers, JeroenBeli±n and Marc Lambrecht

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 1122

The optimal allocation of server time slots over different classes of patients

Stefan Creemers[1,2], Jeroen Beliën[2,3], Marc Lambrecht[2]

[1]IESEG School of Management,
Rue de la Digue 3, 59000 Lille, France
Tel.: +33-3-20545892, Fax: +33-3-20574855
s.creemers@ieseg.fr

[2]Katholieke Universiteit Leuven, Faculty of Business and Economics,
Department of Decision Sciences and Information Management,
Research Center for Operations Management,
Naamsestraat 69, B-3000 Leuven, Belgium
Tel.: +32-16-326958, Fax: +32-16-326624
firstname.lastname@econ.kuleuven.be

[3]Hogeschool-Universiteit Brussel
Center for Modelling and Simulation,
Stormstraat 2, B-1000 Brussel, Belgium
jeroen.belien@hubrussel.be

Abstract

We present a model for assigning server time slots to different classes of patients. The objective is to minimize the total expected weighted waiting time of a patient (where different patient classes may be assigned different weights). A bulk service queueing model is used to obtain the expected waiting time of a patient of a particular class, given a feasible allocation of service time slots. Using the output of the bulk service queueing models as the input of an optimization procedure, the optimal allocation scheme may be identified. For problems with a large number of patient classes and/or a large number of feasible allocation schemes, a step-wise heuristic is developed. A common example of such a system is the allocation of operating room time slots over different medical disciplines in a hospital.

Keywords: Markov processes, OR in health services, Optimal capacity allocation, Bulk service queue, Patient waiting time

1 Introduction

In this article we present a model for assigning a set of predefined server time slots to a number of patient classes. For each patient class and each feasible allocation scheme, a bulk service queueing model can be analyzed. The output of the queueing models serve as the input of an optimization procedure. The optimization procedure minimizes the total expected weighted waiting time of a patient (different classes of patients are allowed to have different weights). For large problem sizes (i.e. problems with a large number of patient classes and/or a large number of feasible allocation schemes), a step-wise heuristic is developed.

The model is applicable to problem settings characterized by the following features:

- The capacity of the server can be divided into discrete blocks of time.
- The allocation of service time slots is a strategic, long-term decision (as such, the model does not deal with the day-to-day, operational allocation problem).
- Patients have to make an appointment in order to receive service in an upcoming service time slot. From the making of an appointment until their arrival at the service facility (e.g. the hospital), patients are introduced into a queue that is also referred to as the "waiting list".
- The waiting time of a patient is an important decision criterion.
- No interaction effect between patient classes takes place (i.e. patients of a particular class can only be served during the service time slots assigned to that class).

A typical example of such a system is the allocation of operating room time slots over different medical disciplines (where each medical discipline represents a patient class). In general, the daily capacity of each operating room may be divided into two service time slots: before noon and after noon (other divisions are of course possible). The allocation of these service time slots to medical disciplines is a strategic decision that has to be made by the hospital management (where the waiting time of patients is an important decision variable). The problem of allocating operating room capacity has been widely studied. Most studies however focus on the operational level of sequencing patients, leveling bed occupancy and optimizing the daily routine of hospital life (refer to Cardoen, Demeulemeester and Beliën (2010) for a recent review of the relevant literature). Only few analytic studies are dedicated to the strategic aspect of capacity allocation in hospitals. This is especially true when it comes to the study of waiting lists (i.e. the queue of patients who are waiting for their appointed date in order to receive treatment). The strategic importance of waiting lists has been stressed by a multitude of authors including Hanning (1996), Goddard and Tavakoli (1998), Besley, Hall and Preston (1999), Martin and Smith (1999), Vanden Bosch and Dietz (2000) and Rotstein and Alter (2006). Albeit some noteworthy works by Worthington (1987) and Green (2008) among others, few analytic efforts have been pledged to address the problem of waiting lists. The model presented in this article addresses the long-term waiting of patients and optimizes its trade-off with the allocation of hospital resources. Next to applications in health care, the model may be adopted to any setting in which a shared resource has to be allocated among a number of competing users. Examples include:

- The allocation of shared class rooms over various faculties at a university.
- The allocation of shelf space in supermarkets.
- . . .

The contribution of this article is threefold: (1) we present a new bulk service queueing model that allows the study of the queueing behavior of a single class of patients that receives service during predefined service time slots; (2) we devise an optimization procedure that allows to determine the optimal allocation (with respect to patient waiting time) of server time slots over a number of patient classes; (3) we develop a step-wise heuristic to cope with large problem sizes. The remainder of this article is organized as follows. Section 2 discusses the bulk service queue that is used to model the queueing behavior of a single patient class that receives service during a given set of time slots. The optimization procedure and the step-wise heuristic are presented in Section 3. Section 4 provides a numerical example and Section 5 concludes.

2 Bulk service queueing model

The bulk service queueing model presented in this article observes the queueing behavior of a single class of patients that is allowed to receive service during a given set of service time slots. Upon the start of such a service time slot, a number of patients is removed from the queue (e.g. the waiting list) in order to receive service at the service facility (e.g. the hospital). The number of patients is either: (1) the number of patients in the queue; (2) the maximum number of patients allowed to receive service during the upcoming service time slot. The model does not observe the time spent waiting for service at the service facility (e.g. the time spent waiting inside the hospital) nor the actual service process itself (refer to Creemers and Lambrecht (2009a) and Creemers (2009c) for models in which these elements are incorporated). Note that these latter elements are of lesser importance when observing the waiting list.

The arrival of patients occurs during arrival time slots (e.g. patients are allowed to make an appointment on working days, during office hours). The characterization of the arrival process during each of the arrival time slots is allowed to differ (i.e. the model allows for time-dependent arrival patterns). Note that the continuous-time case (in which arrivals are allowed to occur at any moment in time) corresponds to the setting in which only a single arrival time slot is assumed.

The main objective of the bulk service queueing model is to assign arriving patients to the first appropriate service time slot (i.e. the first service time slot in which the appropriate class of patients is allowed to receive service) that has capacity available. The properties of the bulk service queueing model may be summarized as follows:

- Patients arrive during arrival time slots.
- Patients are removed from the queue only at the start of a service time slot.
- Patients receive service in the first upcoming service time slot in which capacity is still available.
- Patients are removed instantaneously, in batches and according to a FCFS policy.

- The number of patients removed depends on the current queue size and the maximum number of patients that is allowed to receive service during the upcoming service time slot.

The server operates under a vacation policy. The dynamics of the vacation mechanism are as follows. Upon the start of an appropriate service time slot, the server returns from vacation, instantaneously removes up to a maximum number of patients from the queue and departs on a vacation once more.

Bulk service queues have been receiving a lot of attention during the past decades. For a general reference work on bulk service queues, refer to Chaudhry and Templeton (1983). More recent advances have been made by Gupta and Goswamib (2002), Chaudhry and Gupta (2003), Gupta and Karabi Sikdar (2004), Janssen and van Leeuwaarden (2005), Banik, Gupta and Chaudhry (2009) and Tadj and Abid (2009) among others. Within the literature on vacation models a wide variety of models exist (refer to Doshi (1986), Takagi (1988) and Tian and Zhang (2006) for a general overview). The model presented in this article possesses some unique features, rendering the modeling exercise rather complex:

- Arrival time slots have a unique characterization (i.e. each arrival time slot is allowed to have its own length and patient interarrival distribution).

- The interarrival times of patients follow an i.i.d. phase-type (PH) distribution.

- At the start of each service time slot, there is a maximum number of patients that is removed. As such, the bulk service queueing model has a k-limited service discipline. Note that service itself (i.e. the removal of up to k patients from the waiting list) occurs instantaneously.

- The maximum number of patients removed, depends on the service time slot that is about to start (e.g. the maximum number of patients served during a service time slot on Thursday is allowed to differ from the maximum number of patients served during a service time slot on Friday). As such, the model features time-dependent values of k.

- A vacation is initiated at the following time instances: (1) the start of a service time slot (after the removal of up to k patients); (2) the start of an arrival time slot; (3) the end of an arrival time slot. Since these time instances are fixed moments in time, the intervals in between (i.e. the vacation durations) are of fixed (i.e. deterministic) length as well.

- The deterministic length of a vacation depends on the moment at which the vacation is initiated (e.g. a vacation initiated after a service time slot on Thursday is allowed to have a different length compared to a vacation that is initiated after a service time slot on Friday).

To summarize, we have a bulk service queueing model featuring: (1) arrival time slots that have a unique characterization; (2) a k-limited service discipline; (3) vacations of deterministic length; (4) time-dependent values of k as well as time-dependent vacation lengths. Similar models may be found in Creemers and Lambrecht(2009a), Creemers and Lambrecht (2009b). In this article, we further refine the model presented in Creemers and Lambrecht (2009b) and adopt it in a multi-class setting (refer to Section 3). The refinement concerns a complete overhaul of one of the submodels such that: (1) computational efforts are reduced; (2) numerically exact results are obtained (whereas previously results were approximative).

The performance measure of interest (i.e. the input of the total expected weighted waiting time of a patient) is the expected waiting time of a patient of a particular class. Building on Markov chain theory and through the use of PH distributions, matrix analytical methods and efficient algorithms, we obtain numerically exact results. The validity of these results are supported by simulation studies. Computational experiments show that real-life systems may be assessed (refer to Creemers (2009c) for an assessment of the validity and the computational performance of the bulk service queueing model). In the remainder of this section, we define the basic processes that govern the bulk service queueing model, we characterize the PH distributions that are used to model system processes and establish a counting process to determine the distribution of the number of arriving patients during a vacation. Next we present the bulk service queueing model itself.

2.1 Basic Processes

The service process of a given class of patients is a succession of service time slots during which patients are served. Each service time slot i_s is characterized by the maximum number of patients k_{i_s} allowed to receive service. We assume recurring cycles to be present in the succession of service time slots (e.g. the use of a specific operating room is assigned to the orthopaedics

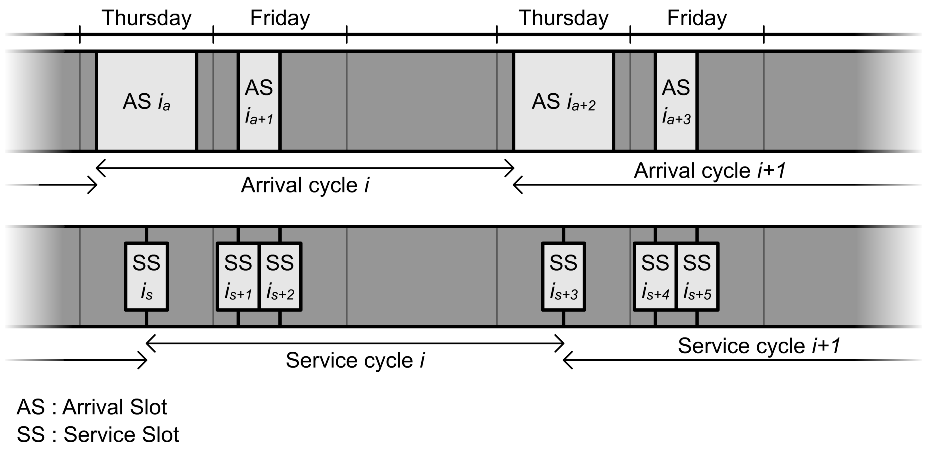

Figure 1: Cyclic nature of the service and arrival process

department every Thursday and Friday afternoon). A cycle of service time slots has length T_{c_s}.

Similarly to the service process, the arrival process is a succession of arrival time slots i_a during which patients of a particular class are allowed to arrive. An arrival time slot i_a is fully characterized by: (1) the length T_{i_a}; (2) the mean interarrival time $\lambda_{i_a}^{-1}$; (3) the variance of interarrival times $\sigma_{i_a}^2$. We assume recurring cycles to be present in the succession of arrival time slots. A cycle of arrival time slots has length T_{c_a}. An illustration of the cyclic nature of service and arrival processes is provided in Figure 1. In building the bulk service queueing model, we will fully exploit the repetitive structure of the service and arrival processes.

Note that we allow the arrival and the service cycle to differ because the arrival of a patient of a particular class does not necessarily coincide with the service time slots that have been assigned to that patient class (e.g. an orthopaedic patient might call to make an appointment on all working days, during office hours whereas the orthopaedics department only performs surgery on Thursday and on Friday afternoon). As such, our model is a generalization of the more common setting in which arrivals are only allowed to take place during the time that service is allowed to take place.

The vacation process is obtained when superimposing both the service and the arrival process. The vacation process is the continuous (i.e. uninterrupted) succession of vacations i_v, of deterministic length T_{i_v}. A new

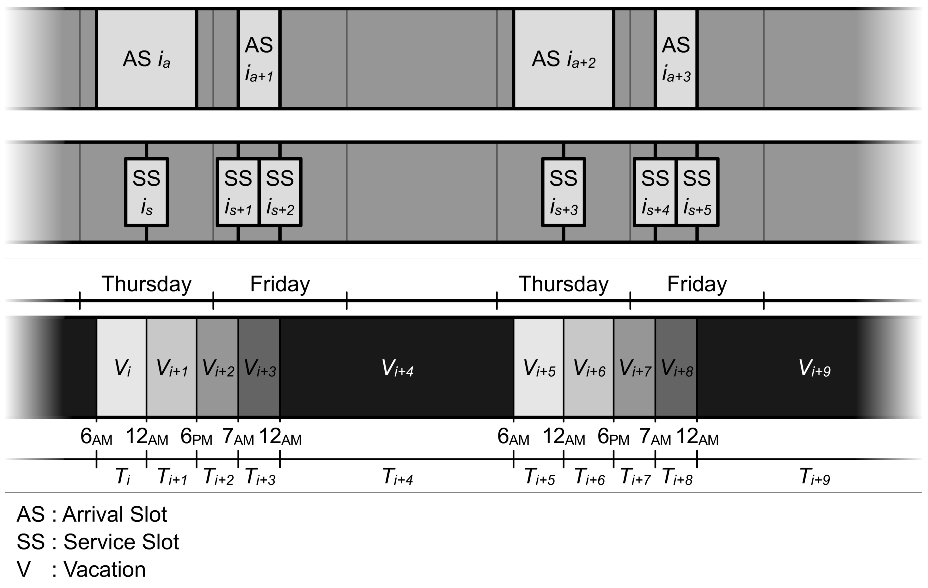
AS i_a
AS i_{a+1}
AS i_{a+2}
AS i_{a+3}
SS i_s
SS i_{s+1}
SS i_{s+2}
SS i_{s+3}
SS i_{s+4}
SS i_{s+5}
Thursday
Friday
Thursday
Friday
V_i V_{i+1} V_{i+2} V_{i+3} V_{i+4} V_{i+5} V_{i+6} V_{i+7} V_{i+8} V_{i+9}
6AM 12AM 6PM 7AM 12AM 6AM 12AM 6PM 7AM 12AM
T_i T_{i+1} T_{i+2} T_{i+3} T_{i+4} T_{i+5} T_{i+6} T_{i+7} T_{i+8} T_{i+9}
AS : Arrival Slot
SS : Service Slot
V : Vacation

Figure 2: The vacation process at the bulk service queueing model

vacation i_v is initiated at each instance in time at which (1) a service time slot starts; (2) an arrival time slot starts; (3) an arrival time slot ends. This observation is used to determine T_{i_v}. We illustrate this procedure in Figure 2. Because service and arrival processes are assumed to be cyclic, the vacation process is cyclic as well. The cycle length of the vacation process T_{c_v} equals the least common multiple of T_{c_s} and T_{c_a} (assuming the ratio of T_{c_s} and T_{c_a} is a rational number). A cycle of vacations contains J vacations (for the remainder of the text, index j is defined as $j \in \{1, 2, \ldots, J\}$). We illustrate these principles in Figure 3. Note that, due to the cyclic nature of the vacation process, a vacation of type $(j + (iJ))$ is also a vacation of type j (for the remainder of the text, index i is defined as $i \in \{0, 1, \ldots\}$). In addition, vacations may be divided into different classes (e.g. arrivals are allowed to take place only during vacations of a particular class). A definition of the different vacation classes is presented in Section 2.4.1.

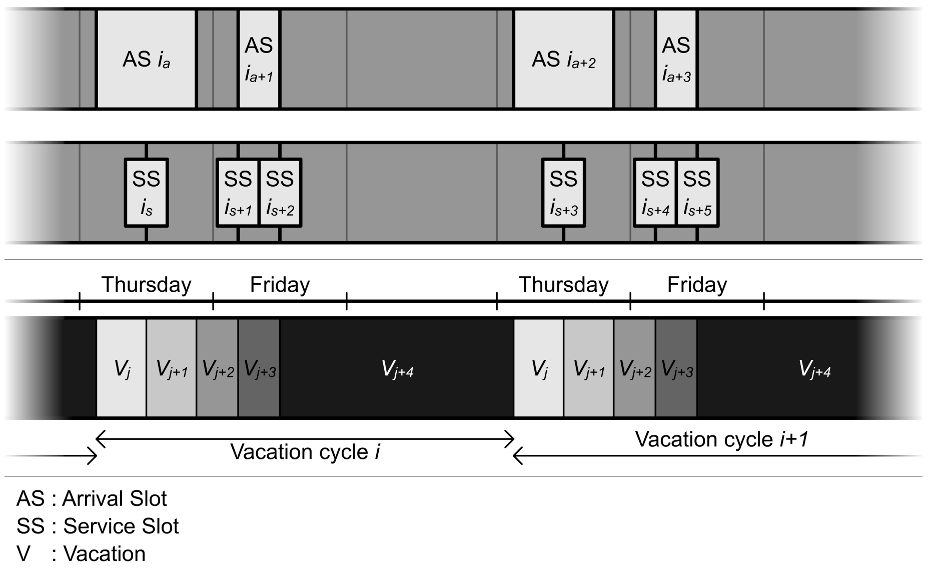

Figure 3: Cyclic nature of the vacation process

2.2 Phase-Type Distributions

In order to model a general i.i.d. arrival process and a deterministic vacation process, we adopt continuous-time PH distributions. Continuous-time PH distributions use exponentially distributed building blocks to approximate (with arbitrary precision) any positive-valued continuous distribution. PH distributions are widely implemented in the queueing literature. For a review on the literature and an introduction on PH distributions we refer to Neuts (1981), Latouche and Ramaswami (1999) and Osogami (2005) among others. A PH distribution is the distribution of time until absorption in a Markov chain with absorbing state 0 and state space $\{0, 1, \ldots, \zeta, \zeta + 1\}$. It is fully characterized by parameters $\boldsymbol{\tau}$ and $\mathbf{Z}$. $\boldsymbol{\tau}$ is the vector of initial probabilities to start the process in any of the $(\zeta + 1)$ transient states and $\mathbf{Z}$ is the matrix containing the transition rates between transient states. The infinitesimal generator of the Markov chain representing the PH distribution is presented below:

$$\mathbf{Q} = \begin{vmatrix} 0 & \mathbf{0} \\ \mathbf{t} & \mathbf{Z} \end{vmatrix},$$

where $\mathbf{0}$ is a matrix of appropriate dimension containing only zeros and ($\mathbf{t} = -\mathbf{Ze}$) (with $\mathbf{e}$ a vector of ones of appropriate size).

In this article we adopt simple PH approximations of the arrival process. Whereas a multitude of approximations are available (ranging from very simple procedures to complex algorithms), we limit ourselves to the matching of the first two moments of the interarrival time distribution (i.e. $\lambda_{i_a}^{-1}$ and $\sigma_{i_a}^2$; the respective mean and variance of the interarrival time distribution are matched by the PH distribution). For notational convenience, let λ_j^{-1} and σ_j^2 denote the mean and variance of the interarrival time distribution of arrivals during a vacation of type j. The two-moment matching procedure developed in this section minimizes M_j, the number of phases required to approximate the arrival process at a vacation of type j. Define $\mathbf{M}_j$; the set containing the different arrival phases of the arrival process at a vacation of type j (as such, ($|\mathbf{M}_j| = M_j$)). Of course, if no arrivals are allowed to occur during a vacation of type j, ($M_j = 0$) and ($\mathbf{M}_j = \{\emptyset\}$). If arrivals are allowed to occur, we make a distinction between three cases: (1) ($C_j^2 = 1$); (2) ($C_j^2 > 1$); (3) ($C_j^2 < 1$) (where ($C_j^2 = \sigma_j^2 \lambda_j^2$) denotes the squared coefficient of variation of interarrival times at a vacation j). This distinction allows us to minimize the number of phases required to match the first two moments of any positive-valued continuous distribution. In the first case, a simple exponential distribution of parameter λ_j suffices to approximate the arrival process. $\boldsymbol{\tau}_j$ and $\mathbf{Z}_j$ are given by:

$$\boldsymbol{\tau}_j = 1\,, \quad \mathbf{Z}_j = -\lambda_j.$$

In the second case, we model the arrival process using a convex mixture of 2 exponential distributions (i.e. using a hyper-exponential distribution). The parameters of the hyper-exponential distribution matching the interarrival time distribution with rate λ_j and variance σ_j^2 are given by:

$$\upsilon_{j_1} = \frac{2}{2 - C_j^4 + C_j^6}, \tag{1}$$

$$\upsilon_{j_2} = 1 - \upsilon_{j_1}, \tag{2}$$

$$\lambda_{j_1}^{-1} = \frac{1}{2\lambda_j}\left(2 - C_j^2 + C_j^4\right), \tag{3}$$

$$\lambda_{j_2}^{-1} = \frac{1}{\lambda_j} - \frac{1}{\lambda_j C_j^2}, \tag{4}$$

where υ_{j_1}, υ_{j_2}, λ_{j_1} and λ_{j_2} denote the probability of having an interarrival time that is exponentially distributed with parameter λ_{j_1}, the probability of having an interarrival time that is exponentially distributed with parameter λ_{j_2}, the parameter of the first exponential distribution and the parameter of the second exponential distribution respectively. $\boldsymbol{\tau}_j$ and $\mathbf{Z}_j$ are defined as:

$$\boldsymbol{\tau}_j = \begin{array}{c|c|} 1 & \upsilon_{j_1} \\ 2 & \upsilon_{j_2} \end{array}, \quad \mathbf{Z}_j = \begin{array}{c|cc|} & 1 & 2 \\ 1 & -\lambda_{j_1} & 0 \\ 2 & 0 & -\lambda_{j_2} \end{array}.$$

With respect to the third case, we model the arrival process using a hypoexponential distribution (a series of exponential distributions whose parameters are allowed to differ; a generalization of the Erlang distribution). The parameters of the hypoexponential distribution matching the interarrival time distribution with rate λ_j and variance σ_j^2 are given by:

$$\zeta_j = \lfloor C_j^{-2} \rfloor - \lfloor C_j^2 \lfloor C_j^{-2} \rfloor \rfloor, \tag{5}$$

$$\lambda_{j_1}^{-1} = \frac{\frac{\zeta_j}{\lambda_j} + \sqrt{-\frac{\zeta_j}{\lambda_j^2} + \frac{C_j^2 \zeta_j}{\lambda_j^2} + \frac{C_j^2 \zeta_j^2}{\lambda_j^2}}}{\zeta_j + \zeta_j^2}, \tag{6}$$

$$\lambda_{j_2}^{-1} = \frac{1}{\lambda_j} - \frac{z_j^2}{\lambda_j\left(\zeta_j + \zeta_j^2\right)} - \frac{\zeta_j \sqrt{-\frac{\zeta_j}{\lambda_j^2} + \frac{C_j^2 \zeta_j}{\lambda_j^2} + \frac{C_j^2 \zeta_j^2}{\lambda_j^2}}}{\zeta_j + \zeta_j^2}, \tag{7}$$

where ζ_j, λ_{j_1} and λ_{j_2} denote the number of phases of exponential duration of parameter λ_{j_1} that occur prior to the last phase, the parameter of the exponentially distributed interarrival times at the first ζ_j phases, the parameter of the exponentially distributed interarrival time at the last phase. $\boldsymbol{\tau}_j$ and $\mathbf{Z}_j$ are presented below:

$$\boldsymbol{\tau}_j = \begin{array}{c|c|} 1 & 1 \\ 2 & 0 \\ \vdots & \vdots \\ \zeta_j & 0 \\ \zeta_j + 1 & 0 \end{array}, \quad \mathbf{Z}_j = \begin{array}{c|ccccc|} & 1 & 2 & \cdots & \zeta_j & \zeta_j + 1 \\ 1 & -\lambda_{j_1} & \lambda_{j_1} & \cdots & 0 & 0 \\ 2 & 0 & -\lambda_{j_1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \zeta_j & 0 & 0 & \cdots & -\lambda_{j_1} & \lambda_{j_1} \\ \zeta_j + 1 & 0 & 0 & \cdots & 0 & -\lambda_{j_2} \end{array}.$$

For the three cases, M_j equals 1,2 and $\zeta_j + 1$ respectively. A summary of the PH distributions, used in this article, is provided in Figure 4.

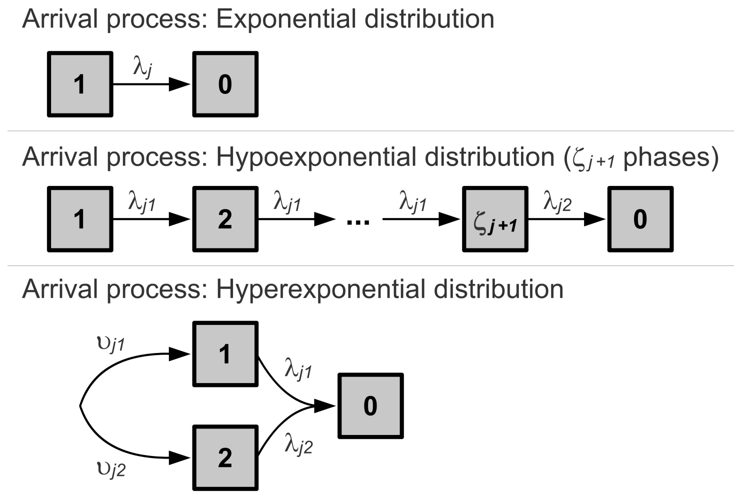
Arrival process: Exponential distribution
1
λj
0
Arrival process: Hypoexponential distribution (ζj+1 phases)
1
λj1
2
λj1
...
λj1
ζj+1
λj2
0
Arrival process: Hyperexponential distribution
υj1
υj2
1
2
λj1
λj2
0

Figure 4: Overview of PH distributions used at the bulk service queueing model

2.3 Counting process

A counting process is established in order to obtain the distribution of the number of patients arrived during a vacation of type j. The counting process developed here builds on the insights presented in Ramaswami (1988). Define $P_j[i, d|0, c]$ as the probability of having i arrivals during a vacation of type j and an arrival process at final phase d ($d \in \mathbf{M}_j$) given: (1) a PH distribution of parameters $\mathbf{Z}_j$ and $\boldsymbol{\tau}_j$; (2) an arrival process at initial phase c ($c \in \mathbf{M}_j$). The distribution of the number of arrivals may be obtained through a counting process of the MAP (Markovian Arrival Process) characterized by $\mathbf{C}_{j_0}$ and $\mathbf{C}_{j_1}$. The counting process has a continuous-time rate matrix:

$$\mathbf{Q}_j = \left| \begin{array}{ccccc} \mathbf{C}_{j_0} & \mathbf{C}_{j_1} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{C}_{j_0} & \mathbf{C}_{j_1} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_{j_0} & \mathbf{C}_{j_1} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}_{j_0} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \ddots \end{array} \right|,$$

where $(\mathbf{C}_{j_0} = \mathbf{Z}_j)$ and $\left(\mathbf{C}_{j_1} = \mathbf{t}_j \boldsymbol{\tau}_j^\top\right)$. The transition probabilities of the counting process during a vacation of deterministic length T_j are given by:

$$\mathbf{C}_j\left(T_j\right) = e^{T_j \mathbf{Q}_j} = \sum_{i=0}^{\infty} \frac{T_j^i}{i!} \mathbf{Q}_j^i. \tag{8}$$

In order to avoid numerical problems and to enhance computational performance, we apply a uniformization argument to the counting process. More specifically, define:

$$\mathbf{P}_j = \frac{\mathbf{Q}_j}{\lambda_j^{max}} + \mathbf{I}, \tag{9}$$

where $\mathbf{I}$ is an identity matrix of appropriate dimension and λ_j^{max} is the largest rate of the PH distribution of parameters $\mathbf{Z}_j$ and $\boldsymbol{\tau}_j$. Building on the insights presented in Tijms (2003), we obtain:

$$\mathbf{C}_j\left(T_j\right) = e^{-T_j \lambda_j^{max}} \sum_{i=0}^{\infty} \frac{(T_j \lambda_j^{max})^i}{i!} \mathbf{P}_j^i. \tag{10}$$

Since we are only interested in transitions moving from states with queue size zero, we only need to observe the first block row of $\mathbf{C}_j\left(T_j\right)$. More specifically, the first block row of $\mathbf{C}_j\left(T_j\right)$ holds the distribution of the number of arrivals during a vacation of type j (i.e. probabilities $P_j[i, d|0, c]$). In order to obtain the first block row of $\mathbf{C}_j\left(T_j\right)$, it suffices to compute $\mathbf{P}_{j_1}^{(i)}$; the first block row of $\mathbf{P}_j^i$ $(\forall i \geq 0)$. $\mathbf{P}_{j_1}^{(i)}$ may be obtained through the simple recursive relationship:

$$\mathbf{P}_{j_1}^{(i)} = \left(\frac{\mathbf{C}_{j_0}}{\lambda_j^{max}} + \mathbf{I}\right) \mathbf{P}_{j_1}^{(i-1)} + \begin{bmatrix} \mathbf{0} & \frac{\mathbf{C}_{j_1}}{\lambda_j^{max}} \mathbf{P}_{j_1}^{(i-1)} \end{bmatrix}. \tag{11}$$

2.4 Model

The bulk service queueing model presented in this article is not a straightforward queueing model. One possible approach would be to construct a Markov chain of four dimensions: (1) the queue size $Q : Q \in \{0, 1, 2, \ldots\}$; (2) the vacation type j; (3) the phase of the arrival process $m : m \in \{1, \ldots, M_j\}$; (4) the phase of the vacation process $v : v \in \{1, \ldots, V\}$.

Unfortunately, the use of multidimensional Markov chains is in general not advisable since it is clear that, as J, M_j or V increase, the resulting statespace grows rapidly. When modeling real life problems, memory- and computational constraints are quickly met. In order to efficiently assess performance measures, we decompose the systems into two subsystems:

- A first subsystem observes the queueing process of patients only at the start of a vacation of type j, prior to the removal of up to k_j patients. We use a set of DTMC X ($X = \{X_1, \dots, X_J\}$ and $X_j = \{X_j\,(t) : t \geq 0\}$) to analyze this first subsystem; where DTMC X_j may be defined as a two-dimensional stochastic process whose statespace can be represented by pairs $(Q, m)_j$. From the analysis of the DTMC X_j, we obtain $\mathcal{Q}_j^{\diamond}$; the expected number of patients in queue during a vacation of type j, given that these patients did already arrive prior to vacation j.

- A second subsystem observes the queueing process of those patients who arrive during a vacation of type j. Using simple arithmetics, we obtain $\mathcal{Q}_j^{p}$; the expected number of patients in queue during a vacation of type j, given that these patients did arrive during vacation j.

We illustrate this decomposition in Figure 5. Decomposing the system significantly improves computational efficiency due to: (1) dimensional reduction of the statespace; (2) avoiding unnecessary computations.

The DTMC X_j observes the queueing behavior of patients at the start of a vacation of type j prior to the removal of up to k_j patients from the queue. Therefore, observation moments coincide with the start of a vacation. The actions taking place in between two successive observation moments (i.e. the start of a vacation of type j and the start of the next vacation of type j) are left unobserved. We refer to Figure 6 for an illustration. In order to take the unobserved alterations of the queueing process into account, one needs to compute all possible outcomes (i.e. resulting queue sizes) and their corresponding probabilities. More specifically, one wants to know the probability of having t patients in the queue at the beginning of a vacation of type j, given that, at the beginning of the previous vacation of type j, s patients were present in the queue. More formally, define $P_{j,n}[t, d|s, c]$ as the probability of moving from a state with queue size s and arrival phase c at the start of a vacation j towards a state with queue size t and arrival phase d at the start of vacation $(j + n)$ ($c \in \mathbf{M}_j \wedge d \in \mathbf{M}_{j+n}$). In order to obtain $P_{j,J}[t, d|s, c]$ (i.e. the transition probabilities of the DTMC X_j), we first determine $P_{j,1}[t, d|s, c]$ by means of a counting process outlined in Section

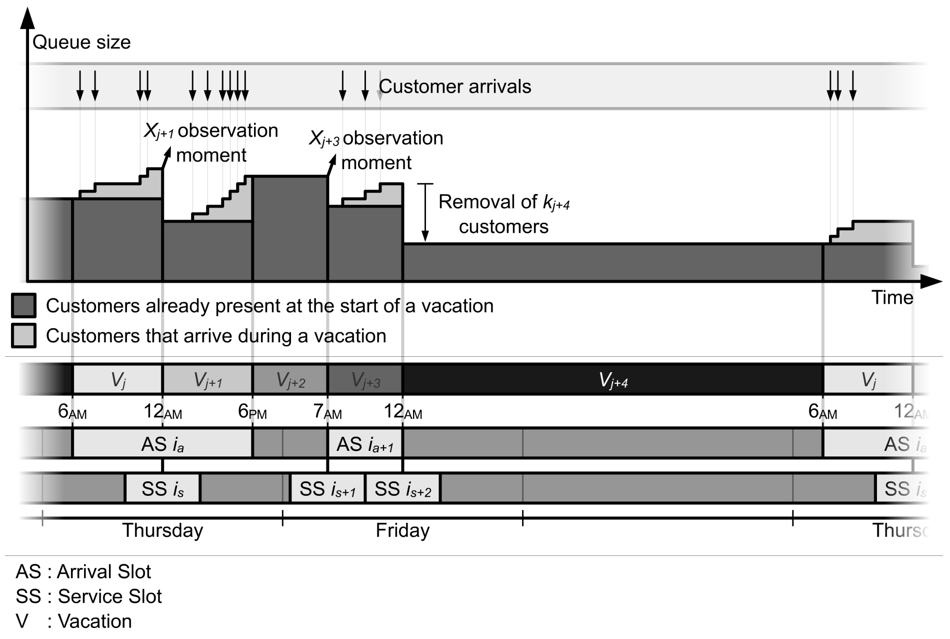

Figure 5: System decomposition at the bulk service queueing model

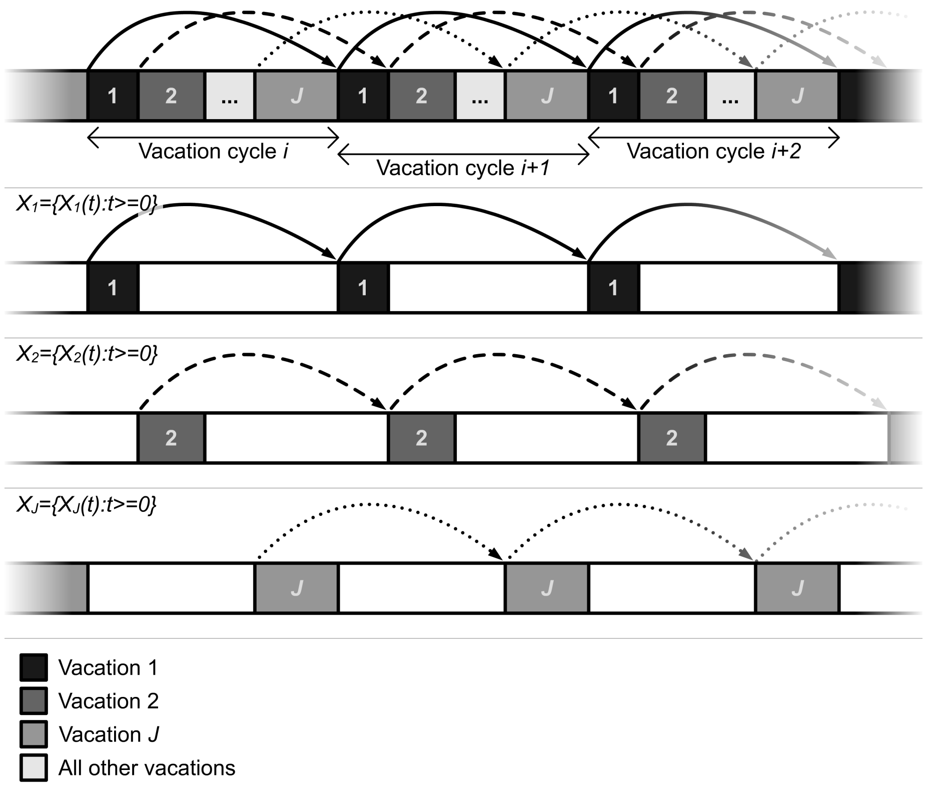

Figure 6: The set of DTMC X

2.3. These latter probabilities serve as the input of an iterative algorithm that yields the required transition probabilities $P_{j,J}[t, d|s, c]$.

In what follows we first provide a classification of the different vacation classes and show how to obtain $P_{j,1}[t, d|s, c]$ for each vacation class. Next we propose an algorithm to compute probabilities $P_{j,J}[t, d|s, c]$ and use these probabilities to construct the DTMC X_j. We use matrix analytical methods to obtain the stationary distribution of the number of patients in the queue at the start of a vacation of type j, prior to the removal of up to k_j patients from the queue (this distribution may be used to determine $\mathcal{Q}_j^{\diamond}$). In addition, we present an alternative algorithm that further improves computational performance. Next, we develop the arithmetics required to compute the expected number of patients in queue during a vacation of type j (i.e. $\mathcal{Q}_j^{*}$). Finally, we aggregate the performance measures (i.e. $\mathcal{Q}_j^{\diamond}$ and $\mathcal{Q}_j^{*}$) to obtain general results.

2.4.1 A Classification of Vacations

A distinction between five classes of vacations may be made:

- Class 1 vacations coincide with the end of an arrival time slot but do not coincide with the start of a service time slot.
- Class 2 vacations coincide with the start of a service time slot, do not coincide with the start of an arrival time slot and no arrival session is still in progress.
- Class 3 vacations coincide with the start of an arrival time slot but do not coincide with the start of a service time slot.
- Class 4 vacations coincide with both the start of an arrival time slot and the start of a service time slot.
- Class 5 vacations coincide with the start of a service time slot, do not coincide with the start of an arrival time slot and an arrival session is still in progress.

We do not allow for (multiple) adjacent arrival time slots because of the limitations involved with the PH distributions adopted in this article. Multiple adjacent arrival time slots however, might be used to take into account additional time-dependencies in the arrival process. The different classes of

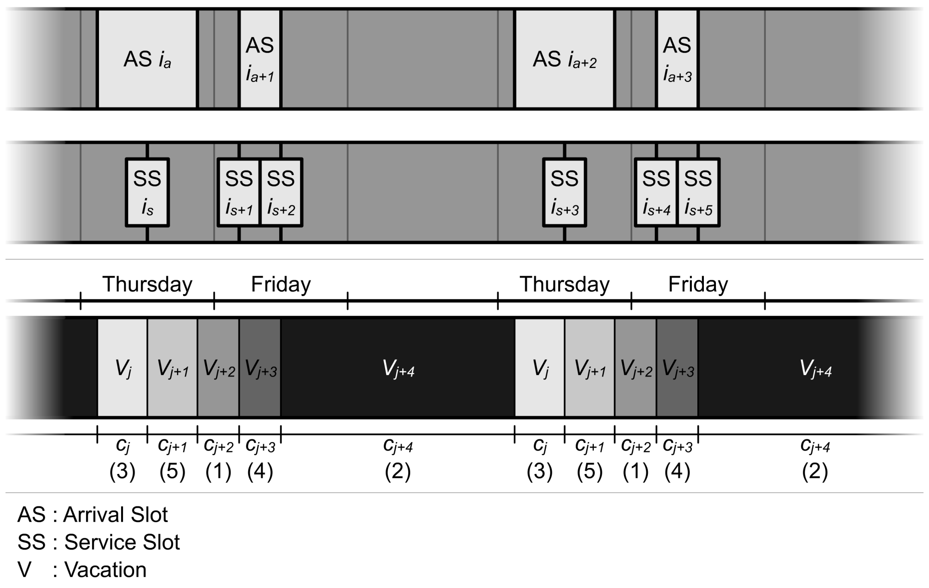
AS i_a
AS i_{a+1}
AS i_{a+2}
AS i_{a+3}
SS i_s
SS i_{s+1}
SS i_{s+2}
SS i_{s+3}
SS i_{s+4}
SS i_{s+5}
Thursday
Friday
Thursday
Friday
V_j V_{j+1} V_{j+2} V_{j+3} V_{j+4}
V_j V_{j+1} V_{j+2} V_{j+3} V_{j+4}
c_j (3) c_{j+1} (5) c_{j+2} (1) c_{j+3} (4) c_{j+4} (2)
c_j (3) c_{j+1} (5) c_{j+2} (1) c_{j+3} (4) c_{j+4} (2)
AS : Arrival Slot
SS : Service Slot
V : Vacation

Figure 7: Overview of different vacation classes

vacations are illustrated in Figure 7. Let $c_j : c_j \in \{1,2,3,4,5\}$ denote the class of a vacation j. Depending on the class, a vacation j is characterized by:

- a deterministic length T_j $(\forall j : c_j \in \{1,2,3,4,5\})$,
- a maximum number of patients k_j that is served instantaneously at the start of a vacation j $(\forall j : c_j \in \{2,4,5\}$ and $(k_j = 0)$ for all $j : c_j \in \{1,3\})$,
- a mean interarrival time λ_j^{-1} and variance σ_j^2 $(\forall j : c_j \in \{3,4,5\})$. Note that $(M_j = 0)$ and $(\mathbf{M}_j = \{\emptyset\})$ for all vacations $j : c_j \in \{1,2\}$.

Each of the vacation classes requires a distinct modeling approach.

Class 1 Vacation

Since no arrivals nor service takes place, the queueing process remains unaltered during class 1 vacations. If vacation $(j+1)$ is of class 2, we have (note that a vacation j of class $c_j \in \{1,2\}$ can never be succeeded by a vacation $(j+1)$ of class $c_{j+1} \in \{1,5\}$):

$$P_{j,1}[t,\emptyset|s,\emptyset] = \begin{cases} 1 & \forall s,t : s=t, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

For $c_{j+1} \in \{3,4\}$, we have:

$$P_{j,1}[t,d|s,\emptyset] = \begin{cases} \boldsymbol{\tau}_{j+1}(d) & \forall s,t : s=t, \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where $\boldsymbol{\tau}_{j+1}(d)$ indicates the probability of starting the arrival process of a vacation $(j+1)$ at an arrival phase d (refer to Section 2.2 for a definition of $\boldsymbol{\tau}_j$).

Class 2 Vacation

Since no arrivals are allowed to take place, the queueing process remains unaltered during class 2 vacations. However, at the start of a class 2 vacation j, a maximum of k_j patients is removed from the queue. If $(c_{j+1}=2)$, we have:

$$P_{j,1}[t,\emptyset|s,\emptyset] = \begin{cases} 1 & \forall s,t : s > k_j \wedge t = s-k_j, \\ 1 & \forall s,t : s \le k_j \wedge t = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

For $c_{j+1} \in \{3,4\}$, we have:

$$P_{j,1}[t,d|s,\emptyset] = \begin{cases} \boldsymbol{\tau}_{j+1}(d) & \forall s,t : s > k_j \wedge t = s-k_j, \\ \boldsymbol{\tau}_{j+1}(d) & \forall s,t : s \le k_j \wedge t = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Class 3 Vacation

For a given vacation j we use the counting process developed earlier to obtain the distribution of the number of patients arrived. No services takes place at a class 3 vacation. For $c_{j+1} \in \{1,2\}$, we have:

$$P_{j,1}[t,\emptyset|s,c] = \begin{cases} \sum\limits_{d \in \mathbf{M}_j} P_j[i,d|0,c] & \forall s,t : t-s=i, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

If $(c_{j+1} = 5)$, we have (note that a vacation j of class $c_j \in \{3,4,5\}$ can never be succeeded by a vacation $(j+1)$ of class $c_{j+1} \in \{3,4\}$):

$$P_{j,1}[t,d|s,c] = \begin{cases} P_j[i,d|0,c] & \forall s,t : t-s=i, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Note that class 3 vacations are not preceded by a vacation during which arrivals are allowed to take place. As such, the initial arrival phase probabilities are given by $\boldsymbol{\tau}_j$. This observation also holds for class 4 vacations.

Class 4 Vacation

Similar to class 3 vacations, we obtain the distribution of the number of patients arrived by means of the counting process discussed previously. At the start of a class 4 vacation j, a maximum of k_j patients is removed from the queue. For $c_{j+1} \in \{1,2\}$, we have:

$$P_{j,1}[t,\emptyset|s,c] = \begin{cases} \sum\limits_{d\in\mathbf{M}_j} P\left[i,d|j,0,c\right] & \forall s,t : s > k_j \wedge t = s+i-k_j, \\ \sum\limits_{d\in\mathbf{M}_j} P_j[i,d|0,c] & \forall s,t : s \le k_j \wedge t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

If $(c_{j+1} = 5)$, we have:

$$P_{j,1}[t,d|s,c] = \begin{cases} P_j[i,d|0,c] & \forall s,t : s > k_j \wedge t = s+i-k_j, \\ P_j[i,d|0,c] & \forall s,t : s \le k_j \wedge t = i, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Class 5 Vacation

Class 5 vacations are identical to class 4 vacations except for their definition of the initial arrival phase probabilities. Whereas the start of a class 4 vacation coincides with the start of a new arrival time slot (i.e. initial arrival phase probabilities are given by $\boldsymbol{\tau}_j$), the start of a class 5 vacation occurs while an arrival process is already in progress. As such, the initial arrival probabilities should reflect the phase of the arrival process at the start of the class 5 vacation (i.e. the initial arrival process phase equals the final arrival process phase of the previous vacation).

2.4.2 Algorithm 1

The algorithm developed in this section serves the purpose of computing $P_{j,J}[t,d|s,c]$; the probability of moving from a state $(s,c)_j$ at the start of a vacation of type j towards a state $(t,d)_{j+J}$ at the start of the next vacation of type j. Define $P_{j,n}[u,e|(s,t),(c,d)]$ as the probability to depart from a state $(s,c)_j$ at the start of a vacation j, to visit state $(t,d)_{j+n-1}$ at the start of vacation $(j+n-1)$ and to end up in state $(u,e)_{j+n}$ at the start of vacation $(j+n)$ $(c \in \mathbf{M}_j \wedge d \in \mathbf{M}_{j+n-1} \wedge e \in \mathbf{M}_{j+n})$. We assume at least two vacations to be present in a vacation cycle (i.e. $(J \geq 2)$; if $(J = 1)$ the requested probabilities $P_{j,J}[t,d|s,c]$ are given by $P_{j,1}[t,d|s,c]$).

Before advancing to the algorithm itself, a number of important properties are established:

Property 1 *When moving from the start of a vacation j to the start of a vacation $(j+J)$, no more than $\left(Q_c = \sum_{j=1}^{J} k_j\right)$ patients can be removed from the queue.*

Property 2 $P_{j,J}[t,d|s,c] = P_{j,J}[(t+i),d|(s+i),c] \quad \forall s : s \geq Q_c.$

One of the practical implications of these properties is that only states with queue sizes up to Q_c have to be evaluated by the algorithm, resulting in a significant reduction of memory and computational requirements.

The algorithm consists of two main steps: (1) iteration; (2) evaluation. In the upcoming sections, we discuss these steps in detail. A final section provides a general outline of the algorithm.

Step 1: Iteration

Prior to starting the first iteration step we initialize a counter $(n = 2)$ and compute all probabilities $P_{j,1}[t,d|s,c]$ for all vacations j and all possible queue sizes and arrival phases. In a first phase of the iteration step, compute all probabilities $P_{j,n}[u,e|(s,t),(c,d)]$ as follows:

$$P_{j,n}[u,e|(s,t),(c,d)] = \begin{cases} P_{j,(n-1)}[t,d|s,c]P_{(j+n-1),1}[u,\emptyset|t,\emptyset] & \forall c_{j+n-1} \in \{1,2\} \wedge \forall c_{j+n} = 2, \\ P_{j,(n-1)}[t,d|s,c]P_{(j+n-1),1}[u,e|t,\emptyset] & \forall c_{j+n-1} \in \{1,2\} \wedge \forall c_{j+n} \in \{3,4\}, \\ P_{j,(n-1)}[t,d|s,c]P_{(j+n-1),1}[u,\emptyset|t,d] & \forall c_{j+n-1} \in \{3,4,5\} \wedge \forall c_{j+n} \in \{1,2\}, \\ P_{j,(n-1)}[t,d|s,c]P_{(j+n-1),1}[u,e|t,d] & \forall c_{j+n-1} \in \{3,4,5\} \wedge \forall c_{j+n} = 5, \end{cases} \tag{20}$$

where $P_{j,(n-1)}[t,d|s,c]$ is either computed in a previous iteration step or is given by $P_{j,1}[t,d|s,c]$.

In the second phase of the iteration step, we aggregate all resulting probabilities over t and d and obtain probabilities $P_{j,n}[u,e|s,c]$ as follows:

$$P_{j,n}[u,e|s,c] = \sum_{t=0}^{\infty} \sum_{d \in \mathbf{M}_{j+n-1}} P_{j,n}[u,e|(s,t),(c,d)]. \tag{21}$$

After aggregation, we proceed to the evaluation step.

Step 2: Evaluation

At the evaluation step we evaluate if $(n = J)$. If the condition holds, we have obtained the required probabilities $P_{j,J}[t,d|s,c]$. If the condition does not hold, we increment the counter n and proceed to another iteration step.

Discussion

A general outline of the algorithm is provided in Algorithm 1.

The computational efficiency of Algorithm 1 follows from the aggregation of probabilities $P_{j,n}[u,e|(s,t),(c,d)]$ and the limitation of initial queue size $s : s \leq Q_c$. When disregarding the complexity resulting from the computation of $P_{j,1}[t,d|s,c]$, the complexity of Algorithm 1 is $O\left((J-2)Q_c Q_{max}^2 M_{max}^3\right)$; where: (1) $M_{max} = \max(M_j) : j \in \{1,\ldots,J\}$; (2) Q_{max} is the maximum number of patients that has a nonzero probability to arrive during any vacation j (note that $\forall s : s \in \{0,\ldots,Q_c\}$, $\forall t,u : t,u \in \{0,\ldots,Q_{max}\}$ and $\forall c,d,e : c,d,e \in \{1,\ldots,M_{max}\}$).

2.4.3 Stationary distribution of the DTMC X_j

To obtain the stationary distribution at a vacation j, we define $\boldsymbol{\mathcal{P}}_j$ as the transition matrix of the DTMC X_j. Following directly from Property 1 and 2 we have that:

$$\begin{array}{rcll} P_{j,J}[t,d|s,c] & = & 0 & \forall s > Q_c \wedge t < (s - Q_c)\,, \\ P_{j,J}[t,d|s,c] & = & P_{j,J}[(t+i),d|(s+i),c] & \forall s \geq Q_c. \end{array} \tag{22}$$

In other words, if $(s > Q_c)$, it is impossible to deplete the queue when moving from a state $(s,c)_j$ at the start of a vacation of type j towards a

Algorithm 1 Algorithm to obtain $P_{j,J}[t,d|s,c]$

```
for all Vacations $j$ do
  for all $s, t, c, d$ do
    Compute $P_{j,1}[t,d|s,c]$
  end for
end for
for all Vacations $j$ do
  Set $n = 2$
  while $n < J$ do
    for all $s, t, u, c, d, e$ do
      $P_{j,n}[u,e|(s,t),(c,d)] =$
      $P_{j,(n-1)}[t,d|s,c]P_{(j+n-1),1}[u,e|t,d]$
    end for
    for all $s, u, c, e$ do
      $P_{j,n}[u,e|s,c] = \sum_{t=0}^{\infty} \sum_{d \in \mathbf{M}_{j+n-1}} P_{j,n}[u,e|(s,t),(c,d)]$
    end for
    Increment $n$
  end while
end for
```

state $(t,d)_{j+J}$ at the start of the next vacation of type j. In addition, if $(s \geq Q_c)$, transition rates moving from states at the start of a vacation of type j towards the start of the next vacation of type j are equal given: (1) equal arrival phases (c,d); (2) equal difference in queue sizes (s,t) and $(s+i,t+i)$. These properties endow the Markov chain $\boldsymbol{\mathcal{P}}_j$ with a special, repetitive structure. More specifically, $\boldsymbol{\mathcal{P}}_j$ may be represented as a non-skip-free $M/G/1$ Markov chain (refer to Gail, Hantler and Taylor (1997)):

$$\boldsymbol{\mathcal{P}}_j = \left| \begin{array}{ccccc} \mathbf{B}_{j,0,0} & \mathbf{B}_{j,0,1} & \mathbf{B}_{j,0,2} & \mathbf{B}_{j,0,3} & \cdots \\ \mathbf{B}_{j,1,0} & \mathbf{B}_{j,1,1} & \mathbf{B}_{j,1,2} & \mathbf{B}_{j,1,3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \\ \mathbf{B}_{j,Q_c-1,0} & \mathbf{B}_{j,Q_c-1,1} & \mathbf{B}_{j,Q_c-1,2} & \mathbf{B}_{j,Q_c-1,3} & \cdots \\ \mathbf{A}_{j,0} & \mathbf{A}_{j,1} & \mathbf{A}_{j,2} & \mathbf{A}_{j,3} & \cdots \\ \mathbf{0} & \mathbf{A}_{j,0} & \mathbf{A}_{j,1} & \mathbf{A}_{j,2} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{j,0} & \mathbf{A}_{j,1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right|,$$

where $\mathbf{A}_{j,i}$ and $\mathbf{B}_{j,s,t}$ are $M_j \times M_j$ matrices that depend on the PH distribution used to model the arrival process at a vacation of type j. If no arrivals are allowed to occur during a vacation of type j (i.e. $c_j \in \{1,2\}$), $\mathbf{A}_{j,i}$ and $\mathbf{B}_{j,s,t}$ are scalars and are given by:

$$\begin{array}{rcl} \mathbf{B}_{j,s,t} & = & P_{j,J}[t,\emptyset|s,\emptyset], \\ \mathbf{A}_{j,i} & = & P_{j,J}[i,\emptyset|Q_c,\emptyset]. \end{array}$$

If arrivals are allowed to occur, we once more make a distinction between 3 cases. In the case of $(C_j^2 = 1)$, $\mathbf{A}_{j,i}$ and $\mathbf{B}_{j,s,t}$ are scalars and are given by:

$$\begin{array}{rcl} \mathbf{B}_{j,s,t} & = & P_{j,J}[t,1|s,1], \\ \mathbf{A}_{j,i} & = & P_{j,J}[Q_c,1|i,1]. \end{array}$$

In the case of $(C_j^2 > 1)$ we have:

$$\mathbf{B}_{j,s,t} = \begin{array}{c|cc|} & 1 & M_j \\ 1 & P_{j,J}[t,1|s,1] & P_{j,J}[t,M_j|s,1] \\ M_j & P_{j,J}[t,1|s,M_j] & P_{j,J}[t,M_j|s,M_j] \end{array},$$

$$\mathbf{A}_{j,i} = \begin{array}{c|cc|} & 1 & M_j \\ 1 & P_{j,J}[i,1|Q_c,1] & P_{j,J}[i,M_j|Q_c,1] \\ M_j & P_{j,J}[i,1|Q_c,M_j] & P_{j,J}[i,M_j|Q_c,M_j] \end{array}.$$

With respect to the hypoexponential case (i.e. $(C_j^2 < 1)$), $\mathbf{B}_{j,s,t}$ is defined as follows:

$$\mathbf{B}_{j,s,t} = \begin{array}{c|ccccc|}
 & 1 & 2 & \cdots & \zeta_j & M_j \\
1 & P_{j,J}[t,1|s,1] & P_{j,J}[t,2|s,1] & \cdots & P_{j,J}[t,\zeta_j|s,1] & P_{j,J}[t,M_j|s,1] \\
2 & P_{j,J}[t,1|s,2] & P_{j,J}[t,2|s,2] & \cdots & P_{j,J}[t,\zeta_j|s,2] & P_{j,J}[t,M_j|s,2] \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\zeta_j & P_{j,J}[t,1|s,\zeta_j] & P_{j,J}[t,2|s,\zeta_j] & \cdots & P_{j,J}[t,\zeta_j|s,z_j] & P_{j,J}[t,M_j|s,\zeta_j] \\
M_j & P_{j,J}[t,1|s,M_j] & P_{j,J}[t,2|s,M_j] & \cdots & P_{j,J}[t,\zeta_j|s,M_j] & P_{j,J}[t,M_j|s,M_j]
\end{array}.$$

and $\mathbf{A}_{j,i}$ is given by:

$$\mathbf{A}_{j,i} = \begin{array}{c|ccccc|}
 & 1 & 2 & \cdots & \zeta_j & M_j \\
1 & P_{j,J}[i,1|Q_c,1] & P_{j,J}[i,2|Q_c,1] & \cdots & P_{j,J}[i,\zeta_j|Q_c,1] & P_{j,J}[i,M_j|Q_c,1] \\
2 & P_{j,J}[i,1|Q_c,2] & P_{j,J}[i,2|Q_c,2] & \cdots & P_{j,J}[i,\zeta_j|Q_c,2] & P_{j,J}[i,M_j|Q_c,2] \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\zeta_j & P_{j,J}[i,1|Q_c,\zeta_j] & P_{j,J}[i,2|Q_c,\zeta_j] & \cdots & P_{j,J}[o,\zeta_j|Q_c,\zeta_j] & P_{j,J}[i,M_j|Q_c,\zeta_j] \\
M_j & P_{j,J}[i,1|Q_c,M_j] & P_{j,J}[i,2|Q_c,M_j] & \cdots & P_{j,J}[i,\zeta_j|Q_c,M_j] & P_{j,J}[i,M_j|Q_c,M_j]
\end{array}.$$

The repetitive structure observed in $\boldsymbol{\mathcal{P}}_j$ may be exploited using matrix analytical methods. Matrix analytical methods have been studied for several decades and have attracted the attention of many researchers in the queueing field. For an overview of literature and an introduction to matrix analytical methods, refer to Latouche and Ramaswami (1999), Riska (2002), Osogami (2005), Bini, Meini and Steffe (2006a and 2006b) among others. In short, matrix analytical methods allow the (numerically) exact analysis of a wide variety of queueing systems featuring some repetitive structure (more specifically, $M/G/1$, $GI/M/1$ and quasi-birth-death processes).

The matrix $\boldsymbol{\mathcal{P}}_j$ can be reblocked into blocks $\boldsymbol{\mathcal{B}}_{j,i}$ and $\boldsymbol{\mathcal{A}}_{j,i}$ of dimensions $M_jQ_c \times M_jQ_c$ as follows:

$$\boldsymbol{\mathcal{P}}_j = \left|\begin{array}{ccccc}
\boldsymbol{\mathcal{B}}_{j,0} & \boldsymbol{\mathcal{B}}_{j,1} & \boldsymbol{\mathcal{B}}_{j,2} & \boldsymbol{\mathcal{B}}_{j,3} & \cdots \\
\boldsymbol{\mathcal{A}}_{j,0} & \boldsymbol{\mathcal{A}}_{j,1} & \boldsymbol{\mathcal{A}}_{j,2} & \boldsymbol{\mathcal{A}}_{j,3} & \cdots \\
\mathbf{0} & \boldsymbol{\mathcal{A}}_{j,0} & \boldsymbol{\mathcal{A}}_{j,1} & \boldsymbol{\mathcal{A}}_{j,2} & \cdots \\
\mathbf{0} & \mathbf{0} & \boldsymbol{\mathcal{A}}_{j,0} & \boldsymbol{\mathcal{A}}_{j,1} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right|,$$

which can be solved as a traditional $M/G/1$ Markov chain. More specifically, this involves the computation of an auxilliary matrix $\mathbf{G}_j$ which is obtained as the solution of (refer to Bini, Meini and Steffe (2006a)):

$$\mathbf{G}_j = \sum_{i=0}^{\infty} \boldsymbol{\mathcal{A}}_{j,i} \mathbf{G}_j^i. \tag{23}$$

However, as is indicated in Gail, Hantler and Taylor (1997), such reblocking might increase computational requirements (since it involves computations of $M_jQ_c \times M_jQ_c$ matrices instead of computations of $M_j \times M_j$ matrices). For non-skip-free $M/G/1$ Markov chains, Gail, Hantler and Taylor (1997) have shown that $\mathbf{G}_j$ may be obtained as follows:

$$\mathbf{G}_j = \mathbf{C}\left(\mathbf{g}_j\right)^{Q_c}, \tag{24}$$

where $(\mathbf{g}_j = (\mathbf{G}_{j,0}, \mathbf{G}_{j,1}, \ldots, \mathbf{G}_{j,Q_c-1}))$ is the first block row of $\mathbf{G}_j$ and $\mathbf{C}\left(\mathbf{g}_j\right)^{Q_c}$ is a companion matrix. $\mathbf{C}\left(\mathbf{g}_j\right)$ may be represented as:

$$\mathbf{C}\left(\mathbf{g}_j\right) = \left|\begin{array}{ccccc} \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \\ \mathbf{G}_{j,0} & \mathbf{G}_{j,1} & \mathbf{G}_{j,2} & \cdots & \mathbf{G}_{j,Q_c-1} \end{array}\right|.$$

As such, once the first block row of $\mathbf{G}_j$ is known, it is a simple matter to compute $\mathbf{G}_j$. Using the functional iteration algorithm outlined in Gail, Hantler and Taylor (1997), we compute $\mathbf{G}_j$. Once we obtained $\mathbf{G}_j$, we are able to compute $\boldsymbol{\pi}_j[i]$ using the recursive formula developed in Ramaswami (1988):

$$\boldsymbol{\pi}_j[n] = \left(\boldsymbol{\pi}_j[0]\boldsymbol{\mathcal{B}}_j^{(n)} + \sum_{i=1}^{n-1} \boldsymbol{\pi}_j[i]\boldsymbol{\mathcal{A}}_j^{(n-i)}\right)\left(\mathbf{I} - \boldsymbol{\mathcal{A}}_j^{(0)}\right)^{-1}, \ \forall n : n \geq 1, \tag{25}$$

where:

$$\begin{aligned} \boldsymbol{\mathcal{A}}_j^{(n)} &= \sum_{i=n}^{\infty} \boldsymbol{\mathcal{A}}_{j,i+1}\mathbf{G}_j^{i-n}, \ \forall n : n \geq 0, \\ \boldsymbol{\mathcal{B}}_j^{(n)} &= \sum_{i=n}^{\infty} \boldsymbol{\mathcal{B}}_{j,i}\mathbf{G}_j^{i-n}, \ \forall n : n \geq 0, \end{aligned}$$

and $\boldsymbol{\pi}_j[0]$ is the solution of (refer to Ramaswami (1988)):

$$\begin{aligned} \boldsymbol{\pi}_j[0] &= \boldsymbol{\pi}_j[0]\boldsymbol{\mathcal{B}}_j^{(0)}, & (26) \\ -\nu_j &= \boldsymbol{\pi}_j[0]\mathbf{b_j} - \nu_j\boldsymbol{\pi}_j[0]\mathbf{e} + \boldsymbol{\pi}_j[0]\left(\mathbf{I} - \boldsymbol{\mathcal{B}}_j\right)\left(\mathbf{I} - \boldsymbol{\mathcal{A}}_j\right)^{\sharp}\mathbf{a}_j, & (27) \end{aligned}$$

where:

- $\boldsymbol{\mathcal{A}}_j = \sum_{i=0}^{\infty} \boldsymbol{\mathcal{A}}_{j,i}$,

- $\boldsymbol{\mathcal{B}}_j = \sum_{i=0}^{\infty} \boldsymbol{\mathcal{B}}_{j,i}$,
- $\mathbf{a}_j = \sum_{i=0}^{\infty} (i-1)\boldsymbol{\mathcal{A}}_{j,i}\mathbf{e}$,
- $\mathbf{b}_j = \sum_{i=0}^{\infty} i\boldsymbol{\mathcal{B}}_{j,i}\mathbf{e}$,
- $\nu_j = \boldsymbol{\alpha}_j^{\top}\mathbf{a}_j$ and $\boldsymbol{\alpha}_j$ is the stationary distribution vector of $\boldsymbol{\mathcal{A}}_j$,
- operator $(\cdot)^{\sharp}$ denotes the group inverse operation.

$\boldsymbol{\pi}_j[i]$ is the vector of stationary probabilities associated with a queue size $s : s \in \{iQ_c, \ldots, ((i+1)Q_c - 1)\}$. More specifically, $\boldsymbol{\pi}_j[i]$ holds the stationary distribution of states $(s,c)_j : \forall s, c : s \in \{iQ_c, \ldots, ((i+1)Q_c - 1)\}$ $\wedge c \in \mathbf{M}_j$. From $\boldsymbol{\pi}_j[i]$ we obtain $\pi_j[s,c]$, the stationary distribution of being in a state $(s,c)_j$; $\forall s, c : s \in \{0, 1; \ldots\} \wedge c \in \mathbf{M}_j$. Note that the stationary distribution $\pi_j[s,c]$ (and all performance measures derived thereof) is computed in a numerically exact manner.

2.4.4 Algorithm 2

In this section, we develop an algorithm that uses the stationary distribution $\pi_j[s,c]$ of a single DTMC X_j to determine the stationary distribution of all other DTMC $X_{j+n} : n \in \{1, \ldots, J-1\}$. Together with probabilities $P_{j,1}[t,d|s,c]$, the stationary distribution $\pi_j[s,c]$ serves as the input of the algorithm outlined below. The algorithm is a simple iterative procedure that consists of two steps: (1) iteration; (2) evaluation. In what follows, we discuss these steps in detail. In a final section, we provide a discussion and give a general outline of the algorithm.

Step 1: Iteration

Prior to starting the first iteration step we initialize a counter ($n = 1$) and compute the stationary distribution $\pi_j[s,c]$ for a single vacation j. In the iteration step, the stationary distribution $\pi_{(j+n)}[t,d]$ is computed as follows:

$$\pi_{(j+n)}[t,d] = \begin{cases} \sum_{s=0}^{\infty} \sum_{c \in \mathbf{M}_{j+n-1}} \pi_{(j+n-1)}[s,c] P_{(j+n-1),1}[t,\emptyset|s,\emptyset] & \forall c_{j+n-1} \in \{1,2\} \wedge \forall c_{j+n} = 2, \\ \sum_{s=0}^{\infty} \sum_{c \in \mathbf{M}_{j+n-1}} \pi_{(j+n-1)}[s,c] P_{(j+n-1),1}[t,d|s,\emptyset] & \forall c_{j+n-1} \in \{1,2\} \wedge \forall c_{j+n} \in \{3,4\}, \\ \sum_{s=0}^{\infty} \sum_{c \in \mathbf{M}_{j+n-1}} \pi_{(j+n-1)}[s,c] P_{(j+n-1),1}[t,\emptyset|s,c] & \forall c_{j+n-1} \in \{3,4,5\} \wedge \forall c_{j+n} \in \{1,2\}, \\ \sum_{s=0}^{\infty} \sum_{c \in \mathbf{M}_{j+n-1}} \pi_{(j+n-1)}[s,c] P_{(j+n-1),1}[t,d|s,c] & \forall c_{j+n-1} \in \{3,4,5\} \wedge \forall c_{j+n} = 5, \end{cases} \tag{28}$$

where $\pi_{(j+n-1)}[s,c]$ is either computed in a previous iteration step or is given by $\pi_j[s,c]$. After the iteration step, we proceed to the evaluation step.

Step 2: Evaluation

At the evaluation step, we evaluate if $(n = J)$. If the condition holds, we have computed all the required stationary distributions. If the condition does not hold, we increment the counter n and proceed to another iteration step.

Discussion

A general outline of the algorithm is provided in Algorithm 2.

Algorithm 2 Optimized algorithm to obtain $P_{j,J}[t,d|s,c]$

```
for all Vacations $j$ do
  for all $s, t, c, d$ do
    Compute $P_{j,1}[t,d|s,c]$
  end for
end for
For a single vacation $j$ compute $\pi_j[s,c]$
Set $n = 1$
while $n < J$ do
  for all $t, d$ do
    $\pi_{(j+n)}[t,d] = \sum_{s=0}^{\infty} \sum_{c \in \mathbf{M}_{j+n-1}} \pi_{(j+n-1)}[s,c] P_{(j+n-1),1}[t,d|s,c]$
  end for
  Increment $n$
end while
```

Given the stationary distribution of a single DTMC X_n, one may may efficiently obtain the stationary distribution of all other DTMC $X_j : j \in$

$\{1, \ldots, J\} \setminus \{n\}$ using Algorithm 2. In order to obtain the stationary distribution of a single DTMC X_n, we resort to matrix analytical methods. To further enhance computational performance, the PH distribution used to model the arrival process at the single DTMC X_n should have as few phases as possible (i.e. select the DTMC X_n for which $M_n = \min(M_j) \forall j : j \in \{1, \ldots, J\}$). When disregarding the complexity resulting from the computation of $P_{j,1}[t, d|s, c]$, the complexity of Algorithm 1 is $O\left((J-1)(Q_c+1)(Q_{max}+1)M_{max}^2\right)$. Note that $\forall s : s \in \{0, \ldots, Q_c\}$, $\forall t : t \in \{0, \ldots, Q_{max}\}$ and $\forall c, d : c, d \in \{1, \ldots, M_{max}\}$.

2.4.5 Aggregation of results

Using the stationary distribution $\pi_j[s, c]$, various performance measures may be derived. In this article, we limit ourselves to the expected queue size at the start of a vacation j. Note that $\pi_j[s, c]$ holds the stationary distribution of the queue size prior to the removal of up to k_j patients from the queue. Because we are interested in the expected number of patients after the removal of up to k_j patients, a shifting operation is required:

$$\mathcal{Q}_j^{\diamond} = \sum_{s=k_j}^{\infty} \sum_{c \in \mathbf{M}_j} (s - k_j)\pi_j[s, c]. \tag{29}$$

In order to compute $\mathcal{Q}_j^*$ we resort to probabilities $P_j[i, d|0, c]$ (i.e. the probabilities resulting from the counting process; the probabilities of having a number of arrivals during a vacation of type j). When assuming a time-independent arrival process, the expected number of patients in queue during a vacation of type j (given that these patients arrive during vacation j itself) is given by:

$$\mathcal{Q}_j^* = \frac{\eta_j}{2} \tag{30}$$

where η_j is the expected number of arrivals during a vacation of type j and is computed as follows:

$$\eta_j = \sum_{i=0}^{\infty} \sum_{c \in \mathbf{M}_j} \sum_{d \in \mathbf{M}_j} i P_j[i, d|0, c]. \tag{31}$$

Note that:

- The problem of identifying $\mathcal{Q}_j^*$ corresponds to the computation of the average production over a production cycle, the average demand over a demand cycle, the average inventory over an inventory cycle (disregarding the presence of safety stock), ...
- Probabilities $P_j[i, d|0, c]$ have already been determined in the process of computing $\pi_j[s, c]$. As such, limited additional computational effort is required to compute $\mathcal{Q}_j^*$.
- $(\mathcal{Q}_j^* = 0)$ for all $c_j \in \{1, 2\}$.

The expected number of patients in queue during a vacation of type j is:

$$\mathcal{Q}_j = \mathcal{Q}_j^\diamond + \mathcal{Q}_j^*. \tag{32}$$

Performance measures $\mathcal{Q}_j^\diamond$, $\mathcal{Q}_j^*$ and hence $\mathcal{Q}_j$ are computed in in a numerically exact manner.

From the previous sections we obtained the expected queue size during a vacation of type j. We can aggregate these results to obtain the expected queue size at the bulk service queueing model (defined as $\mathcal{Q}$) and the expected waiting time of a patient (defined as $\mathcal{W}$).

The average queue size at the bulk service queueing model is given by:

$$\mathcal{Q} = \sum_{j=1}^{J} p_j \mathcal{Q}_j, \tag{33}$$

where p_j is the probabilities of finding oneself at a vacation of type j:

$$p_j = \frac{T_j}{\sum\limits_{j=1}^{J} T_j}. \tag{34}$$

Using Little's law, we can compute the expected waiting time of a patient:

$$\mathcal{W} = \frac{\mathcal{Q}}{\sum\limits_{j=1}^{J} \frac{\eta_j}{T_j}}, \tag{35}$$

In this article we limit ourselves to the computation of the expected waiting time of a patient. Note however that higher moments may easily be derived.

3 Optimal capacity allocation model

From the bulk service queueing model we obtain the expected waiting time of a given class of patients and a given allocation scheme. More formally, define $\mathbf{Y} = \{1, 2, \ldots, Y\}$, the set of all patient classes. Let w_y denote the weight of the expected waiting time of a patient of class y (for the remainder of the text, index y is defined as $y \in \{1, 2, \ldots, Y\}$). In addition, define $\mathbf{H} = \{1, 2, \ldots, H\}$ to be the set of all available service time slots. Each service time slot h is characterized by a duration L_h (for the remainder of the text, index h is defined as $h \in \{1, 2, \ldots, H\}$). Let $f(h, y)$ denote the allocation of time slot h to class y. An allocation scheme is a function $f(\cdot)$ that maps all elements of $\mathbf{S}$ onto elements of $\mathbf{Y}$ (i.e. a function that allocates service time slots to patient classes). An allocation scheme $f(\cdot)$ can be seen as a set of allocations $f(h, y)$ such that every time slot h is allocated to a patient class y. The set of feasible allocation schemes is denoted by $\mathbf{F}$. The cardinality of $\mathbf{F}$ (defined as F) depends on: (1) the number of patient classes; (2) the number of service time slots to be allocated; (3) the restrictions imposed upon allocating service time slots over patient classes. For each patient class $y : y \in \mathbf{Y}$ and each feasible allocation scheme $f(\cdot) : f(\cdot) \in \mathbf{F}$, a bulk service queueing model may be analyzed. Let $\mathcal{W}_{y,f(\cdot)}$ denote the expected waiting time of a patient of class y if an allocation scheme $f(\cdot)$ is imposed. The total expected weighted waiting time function given an allocation scheme $f(\cdot)$ may be formulated as follows:

$$TW_{f(\cdot)} = \sum_{y=1}^{Y} w_y \mathcal{W}_{y,f(\cdot)}. \tag{36}$$

For smaller problems an enumerative search of the solution space may be performed. for large Y and/or large F, such an enumerative approach might prove to be unfeasible. Therefore, a step-wise heuristic is developed. Note that:

- While the theoretical number of possible solutions quickly grows beyond bounds, practical limitations (e.g. strategic choices made by hospital management, preferences of medical staff etc.) significantly reduce the size of the solution space.

- Whereas the enumerative search yields an optimal solution, the step-wise heuristic is not necessarily optimal.

In what follows, we discuss both the enumerative search as well as the step-wise heuristic.

3.1 Enumerative search

When minimizing $TW_{f(\cdot)}$ over all feasible allocation schemes $f(\cdot) : f(\cdot) \in \mathbf{F}$, we obtain the optimal solution of the optimization problem (i.e. the allocation of service time slots over patient classes such that the total expected weighted waiting time of a patient is minimized). The objective function is:

$$\min_{f(\cdot)\in\mathbf{F}} TW_{f(\cdot)} = \sum_{y=1}^{Y} w_y \mathcal{W}_{y,f(\cdot)}. \tag{37}$$

Note that additional constraints (e.g. strategic limitations, doctor preferences etc.) can easily be introduced.

3.2 Step-wise heuristic

The step-wise heuristic contains two steps. The first step involves finding a good (at least feasible) starting solution. We propose the use of a feasible schedule that: (1) is evenly spread across time; (2) takes into account the weights of the patient waiting times; (3) maximizes the number of unassigned service time slots (i.e. each patient class is assigned the minimum number of service time slots required to avoid infinite queues). The procedure builds on the principle that patients have to wait longer in a system in which service moments are located far away from each other in time (e.g. patients will have to wait longer in a system in which they receive service only once a month when compared to a system in which weekly service is issued).

After a starting solution has been obtained, a steepest descent algorithm is initiated in order to assign the unassigned service time slots. In what follows we discuss both the starting solution as well as the steepest descent algorithm.

3.2.1 Starting solution

The objective of the starting solution is to define a feasible allocation scheme in which each patient class is provided with the minimum number of time slots required and these time slots are scheduled as evenly as possible across time. Since an occupancy of 100% or more leads to infinite waiting times, the minimum number of time slots for each patient class is the smallest number for which the occupancy is less than 100%. The remaining time slots are assigned in the steepest descent algorithm described hereafter. In order to become a starting solution in which the time slots are distributed as evenly as possible, our objective will minimize the largest 'distance' (in time) between two succeeding time slots for each class. Due to a limited number of available time slots per time period, a perfectly spread distribution of time slots for each patient class is very unlikely. In that case, priority is given to the more important patient classes by adding the weights of each class in the objective function. More formally, the problem we solve to find a good starting solution can be stated as follows.

First, let R_y denote the minimum number of time slots required for class y. We define the binary decision variable x_{yrh} to be 1 if time slot h is assigned to class y in the r-th position ($r \in 1, 2, ..., R_y$) and 0 otherwise. Let $d_{h_1 h_2}$ be a parameter indicating the difference in time between two time slots h_1 and h_2 (for the remainder of the text, indices h_1 and h_2 are defined as $h_1, h_2 \in \{1, 2, \ldots, H\}$). For instance, if h_1 and h_2 correspond to the afternoon time slots of operating rooms 1 and 2, $d_{h_1 h_2}$ is equal to 0 hours (both time slots occur at the same time). On the other hand, if h_1 and h_2 correspond to afternoon time slots on Monday and Tuesday respectively, $d_{h_1 h_2}$ is equal to 24 hours. Note that $d_{h_1 h_2}$ is also defined for $h_2 < h_1$. In this case, the difference in time is dependent on the length of the cycle time. For instance, if the cycle time is one week and h_1 and h_2 correspond to afternoon time slots on Friday and Tuesday respectively, $d_{h_1 h_2}$ is equal to 24*4=96 hours. Let $z_{yr(r+1)}$ be a help variable representing the difference in time between the r-th and the (r+1)-th time slot assigned to class y and let z_y^{max} denote the maximal difference in time over all pairs of succeeding time slots assigned to class y. The model is then as follows:

$$Minimize \sum_{y=1}^{Y} w_y z_y^{max} \tag{38}$$

subject to

$$\sum_{y=1}^{Y} \sum_{r=1}^{R_y} x_{yrh} \leq 1 \qquad \forall h \in \{1, 2, \ldots, H\} \tag{39}$$

$$\sum_{h=1}^{H} x_{yrh} = 1 \qquad \forall y \in \{1,2,\dots,Y\} \qquad \forall r \in \{1,\dots,R_y\} \tag{40}$$

$$\sum_{h_1=1}^{h_2-1} x_{yrh_1} \geq x_{y(r+1)h_2} \forall y \in \{1,2,\dots,Y\} \forall r \in \{1,\dots,R_y-1\} \forall h_2 \in \{2,\dots,H\} \tag{41}$$

$$z_{yr(r+1)} = \sum_{h_1=1}^{H} \sum_{h_2=h_1+1}^{H} d_{h_1h_2} x_{yrh_1} x_{y(r+1)h_2} \forall y \in \{1,2,\dots,Y\} \forall r \in \{1,\dots,R_y-1\} \tag{42}$$

$$z_{y(R_y)1} = \sum_{h_1=1}^{H} \sum_{h_2=1}^{H} d_{h_1h_2} x_{y(R_y)h_1} x_{y1h_2} \qquad \forall y \in \{1,2,\dots,Y\} \tag{43}$$

$$z_y^{max} \geq z_{yr(r+1)} \qquad \forall y \in \{1,2,\dots,Y\} \qquad \forall r \in \{1,\dots,R_y-1\} \tag{44}$$

$$z_y^{max} \geq z_{y(R_y)1} \qquad \forall y \in \{1,2,\dots,Y\} \tag{45}$$

$$x_{yrh} \in \{0,1\} \qquad \forall y \in \{1,2,\dots,Y\} \forall r \in \{1,\dots,R_y\} \forall h \in \{1,2,\dots,H\} \tag{46}$$

$$z_{yr(r+1)} \geq 0 \qquad \forall y \in \{1,2,\dots,Y\} \qquad \forall r \in \{1,\dots,R_y-1\} \tag{47}$$

$$z_{y(R_y)1} \geq 0 \qquad \forall y \in \{1,2,\dots,Y\} \tag{48}$$

$$z_y^{max} \geq 0 \qquad \forall y \in \{1,2,\dots,Y\} \tag{49}$$

Problem (38)-(49) is a min-max quadratic programming problem (QP). The objective function (38) minimizes the maximal difference between two succeeding time slots (weighted over all classes) in order to obtain a schedule in which the time slots are distributed as evenly as possible over time. Constraint set (39) ensures that each time slot is assigned at most once, while constraint set (40) ensures that each class receives its minimal number of time slots. Constraint set (41) models the precedence constraints. Constraint sets (42) and (43) calculate the difference (in time) between two succeeding time slots. Constraint sets (44) and (45) make sure that the variable z_y^{max} is set to the maximal difference in time between any two succeeding time slots for each class y. Finally, constraint sets (46) to (49) define the decision variables.

Problem (38)-(49) can be solved with a commercial optimization package that allows for solving QPs, e.g. IBM-ILOG $CPLEX^{©}$. This works reasonably well for small problem dimensions, which is a realistic assumption because the total minimal number of time slots R_y is often much smaller than the available number of time slots. The solution to problem (38)-(49) yields a feasible allocation scheme $f(\cdot)$ that is used as a starting solution in the steepest descent algorithm (described hereafter). In the unlikely case that problem (38)-(49) could not be solved to optimality within a reasonable time limit, one can always end the computation and work with a non-optimal but still feasible solution to problem (38)-(49).

3.2.2 Steepest descent algorithm

Once we obtain the feasible starting solution, the steepest descent algorithm may be initiated. The steepest descent algorithm assigns all the unassigned time slots, i.e. the total available time slots minus the total minimal numbers of time slots (over all classes). The latter are already allocated in the starting solution and are frozen for the remainder of the search. In the steepest descent algorithm, each iteration involves assigning one unassigned service time slot to a service class in a greedy way, i.e. such that the resulting weighted waiting time is minimized. An outline of the steepest descent algorithm is provided in Algorithm 3.

In what follows we present a numerical example to illustrate both the enumerative and the heuristic solution approach.

4 Numerical example

Consider a small hospital that has one operating room at its disposal. The cycle time is one week and the operating room is only available for service during weekdays. Each weekday may be divided into two service time slots: before noon and after noon. All service time slots are assumed to be of equal length (i.e. $L_{h_1} = L_{h_2} : \forall h_1, h_2 \in \{1, 2, \ldots, H\}$). The hospital is specialized in two major medical disciplines: orthopaedics and neurology. These disciplines account for 90% of the patient population, the remaining 10% of the patients are of lesser strategic importance and are served by a variety of smaller departments. As such, the ten available time slots (per week) are to be allocated among the orthopaedic, the neurology and the

Algorithm 3 Steepest descent algorithm

```
Set $f(\cdot)^{MIN}$ corresponding to the allocation scheme in the starting solution
for all unassigned time slots $h$ do
    Set $TW^{MIN} = +\infty$
    Set $y = 1$
    while $y \leq Y$ do
        Set $f(\cdot) = f(\cdot)^{MIN}$
        Assign time slot $h$ to patient class $y$
        Set $f(\cdot) = f(\cdot) \bigcup f(h, y)$
        Compute $TW_{f(\cdot)}$
        if $TW_{f(\cdot)} < TW^{MIN}$ then
            Set $TW^{MIN} = TW_{f(\cdot)}$
            $h^* = h$
            $y^* = y$
        end if
        Increment $y$
    end while
    Set $f(\cdot)^{MIN} = f(\cdot)^{MIN} \bigcup f(h^*, y^*)$
    Increment $h$
end for
```

smaller departments. Three patient classes may be discerned (i.e. $(Y = 3)$):

y	Class
1	Neurologic patients
2	Orthopaedic patients
3	Other patients

Assume that hospital management wants to prioritize neurologic patients over orthopaedic and other patients and orthopaedic patients over other patients. The weights are:

y	w_y	Class
1	0.8	Neurologic patients
2	0.5	Orthopaedic patients
3	0.1	Other patients

In order to maintain transparency, we assume that arrivals are allowed to occur at any moment in time (i.e. we assume a single arrival session for all classes of patients). Furthermore assume that four patients can be served during each service time slot, regardless of patient class (i.e. $k = 4 \; \forall y \; : \; y \in \mathbf{Y}$).

An example of a feasible allocation scheme is provided below:

	Monday	Tuesday	Wednesday	Thursday	Friday
Before noon	2	1	1	1	1
After noon	2	3	3	3	3

The number in each time slot indicates the assigned patient class (1, 2 or 3). All that remains is to characterize the arrival processes of the different classes of patients. The data are presented in the table below (all time-related variables are expressed in terms of hours):

y	λ^{-1}	C^2	λ_1^{-1}	λ_2^{-1}	ζ	ν_1	ν_2
1	$^{200}/_{18}$	1.5	15.28	3.70	1	0.64	0.36
2	$^{200}/_{18}$	0.33	3.70	3.70	2		
3	50	1	50				

As such, the arrival process of neurologic patients is the most variable and is modeled using a hyper-exponential distribution with a mean interarrival time of $^{200}/_{18}$ hours (i.e. neurologic patients arrive at the queue every $^{200}/_{18}$ hours on average; refer to Section 2.2 for an account on the PH distributions used to model the arrival processes of the different patient classes). For each patient class, we summarize the data on the required number of service time slots:

Patient class	1	2	3
Number of arriving patients	15.12	15.12	3.36
Number of patients served during a service time slot	4	4	4
Required number of service time slots	3.78	3.78	0.84
Integer required number of service time slots	4	4	1

We now have all the data needed to initiate the step-wise heuristic described above. First, nine (4+4+1) out of the ten available time slots are assigned to the three patient classes by solving the min-max QP for finding a good starting solution (step 1). The resulting optimal allocation scheme is presented below:

	Monday	Tuesday	Wednesday	Thursday	Friday
Before noon	1	1		1	1
After noon	2	2	3	2	2

Second, starting from this initial solution, the steepest descent algorithm will complete the schedule. For this example, this second step is rather easy, as it only involves assigning the single remaining time slot to one of the three patient classes such that the (weighted) total waiting time decreases the most. The resulting allocation scheme is presented below:

	Monday	Tuesday	Wednesday	Thursday	Friday
Before noon	1	1	1	1	1
After noon	2	2	3	2	2

This allocation scheme turns out to be the optimal one (checked by complete enumeration) having an optimal objective value of 61.66 (the worst feasible allocation scheme has an objective value of 314.77). As both the initial solution and the steepest descent algorithm are heuristic methods with respect to the real objective, that is minimizing the weighted total waiting time, there is no guarantee that our procedure results in the optimal solution. Hence, the detection of the optimal solution in this small example is coincidence. The smaller the problem instance, the larger the chance of finding the optimum (as there is a limited number of solutions). For larger problems, complete enumeration cannot longer be used to check the optimality of the solution within a reasonable time limit. We expect, however, that the optimality gap (relative difference between the optimal solution value and the solution value resulting from applying our procedure) decreases with growing problem dimensions. The average waiting times of a patient of the neurology, orthopaedics and other class respectively, are provided below:

y	w_y	$\mathcal{W}_y$	$w_y\mathcal{W}_y$
1	0.8	54.53	43.62
2	0.5	31.36	15.68
3	0.1	23.59	2.36
Total			61.66

Note that the model may also be used to answer additional capacity-related strategic questions (e.g. should the hospital expand capacity, does the hospital overprioritize the neurologic department, ...).

5 Conclusion

In this article we present a model that allows the optimal allocation of service time slots (i.e. the server capacity) over a number of patient classes. The optimal allocation minimizes the total expected weighted waiting time of a patient (different classes of patient may be assigned different weights). The inputs of the optimization procedure are drawn from a bulk service queueing model that observes the queueing behavior of a single class of patients, given a particular capacity allocation over all patient classes. As such, a bulk service queueing model is analyzed for: (1) each class of patients; (2) each feasible capacity allocation scheme. We build on the model presented in Creemers and Lambrecht (2009b) and make some important improvements (we improve both on computational requirements as well as on model accuracy).

The model presented in this article is inspired on a practical problem experienced in many hospitals: how to allocate operating room capacity over a variety of medical disciplines. Few studies have addressed this problem, let alone tried to solve it. We present a model that allows hospital decision makers to assess the impact of the allocation of operating room capacity on the waiting time of different classes of patients. Since waiting time is only one aspect in the decision-making process, there is still a lot of room for further research.

Future research should focus on incorporating hospital-side elements into the optimization procedure. Such elements include but are not limited to: staff overtime, unused operating room capacity and additional stochastic behavior (e.g. set-up of medical equipment, late arrival of medical staff, failure/shortage of hospital resources). Next to applications in health care, the model may also be adopted in other services and industries. Future research should explore the possibilities with respect to the allocation of class rooms over faculty or school departments, the allocation of shelf space over consumer products, ...

Acknowledgements

The authors wish to thank the Fund for Scientific Research Flanders (FWO), project G.0333.10N for the financial support.

References

[Banik2009] Banik, A.D., Gupta, U.C., & Chaudhry, M.L. (2009). Finite-Buffer Bulk Service Queue Under Markovian Service Process: $GI/MSP^{a,b}/1/N$. *Stochastic Analysis and Applications*, *27(3)*, pp 500–522.

[Besley1999] Besley, T., Hall, J. & Preston, I. (1999). The demand for private health insurance : do waiting lists matter?. *Journal of public economics*, *72(2)*, 155–181.

[Bini2006] Bini, D., Meini, B., Steffe, S., & Van Houdt, B. (2006a). Structured Markov chains solver: algorithms. In ACM international conference proceeding series, *proceedings of SMCtools*. Pisa, Italy.

[Bini2006] Bini, D., Meini, B., Steffe, S., & Van Houdt, B. (2006b). Structured Markov chains solver: software tools. In ACM international conference proceeding series, *proceedings of SMCtools*. Pisa, Italy.

[Cardoen2010] Cardoen, B., Demeulemeester, E., & Beliën J. (2010). Operating room planning and scheduling: a literature review. *European Journal of Operational Research*, *201*, 921–932.

[Chaudhry1983] Chaudhry, M.L., & Templeton, J.G.C. (1983). *A first course in bulk queues*. New York: John Wiley & Sons.

[Chaudhry2003] Chaudhry, M.L., & Gupta, U.C. (2003). Analysis of a finite-buffer bulk-service queue with discrete-Markovian arrival process: $D-MAP/G^{a,b}/1/N$. *Naval Research Logistics*, *50(4)*, pp 345–363.

[Creemers2009] Creemers, S., & Lambrecht, M.R. (2009a). Queueing models for appointment-driven systems. *Annals of Operations Research*, 10.1007/s10479-009-0646-9.

[Creemers2009] Creemers, S., & Lambrecht, M.R. (2009b). An advanced queueing model to analyze appointment-driven service systems. *Computers and Operations Research* , *36(10)*, 2773–2785.

[Creemers2009] Creemers, S. (2009c). *Appointment-driven queueing systems.* PhD thesis, Department of Decision Sciences & Information Management, K.U. Leuven.

[Doshi1986] Doshi, B.T. (1986). Queueing systems with vacations - a survey. *Queueing Systems*, *1*, 29–66.

[Gail1997] Gail, H.R., Hantler, S.L. & Taylor, B.A. (1997). Non-skip-free $M/G/1$ and $G/M/1$ type Markov chains. *Advances in Applied Probability*, *29(3)*, 733–758.

[Galassi2006] Galassi, M., Davies, J., Theiler, J., Gough, G., Jungman, B., Booth, M. & Rossi, F. (2006). *GNU Scientific Library Reference Manual.* Network Theory Limited.

[Goddard1998] Goddard, J.A. & Tavakoli, M. (1998). Referral rates and waiting lists: some empirical evidence. *Health Economics*, *7(6)*, 545–549.

[Green2008] Green, L.V. (2008). Using Operations Research to Reduce Delays for Healthcare. *Tutorials in Operations Research*, *1*, 1–16.

[Gupta2002] Gupta, U.C., & Goswamib, V. (2002). Performance analysis of finite buffer discrete-time queue with bulk service. *Computers and Operations Research*, *29(10)*, pp 1331–1341.

[Gupta2004] Gupta, U.C., & Karabi Sikdar (2004). The finite-buffer $M/G/1$ queue with general bulk-service rule and single vacation. *Performance Evaluation*, *57(2)*, pp 199–219.

[Hanning1996] Hanning, M. (1996). Maximum waiting-time guarantee – an attempt to reduce waiting lists in Sweden. *Health Policy*, *36(1)*, 17–35.

[Janssen2005] Janssen, A.J.E.M., & van Leeuwaarden, J.S.H. (2005). Analytic Computation Schemes for the Discrete-Time Bulk Service Queue. *Queueing Systems*, *50*, pp 141–163.

[Latouche1999] Latouche, G., & Ramaswami, V. (1999). *Introduction to Matrix Analytic Methods in Stochastic Modeling.* Philadelphia: ASA-SIAM Series on Statistics and Applied Probability.

[Osogami2005] Osogami, T. (2005). *Analysis of multiserver systems via dimensionality reduction of Markov chains.* PhD thesis, School of Computer Science, Carnegie Mellon University.

[Martin1999] Martin, S. & Smith, P.C. (1999). Rationing by waiting lists: an empirical investigation. *Journal of Public Economics*, *71(1)*, 141–164.

[Neuts1981] Neuts, M.F. (1981). *Matrix-geometric solutions in stochastic models.* Baltimore: Johns Hopkins University Press.

[Ramaswami1988] Ramaswami, V. (1988). A stable recursion for the steady state vector in Markov chains of $M/G/1$ type. *Stochastic Models*, *4(1)*, 183–189.

[Riska2002] Riska, A. (2002). *Aggregate matrix analytic techniques and their applications.* PhD Dissertation, The College of William and Mary.

[Rotstein2006] Rotstein, D.L. & Alter, D.A. (2006). Where does the waiting list begin? A short review of the dynamics and organization of modern waiting lists. *Social Science & Medicine*, *62*, 3157–3160.

[Tadj2009] Tadj, L., & Abid, C. (2009). Optimal management policy for a single and bulk service queue under Bernoulli vacation schedules. *International Journal of Applied Decision Sciences*, *2(3)*, pp 262–274.

[Takagi1988] Takagi, H. (1988). Queueing analysis of polling models. *ACM Computing Surveys*, *20*, 5–28.

[Tian2006] Tian, N., & Zhang, Z. (2006). *Vacation queueing models*. New York: Springer Science.

[Tijms2003] Tijms, H.C. (2003). *A first course in stochastic models*. Chichester: John Wiley & Sons.

[Vanden Bosch2000] Vanden Bosch, P.M., & Dietz, D. C. (2000). Minimizing expected waiting in a medical appointment system. *IIE Transactions*, *32*, 841–848.

[Worthington1987] Worthington, D.J. (1987). Queueing models for hospital waiting lists. *The Journal of the Operational Research Society*, *38(5)*, pp 413–422.