Simultaneous transient analysis of QBD Markov chains for all initial configurations using a level-based recursion

Simultaneous Transient Analysis of QBD Markov Chains for all Initial Configurations using a Level Based Recursion

J. Van Velthoven, B. Van Houdt* and C. Blondia

University of Antwerp
Middelheimlaan 1
B-2020 Antwerp, Belgium

ABSTRACT

A new algorithm to assess transient performance measures for every possible initial configuration of a Quasi-Birth-and-Death (QBD) Markov chain is introduced. We make use of the framework termed QBDs with marked time epochs that transforms the transient problem into a stationary one by applying a discrete Erlangization and constructing a reset Markov chain. To avoid the need to repeat all computations for each initial configuration, we propose a level based recursive algorithm that uses intermediate results obtained for initial states belonging to levels $0, \ldots, r-1$ to compute the transient measure when the initial state is part of level r. Also, the computations for all states belonging to level r are performed simultaneously. A key property of our approach lies in the exploitation of the internal structure of the block matrices involved, avoiding any need to store *large* matrices. A flexible Matlab implementation of the proposed algorithm is available online.

1. INTRODUCTION

The study of Quasi-Birth-and-Death (QBD) Markov chains has a long tradition dating back to the early work by Neuts in the 1960s. Today this topic is still very actively pursued by many researchers. Early contributions to the transient analysis of QBDs were made by Ramaswami [11] and, Zhang and Coyle [17], mostly relying on Laplace transforms. More recent works include those by Le Ny and Sericola [10] and Remke et al. [13], which both combine uniformization techniques with a recursive approach to tackle the resulting discrete time problem. Their strength lies mostly in solving transient problems over short time scales, as the computation times increase as a function of the time epoch of the event of interest. The approach taken in this paper is very different and more general in nature and builds on earlier work [14], [15].

A new approach that reduces a variety of transient QBD problems to stationary ones, was introduced in [14] for the special case of a D-BMAP/PH/1 queue. The two main steps introduced were the use of a discrete Erlangization and the construction of a reset Markov chain. Although the reset Markov chain contained considerably more states, compared to the original QBD, computation times were limited by exploiting the internal structure of the block matrices that characterize its transition matrix.

*B. Van Houdt is a post-doctoral fellow of the FWO-Flanders.

The underlying ideas presented in [14] gave rise to the development of a new framework, termed QBDs with marked time epochs [15], to assess transient measures in a plug-and-play manner for any QBD, where a significant improvement over [14] was also achieved in terms of the required computational resources. An extension to the class of tree-like processes [2], a generalization of the QBD paradigm, was discussed in [16] and applied to analyze a set of random access algorithms.

Both [15] and [16] were limited to the case where the initial state of the Markov chain was part of the boundary level. That is, the i-th component of some vector α_{ini} gave the initial probability of being in the i-th boundary state at time 0. An approach to deal with more general initial conditions was included in [14]. This approach can even be extended to include general (bounded) initial distributions, meaning, the set of all possible initial states does not need to be a subset of a specific level. Although useful, this approach is restrictive in the sense that all computations need to be redone when considering a different initial state (distribution). This is especially problematic when we wish to compute transient results for each possible initial state as is often a measure of interest in the area of model checking [12]. Although the material presented in this paper is for discrete-time systems only, it should be fairly easy to adapt it for continuous time systems.

In this paper we provide a novel approach to deal with more general initial conditions. Actually, a level based recursive algorithm is proposed that computes the transient performance measure for all possible initial states. That is, having performed the necessary computations when the initial state is part of the first $r-1$ levels, we demonstrate how to obtain the transient measure when the initial state is part of level r (for each of the states belonging to level r). During this step we heavily rely upon previously computed intermediate results, avoiding the need to redo all computations for each new initial state considered. Notice, the recursion is *not* on the time epoch, as in many other studies, but on the level of the initial state. Furthermore, if we know the transient measure for any single initial state (up to some level N), the results are readily available for any initial distribution (bounded to the first N levels). A flexible Matlab implementation of the algorithm, taking among others the 6 matrices that characterize the QBD with marked time epochs as its input, is available online.

To some extent, the contribution of this paper is related to the following. Many researchers that need to compute the system state at time n for some discrete time Markov chain, will often take the initial probability vector and repeatedly multiply this with the transition matrix. Such an approach is fruitful, but the computation needs to be repeated each time we consider a different initial distribution. This can, in principle, be avoided by computing the n-th power of the transition matrix, but in practice this is often too time consuming. Our approach, although completely different in methodology, can intuitively be understood as intermediate to these two approaches. We compute the state for some initial vectors and use some of the intermediate results, to drastically speed-up further computations.

The paper is structured as follows. We start by giving some background information on QBDs with marked time epochs and we introduce the reset Markov chain of interest (Section 2). In Section 3 we convert this reset process into a level dependent QBD with a generalized boundary condition. A key role for the computation of the steady state probabilities of this QBD is played by a set of R-matrices [6] that can be computed recursively as demonstrated in Section 3.1. Next, the steady state probabilities are obtained from these R-matrices (Section 3.2) and to summarize, an algorithmic overview is given in Section 3.3. In Section 4 we illustrate our algorithm with some numerical examples.

2. QBDS WITH MARKED TIME EPOCHS: A REVIEW

We are interested in the transient behavior of a QBD Markov chain (MC) characterized by the transition matrix

$$\bar{P} = \begin{bmatrix} \bar{B}_1 & \bar{B}_0 & 0 & 0 & \cdots \\ \bar{B}_2 & \bar{A}_1 & \bar{A}_0 & 0 & \ddots \\ 0 & \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \qquad (1)$$

where $\bar{B}_i$ and $\bar{A}_i$ are square matrices of dimension h, for $i = 0, 1$ and 2. The state space of this infinite Markov chain is partitioned into an infinite number of sets, which we call levels. Transitions among the levels are described by the $\bar{B}_i$ and $\bar{A}_i$ matrices. To avoid the notations from becoming unnecessarily complex, we assume that level zero of the QBD has the same number of states as all other levels. However, the method described in this paper does not rely on this assumption and therefore may be relaxed if needed. Our Matlab tool does not have this restriction, as illustrated in Section 4. Furthermore, we denote the states of this MC as $\langle i, j\rangle$, where $i \geq 0$ denotes the level and $1 \leq j \leq h$ identifies the state within the level.

To obtain various transient performance measures in a unified manner, QBDs with marked time epochs (QBD^m) were introduced in [15]. Such a QBD^m is fully characterized by two sets of nonnegative matrices: a set with superscript ^{m} and one where all matrices have superscript ^{u}. These matrices have the additional property that the matrices defined as

$$\bar{B}_i = \bar{B}_i^u + \bar{B}_i^m, \quad \bar{A}_i = \bar{A}_i^u + \bar{A}_i^m,$$

for $i = 0, 1$ and 2, characterize a QBD Markov chain. The probabilistic interpretation is as follows. The $(j, j')^{th}$ entry of the matrix $\bar{A}_i^m$ contains the probability that at time t a transition occurs from state j of level s to state j' of level $s - i + 1$ and time epoch t is *marked*. The probabilities of the corresponding events without marking time t are given by the matrix $\bar{A}_i^u$.

Based on the transient performance measure we want to study, we mark part of the time epochs. For example, to obtain the system state at time n, we simply mark all time epochs. To compute the waiting time of the n^{th} customer in some queueing system that can be modeled as a QBD, we mark each time epoch in which an arrival occurs, etc. For more examples, see [14] and [15]. Any transient problem that can be formulated in terms of the n-th marking of a QBD^m, can be solved in a plug-and-play manner. The technique used to obtain the system state at the n-th marking consists of two steps.

Step 1: Discrete Erlangization. Denote $\pi^m(n)$ as the probability vector associated with the n^{th} marked time epoch, i.e., $\pi^m(n) = (\pi_0^m(n), \pi_1^m(n), \ldots)$, where $\pi_i^m(n)$ is of size h for $i \geq 0$. To speed-up the computations, the system state at the n^{th} marking $t^m(n)$ is approximated by considering the system state at the $Z_{k,n}$-th marked time epoch $t^m(Z_{k,n})$, where $Z_{k,n}$ is a negative binomially distributed (NBD) random variable with k phases and mean n, for k ($\leq n$) sufficiently large. The larger k, the lower the variation of $Z_{k,n}$ and the better the approximation becomes. Setting $k = n$ provides us with exact results, however, k cannot always be set equal to n as the reset MC might become periodic.

To compute the system state at time $t^m(Z_{k,n})$, an expanded Markov chain, called a reset Markov chain, was introduced. The key feature of such a Markov chain is that it reformulates the transient problem of computing the state at time $t^m(Z_{k,n})$ into a steady state analysis. Using the NBD as a reset time implies, among others, that the transition blocks of this reset MC have a special structure that can be exploited when computing its steady state probabilities.

Step 2: Reset Markov chains. Consider the stochastic process that evolves according to the transition matrix $\bar{P}$, but that is repeatedly reset when leaving the $Z_{k,n}$-th marked time epoch. A reset event corresponds to a transition from the current state to the initial state of the Markov chain. Denote the initial level as r and the initial state within this level as l. A reset event therefore corresponds to a transition from the current state to state $\langle r, l\rangle$. If we perform a Bernoulli trial with parameter $p = k/n$ each time we have a transition out of a marked time epoch, the system is reset whenever k successes have occurred. We define the reset counter as being the number of pending successes before the next reset event. It is clear that this reset counter takes values in the range $\{1, 2, \ldots, k\}$. We will add the reset counter as an auxiliary variable to the Markov chain characterized by $\bar{P}$ and label its states as (c, j) with $1 \leq c \leq k$ and $1 \leq j \leq h$. As explained further on, if we succeed in computing the stationary behavior of the expanded MC, we can readily compute the system state just prior to a reset event, which is exactly the system state at time $t^m(Z_{k,n})$.

The work presented in [15] was restricted to QBDs whose initial state was part of level zero. Although the algorithm in [14], for more general initial conditions, is easily extended to the QBD^m framework, it requires that all computations need to be redone when considering a different initial state. In this paper we therefore present an efficient algorithm to obtain transient performance measures for every possible initial configuration, without the need to repeat all computations for each configuration. We assume that at time zero the QBD resides in state l ($1 \leq l \leq h$) of level r. The reset process is characterized by the transition matrix $P_{k,n}$:

$$P_{k,n} = \mathcal{Q}_{k,n} + \mathcal{C}_{k,n}, \tag{2}$$

with

$$\mathcal{Q}_{k,n} = \begin{bmatrix} B_1^{k,n} & B_0^{k,n} & 0 & 0 & \cdots \\ B_2^{k,n} & A_1^{k,n} & A_0^{k,n} & 0 & \ddots \\ 0 & A_2^{k,n} & A_1^{k,n} & A_0^{k,n} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{3}$$

$$\mathcal{C}_{k,n} = \begin{bmatrix} 0 & \cdots & 0 & C_0^{k,n} & 0 & \cdots \\ 0 & \cdots & 0 & C_1^{k,n} & 0 & \cdots \\ 0 & \cdots & 0 & C_1^{k,n} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{4}$$

where the matrices $C_0^{k,n}$ and $C_1^{k,n}$ appear on the $(r+1)^{th}$ block column and where

$$A_i^{k,n} = (I_k \otimes (\bar{A}_i^u + (1-p)\bar{A}_i^m)) + (M_0^k \otimes p\bar{A}_i^m) \tag{5}$$

$$B_i^{k,n} = (I_k \otimes (\bar{B}_i^u + (1-p)\bar{B}_i^m)) + (M_0^k \otimes p\bar{B}_i^m) \tag{6}$$

$$C_0^{k,n} = M_1^k \otimes (p(\bar{B}_0^m e + \bar{B}_1^m e)\alpha_l) \tag{7}$$

$$C_1^{k,n} = M_1^k \otimes (p\bar{A}^m e\alpha_l), \tag{8}$$

for $i = 0$, 1 or 2 and $\bar{A}^m = \bar{A}_0^m + \bar{A}_1^m + \bar{A}_2^m$. The $k \times k$ matrix M_0^k has ones on the first diagonal below its main diagonal and all other entries equal to zero, while M_1^k has only one entry differing from zero being its last entry on the first row, which equals one. Further, e is a column vector with all its entries equal to one and α_l is a $1 \times h$ row vector with all entries equal to zero, except for the l^{th} entry which equals one.

Let $\pi^{k,n} = (\pi_0^{k,n}, \pi_1^{k,n}, \pi_2^{k,n}, \ldots)$ be the steady state vector of $P_{k,n}$. Moreover, let $\pi_i^{k,n} = (\pi_{i,1}^{k,n}, \ldots, \pi_{i,k}^{k,n})$, partitioned in the obvious manner. Provided that the vector $\pi^{k,n}$ exists, the system state $\pi^m(Z_{k,n})$ at time $t^m(Z_{k,n})$, used as an approximation to $\pi^m(n)$, is the stochastic vector proportional to $(\pi_{0,1}^{k,n}.\phi_0, \pi_{1,1}^{k,n}.\phi_1, \pi_{2,1}^{k,n}.\phi_1, \ldots)p$. Here, ϕ_0 and ϕ_1 are the transposed vectors of $\left(\bar{B}_0^m e + \bar{B}_1^m e\right)$ and $\bar{A}^m e$, respectively and '.' denotes the point-wise vector product. That is, a reset will occur if the value of the reset counter equals one and the Bernoulli trial with parameter p results in a success. To compute the steady state vector of $P_{k,n}$ we shall construct a level dependent QBD Markov chain with a generalized initial condition of which the transition matrix is denoted by P_{QBD}. It is in this respect that the current paper differs from the approach taken in [14], where a different QBD reduction method was used to compute $\pi^{k,n}$. The novel construction allows us to simplify the calculation of the steady state vectors significantly when computing $\pi^{k,n}$ for every initial setting $\langle r, l\rangle$ with $r \geq 0$ and $1 \leq l \leq h$.

3. QBD REDUCTION

In this section, we show how to convert the reset MC, introduced Step 2 of Section 2, into a level dependent QBD with a generalized boundary condition. For this purpose, we introduce h additional states to each of the levels $0, \ldots, r$ and we refer to these states as artificial states. Meaning, we increase the number of states for each of the first $r+1$ levels by a factor $(k+1)/k$. When a reset event occurs, the reset Markov chain characterized by $P_{k,n}$ makes a transition to state $\langle r, (k, l)\rangle$. In the reduced QBD approach, we split such a transition into $r+2$ steps:

- **Step 1:** A transition to artificial state l of level zero takes place.
- **Step 2 to r+1:** Next, transitions between artificial state l of level i and artificial state l of level $i+1$ will follow, for $i = 0, \ldots, r-1$.
- **Step r+2:** Finally, when the state equals artificial state l of level r, a transition to the non-artificial state (k, l) of level r is made.

Before we discuss the transition matrix, let us reflect a bit on the reduction choices made. The key property of this reduction is that every reset event causes the MC to jump to level zero. Thus, as opposed to the technique used in [14], even if the reset event occurs at some level s, with $0 < s < r$, the path to state $\langle r, (k, l)\rangle$ goes through level zero. Therefore, the MC becomes level independent starting from level $r+1$ and requires no artificial states on the levels $s > r$.

The steady state vector of a QBD that becomes level independent at some level can be determined by a finite set of R-matrices [6]. By immediately selecting state l while visiting level zero, a single set of R-matrices suffices to compute the transient performance measures for all h initial states of the form $\langle r, l\rangle$, with $1 \leq l \leq h$. Furthermore, these R-matrices can be reused when we look at initial states of the form $\langle r', l\rangle$, for $r' > r$. More specifically, when we increase r by one, the proposed reduction method only requires us to compute one additional R-matrix. This is a significant improvement over [14], where the entire set of R-matrices had to be recomputed for each new level. Finally, we shall see that these R-matrices have a useful block-structure and are identical to one another, except for the last block row.

It is possible to shorten the reduction procedure to $r+1$ steps by removing the h artificial states from level r (and by returning to a non-artificial state when going from level $r-1$ to level r). This causes no true performance gain (even though we need one R-matrix less), however, for uniformity reasons we decided to include step $r+2$.

The state space of the constructed QBD MC is organized such that whenever we visit an artificial state, we temporarily set the reset counter equal to $k+1$. This gives rise to the transition matrix P_{QBD} presented in Equation 9. In this equation $A_i^{>r} = A_i^{k,n}$, for $i = 0, 1, 2$,

$$P_{QBD} = \begin{bmatrix} B_1^r + C_0^r & B_0^r & 0 & \cdots & 0 & 0 & 0 & \cdots \\ B_2^r + C_1^{\leq r} & A_1^{<r} & A_0^{<r} & \cdots & 0 & 0 & 0 & \cdots \\ C_1^{\leq r} & A_2^{\leq r} & A_1^{<r} & \ddots & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots \\ C_1^{\leq r} & 0 & 0 & \ddots & A_1^{<r} & A_0^{<r} & 0 & \cdots \\ C_1^{\leq r} & 0 & 0 & \ddots & A_2^{\leq r} & A_1^r & A_0^r & \cdots \\ C_1^{>r} & 0 & 0 & \ddots & 0 & A_2^{r+1} & A_1^{>r} & \cdots \\ C_1^{>r} & 0 & 0 & \ddots & 0 & 0 & A_2^{>r} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{9}$$

$$B_0^r = \begin{bmatrix} B_0^{k,n} & 0 \\ 0 & I_h \end{bmatrix}, \quad A_0^{<r} = \begin{bmatrix} A_0^{k,n} & 0 \\ 0 & I_h \end{bmatrix},$$

$$A_0^r = \begin{bmatrix} A_0^{k,n} \\ 0 \end{bmatrix}, \quad B_1^r = \begin{bmatrix} B_1^{k,n} & 0 \\ 0 & 0 \end{bmatrix},$$

$$A_1^{<r} = \begin{bmatrix} A_1^{k,n} & 0 \\ 0 & 0 \end{bmatrix}, \quad A_1^r = \begin{bmatrix} A_1^{k,n} & 0 \\ J & 0 \end{bmatrix},$$

$$B_2^r = \begin{bmatrix} B_2^{k,n} & 0 \\ 0 & 0 \end{bmatrix}, \quad A_2^{\leq r} = \begin{bmatrix} A_2^{k,n} & 0 \\ 0 & 0 \end{bmatrix},$$

and

$$A_2^{r+1} = \begin{bmatrix} A_2^{k,n} & 0 \end{bmatrix},$$
$$C_0^r = M_1^{k+1} \otimes (p(\bar{B}_0^m e + \bar{B}_1^m e)\alpha_l),$$
$$C_1^{\leq r} = M_1^{k+1} \otimes (p\bar{A}^m e\alpha_l),$$
$$C_1^{>r} = M_2^{k+1} \otimes (p\bar{A}^m e\alpha_l).$$

In these equations J is an $h \times hk$ matrix given by

$$J = \begin{bmatrix} 0 & \cdots & 0 & I_h \end{bmatrix}. \tag{10}$$

The matrix M_2^{k+1} is a $k \times (k+1)$ matrix with all entries equal to zero, except for the last entry on the first row which equals one.

A key role in the computation of the invariant vector $\hat{\pi}$ of P_{QBD} is played by a set of R-matrices [6], which we denote as $R_0\{r\}, R_1\{r\}, \ldots, R_r\{r\}$ and $R_{>r}\{r\}$ (see Section 3.2 for details). We briefly outline the case in which the Markov chain starts in state l of level zero, as this case was already discussed in detail in [15].

3.1 Computing the R-matrices

3.1.1 Reset to level zero

Recall that there is no need to add step $r+2$ in the reduction process, which explains why there are no artificial states in [15], where the reset level equals zero. We first use the algorithm described in [15, Section 5] to solve the equation

$$R = A_0^{>r} + RA_1^{>r} + R^2 A_2^{>r}. \tag{11}$$

This algorithm reduces the above quadratic matrix equation involving large matrices (size kh) to a single quadratic matrix equation and a set of Sylvester matrix equations of a much smaller dimension (size h). The Cyclic Reduction (CR) algorithm [9, 1] can be used to solve the quadratic matrix equation, whereas a solution to the Sylvester equations can be found using a Hessenberg algorithm [4]. This reduction can be applied because the matrices $A_i^{>r}$, for $i = 0, 1$ and 2, have a block triangular block Toeplitz (btbT) structure. A btbT matrix X is fully characterized by its first block column as follows:

$$X = \begin{bmatrix} X_1 & 0 & \cdots & 0 & 0 \\ X_2 & X_1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ X_{k-1} & X_{k-2} & \ddots & X_1 & 0 \\ X_k & X_{k-1} & \cdots & X_2 & X_1 \end{bmatrix}. \tag{12}$$

Actually, only X_1 and X_2 differ from zero in case of the $A_i^{>r}$ matrices. Since the btbT structure is preserved by the matrix multiplication, R also has a btbT structure, which allows us to compute the matrix R in a time and memory complexity of $O(h^3k^2)$ and $O(h^2k)$, respectively. Next, the matrix $R_0\{0\}$ defined as

$$R_0\{0\} = B_0^{k,n}(I - A_1^{>r} - RA_2^{>r})^{-1}, \tag{13}$$

can be computed efficiently by exploiting the btbT structure.

3.1.2 Reset to an arbitrary level $r > 0$

The fundamental idea of our algorithm is to compute the steady state probabilities for the other initial configurations recursively. To achieve this, we establish a simple recursion on r among the R-matrices $R_0\{r\}, R_1\{r\}, \ldots, R_r\{r\}$ and $R_{>r}\{r\}$ needed to obtain the steady state vector of interest. Thus, after obtaining the two R-matrices R and $R_0\{0\}$ for level zero, we continue with the R-matrices needed when the initial level equals one, two, etc. The recursion can be understood by looking at the probabilistic interpretation of the R-matrices involved. More specifically, the $(u, v)^{th}$ entry of the matrix $R_i\{r\}$ denotes the expected number of visits to state v of level $i+1$, starting from state u of level i under taboo of the levels $0, \ldots, i$ and under the assumption that the reset level of the Markov chain equals r.

As explained below the following equalities hold:

$$R_i\{r\} = \begin{bmatrix} R & 0 \\ JG^{r-i-1}(I_{kh}-U)^{-1} & I_h \end{bmatrix}, \quad (14)$$

$$R_r\{r\} = \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad (15)$$

$$R_{>r}\{r\} = R. \quad (16)$$

for $r \geq 1$ and $0 < i < r$. The btbT matrices U and G are given by

$$U = A_1^{k,n} + RA_2^{k,n}, \quad (17)$$

$$G = (I_{kh}-U)^{-1}A_2^{k,n}. \quad (18)$$

The expression for $R_{>r}\{r\}$ is immediate, as the expected number of visits to a state of level $i+1$ under taboo of the levels $0,\ldots,i$ does not depend on the actual value of $i > r$ since the transition matrix P_{QBD} is level independent from level $r+1$ onwards.

The equality for $R_i\{r\}$, for $i = 1,\ldots,r$, can be justified in two steps. First, consider the situation in which the starting state u is a non-artificial state of level i. All sample paths that avoid levels 0 to i do not contain a reset event, as all reset events result in a visit to level zero. Hence, all these paths evolve according to the matrices $A_0^{k,n}, A_1^{k,n}$ and $A_2^{k,n}$. Therefore, there are no visits to any of the artificial states of level $i+1$, while the expected number of visits to level $i+1$ is determined by R.

Second, if the starting state u is one of the h artificial states of level i, the next $r-i+1$ transitions are fixed. The first $r-i$ transitions are between the two corresponding artificial states of level j and $j+1$, for $i \leq j < r$, and the last transition between an artificial state l and its corresponding non-artificial state (k,l) both of level r. The expected number of visits to level $i+1$ will therefore solely depend on the number of levels between i and r. The (u,v)-th entry of G gives us the probability that the first visit to level i is a visit to state v, provided that we started in state u of level $i+1$, without the occurrence of a reset event. Therefore, the first visit to level $i+1$ under taboo of the levels 0 to i, starting from a state of level r is determined by the matrix G^{r-i-1}. The (w,w')-th entry of the matrix $(I_{kh}-U)^{-1}$ holds the expected number of visits to state w' of level $i+1$, starting from state w of level $i+1$, under taboo of level 0 to i. Furthermore, starting from an artificial state of level $i < r$ implies that we visit exactly one artificial state of level $i+1$ before returning to level zero (hence, the I_h matrix in the lower right corner).

The matrix that remains to be determined is $R_0\{r\}$. In the case where the starting state u of level zero is not an artificial state, all sample paths that avoid level zero evolve according to the matrices $A_0^{k,n}, A_1^{k,n}$ and $A_2^{k,n}$, except for the first transition which is characterized by the matrix B_0^r. As a result, the expected number of visits to level 1 are determined by $R_0\{0\}$. If, on the other hand, the starting state is an artificial start, we may rely on the same arguments as for $i > 0$. Hence,

$$R_0\{r\} = \begin{bmatrix} R_0\{0\} & 0 \\ JG^{r-1}(I_{kh}-U)^{-1} & I_h \end{bmatrix}. \quad (19)$$

Equations (14) and (19) naturally lead to a simple recursion as

$$JG^{(r+1)-i-1}(I_{kh}-U)^{-1} = JG^{r-i-1}(I_{kh}-U)^{-1}\left(A_2^{k,n}(I_{kh}-U)^{-1}\right). \quad (20)$$

In conclusion, to compute all the R-matrices needed to assess the transient performance for all initial states $\langle r',l\rangle$ with $r' \leq r$, it suffices to compute

1. The first block column of the btbT matrices R and $R_0\{0\}$.
2. The last block row of the r matrices $R_0\{r\},\ldots,R_{r-1}\{r\}$, which can be found recursively using a single btbT product (see (20)). Notice, $R_i\{r'\}$ equals $R_{i+r-r'}\{r\}$ for $r' \leq r$.

3.2 Computing the steady state probabilities

3.2.1 Reset to level zero

This section briefly describes how to obtain the steady state probability vector $\hat{\pi}\{0,l\}$ of P_{QBD} when the initial state $\langle r,l\rangle = \langle 0,l\rangle$, for $1 \leq l \leq h$, using the matrices R and $R_0\{0\}$:

$$\hat{\pi}_0\{0,l\} = \hat{\pi}_0\{0,l\}(B_1^{k,n} + C_0^{k,n} + R_0\{0\}(B_2^{k,n} + (I-R)^{-1}C_1^{k,n})), \quad (21)$$

$$\hat{\pi}_1\{0,l\} = \hat{\pi}_0\{0,l\}R_0\{0\}, \quad (22)$$

$$\hat{\pi}_i\{0,l\} = \hat{\pi}_{i-1}\{0,l\}R, \quad (23)$$

for $i > 1$, while $\hat{\pi}_0\{0,l\}$ and $\hat{\pi}_1\{0,l\}$ are normalized as $\hat{\pi}_0\{0,l\}e + \hat{\pi}_1\{0,l\}(I-R)^{-1}e = 1$. The matrix in Eq. (21) of which $\hat{\pi}_0\{0,l\}$ is an eigenvector with eigenvalue one, is the sum of a btbT matrix and a matrix with all entries equal to zero, except for the last block column. In order to compute $\hat{\pi}_0\{0,l\}$ efficiently, we can therefore rely on the algorithm presented in [15, Section 5.2], the time and space complexity of which equal $O(h^3k^2)$ and $O(h^2k)$, respectively. This algorithm solves a linear system of hk equations of the following form:

$$\hat{\pi}_0\{0,l\} = \hat{\pi}_0\{0,l\}\begin{bmatrix} Y_1 & 0 & \ldots & 0 & Z_1 \\ Y_2 & Y_1 & \ddots & 0 & Z_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ Y_{k-1} & Y_{k-2} & \ldots & Y_1 & Z_{k-1} \\ Y_k & Y_{k-1} & \ldots & Y_2 & Y_1+Z_k \end{bmatrix}.$$

Instead of plugging in the h different α_l vectors in the algorithm presented in [15, Section 5.2] (where the vector α_l has a one on the l-th position and zeros elsewhere), we can actually compute the h vectors $\hat{\pi}_0\{0,l\}$ simultaneously as follows.

The algorithm in [15] first computes a set of matrices $G_1,\ldots,G_k$, which is by far the most expensive part of the algorithm. However, changing the initial state $\langle 0,l\rangle$ does not alter the matrices $G_2,\ldots,G_k$, but only affects G_1. Moreover, keeping in mind that G_1 has to be stochastic, one can show that

$$G_1 = Y_1 + \alpha_l \otimes (e - Y_1 e), \quad (24)$$

where Y_1 does not depend on l. The remaining steps of the algorithm to obtain the $\hat{\pi}_0\{0,l\}$ vector from $G_1,\ldots,G_k$

remain identical. Notice that the matrices R and $R_0\{0\}$ do not depend on the initial state l and need to be computed only once.

3.2.2 Reset to an arbitrary level $r > 0$

Having obtained the matrices $R_0\{r\}, \ldots, R_r\{r\}$ and R, we immediately have the following expression for the steady state probability vector $\hat{\pi}$ of P_{QBD} (for the system where the reset level equals $r > 0$), due to [6]:

$$\hat{\pi}_i\{r,l\} = \begin{cases} \hat{\pi}_{i-1}\{r,l\}R_{i-1}\{r\} & \text{for } i = 1,\ldots,r+1, \\ \hat{\pi}_{i-1}\{r,l\}R, & \text{for } i > r+1. \end{cases} \tag{25}$$

Thus, it suffices to determine $\hat{\pi}_0\{r,l\}$, which obeys the following equation:

$$\hat{\pi}_0\{r,l\} = \hat{\pi}_0\{r,l\}N, \tag{26}$$

with

$$N = B_1^r + C_0^r + R_0\{r\}B_2^r + R_S^r C_1^{\leq r} + R_P^r (I-R)^{-1} C_1^{>r}, \tag{27}$$

and

$$R_S^r = \sum_{k=0}^{r-1} \left(\prod_{i=0}^{k} R_i\{r\} \right), \qquad R_P^r = \left(\prod_{i=0}^{r} R_i\{r\} \right). \tag{28}$$

To speed-up the computation of $\hat{\pi}_0\{r,l\}$, for $1 \leq l \leq h$, we exploit the structure of the square N matrix, the dimension of which equals $h(k+1)$. It might seem odd that we are considering a system of $h(k+1)$ linear equations, instead of just $hk+1$ as only one of the last h entries of $\hat{\pi}_0\{r,l\}$ will differ from zero (being the l-th). However, considering this somewhat larger N matrix, enables us to compute the h different $\hat{\pi}_0\{r,l\}$ vectors simultaneously.

Write the matrix N as

$$N = \begin{bmatrix} N^B & N^C \end{bmatrix}, \text{ with } N^B = \begin{bmatrix} N^T \\ N_r^R \end{bmatrix},$$

where N^T is an $hk \times hk$ matrix corresponding to transitions between non-artificial states, N_r^R an $h \times hk$ matrix and N^C an $h(k+1) \times h$ matrix.

Only the last block column of $C_0^r, C_1^{\leq r}$ and $C_1^{>r}$ differs from zero; therefore, N^T is a btbT matrix given by

$$N^T = B_1^{k,n} + R_0\{0\}B_2^{k,n}, \tag{29}$$

meaning N^T does not depend on the initial level r (and state l) of the QBD and needs to be computed only once. The block row N_r^R does depend on the initial level r and can be computed as

$$N_r^R = JG^{r-1}(I_{kh} - U)^{-1} B_2^{k,n}. \tag{30}$$

As we increase the initial level r, this matrix changes. However, we can easily compute it recursively by relying on Equation (20) with $i = 0$. The matrix N^C will only contain one non-zero column, which does not need to be computed to determine $\hat{\pi}_0\{r,l\}$.

Let us now discuss how the vectors $\hat{\pi}_0\{r,l\}$, for $1 \leq l \leq h$, can be obtained simultaneously from the matrices N^T and N_r^R. Denote N_j^T as the j-th $h \times h$ block appearing on the first block column of the btbT matrix N^T and denote $N_{r,j}^R$ as the j-th block of the block vector N_r^R. The algorithm discussed below is a variant of the one in [15, Section 5] used to compute the invariant vector of a matrix of the form seen in Section 3.2.1. N has a slightly different structure, causing some minor modifications to the algorithm, however as in [15, 5] it stems from a repeated censoring argument. First, we compute the k matrices G_2 until G_{k+1}:

1. Set $G_i = N_{r,k-i+2}^R$, for $i = 2, \ldots, k+1$,
2. for $i = 3$ to $k+2$ do
   $G_{i-1} = G_{i-1}(I_h - N_1^T)^{-1}$
   for $j = i$ to $k+1$ do
   $G_j = G_j + G_{i-1}N_{j-(i-1)+1}^T$
   end
   end

The time and memory complexity of this step equals $O(h^3k^2)$ and $O(h^2k)$ respectively. The vectors

$$\hat{\pi}_0\{r,l\} = (\hat{\pi}_{0,1}\{r,l\}, \ldots, \hat{\pi}_{0,k+1}\{r,l\})$$

can be retrieved from these matrices as follows:

1. $\hat{\pi}_{0,k+1}\{r,l\} = \alpha_l$,
2. for $i = 2$ to $k+1$
   $\hat{\pi}_{0,k-i+2}\{r,l\} = \alpha_l G_i$
   end

Recall, α_l is a $1 \times h$ vector with a one in position l and zeros elsewhere. Thus, $\alpha_l G_i$ corresponds to the l-th row of the matrix G_i.

Finally, due to Equation (25), the vector $\hat{\pi}_0\{r,l\}$ is normalized as

$$\hat{\pi}_0\{r,l\}e + \hat{\pi}_0\{r,l\}\left(R_S^r e + R_P^r (I-R)^{-1} e\right) = 1. \tag{31}$$

Remark that for the btbT part of the matrices appearing in this equation, we only store the first block column.

Having obtained the steady state vector $\hat{\pi}\{r,l\}$ of P_{QBD}, we can derive $\pi^{k,n}\{r,l\}$, the invariant vector of $P_{k,n}$ as follows. Write $\hat{\pi}_j\{r,l\}$, for $j = 0, \ldots, r$, as $(\hat{\pi}_j^f\{r,l\}, \hat{\pi}_j^{art}\{r,l\})$, where $\hat{\pi}_j^{art}\{r,l\}$ is a $1 \times h$ vector. Notice, there are no artificial states from level $r+1$ onwards, therefore we may write $\hat{\pi}_j\{r,l\} = \hat{\pi}_j^f\{r,l\}$, for $j > r$. When censored on the non-artificial states, P_{QBD} coincides with $P_{k,n}$, consequently,

$$\pi_j^{k,n}\{r,l\} = \hat{\pi}_j^f\{r,l\}/(1-c), \tag{32}$$

where $j \geq 0$ and $c = \sum_{j=0}^{r} \hat{\pi}_j^{art}\{r,l\}$. An approximation for $\pi_m(n)$, the system state at the n^{th} marking can now be computed as described in Step 2 of Section 2.

The algorithm discussed in this section allows us to compute the steady state vectors $\pi^{k,n}\{r',l\}$, for $r' \leq r$ and $1 \leq l \leq h$, from the matrices $R_0\{r\}, \ldots, R_r\{r\}$ and R. As we are looking at a transient event, choosing r sufficiently large, guarantees that $\pi_{0,1}^{k,n}\{r,l\}$ decreases to zero, because the probability of reaching level zero before a reset occurs decreases to zero. Thus, for r sufficiently large, $\pi^{k,n}\{r+1,l\}$ can be obtained from $\pi^{k,n}\{r,l\}$ by shifting its subvectors by

one position. To determine the transient behavior for all initial states, we dynamically increase r until $\pi_{0,1}^{k,n}\{r,l\}$ is negligible or until a predefined upper bound on r is reached. It should be noted that this idea is somewhat similar to the technique used in [13].

3.3 Algorithmic overview

To conclude this section, we will present a short algorithmic overview of the discussed results. To obtain the transient performance measures of interest for every initial setting $\langle r,l\rangle$, implement the following steps:

1. Mark time epochs in correspondence with the desired performance measure and determine the matrices introduced in (9), using the equations (5) – (8).[1]
2. Compute the matrices R from (11) and $R_0\{0\} = B_0^{k,n}(I - A_1^{>r} - RA_2^{>r})^{-1}$.
3. Use (21) – (23) to compute the steady state vector $\hat{\pi}\{0,l\}$. This can be done simultaneously for $l = 1,\ldots,h$ as described in Section 3.2.1.
4. Compute $N^T = B_1^{k,n} + R_0\{0\}B_2^{k,n}$ and $U = A_1^{k,n} + RA_2^{k,n}$.
5. Set $R_0^*\{1\} = J(I_{kh} - U)^{-1}$.
6. For each initial level $r > 0$ do
   - If $r > 1$ calculate
   $$R_0^*\{r\} = R_0^*\{r-1\}\left(A_2^{k,n}(I_{kh} - U)^{-1}\right).$$
   - Set $N_r^R = R_0^*\{r\}B_2^{k,n}$.
   - Use the algorithm described in Section 3.2.2 to obtain $\hat{\pi}_0\{r,l\}$ from N^T and N_r^R simultaneously for $1 \le l \le h$.
   - Normalize the vector $\hat{\pi}_0\{r,l\}$ using Eq. (31), where R_S^r and R_P^r can be computed recursively from R_S^{r-1} and R_P^{r-1} as indicated by Eq. (28).
   - Compute $\hat{\pi}_i\{r,l\} = \hat{\pi}_{i-1}\{r,l\}R_{i-1}\{r\}$ for $0 < i \le r+1$ and $\hat{\pi}_i\{r,l\} = \hat{\pi}_{i-1}\{r,l\}R$ for $i > r+1$, where the R-matrices are described in Section 3.1.2.[2]
   - Finally, $\pi^{k,n}\{r\}$ can be found from Eq. (32).

   until $\pi_{0,1}^{k,n}\{r,l\}$ is negligible or until a predefined upper bound on r is reached.

4. NUMERICAL EXAMPLES

In this section we will demonstrate our approach with some numerical examples and compare the efficiency with the algorithm presented in [14]. Therefore, we will compute the queue length distribution of a D-MAP/PH/1 queue at some time t as well as the waiting time distribution of the n-th customer in this queue.

[1]Note that for every matrix with a btbT structure only the first block column has to be stored.

[2]The rows of $R_i\{r\}$ corresponding to the artificial states are given by $[R_0^*\{r-i\}\ I_h]$. Also, the product between a vector and a btbT matrix can be implemented using fast Fourier transforms, see [15, Section 5.3], as the remaining rows of $R_{i-1}\{r\}$ are identical to R, this results in a significant gain in time.

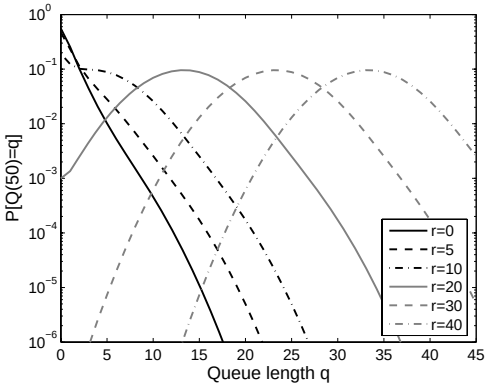

Figure 1: Queue length distributions of the D-MAP/PH/1 queue with $t = 50$, $\alpha = 1$

The D-MAP arrival process, introduced in [3], is the discrete time version of the Markovian arrival process, MAP [7, 8]. It is characterized by two $h \times h$ matrices D_0 and D_1, with h a positive integer. These matrices contain the transition probabilities of the underlying Markov chain when either a customer arrives (covered by D_1) or not (D_0). In this example we consider a 2-state D-MAP that generates an arrival with probability 0.1 while in state 1 and with probability 0.25 when the state of the underlying Markov chain equals 2. The average sojourn time is 500 slots in state 1 and 1000 slots in state 2, resulting in an arrival rate $\lambda = 0.2$. Hence,

$$D_0 = \begin{bmatrix} 0.89820 & 0.00180 \\ 0.00075 & 0.74925 \end{bmatrix},\ D_1 = \begin{bmatrix} 0.09980 & 0.00020 \\ 0.00025 & 0.24975 \end{bmatrix}.$$

Denote the initial state of the D-MAP process by α. The service time of a customer follows a discrete-time phase-type (PH) distribution with a matrix representation (m,β,T), where m is a positive integer. The vector β contains the probabilities that the service of a customer starts in a given phase. The probability that a customer continues his service in phase j at time $t+1$, given the phase at time t equals i, is represented by the $(i,j)^{th}$ entry of the $m \times m$ substochastic matrix T. For this example we take $m = 3$, $\beta = (3/4, 1/8, 1/8)$ and the matrix T is given by

$$T = \begin{bmatrix} 4/5 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}.$$

Furthermore, define $T^* = e - Te$. Remark, the mean service time equals $E[S] = \beta(I-T)^{-1}e = 4.1666$ slots, resulting in a total system load $\rho = 0.8333$. This gives rise to a QBD Markov chain, characterized by the transition matrix $\bar{P}$ (as in Equation (1)), with $\bar{B}_0 = D_1 \otimes \beta$, $\bar{B}_1 = D_0$, $\bar{B}_2 = D_0 \otimes T^*$, $\bar{A}_0 = D_1 \otimes T$, $\bar{A}_1 = (D_0 \otimes T) + (D_1 \otimes T^*\beta)$, and $\bar{A}_2 = D_0 \otimes T^*\beta$. Since we are interested in the system state at some time t, we mark every time instant, meaning $\bar{A}_i^m = \bar{A}_i$ and $\bar{B}_i^m = \bar{B}_i$, for $i = 0,1,2$. For a detailed description about how the queue length distribution of a D-MAP/PH/1 queue at some time t can be obtained using a QBD with marked

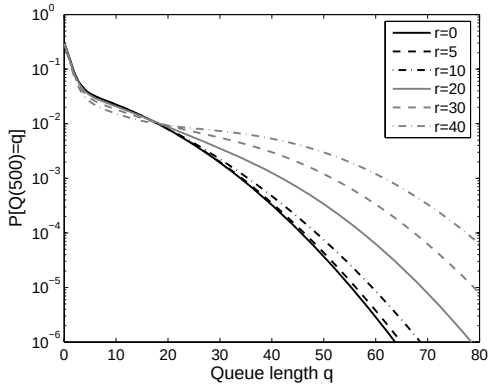

Figure 2: Queue length distributions of the D-MAP/PH/1 queue with $t = 500$, $\alpha = 1$

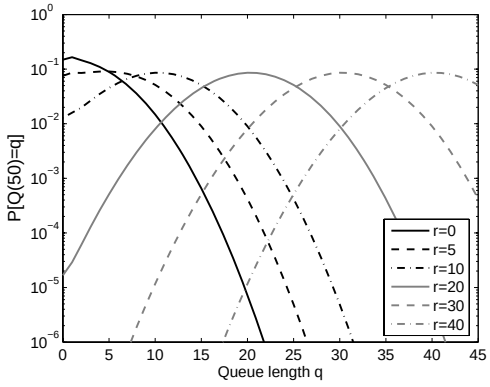

Figure 4: Queue length distributions of the D-MAP/PH/1 queue with $t = 50$, $\alpha = 2$

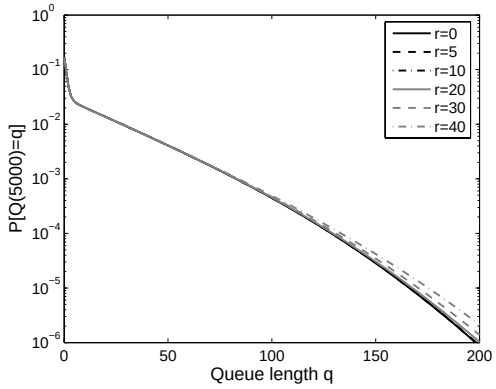

Figure 3: Queue length distributions of the D-MAP/PH/1 queue with $t = 5000$, $\alpha = 1$

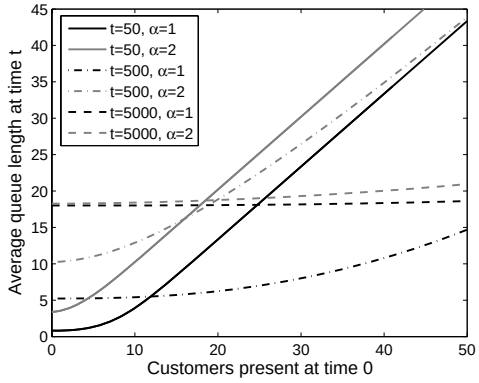

Figure 5: Average queue length of the D-MAP/PH/1 queue

time epochs, we refer to [14]. Notice, in this example, level 0 has fewer states than level $r > 0$, demonstrating that the algorithm in the paper is easily generalized to such cases.

Take $\alpha = 1$, then Figure 1 shows the queue length distribution for respectively $0, 5, 10, 20, 30$ and 40 customers in the system at time 0. If the system is initially not empty, it is assumed that the first customer starts his service at time 0. Thus, for $r > 0$, all curves plotted for $\alpha = 1$ or 2 are actually the weighted sum of three distributions, where the initial service phase equals 1, 2 and 3 and where the weights are given by β.

We observe a large influence of the number of customers originally present on the queue length distribution. As could be expected this influence becomes smaller for larger values of t as can be observed from Figures 2 and 3, for $t = 500$, resp. $t = 5000$, and $k = 100$. Remark that after choosing the values for t and k, we need to execute our recursive algorithm only once to obtain the results for every possible combination of r and α.

Figure 4 shows, for $t = 50$, the queue length distributions for the same number of customers in the system at time 0, but with $\alpha = 2$. Since the arrival rate in state 2 of the D-MAP is larger than the arrival rate in state 1, we observe higher probabilities for the larger queue lengths in this example. From Figure 5 it can be observed that for a small value of t the average queue length increases almost linearly with the number of customers present in the queueing system at time 0. In addition, the influence of the number of customers initially present decreases when the time t at which we evaluate the queue length increases.

A different behavior can be observed with relation to the initial state of D-MAP arrival process. Denote by $q_1^{r,t}$, resp. $q_2^{r,t}$, the average queue length at time t, given that r customers were present at time 0 and the initial state of the D-MAP process equals state 1, resp. state 2. Next, let us define $Diff^{r,t} = q_2^{r,t} - q_1^{r,t}$. From Figure 5 we notice for example that $Diff^{r,50} < Diff^{r,500}$, while $Diff^{r,500} > Diff^{r,5000}$, for $1 \leq r \leq 50$. A better insight in the behavior of $Diff^{r,t}$ for different values of r and t can be obtained by observing Figure 6. For $1 \leq r \leq 50$ we can see that the difference between $q_2^{r,t}$ and $q_1^{r,t}$ increases for small values of t and decreases later on. This effect can be explained by looking at the arrival

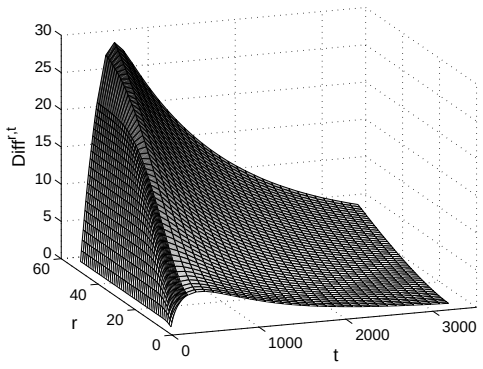

Figure 6: Difference $Diff^{r,t}$ between the average queue lengths $q_2^{r,t}$ and $q_1^{r,t}$

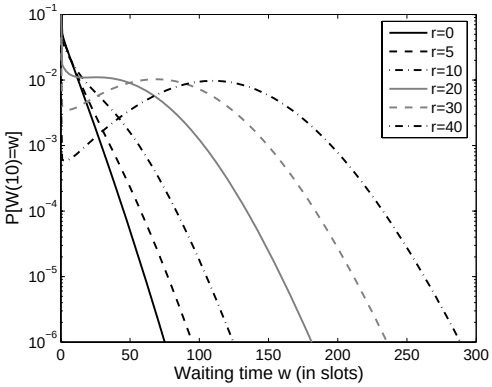

Figure 7: Waiting time distribution of the 10-th customer in a D-MAP/PH/1 queue

rates in both states of the D-MAP process. Recall, while in state 1, resp. state 2, there is an arrival in an arbitrary time slot with probability 0.1, resp. 0.25. For values of t close to zero, the difference $Diff^{r,t}$ will be relatively small since the average number of customers present at time t, is mainly influenced by r, the number of customers initially present. As long as there is no transition of the D-MAP arrival process, the difference between the average queue lengths $q_2^{r,t}$ and $q_1^{r,t}$ increases with t, due to the different arrival rates in the corresponding states. After a state transition of the D-MAP process, $Diff^{r,t}$ will start to decrease and eventually, this difference will diminish to zero when t approaches ∞ as the system has a steady state for $\rho = 0.8333$.

time	new algorithm		original algorithm		
t	R's	total	all states	level 50	$\langle 50, l\rangle$
50	0.46	9	2310	121	20
500	1.94	53	31595	1052	175
5000	1.95	133	33499	1080	180

Table 1: Execution times (s) for $r \leq 50$

In Table 1 the time needed to compute the queue length distribution for every initial configuration is compared with the execution time of the original algorithm [14]. The measurements were made using the Matlab Profiler on an Intel Pentium M 1.70GHz processor with 1GB of memory. We computed the queue length distribution at time $t = 50$, 500 and 5000, with a predefined upper bound of $r = 50$ on the initial level. For the new algorithm, the time needed to compute the R-matrices is presented as well as the total execution time. For the original algorithm we show the time needed to compute the queue length distributions for all initial configurations, for every state $1 \leq l \leq 6$ of level 50 and for a single initial configuration $\langle 50, l\rangle$.

As before, we set $k = t - 1$ for $t = 50$, for both $t = 500$ and $t = 5000$, k is set to 100. In the first scenario where $t = 50$, the algorithm stops after the initial level reached $r = 39$ because $\pi_{0,1}^{k,n}\{r, l\}$ becomes negligible for larger values of r. For $t > 50$ the predefined maximum of $r = 50$ was reached. The same stopping criterion was used by the original algorithm.

These results show that the computation time is reduced considerably by the new algorithm. The computation time of the level based recursion to compute the queue length distribution for every possible initial setting is even below the time the original algorithm needs for a single initial state. With the new algorithm, the dominant part of the computation lies in obtaining the $\hat{\pi}\{r, l\}$ vectors. This also explains the smaller difference between the total computation time of the new and the single state computation time of the old algorithm in the $t = 5000$ result, since a larger value of u is required such that $1 - \sum_{i=0}^{u} \hat{\pi}_i\{r, l\} < \epsilon$. Of course, if we were interested in just one single initial state $\langle 50, l\rangle$, the time consuming iteration for the $\hat{\pi}\{r, l\}$ vectors is performed only once (as opposed to the 302 times for all states up to level $r = 50$).

In the next example, we consider the same D-MAP/PH/1 queue, however we are now interested in the waiting time distribution of the n-th customer, which can also be obtained using the presented algorithm. To compute this waiting time it suffices to know the system state at the n-th arrival epoch. Therefore, we mark only those time epochs at which an arrival occurs (instead of marking all time epochs as needed to obtain the queue length distribution at some time t). We have $\bar{A}_0^m = \bar{A}_0$, $\bar{A}_1^m = D_1 \otimes T^* \beta$, $\bar{A}_2^m = 0$, $\bar{B}_0^m = \bar{B}_0$, $\bar{B}_1^m = 0$, and, $\bar{B}_2^m = 0$.

Executing our algorithm with these matrices gives us an approximation of the system state at the n-th arrival. From this system state we can obtain the waiting time distribution of the n-th customer by considering the number of customers present at this n-th arrival, together with the sum of their expected (residual) service times. More details about the calculation of this waiting time distribution can be found in [14, Section 4].

Figure 7 shows the waiting time distribution of the 10-th customer with $\alpha = 1$ for different values of r. If we compare these results with the waiting time distributions of the 50-th customer, presented in Figure 8, we observe that r has a larger influence in the first case, as one could expect. Also, for the average waiting times, presented in Figure 9, we can draw similar conclusions as for the average queue

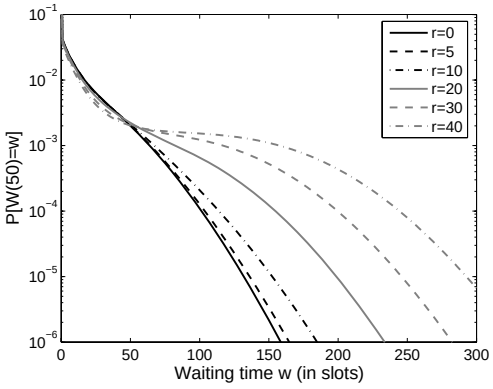

Figure 8: Waiting time distribution of the 50-th customer in a D-MAP/PH/1 queue

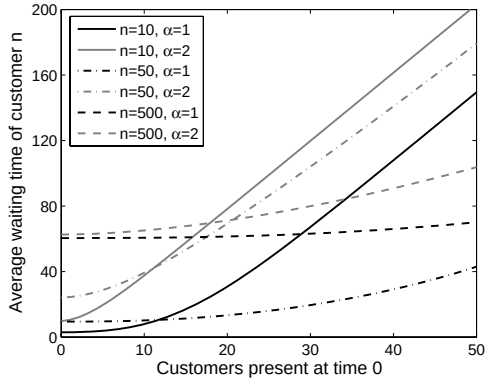

Figure 9: Average waiting time of the n-th customer in a D-MAP/PH/1 queue

length. That is, as n increases, the influence of the number of customers initially present on the average waiting time of the n-th customer decreases. Moreover, for small n, the difference that is caused by the initial D-MAP state increases together with n and it decreases hereafter.

5. CONCLUSIONS

In this paper, we discussed a new method to obtain transient performance measures of a discrete time QBD Markov chain for every possible initial configuration. First, we discussed the approach for an initial level equal to zero. For the other initial levels, we developed a recursive algorithm that makes use of the computations done for the previous levels and that exploits the btbT structure of the matrices occurring in the calculations. Some numerical examples were presented to illustrate the novel algorithm.

6. REFERENCES

[1] D. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chain solver: software tools. In *Proc. of the SMCTools workshop*, Pisa, Italy, 2006.

[2] D.A. Bini, G. Latouche, and B. Meini. Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra Appl.*, 366:39–64, 2003.

[3] C. Blondia. A discrete-time batch Markovian arrival process as B-ISDN traffic model. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(3,4):3–23, 1993.

[4] N.J. Higham and H. Kim. Solving a quadratic matrix equation by Newton's method with exact line search. *SIAM J. Matrix Anal. Appl.*, 23:303–316, 2001.

[5] G. Latouche, P.A. Jacobs, and D.P. Gaver. Finite Markov chain models skip-free in one direction. *Naval Research Logistics Quarterly*, 31:571–588, 1984.

[6] G. Latouche, C.E.M. Pearce, and P.G. Taylor. Invariant measure for quasi-birth-and-death processes. *Stochastic Models*, 14(1&2):443–460, 1998.

[7] D.M. Lucantoni. New results on the single server queue with a batch Markovian arrival process. *Stochastic Models*, 7(1):1–46, 1991.

[8] D.M. Lucantoni, K.S. Meier-Hellstern, and M.F. Neuts. A single server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*, 22:676–705, 1990.

[9] B. Meini. Solving QBD problems: the cyclic reduction algorithm versus the invariant subspace method. *Advances in Performance Analysis*, 1:215–225, 1998.

[10] L.M. Le Ny and B. Sericola. Transient analysis of the BMAP/PH/1 queue. *I.J. of Simulation*, 3(3-4):4–14, 2003.

[11] V. Ramaswami. The busy period of queues which have a matrix-geometric steady state probability vector. *Operations Research*, 19(4):238–261, 1982.

[12] A. Remke, B.R. Haverkort, and L. Cloth. Model checking infinite-state Markov chains. In *TACAS 2005, LNCS 3440, Springer*, pages 237–252, Feb 2005.

[13] A. Remke, B.R. Haverkort, and L. Cloth. Uniformization with representatives: comprehensive transient analysis of infinite-state QBDs. In *Proc. of SMCTools workshop*, Pisa, Italy, Oct 2006. ACM.

[14] B. Van Houdt and C. Blondia. Approximated transient queue length and waiting time distributions via steady state analysis. *Stochastic Models*, 21(2-3):725–744, 2005.

[15] B. Van Houdt and C. Blondia. QBDs with marked time epochs: a framework for transient performance measures. In *Proc. of QEST 2005*, pages 210–219. IEEE Computer Society, 2005.

[16] J. Van Velthoven, B. Van Houdt, and C. Blondia. Transient analysis of tree-like processes and its application to random access systems. *ACM Sigmetrics Performance Evaluation Review*, 34:181–190, June 2006.

[17] J. Zhang and E.J. Coyle. Transient analysis of quasi-birth-death processes. *Stochastic Models*, 5(3):459–496, 1989.