J. Appl. Prob. **50,** 151–165 (2013)
Printed in England

COMPUTING STATIONARY EXPECTATIONS IN LEVEL-DEPENDENT QBD PROCESSES

HENDRIK BAUMANN * AND
WERNER SANDMANN,* ** *Clausthal University of Technology*

Abstract

Stationary expectations corresponding to long-run averages of additive functionals on level-dependent quasi-birth-and-death processes are considered. Special cases include long-run average costs or rewards, moments and cumulants of steady-state queueing network performance measures, and many others. We provide a matrix-analytic scheme for numerically computing such stationary expectations without explicitly computing the stationary distribution of the process, which yields an algorithm that is as quick as its counterparts for stationary distributions but requires far less computer storage. Specific problems arising in the case of infinite state spaces are discussed and the application of the algorithm is demonstrated by a queueing network example.

Keywords: Level-dependent quasi-birth-and-death process; matrix-analytic computation; long-run average additive functional; stationary expectation; memory-efficient algorithm

2010 Mathematics Subject Classification: Primary 60J22
Secondary 60J27; 60J28

1. Introduction

Suppose that we are given an ergodic continuous-time Markov chain (CTMC) $(X_t)_{t\geq 0}$ with discrete state space $\mathcal{S}$ and generator matrix $Q = (q_{ij})_{i,j\in\mathcal{S}}$. Then the stationary distribution is the unique probability measure $\pi = (\pi_i)_{i\in S}$ satisfying $\pi Q = 0$ and it coincides with the limiting distribution. Hence, $(X_t)_{t\geq 0}$ converges to a random variable X_∞ with distribution π. Furthermore, if, for a function $f : \mathcal{S} \to \mathbb{R}$, which may be written as a function in the usual way or as a column vector,

$$\mathbb{E}_\pi[|f|] := \sum_{i\in\mathcal{S}} \pi_i |f(i)| = \pi|f| < \infty,$$

then the ergodic theorem

$$\lim_{t\to\infty} \frac{1}{t}\int_0^t f(X_s)\,\mathrm{d}s = \sum_{i\in\mathcal{S}} \pi_i f(i) = \pi f = \mathbb{E}_\pi[f] \quad \text{with probability 1} \tag{1}$$

holds, which is actually a strong law of large numbers for CTMCs [1, pp. 52–54], [15, pp. 264–265].

In many cases where the CTMC describes the behavior of a complex real-world system, one is not primarily interested in the limiting or stationary distribution π but rather in certain associated steady-state system measures expressed by (1). For instance, if f is interpreted

Received 15 May 2012; revision received 5 July 2012.
* Postal address: Department of Applied Stochastics and Operations Research, Clausthal University of Technology, Erzstr. 1, D-38678 Clausthal–Zellerfeld, Germany.
** Email address: werner.sandmann@tu-clausthal.de

as a cost or reward function then (1) gives the long-run average cost or reward rate, which is often of higher practical interest than π. Similarly, performance or availability measures in queueing networks or reliability models can be expressed by f. More abstractly, when analyzing additive functionals on (X_t), that is, functions of the form $\int_0^t f(X_s)\,\mathrm{d}s$, the value of the long-run average $\mathbb{E}_\pi[f]$ is often of primary interest. For example, in the case $\mathcal{S} \subseteq \mathbb{N}$, by a suitable choice of f, moments or cumulants of the limiting distribution of (X_t) can be computed. It is straightforward to generalize the scope to vector-valued functions $f: \mathcal{S} \to \mathbb{R}^m$ and expectations $\mathbb{E}_\pi[f^{(1)}], \ldots, \mathbb{E}_\pi[f^{(m)}]$, or, compactly written, $\mathbb{E}_\pi[f]$.

An obvious approach for determining $\mathbb{E}_\pi[f]$ if no analytical expressions are available is to compute at first the stationary distribution numerically and then to calculate $\mathbb{E}_\pi[f]$ according to the sum in (1), but, in general, if the Markov chain does not have a specific structure that can be exploited, due to prohibitively large state spaces, the requirements of the numerical computations may easily exhaust the computer storage capacity. In some cases one may be content with bounds on the stationary expectations [7], [11], [13]. An alternative is to estimate $\mathbb{E}_\pi[f]$ by simulation, but simulation is inherently costly and only provides estimates that are subject to statistical uncertainty. Depending on the characteristics of the Markov chain and the function f, an extremely large number of simulation runs may be required in order to obtain estimates with reasonable statistical accuracy.

In this paper we provide an efficient algorithm for computing stationary expectations of functions on the (typically multidimensional) state space of Markov chains with a specific structure, namely level-dependent quasi-birth-and-death (LDQBD) processes, where according to an appropriate ordering of the states the generator matrix is block tridiagonal. Our algorithm builds upon matrix-analytic methods for computing the stationary distribution of LDQBD processes [2], [4], [5], [6], [9], [17], but rather than directly invoking these methods as a first computational step, we develop a matrix-analytic scheme for numerically computing the stationary expectations without at first explicitly computing the stationary distribution of the process. Thereby, the costly storage of a large family of matrices as required by matrix-analytic computations of the stationary distribution is avoided, yielding a memory-efficient algorithm that is as quick as its counterparts for stationary distributions but requires far less computer storage. Note that though we have formulated the problem statement for processes in continuous time and will continue to focus on continuous-time LDQBD processes, an ergodic theorem analogous to (1) holds for discrete-time Markov chains, too [1, pp. 16–19], [15, pp. 45–47]. Accordingly, with only slight modifications, our algorithm can also be applied to compute stationary expectations in discrete-time LDQBD processes, as will be demonstrated later.

In the next section we provide the necessary background on LDQBD processes and matrix-analytic computations of their stationary distributions, which serves as the basis of our algorithm as well as the starting point for improvements with regard to the memory requirements when computing stationary expectations. In Section 3 we present our algorithm where its development and description show that it indeed obviously requires far less computer storage than the matrix-analytic computations of stationary distributions. In Section 4 we demonstrate how moments and cumulants of LDQBD process components can be computed as special cases of our general framework. In Section 5 we discuss how to start the computations in the case of infinite state spaces that have to be truncated in some manner. In particular, we address the choice of the matrix that is needed as an initial value of the matrix-analytic computations. In Section 6 we consider an example of a tandem queueing network with Markovian arrival processes and phase-type-distributed service times in order to demonstrate how our algorithm applies to model analysis, where we first show how to describe the model and its performance measures

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

appropriately within our LDQBD framework and then give some numerical results for specific settings. Finally, in Section 7 we conclude the paper and outline further research directions.

2. Stationary distributions of QBD processes

Suppose that the state space $\mathcal{S}$ of $(X_t)_{t\geq 0}$ is partitioned such that

$$\mathcal{S} = \bigcup_{n=0}^{\infty} \mathcal{S}^{(n)}, \qquad \mathcal{S}^{(n)} = \{s_1^{(n)}, s_2^{(n)}, \dots, s_{d_n}^{(n)}\}, \qquad \mathcal{S}^{(m)} \cap \mathcal{S}^{(n)} = \varnothing \quad \text{for } m \neq n,$$

and

$$\mathbb{P}(X_{t+h} \in \mathcal{S}^{(m)} \mid X_t \in \mathcal{S}^{(n)}) = o(h) \quad \text{for } |m-n| > 1.$$

Then the states can be arranged such that the generator matrix is of block-tridiagonal form, that is,

$$Q = \begin{pmatrix} Q_{00} & Q_{01} & & & \\ Q_{10} & Q_{11} & Q_{12} & & \\ & Q_{21} & Q_{22} & Q_{23} & \\ & & \ddots & \ddots & \ddots \end{pmatrix},$$

$Q_{ij} \in \mathbb{R}^{d_i \times d_j}$. In this case, $(X_t)_{t\geq 0}$ is said to be a quasi-birth-and-death (QBD) process [12, pp. 129–130], [14, pp. 81–83], and, for all $n \in \mathbb{N}$, the subset $\mathcal{S}^{(n)}$ of states is referred to as the process level with level number n. It is important to note that a QBD process can have an infinite state space of potentially arbitrary dimension. In order to obtain the desired structure, the levels can be defined in such a manner that the level number is determined by a function of the values of multiple components of the state variables; cf., e.g. [5] and Section 6.

When computing the stationary distribution π of a QBD process, it is quite natural to use a partition $\pi = (\pi_0, \pi_1, \pi_2, \dots)$ with $\pi_n = (\pi_{n,1}, \pi_{n,2}, \dots, \pi_{n,d_n})$. In the case of a level-independent QBD process, the transition rates do not depend on the level and the stationary distribution is simply given by $\pi_n = \pi_0 R^n$, where the matrix R is defined by the matrix quadratic equation

$$Q_{01} + RQ_{11} + R^2 Q_{21} = 0,$$

and π_0 is a solution of $\pi_0(Q_{00} + RQ_{10}) = 0$, normalized by $\pi_0(I-R)^{-1}e = 1$ with identity matrix I and vector $e = (1, 1, \dots, 1)^\top$, both of appropriate dimension. See [12, pp. 141–142] and [14, pp. 30–33, 81–83] for details.

In the more general case of a level-dependent QBD (LDQBD) process, sometimes also referred to as nonhomogeneous QBD [12, Chapter 12], (some of) the transition rates depend on the level and the stationary distribution satisfies

$$\pi_{n+1} = \pi_n R_n \tag{2}$$

with nonnegative matrices $R_n \in \mathbb{R}^{d_n \times d_{n+1}}$ that depend on the level. The basic idea behind the efficient numerical computation of matrix-analytic methods [2], [4], [5], [6], [9], [17] is to exploit the fact that these matrices satisfy the infinite recurrence scheme

$$R_n = -Q_{n,n+1}(Q_{n+1,n+1} + R_{n+1}Q_{n+2,n+1})^{-1} \tag{3}$$

and π_0 is a solution of the equation

$$\pi_0(Q_{00} + R_0 Q_{10}) = 0. \tag{4}$$

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

Using a physical interpretation of the entries of the matrices R_n (as conditional expected state sojourn times), it can be shown that the inverse matrices in (3) exist (irreducibility of the Markov chain is required), and that $Q_{00} + R_0 Q_{10}$ has the characteristics of a generator matrix of an irreducible CTMC with finite state space (ergodicity is required); see [4]. Thus, the solution π_0 of (4) is unique up to a multiplicative constant and can be chosen to be nonnegative.

For finite state spaces with maximum level N, (3) must be modified for $n = N - 1$ by

$$R_{N-1} = -Q_{N-1,N} Q_{NN}^{-1}. \tag{5}$$

Thus, an exact (up to numerical errors) algorithm for computing the stationary distribution of finite LDQBDs proceeds as follows. Determine R_{N-1} by (5), then compute (3) for $n = N-2, \ldots, 0$, determine π_0 by choosing a nontrivial solution of (4), and use (2) for computing $\pi_1, \pi_2, \ldots, \pi_N$. Finally, normalize π in the unit 1-norm.

If the state space is infinite, it has to be truncated. The simplest way consists of using the 'dishonest' generator

$$Q = \begin{pmatrix} Q_{00} & Q_{01} & & & \\ Q_{10} & Q_{11} & Q_{12} & & \\ & Q_{21} & Q_{22} & Q_{23} & \\ & & \ddots & \ddots & \ddots \\ & & & Q_{N,N-1} & Q_{NN} \end{pmatrix}$$

and the above algorithm for finite state spaces. To make things as simple as possible, here we define $R_N = 0 \in \mathbb{R}^{d_N \times d_{N+1}}$ and use (3) for computing R_{N-1}, too. Instead of setting $R_N = 0$, there are approaches to approximating a suitable R_N [2], [4], [9] that we will discuss in Section 5 with a special focus on our goal of computing stationary expectations rather than explicitly computing the stationary distribution. Altogether, the corresponding algorithm for computing the stationary distribution works as follows.

Algorithm 1. (*Computing stationary distributions of LDQBD processes.*)

- Choose N large and define $R_N = 0 \in \mathbb{R}^{d_N \times d_{N+1}}$.
- For $n = N-1, N-2, \ldots, 0$, compute and store

$$R_n = -Q_{n,n+1}(Q_{n+1,n+1} + R_{n+1} Q_{n+2,n+1})^{-1}.$$

- Determine a nontrivial solution of

$$x_0(Q_{00} + R_0 Q_{10}) = 0.$$

- For $n = 0, 1, \ldots, N-1$, compute $x_{n+1} = x_n R_n$.
- By normalizing $x = (x_0, \ldots, x_N)$, determine π, that is,

$$\pi_n = \frac{x_n}{c}, \qquad c = \sum_{n=0}^{N} \|x_n\|,$$

where $\|\cdot\|$ is the row sum norm.

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

Note that all the matrices $R_{N-1}, R_{N-2}, \ldots, R_0$ have to be stored. Furthermore, note that, whether $R_N = 0$ is used or any approximation, we will not figure out the exact matrix R_N. Thus, the approximations $R_N^*, R_{N-1}^*, \ldots, R_0^*$ are used rather than the exact matrices $R_N, R_{N-1}, \ldots, R_0$. Using the approximation R_0^* instead of R_0, (4) has no exact solution, the matrix $Q_{00} + R_0^* Q_{10}$ has full rank d_0. In practice, one of the scalar equations is ignored and one of the components is fixed. For $N \to \infty$, the matrices $R_0^*, R_1^*, R_2^*, \ldots$ and, thus, $Q_{00} + R_0^* Q_{10}$ tend to their exact values $R_0, R_1, R_2, \ldots$ and $Q_{00} + R_0 Q_{10}$, entailing that the computed approximation $\pi^* = (\pi_0^*, \pi_1^*, \ldots, \pi_N^*, 0, \ldots)$ actually converges to the stationary distribution $\pi = (\pi_0, \pi_1, \pi_2, \ldots)$.

The stated matrix-analytic algorithm is very quick and definitely outperforms all standard methods for numerically computing stationary distributions of Markov chains [16, pp. 121–230] as long as the block sizes are small compared to the maximum level or the truncation level. Usually, the block matrices themselves are sparse, entailing that iterative methods get by with smaller memory. In [2] a variant of the matrix-analytic algorithm is discussed using smaller memory ($O(\sqrt{N}d^2)$ for constant block dimensions $d_i = d$), but paying the price of nearly double computing time. Nevertheless, the regular version still outperforms standard methods with regard to memory capacity and computing time in many examples.

3. Stationary expectations of LDQBD processes

Now we are prepared for approaching the task of computing $\mathbb{E}_\pi[f]$ for some function $f = (f^{(1)}, \ldots, f^{(m)})\colon \mathcal{S} \to \mathbb{R}^m$, where π is the stationary distribution of an LDQBD process. The goal is to obtain $\mathbb{E}_\pi[f]$ without explicitly computing π and without the need for storing all the matrices $R_N, R_{N-1}, \ldots, R_0$ as necessary for computing π by Algorithm 1.

Since the state space $\mathcal{S}$ is discrete, we can interpret $f^{(\mu)}$ as a column vector, partitioned into

$$f^{(\mu)} = (f_0^{(\mu)}, f_1^{(\mu)}, \ldots)^\top \quad \text{with} \quad f_n^{(\mu)} = (f_{n,1}^{(\mu)}, \ldots, f_{n,d_n}^{(\mu)})^\top,$$

using the notation $f_{n,v}^{(\mu)} = f^{(\mu)}(s_v^{(n)})$, where $s_v^{(n)} \in \mathcal{S}$. Hence, for $\mu = 1, \ldots, m$, we can write

$$\mathbb{E}_\pi[f^{(\mu)}] = \pi f^{(\mu)} = \sum_{n=0}^{\infty} \sum_{v=1}^{d_n} \pi_{n,v} f^{(\mu)}(s_v^{(n)}) = \sum_{n=0}^{\infty} \pi_n f_n^{(\mu)}$$

and, thus,

$$\mathbb{E}_\pi[f] = \sum_{n=0}^{\infty} \pi_n f_n \quad \text{with} \quad f_n = (f_n^{(1)}, \ldots, f_n^{(m)}).$$

Computing the exact value is impossible as long as the state space is infinitely large and there is no analytic expression for π. Thus, the sum has to be truncated, that is, we compute $\sum_{n=0}^{N} \pi_n f_n$ for some $N \in \mathbb{N}$, implicitly using the approximation for π given above. But, instead of first computing π and then summing up, we use a Horner-type scheme, which is normally used to evaluate polynomials at linear time [8, pp. 568–569], [10, pp. 486–489]. Applying (2) yields

$$\begin{aligned}\sum_{n=0}^{N} \pi_n f_n &= \pi_0 \sum_{n=0}^{N} \left(\prod_{k=0}^{n-1} R_k \right) f_n \\ &= \pi_0 (f_0 + R_0 (f_1 + R_1 (f_2 + R_2 (\cdots + R_{N-1} (f_{N-1} + R_{N-1} f_N))))).\end{aligned}$$

The term

$$f_0 + R_0 (f_1 + R_1 (f_2 + R_2 (\cdots + R_{N-1} (f_{N-1} + R_{N-1} f_N))))$$

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

can be computed by $Z = f_n + R_n Z$ for $n = N-1, N-2, \ldots, 0$, starting from $Z = f_N$. With c being the same normalization constant as in Algorithm 1 for computing the stationary distribution, using (4), we can determine $x_0 = c\pi_0$, and, thus, by updating $Z = x_0 Z$, we obtain $Z = c\mathbb{E}_\pi[f]$.

Note that, for an algorithm corresponding to this just developed computational scheme, there is indeed no need for storing the matrices R_n since they are only used once for updating Z and computing the next matrix R_{n-1}.

The remaining problem is to determine the normalization constant c, which is obtained by summing up $\|\pi_n\|$ when computing the stationary distribution, but now, we do not want to compute π_n via (2) anymore. In fact, the normalization constant c can be determined in another way as follows. We simply extend the function f to $\bar{f} = (f^{(0)}, f^{(1)}, \ldots, f^{(m)})$ with $f^{(0)}_{n,v} = 1$. Then the normalization condition can be expressed as $\mathbb{E}_\pi[f^{(0)}] = 1$. Applying our algorithm to $\bar{f}$ we obtain a vector $(z^{(0)}, z^{(1)}, \ldots, z^{(m)}) = c\mathbb{E}_\pi[\bar{f}]$, and, thus, our final result will be

$$\mathbb{E}_\pi[f] = \left(\frac{z^{(1)}}{z^{(0)}}, \ldots, \frac{z^{(m)}}{z^{(0)}} \right).$$

Hence, let Q be the generator matrix of an ergodic LDQBD and let $f_n^{(\mu)} \in \mathbb{R}^{d_n}$ be given for $n \in \mathbb{N}$ and $\mu = 1, \ldots, m$. Define $z^{(\mu)} = \sum_{n=0}^{\infty} \pi_n f_n^{(\mu)}$. Then approximations for $z^{(1)}, \ldots, z^{(m)}$ can be obtained by the following algorithm.

Algorithm 2. (*Computing stationary expectations of LDQBD processes.*)

- Define $f_n^{(0)} = (1, 1, \ldots, 1)^\top \in \mathbb{R}^{d_n}$ and $\bar{f}_n = (f_n^{(0)}, f_n^{(1)}, \ldots, f_n^{(m)}) \in \mathbb{R}^{d_n \times (m+1)}$.
- Choose N large, $R = 0 \in \mathbb{R}^{d_N \times d_{N+1}}$, $Z = f_N$.
- For $n = N-1, N-2, \ldots, 1, 0$,
  - update $R = -Q_{n,n+1}(Q_{n+1,n+1} + RQ_{n+2,n+1})^{-1}$,
  - update $Z = f_n + RZ$.
- Compute a nontrivial vector x_0 by solving $d_0 - 1$ scalar equations of the system $x_0(Q_{00} + RQ_{10}) = 0$ of size $d_0 \times d_0$ and update $Z = x_0 Z \in \mathbb{R}^{1 \times (m+1)}$.
- Define $Z = (z^{(0)}, z^{(1)}, \ldots, z^{(m)})$ and replace $Z = Z/z^{(0)}$.

As stated above, the main advantage of this algorithm is that it is memory efficient. Since just R and Z are stored and immediately substituted by their new values, the maximum number of real numbers to be stored at the same time is

$$\max_{n=0,\ldots,N} (d_n d_{n+1} + d_n(m+1)).$$

The main time cost is updating R, which is also part of the algorithm for computing stationary distributions. Thus, the new algorithm needs a similar amount of time for computing stationary expectations, but clearly outperforms the algorithm for computing stationary distributions with regard to memory efficiency. Especially for applications in which the block sizes d_n are small but a very high truncation level N has to be chosen, the new algorithm is highly efficient.

Remarks. 1. Similarly as in [4] and [9] in the context of computing the stationary distribution, instead of choosing $R = 0$ at the beginning, approximations for R_N may be used. As stated before, we will discuss these approaches in Section 5.

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

2. An analogous algorithm for discrete-time Markov chains with transition probability matrix P of block-tridiagonal form can easily be deduced. Only slight modifications have to be made. The update for R is given by

$$P_{n,n+1}(I_{d_{n+1}} - P_{n+1,n+1} - RP_{n+2,n+1})^{-1}$$

and x_0 is determined by

$$x_0(I_{d_0} - P_{00} - RP_{10}) = 0,$$

where I_d denotes the identity matrix of size $d \times d$ and the P_{ij} are the blocks of the transition probability matrices.

4. Special case: moments and cumulants

Consider an ergodic p-dimensional LDQBD $(X_t)_{t\geq 0}$, $X_t = (X_t^{(1)}, \ldots, X_t^{(p)})$, with state space

$$\mathcal{S} = \bigcup_{n=0}^{\infty} \mathcal{S}^{(n)} \subseteq \mathbb{N}^p.$$

A natural goal is to obtain moments and cumulants of the various components. This task can be simply cast in our framework and solved by means of our general algorithm for calculating stationary expectations. For computing the qth steady-state moment of the kth component, we choose $f_{n,v} = (s_v^{(n)})_k^q$. Then our algorithm yields

$$\mathbb{E}_\pi[f] = \sum_{n=0}^{\infty} \pi_n f_n = \sum_{n=0}^{\infty} \sum_{v=1}^{d_n} \pi_{n,v} (s_v^{(n)})_k^q = \mathbb{E}[(X_\infty^{(k)})^q].$$

Cumulants can be easily composed through moments. Of course, when computing moments and cumulants of the components, we have to compute all the moments we need at the same time, since, otherwise, we would solve for the R in the algorithm more than once.

For example, consider an ergodic two-dimensional LDQBD $(L_t, P_t)_{t\geq 0}$ on the state space $\mathcal{S} \subseteq \mathbb{N} \times \mathbb{N}$, where the level number is defined as the value of the state variable of the first component. Choose

$$\bar{f}_n = \begin{pmatrix} 1 & n & n^2 & n^3 & n^4 & (s_1^{(n)})_2 & (s_1^{(n)})_2^2 & (s_1^{(n)})_2^3 & (s_1^{(n)})_2^4 \\ 1 & n & n^2 & n^3 & n^4 & (s_2^{(n)})_2 & (s_2^{(n)})_2^2 & (s_2^{(n)})_2^3 & (s_2^{(n)})_2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & n & n^2 & n^3 & n^4 & (s_{d_n}^{(n)})_2 & (s_{d_n}^{(n)})_2^2 & (s_{d_n}^{(n)})_2^3 & (s_{d_n}^{(n)})_2^4 \end{pmatrix}.$$

Then after performing our algorithm we obtain

$$\mathbb{E}[L_\infty] = z^{(1)}, \quad \mathbb{E}[L_\infty^2] = z^{(2)}, \quad \mathbb{E}[L_\infty^3] = z^{(3)}, \quad \mathbb{E}[L_\infty^4] = z^{(4)},$$
$$\mathbb{E}[P_\infty] = z^{(5)}, \quad \mathbb{E}[P_\infty^2] = z^{(6)}, \quad \mathbb{E}[P_\infty^3] = z^{(7)}, \quad \mathbb{E}[P_\infty^4] = z^{(8)}.$$

Since the algorithm is based on a finite vector π^* as an approximation for the stationary distribution π, it does not verify whether or not these moments exist, but it will provide results in either case. Thus, in order to check the existence of the moments, if not guaranteed by certain

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

knowledge of the underlying system that is modeled, other methods may be used in advance before invoking our algorithm. For instance, one may consider bounds as given in [7], [11], and [13]. In particular, if there is a finite upper bound on a moment, the respective moment exists. However, this is not the scope of the present paper.

Using the first four moments, we can compose cumulants, i.e.

$$\begin{aligned}\kappa_{L_\infty,2} &= z^{(2)} - (z^{(1)})^2,\\ \kappa_{L_\infty,3} &= z^{(3)} - 3z^{(2)}z^{(1)} + 2(z^{(1)})^3,\\ \kappa_{L_\infty,4} &= z^{(4)} - 4z^{(3)}z^{(1)} - 3(z^{(2)})^2 + 12z^{(2)}(z^{(1)})^2 - 6(z^{(1)})^4,\end{aligned}$$

and, thus, variance, skewness, and kurtosis of L_∞:

$$\begin{aligned}\operatorname{var}[L_\infty] &= \kappa_{L_\infty,2} = z^{(2)} - (z^{(1)})^2,\\ \operatorname{skew}[L_\infty] &= \frac{\kappa_{L_\infty,3}}{\kappa_{L_\infty,2}^{3/2}} = \frac{z^{(3)} - 3z^{(2)}z^{(1)} + 2(z^{(1)})^3}{(z^{(2)} - (z^{(1)})^2)^{3/2}},\\ \operatorname{kurt}[L_\infty] &= \frac{\kappa_{L_\infty,4}}{\kappa_{L_\infty,2}^{2}} = \frac{z^{(4)} - 4z^{(3)}z^{(1)} - 3(z^{(2)})^2 + 12z^{(2)}(z^{(1)})^2 - 6(z^{(1)})^4}{(z^{(2)} - (z^{(1)})^2)^2}.\end{aligned}$$

Variance, skewness, and kurtosis of P_∞ can be obtained analogously using $z^{(5)}, \ldots, z^{(8)}$ instead of $z^{(1)}, \ldots, z^{(4)}$.

To further illustrate the choice of $\bar{f}_n$, note that if the d_n states contained in level n take the values $1, \ldots, d_n$ then $\bar{f}_n$ is simply

$$\bar{f}_n = \begin{pmatrix} 1 & n & n^2 & n^3 & n^4 & 1 & 1^2 & 1^3 & 1^4 \\ 1 & n & n^2 & n^3 & n^4 & 2 & 2^2 & 2^3 & 2^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & & & \\ 1 & n & n^2 & n^3 & n^4 & d_n & d_n^2 & d_n^3 & d_n^4 \end{pmatrix}.$$

5. Getting started in the case of infinite state spaces

The remaining question for the algorithm is how to choose the starting matrix R, that is, the approximation R_N^* for R_N. The method suggested by Bright and Taylor [4] consists of choosing $L \geq 0$ and computing

$$R_N^* = R_N(L) := \sum_{\ell=0}^{L} U_N^{(\ell)} \sum_{i=0}^{\ell-1} D_{N+2^{\ell-i}}^{(\ell-1-i)},$$

where

$$U_k^{(0)} = Q_0^{(k)}(-Q_1^{(k+1)})^{-1}, \tag{6}$$

$$D_{k+2}^{(0)} = Q_2^{(k+2)}(-Q_1^{(k+1)})^{-1}, \tag{7}$$

$$U_k^{(\ell)} = U_k^{(\ell-1)} U_{k+2^{\ell-1}}^{(\ell-1)} (I - U_{k+2^\ell}^{(\ell-1)} D_{k+3\cdot 2^{\ell-1}}^{(\ell-1)} - D_{k+2^\ell}^{(\ell-1)} U_{k+2^{\ell-1}}^{(\ell-1)})^{-1}, \tag{8}$$

$$D_{k+2^{\ell+1}}^{(\ell)} = D_{k+2^{\ell+1}}^{(\ell-1)} D_{k+3\cdot 2^{\ell-1}}^{(\ell-1)} (I - U_{k+2^\ell}^{(\ell-1)} D_{k+3\cdot 2^{\ell-1}}^{(\ell-1)} - D_{k+2^\ell}^{(\ell-1)} U_{k+2^{\ell-1}}^{(\ell-1)})^{-1}. \tag{9}$$

They proved that, for $L \to \infty$, the approximation $R_N(L)$ tends to R_N and gave an algorithm exploiting the facts that the inverse matrices in (6) and (7) are the same and the inverse matrices in (8) and (9) are the same. By avoiding to compute the same inverse twice, the computational effort of their method was dominated by $4 \cdot 2^L$ matrix inversions to be done.

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

In fact, this method is equivalent to a preprocessing step associated with the truncation level. Another method of preprocessing was suggested in [9] in the context of computing stationary distributions. Let us denote R_N^* as the matrix defined by choosing $R_{N^*} = 0$ for some $N^* > N$ and then computing $R_{N^*-1}, R_{N^*-2}, \ldots, R_N$ using recursion (3). Of course, this method delivers a better approximation for R_N too, since, for $N^* \to \infty$, we have $R_N^* \to R_N$. Actually, one can prove (by induction) that $R_N^* = R_N(L)$ when choosing $N^* = N + 2^{L+1} - 1$. Thus, using (3) for a preprocessing, we need 2^{L+1} matrix inversions, that is, about as many as in the more complicated (with regard to the formulae) version discussed in [4]. In some examples there is a difference with regard to numerical stability of the matrix inversions.

We now discuss the influence of a preprocessing step associated with the truncation level. For this purpose, we consider the example of a simple birth–death process with constant rates, that is,

$$Q = \begin{pmatrix} -\lambda & \lambda & & & \\ \mu & -(\lambda+\mu) & \lambda & & \\ & \mu & -(\lambda+\mu) & \lambda & \\ & & \ddots & \ddots & \ddots \end{pmatrix}.$$

We neglect any numerical errors and focus on the influence of the parameters N (truncation level) and N^* (referred to as the preprocessing level). In this analysis, we do not need to distinguish which variant of the matrix-analytic algorithm we use. First computing an approximation $(\pi_0^*, \ldots, \pi_N^*)$ and then calculating steady-state system measures will yield the same result as our direct algorithm.

In this simple example we have $d_n = 1$ and, of course, we know the exact stationary distribution, which is given by $\pi_n = (1-\rho)\rho^n$ with $\rho = \lambda/\mu$, provided $\rho < 1$ such that a unique stationary distribution exists. Define L as the stationary population size (or number of customers in an M/M/1 queue). Then $\mathbb{E}[L] = \rho/(1-\rho)$ and $\mathrm{var}[L] = \rho/(1-\rho)^2$. Now, we determine the values that our algorithm will yield for $\mathbb{E}[L]$ (up to numerical errors) and compare the variants without preprocessing, that is, $R_N = 0$, and with preprocessing, that is, $R_{N^*} = 0$ for some $N^* > N$.

With preprocessing we have $R_{N^*} = 0$ and

$$R_n = \lambda(\lambda + \mu - R_{n+1}\mu)^{-1} = \frac{\rho}{1 + \rho - R_{n+1}}, \qquad n < N^*,$$

and simple induction shows that

$$R_n = \rho \frac{1 - \rho^{N^*-n}}{1 - \rho^{N^*+1-n}}.$$

By another simple induction we see that the approximation for the stationary distribution is given by $\pi_n^* = x_n/S$ with

$$x_n = \frac{\rho^n - \rho^{N^*+1}}{1 - \rho^{N^*+1}}, \qquad S = \sum_{n=0}^{N} x_n = \frac{1 - \rho^{N+1} - (N+1)(1-\rho)\rho^{N^*+1}}{(1-\rho)(1-\rho^{N^*+1})}.$$

Theoretically, x_0 as a multiple of π_0 has to satisfy $x_0(-\lambda + R_0\mu) = 0$, and, since $R_0 \neq \rho$, this yields $x_0 = 0$. But, as stated above, due to the truncation procedure, the determination of x_0 has to be modified where one scalar equation is ignored and one component of x_0 is fixed. Hence, in the scalar case we can choose an arbitrary $x_0 \neq 0$, and for convenience we choose $x_0 = 1$.

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

Some simple algebra yields

$$\sum_{n=0}^{N} n\pi_n^* = \rho \frac{2 - 2(N+1)\rho^N + 2N\rho^{N+1} - N(N+1)\rho^{N^*}(1 - 2\rho + \rho^2)}{2(1-\rho)(1 - \rho^{N+1} - (N+1)(1-\rho)\rho^{N^*+1})}$$

as an approximation for $\mathbb{E}[L]$ and, thus, the error is

$$\mathrm{err}(N, N^*) = \mathbb{E}[L] - \sum_{n=0}^{N} n\pi_n^* = \frac{N+1}{2}\rho^{N+1} \frac{2 + N\rho^{N^*-N} - (N+2)\rho^{N^*+1-N}}{1 - \rho^{N+1} - (N+1)(1-\rho)\rho^{N^*+1}}.$$

Studying this function depending on N and N^* we observe that, for $N^* \to \infty$, we have monotonically decreasing convergence to $(N+1)\rho^{N+1}/(1-\rho^{N+1})$ and, for $N \to \infty$, we have monotonically decreasing convergence to 0. Thus, the influence of a higher truncation level N is much stronger than that of a higher preprocessing level N^*. Hence, N has to be chosen as large as possible, and without additional effort the best choice is $N = N^*$, yielding the error

$$\mathrm{err}(N, N) = \frac{(N+1)(N+2)}{2} \frac{(1-\rho)\rho^{N+1}}{1 - (N+2)\rho^{N+1} + (N+1)\rho^{N+2}}.$$

In the general case of an arbitrary LDQBD process and an arbitrary function f the error functions may be more complicated, but it seems reasonable that instead of choosing a preprocessing level $N^* > N$ it is more efficient to increase the truncation level N itself. With and without preprocessing, most of the algorithm's computational time is spent calculating R_n. In the case of computing the stationary distribution a preprocessing might thus make sense with regard to memory capacity. Preprocessing according to [4] requires the storage of $N^* - N + 1$ pairs of U and D matrices as given in (8) and (9), but the matrices $R_{N^*}, \ldots, R_N$ calculated within the preprocessing using $R_{N^*} = 0$ and (3) as suggested in [9] do not have to be stored, whereas $R_{N-1}, R_{N-2}, \ldots, R_0$ do.

In our algorithm for computing stationary expectations no matrix R_n has to be stored for more than one computational step and, thus, a preprocessing does not make much sense. While choosing large N seriously influences the memory requirement when computing the stationary distribution, our algorithm does not seriously suffer from a large N in that it requires storing a matrix of corresponding size only once.

6. Tandem queueing network example

We now demonstrate how the performance analysis of a queueing network can be cast in our framework for computing stationary expectations of LDQBD processes. First we consider a general description of an open tandem queueing network with Markovian arrival processes and phase-type-distributed service times. Then we present numerical results for a model with specific choices of the arrival processes and the service time distributions.

6.1. LDQBD modeling framework

We consider a tandem queueing network with two single-server queues of infinite capacity, phase-type-distributed service times, and Markovian arrival processes (MAPs) at both nodes. After service completion at the first node, customers move to the second node with probability $p(n_2)$ and leave the network with probability $1 - p(n_2)$, where n_2 is the number of customers at the second node. Since leaving can be interpreted as customers being discouraged by the length of the second queue, it makes sense to assume that $p(n_2)$ is decreasing.

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

A MAP itself is a special case of a QBD process, namely a two-dimensional level-independent QBD with block-tridiagonal generator matrix

$$\begin{pmatrix} D & E & & \\ & D & E & \\ & & \ddots & \ddots \end{pmatrix}.$$

The first component—corresponding to the level—describes the number of arrivals and the second component is some kind of 'environmental state'. If E is a diagonal matrix, the MAP reduces to a Markov-modulated Poisson process (MMPP) [12, pp. 72–75].

Phase-type distributions [12, Chapter 2], [14, Chapter 2], defined as distributions of the time to absorption in an absorbing Markov chain, are characterized by a probability distribution giving the initial distribution for the transient phases, and a dishonest generator giving the transition rates of the transient phases. More precisely, for a CTMC with finite state space $\{1, \ldots, n, n+1\}$ and initial distribution $\nu = (\nu_1, \ldots, \nu_{n+1})$, where the states $1, \ldots, n$ are transient and the state $n+1$ is absorbing, the generator matrix Q of the chain can be represented as

$$Q = \begin{pmatrix} T & t \\ 0 & 0 \end{pmatrix}, \qquad T \in \mathbb{R}^{n \times n},\ t \in \mathbb{R}^n,$$

and $((\nu_1, \ldots, \nu_n), T)$ is a representation of the phase-type distribution.

In our model we assume a (D_1, E_1)-MAP for arrivals at the first queueing node, where the service time is (α_1, T_1)-phase-type distributed, and a (D_2, E_2)-MAP for arrivals at the second queueing node, where the service time is (α_2, T_2)-phase-type distributed. Altogether, we obtain a six-dimensional Markov chain, in which a state $(n_1, n_2, u_1, u_2, v_1, v_2)$ consists of

- the number n_1 of customers at the first node,
- the number n_2 of customers at the second node,
- the state u_1 of the environment for the first MAP,
- the state u_2 of the environment for the second MAP,
- the phase v_1 of the service time at the first node,
- the phase v_2 of the service time at the second node.

The state space is

$$\begin{aligned} \mathcal{S} = {} & \mathbb{N}_{>0} \times \mathbb{N}_{>0} \times \{1, \ldots, e_1\} \times \{1, \ldots, e_2\} \times \{1, \ldots, m_1\} \times \{1, \ldots, m_2\} \\ & \cup \mathbb{N}_{>0} \times \{0\} \times \{1, \ldots, e_1\} \times \{1, \ldots, e_2\} \times \{1, \ldots, m_1\} \times \{1\} \\ & \cup \{0\} \times \mathbb{N}_{>0} \times \{1, \ldots, e_1\} \times \{1, \ldots, e_2\} \times \{1\} \times \{1, \ldots, m_2\} \\ & \cup \{0\} \times \{0\} \times \{1, \ldots, e_1\} \times \{1, \ldots, e_2\} \times \{1\} \times \{1\}, \end{aligned}$$

where e_i denotes the number of environmental states for the ith MAP and m_i denotes the number of transient phases for the ith phase-type distribution. With $t_i = -T_i e$, $i = 1, 2$, we have transitions as given in Table 1.

Now let us choose $\mathcal{S}^{(n)} = \{(n_1, n_2, u_1, u_2, v_1, v_2) \in \mathcal{S} : n_1 + n_2 = n\}$. Then we have a QBD process, which is level dependent due to the probability $p(n_2)$ and because of the different numbers of states at each level. Within the levels we enumerate the states in lexicographic order, that is,

$$s_1^{(n)} = (0, n, 1, 1, 1, 1), \ldots, s_{d_n}^{(n)} = (n, 0, e_1, e_2, m_1, 1).$$

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

TABLE 1: State transitions of the tandem queueing network model.

From state	To state	Rate	Condition
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2, u_1', u_2, v_1, v_2)$	$(D_1)_{u_1,u_1'}$	
$(0, n_2, u_1, u_2, 1, v_2)$	$(1, n_2, u_1', u_2, v_1', v_2)$	$(E_1)_{u_1,u_1'}(\alpha_1)_{v_1'}$	
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1+1, n_2, u_1', u_2, v_1, v_2)$	$(E_1)_{u_1,u_1'}$	$n_1 > 0$
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2, u_1, u_2', v_1, v_2)$	$(D_2)_{u_2,u_2'}$	
$(n_1, 0, u_1, u_2, v_1, 1)$	$(n_1, 1, u_1, u_2', v_1, v_2')$	$(E_2)_{u_2,u_2'}(\alpha_2)_{v_2'}$	
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2+1, u_1, u_2', v_1, v_2)$	$(E_2)_{u_2,u_2'}$	$n_2 > 0$
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2, u_1, u_2, v_1', v_2)$	$(T_1)_{v_1,v_1'}$	$n_1 > 0$
$(1, 0, u_1, u_2, v_1, 1)$	$(0, 1, u_1, u_2, 1, v_2')$	$(t_1)_{v_1}(\alpha_2)_{v_2'}p(0)$	
$(n_1, 0, u_1, u_2, v_1, 1)$	$(n_1-1, 1, u_1, u_2, v_1', v_2')$	$(t_1)_{v_1}(\alpha_1)_{v_1'}(\alpha_2)_{v_2'}p(0)$	$n_1 > 1$
$(1, n_2, u_1, u_2, v_1, v_2)$	$(0, n_2+1, u_1, u_2, 1, v_2)$	$(t_1)_{v_1}p(n_2)$	$n_2 > 0$
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1-1, n_2+1, u_1, u_2, v_1', v_2)$	$(t_1)_{v_1}(\alpha_1)_{v_1'}p(n_2)$	$n_1 > 1, n_2 > 0$
$(1, n_2, u_1, u_2, v_1, v_2)$	$(0, n_2, u_1, u_2, 1, v_2)$	$(t_1)_{v_1}(1-p(n_2))$	
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1-1, n_2, u_1, u_2, v_1', v_2)$	$(t_1)_{v_1}(\alpha_1)_{v_1'}(1-p(n_2))$	$n_1 > 1$
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2, u_1, u_2, v_1, v_2')$	$(T_2)_{v_2,v_2'}$	$n_2 > 0$
$(n_1, 1, u_1, u_2, v_1, v_2)$	$(n_1, 0, u_1, u_2, v_1, 1)$	$(t_2)_{v_2}$	
$(n_1, n_2, u_1, u_2, v_1, v_2)$	$(n_1, n_2-1, u_1, u_2, v_1, v_2')$	$(t_2)_{v_2}(\alpha_2)_{v_2'}$	$n_2 > 1$

Important system measures are the mean steady-state queue lengths, the mean queueing time (computed by Little's law), and the probability of 'loss', that is, the probability that a customer leaves the network after service completion at the first node. Since the output of the first node is not a Poisson process, this probability does not depend only on the queue length distribution of the second node, but it also depends on the service completion times at the first node.

It is quite easy to find functions $\tilde{n}_1(n, v), \tilde{n}_2(n, v), \tilde{u}_1(n, v), \ldots$ for recomputing $n_1, n_2, u_1, \ldots$ for the vth state of level n. When defining

$$\begin{aligned}
\bar{f}_{n,v}^{(0)} &= 1,\\
\bar{f}_{n,v}^{(1)} &= \tilde{n}_1(n, v),\\
\bar{f}_{n,v}^{(2)} &= \tilde{n}_2(n, v),\\
\bar{f}_{n,v}^{(3)} &= \mathbf{1}_{\{\tilde{n}_1(n,v)>0\}}\tilde{v}_1(n, v)p(\tilde{n}_2(n, v)),\\
\bar{f}_{n,v}^{(4)} &= \mathbf{1}_{\{\tilde{n}_1(n,v)>0\}}\tilde{v}_1(n, v),\\
\bar{f}_{n,v}^{(5)} &= 1 - p(\tilde{n}_2(n, v)),\\
\bar{f}_{n,v}^{(6)} &= \sum_{i=1}^{e_1}(E_1)_{\tilde{u}_1(n,v),i},\\
\bar{f}_{n,v}^{(7)} &= \sum_{i=1}^{e_2}(E_2)_{\tilde{u}_2(n,v),i},
\end{aligned}$$

our algorithm provides

- $z^{(1)} = \mathbb{E}[N_1]$: the stationary mean number of customers at the first node,
- $z^{(2)} = \mathbb{E}[N_2]$: the stationary mean number of customers at the second node,

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

- $z^{(3)}$: the stationary rate of customers joining the second node after having been served at the first node,
- $z^{(4)}$: the stationary total rate of customers served at the first node,
- $p_{\text{loss,Poi}} = z^{(5)}$: the stationary probability for discourage when arriving at the second node according to a Poisson process,
- $z^{(6)}$: the stationary total arrival rate (from outside) at the first node,
- $z^{(7)}$: the stationary total arrival rate (from outside) at the second node.

For model analysis, we additionally compute

- $\mathbb{E}[R_1] = z^{(1)}/z^{(6)}$: the stationary mean residence time at the first node,
- $\mathbb{E}[R_2] = z^{(2)}/(z^{(3)} + z^{(7)})$: the stationary mean residence time at the second node,
- $p_{\text{loss}} = 1 - z^{(3)}/z^{(4)}$: the stationary probability for customers leaving after having been served at the first node.

The value $p_{\text{loss,Poi}}$ is just computed for means of comparison to p_{loss}. The stationary arrival rates, that is, $z^{(6)}$ and $z^{(7)}$, could be computed in an alternative manner: $D_1 + E_1$ is a finite generator of an irreducible Markov chain. Therefore, a unique stationary distribution ψ exists, which is the stationary marginal distribution of the first environmental state. Thus, $z^{(6)} = \psi E_1 e$. Analogously, $z^{(7)}$ can be expressed in terms of the stationary distribution of $D_2 + E_2$. Usually, this way of determining the arrival rates is more efficient, but the additional effort for computing $z^{(6)}$ and $z^{(7)}$ is small. More importantly, in a more general framework in which the arrival rates may depend on the number of customers waiting, the approach used for computing $z^{(6)}$ and $z^{(7)}$ is still valid, too.

6.2. Numerical example

As a specific numerical example, we consider an MMPP with

$$e_1 = 2, \qquad D_1 = \begin{pmatrix} -3.1 & 0.1 \\ 0.1 & -2.1 \end{pmatrix}, \qquad E_1 = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}$$

for arrivals at the first node, an MMPP with

$$e_2 = 3, \qquad D_2 = \begin{pmatrix} -3.4 & 0.3 & 0.1 \\ 0.5 & -2.8 & 0.3 \\ 1.0 & 1.0 & -12.0 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$

for arrivals at the second node, a hyperexponential distribution with

$$m_2 = 2, \qquad \alpha_2 = (0.9, 0.1), \qquad T_2 = \begin{pmatrix} -6 & 0 \\ 0 & -3 \end{pmatrix}, \qquad t_2 = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

for the service times at the second node, and probabilities

$$p(n_2) = \begin{cases} 1, & n_2 = 0, \\ \dfrac{1}{n_2}, & n_2 \geq 1, \end{cases}$$

https://doi.org/10.1239/jap/1363784430 Published online by Cambridge University Press

TABLE 2: Numerical examples for the tandem queueing network model.

k	$\mathbb{E}[N_1]$	$\mathbb{E}[N_2]$	$\mathbb{E}[R_1]$	$\mathbb{E}[R_2]$	p_{loss}	$p_{\text{loss,Poi}}$
1	1.869 18	3.743 45	0.747 668	0.830 160	0.486 578	0.481 788
2	1.574 54	3.740 89	0.629 811	0.825 137	0.476 849	0.481 493
4	1.424 49	3.738 15	0.569 794	0.822 171	0.471 638	0.480 993
6	1.374 00	3.736 96	0.549 595	0.821 103	0.469 854	0.480 756
8	1.348 65	3.736 31	0.539 456	0.820 553	0.468 953	0.480 623

for moving to the second node after having been served at the first node. At the first node we assume Erlang-distributed service times with k phases and parameter $4k$, that is, the mean service time is $\frac{1}{4}$, and, for large k, the variance becomes small, implying that the service time becomes more and more deterministic. Varying k, we can compute the stationary characteristics described above. Numerical examples are given in Table 2.

The truncation level was chosen as 50 for $k = 1$, we used the matrix-analytic algorithm as described in Section 2 for determining the stationary distribution π and observed a residual error norm of $\|\pi Q\| \approx 10^{-3.5}$. For this algorithm, in general, we have to store approximately $144k^2N^3/3$ floating-point numbers, whereas for our new algorithm, the maximum number of floating-point numbers to be stored at the same time is approximately $144k^2N^2$. The running time is similar for both algorithms, the order of magnitude is $O(k^3N^4)$.

The numerical results are consistent with intuition. When increasing the number of phases, and, thus, decreasing the variance of the service times at the first node, both the mean number of customers and the mean residence time in the first node decrease. Hence, the customer flow from the first to the second node becomes more and more deterministic. We can observe a similar effect at the second node where the effect is weaker since the external arrivals at the second node are not influenced by the service time distribution at the first node. Naturally, since the number of customers at the second node decreases, the probability of discouraging customers after having been served at the first node slowly decreases, too.

7. Conclusion

We have shown how stationary expectations corresponding to long-run averages of additive functionals on LDQBD processes can be efficiently computed by matrix-analytic methods. Applications include long-run average costs, reward rates, steady-state queueing network performance measures, and availability measures in reliability models, amongst many others.

Our new matrix-analytic algorithm computes stationary expectations directly without at first computing the stationary distribution. In this way, compared to matrix-analytic computations of the stationary distribution, the run time of our algorithm is of the same order of magnitude, but we achieve a significant reduction of computer storage requirements by avoiding the storage of a large family of matrices. Similarly as in the case of computing stationary distributions, if the state space is infinite, it must be truncated at a certain level and an appropriate initial matrix for the computations has to be chosen, but rather than performing some kind of preprocessing step associated with the truncation level, we recommend a high truncation level. The costs (in terms of run time and memory requirements) for increasing the truncation level and for preprocessing are similar in our algorithm, but increasing the truncation level appears to be more effective than a preprocessing step associated with the truncation level.

Currently, ongoing research is concerned with obtaining error bounds where techniques based on Lyapunov functions [5], [7] are a potentially viable approach to determine truncation levels dependent on a prescribed maximum error. Furthermore, extensions of the algorithm beyond LDQBD processes are considered, for example, along the lines of [3], in which matrix-analytic methods were applied to compute stationary distributions of LDQBD processes with catastrophes where in each state the level component may drop to zero such that the generator matrix deviates from the block-tridiagonal form in its first block column.

References

[1] Asmussen, S. (2003). *Applied Probability and Queues*, 2nd edn. Springer, New York.

[2] Baumann, H. and Sandmann, W. (2010). Numerical solution of level dependent quasi-birth-and-death processes. *Procedia Comput. Sci.* **1,** 1561–1569.

[3] Baumann, H. and Sandmann, W. (2012). Steady state analysis of level dependent quasi-birth-and-death processes with catastrophes. *Comput. Operat. Res.* **39,** 413–423.

[4] Bright, L. and Taylor, P. G. (1995). Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes. *Commun. Statist. Stoch. Models* **11,** 497–525.

[5] Dayar, T., Sandmann, W., Spieler, D. and Wolf, V. (2011). Infinite level-dependent QBD processes and matrix-analytic solutions for stochastic chemical kinetics. *Adv. Appl. Prob.* **43,** 1005–1026.

[6] Gaver, D. P., Jacobs, P. A. and Latouche, G. (1984). Finite birth-and-death models in randomly changing environments. *Adv. Appl. Prob.* **16,** 715–731.

[7] Glynn, P. W. and Zeevi, A. (2008). Bounding stationary expectations of Markov processes. In *Markov Processes and Related Topics: A Festschrift for Thomas G. Kurtz* (Inst. Math. Statist. Collect. **4**), Institute of Mathematical Statistics, Beachwood, OH, pp. 195–214.

[8] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*, 3rd edn. Johns Hopkins University Press, Baltimore, MD.

[9] Hanschke, T. (1999). A matrix continued fraction algorithm for the multiserver repeated order queue. *Math. Comput. Modelling* **30,** 159–170.

[10] Knuth, D. E. (1998). *The Art of Computer Programming*, Vol. 2, 3rd edn. Addison-Wesley.

[11] Kumar, S. and Kumar, P. R. (1994). Performance bounds for queueing networks and scheduling policies. *IEEE Trans. Automatic Control* **39,** 1600–1611.

[12] Latouche, G. and Ramaswami, V. (1999). *Introduction to Matrix Analytic Methods in Stochastic Modeling*. SIAM, Philadelphia, PA.

[13] Morrison, J. R. and Kumar, P. R. (2008). Computational performance bounds for Markov chains with applications. *IEEE Trans. Automatic Control* **53,** 1306–1311.

[14] Neuts, M. F. (1981). *Matrix-Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, Baltimore, MD.

[15] Serfozo, R. (2009). *Basics of Applied Stochastic Processes*. Springer, Berlin.

[16] Stewart, W. J. (1994). *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press.

[17] Thorne, J. (1997). An investigation of algorithms for calculating the fundamental matrices in level dependent quasi birth death processes. Honours thesis, The University of Adelaide.