Performance Evaluation 70 (2013) 663–681

Contents lists available at ScienceDirect

Performance Evaluation

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

On the numerical solution of Kronecker-based infinite level-dependent QBD processes

H. Baumann [a], T. Dayar [b,*], M.C. Orhan [b], W. Sandmann [c]

[a] Department of Applied Stochastics and Operations Research, Clausthal University of Technology, D-38678 Clausthal–Zellerfeld, Germany

[b] Department of Computer Engineering, Bilkent University, TR-06800 Bilkent, Ankara, Turkey

[c] Campus E1 3, Room 325, Saarland University, D-66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Available online 11 June 2013

Keywords:
Markov chain
Level-dependent QBD process
Kronecker product
Matrix analytic method
Steady-state expectation
Call center

ABSTRACT

Infinite level-dependent quasi-birth-and-death (LDQBD) processes can be used to model Markovian systems with countably infinite multidimensional state spaces. Recently it has been shown that sums of Kronecker products can be used to represent the nonzero blocks of the transition rate matrix underlying an LDQBD process for models from stochastic chemical kinetics. This paper extends the form of the transition rates used recently so that a larger class of models including those of call centers can be analyzed for their steady-state. The challenge in the matrix analytic solution then is to compute conditional expected sojourn time matrices of the LDQBD model under low memory and time requirements after truncating its countably infinite state space judiciously. Results of numerical experiments are presented using a Kronecker-based matrix-analytic solution on models with two or more countably infinite dimensions and rules of thumb regarding better implementations are derived. In doing this, a more recent approach that reduces memory requirements further by enabling the computation of steady-state expectations without having to obtain the steady-state distribution is also considered.

1. Introduction

Continuous-time infinite level-dependent quasi-birth-and-death (LDQBD) processes [1–3] are continuous-time Markov chain (CTMC) processes that have block tridiagonal transition rate matrices of the form

$$Q = \begin{pmatrix} Q_{0,0} & Q_{0,1} & & & & \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & & & \\ & \ddots & \ddots & \ddots & & \\ & & Q_{l,l-1} & Q_{l,l} & Q_{l,l+1} & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}$$

when their states are appropriately ordered. This study is about their numerical steady-state analyses, and hereafter, we shall be omitting the terms continuous-time and infinite in designating the LDQBD models we consider since they are all continuous-time and infinite.

* Corresponding author. Tel.: +90 312 290 1981; fax: +90 312 266 4047.
E-mail addresses: hendrik.baumann@tu-clausthal.de (H. Baumann), tugrul@cs.bilkent.edu.tr (T. Dayar), morhan@cs.bilkent.edu.tr (M.C. Orhan), sandmann@cs.uni-saarland.de (W. Sandmann).

0166-5316/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.peva.2013.05.001

In the following, all vectors are column vectors as in linear algebra, except state vectors, consistent with the conventional definition of state probability vectors as row vectors. e represents a column vector of 1's. e_i represents the ith column of the identity matrix. diag(a) denotes a diagonal matrix with the entries of vector a along its diagonal. The lengths of the vectors are determined by the context in which they are used, ^{T} denotes transposition, $\times$ when used with sets denotes Cartesian product, $\otimes$ denotes Kronecker product [4], $\mathbb{1}$ denotes the indicator function, $\mathbb{Z}$ and $\mathbb{R}$ stand respectively for the sets of integers and real numbers, whereas $\mathbb{Z}_+$ and $\mathbb{R}_{\geq 0}$ denote their nonnegative subsets.

Now, let $\mathcal{S}$ denote the irreducible state space of the LDQBD process under consideration, the state of the LDQBD process at time t be given by $X(t) = (X_1(t), \ldots, X_n(t)) \in \mathcal{S}$, and $\{X(t) \in \mathcal{S},\ t \geq 0\}$ be the associated n-dimensional CTMC process. Furthermore, let the probability of the LDQBD process being in state $x = (x_1, \ldots, x_n) \in \mathcal{S}$ at time t be expressed as $Pr\{X(t) = x\} = \Pr\{X_1(t) = x_1, \ldots, X_n(t) = x_n\}$.

Levels define a partition of $\mathcal{S}$; that is, $\mathcal{S} = \bigcup_{l=0}^{\infty} \mathcal{S}^{(l)}$ and $\mathcal{S}^{(l)} \cap \mathcal{S}^{(m)} = \emptyset$ for $l \neq m$ and $l, m \in \mathbb{Z}_+$, where $\mathcal{S}^{(l)}$ is the subset of states corresponding to level l. Therefore, the nonzero blocks at level l satisfy

$$Q_{l,l-1} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}^{(l)}| \times |\mathcal{S}^{(l-1)}|}, \qquad Q_{l,l} \in \mathbb{R}^{|\mathcal{S}^{(l)}| \times |\mathcal{S}^{(l)}|}, \qquad Q_{l,l+1} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}^{(l)}| \times |\mathcal{S}^{(l+1)}|}$$

with the exception that there are only two boundary blocks at level 0.

We consider ergodic LDQBD processes and investigate the numerical computation [5,6] of their steady-state probability measures. The steady-state probability distribution row vector of an ergodic LDQBD process is defined as $\pi = \lim_{t \to \infty} Pr\{X(t)\}$, and it satisfies

$$\pi Q = 0, \qquad \sum_{x \in \mathcal{S}} \pi(x) = 1.$$

In particular, the steady-state vector of an ergodic LDQBD process can be accordingly partitioned and expressed as

$$\pi = (\pi^{(0)}, \pi^{(1)}, \ldots),$$

and its subvector at level $(l+1)$ can be obtained from the matrix analytic equation

$$\pi^{(l+1)} = \pi^{(l)} R_l$$

as shown in [1], once the conditional expected sojourn time matrix at level l

$$R_l = Q_{l,l+1}(-Q_{l+1,l+1} - R_{l+1}Q_{l+2,l+1})^{-1}$$

is determined for $l \in \mathbb{Z}_+$. Note that $\pi^{(l)} \in \mathbb{R}_{\geq 0}^{1 \times |\mathcal{S}^{(l)}|}$, $\pi^{(l+1)} \in \mathbb{R}_{\geq 0}^{1 \times |\mathcal{S}^{(l+1)}|}$, and $R_l \in \mathbb{R}_{\geq 0}^{|\mathcal{S}^{(l)}| \times |\mathcal{S}^{(l+1)}|}$. LDQBD processes are a generalization of QBD processes originally proposed in [7,8] and improved over the years [2,9] with two quadratically convergent algorithms for their steady-state analyses. These algorithms are logarithmic reduction [10] and cyclic reduction [11]. In the level dependent context, the situation is more complicated since the conditional expected sojourn time matrix is not constant and changes from level to level.

In many cases, the ergodicity of an LDQBD process can be established by relatively easy to check conditions on the 1-dimensional CTMC defined over its levels [12]. For computational purposes however, it is preferable to consider Lyapunov function methods as discussed in [13,14], so that lower and higher level numbers (called *Low* and *High*, respectively) can be computed as in [15–17] between which a specified percentage of the steady-state probability mass lies when the LDQBD is ergodic. It is the latter approach we follow here. In this way, steady-state measures can be computed exactly up to machine precision by choosing (*High*–*Low*) sufficiently large as discussed in [17].

Systems of stochastic chemical kinetics [18–21] and queueing networks [2,5,22–24] are two classes of problems that can be modeled as LDQBD processes using the described n-dimensional state representation with some countably infinite variables. For the former class of problems, countably infinite variables represent numbers of molecules of each chemical species existing in the system. The remaining variables, if any, are finite. Up until recently, stochastic simulation [25,26] seemed to be the only viable approach that would yield relatively accurate results for this class of problems. However, it has been shown in [17] that systems of stochastic chemical kinetics can be modeled as LDQBD processes with the level number determined by the maximum value among the countably infinite variables. Inspired by hierarchical Markovian models (HMMs) introduced in [22], the result in [17] has been taken one step further in [16] by providing a Kronecker-based representation for the nonzero blocks $Q_{l,l-1}$, $Q_{l,l}$, $Q_{l,l+1}$ at each level to cope with the multidimensionality of the product state space of variables and its reachability. As suggested in [27] and as observed to be the best overall choice in [17], again R_{High} is set to 0 to initiate the computational procedure associated with the matrices of conditional expected sojourn times. Therein, a comparative study between stochastic simulation and the matrix analytic approach has also been undertaken. Naturally, the matrix analytic approach yields an accuracy measure obtained by computing the residual norm of the solution which is not possible with simulation.

It is the latter class of problems on which we concentrate in this paper by considering various queueing network models of call centers with multiple types of customers. In this context, the countably infinite variables represent occupancies of queues with unbounded waiting space and finite variables represent occupancies of server pools. The interplay between finite variables and countably infinite variables in these models is more intricate than that of models of stochastic chemical kinetics systems. This requires us to extend the form of the transition rates used in [16]. To that end, we resort to generalized

functional transitions of stochastic automata networks (SANs) [28–30], and let the form of the dependency of the transition rate on the values of the variables be more general. As we shall see, the use of such transition rates does not pose a computational efficiency problem, since a direct method in which each nonzero block is processed once is employed in the matrix analytic solution. In other words, each function is evaluated once for each state in the truncated state space $\bigcup_{l=Low}^{High} \mathcal{S}^{(l)}$ during analysis. An implication of this extension is that not only do we need to compute partitions of state spaces of countably infinite variables as in [16], but now we also need to compute subpartitions of these partitions and partitions of state spaces of finite variables to obtain a Kronecker-based representation for the nonzero blocks of Q.

We also consider the more recent approach in [31] that reduces memory requirements further by enabling the computation of steady-state expectations without having to obtain the steady-state distribution. The approach is inspired by a Horner-like computational scheme in which only the conditional expected sojourn time matrix at level l needs to be allocated in step l; otherwise, there are no time savings. In other words, not all R_l matrices need to be stored simultaneously. With this memory efficient approach, a tandem queueing network of two single server queues with infinite waiting spaces and customers departing from the first queue leaving the system with a probability depending on the length of the second queue is analyzed. The steady-state measures computed are those that are based on average costs or rewards, moments, and cumulants. However, there is one drawback; it is the loss of the accuracy measure because the steady-state distribution is no longer available. Ongoing work is concerned with obtaining accuracy measures for this approach as well, but they are not yet available.

Taking into account the experience gained from the Matlab implementation used in [16], we provide an implementation in C based on the Nsolve package of the Abstract Petri Net Notation (APNN) Toolbox [32]. Our objective in doing so is to have more control over memory usage and timings. Since the number of states within each level increases with increasing level number in the models we consider, we conduct numerical experiments with timing results to see how and when memory should be allocated and deallocated, and whether the conditional sojourn time matrices should be stored as full or sparse matrices. With the existing continuous improvement in computer technology, we believe we have reason to invest in numerical analysis approaches for multidimensional Markovian models as those discussed here to obtain more accurate results at a finer level of detail.

In the next section, we extend the specification for the class of models considered to build a Kronecker representation of the nonzero blocks in Q and introduce a running example. Due to space limitations, we refrain from discussing the models of stochastic chemical kinetics systems used before and refer to [33]. We also introduce two other models of call centers at the end of Section 2. In Section 3, we discuss implementation issues including the choice of Lyapunov functions and provide results regarding memory usage for different implementations after giving the algorithm for the computation of the steady-state distribution of LDQBD processes. In Section 4, we report on the results of numerical experiments with the matrix analytic approach on the models introduced earlier to identify better implementations. In Section 5, we conclude.

2. Kronecker representation

We consider Markovian systems with interacting subsystems. The state space $\mathcal{S}$ is irreducible, countably infinite, and n-dimensional with n_I countably infinite state variables and n_F finite state variables, where $n_I \geq 2$, $n_F \geq 0$, and $n = n_I + n_F$. Hereafter, we shall be omitting the word 'state' and referring to state variables as variables. The state space of variable x_h is denoted by $\mathcal{S}_h$ and $\mathcal{S}_h \subseteq \mathbb{Z}_+$ for $h = 1, \ldots, n$. Without loss of generality, we assume that the first n_I indices correspond to countably infinite variables. Hence, $\mathcal{S}_h$ is countably infinite for $h = 1, \ldots, n_I$ and finite for $h = n_I + 1, \ldots, n$. Clearly, not all states in the product state space $\times_{h=1}^{n} \mathcal{S}_h$ are necessarily reachable. However, each state in $\mathcal{S}$ is reachable from every other state in $\mathcal{S}$ due to our assumption of irreducibility. In many cases, $\mathcal{S}$ is a proper subset of the product state space (i.e., $\mathcal{S} \subset \times_{h=1}^{n} \mathcal{S}_h$). Indeed, it is as such in the models of call centers introduced in the next section.

The n-dimensional Markovian models we consider are defined by a set of K transition classes over $\mathcal{S}$ and $x = (x_1, \ldots, x_n) \in \mathbb{Z}_+^{1\times n}$ denotes a state in $\mathcal{S}$. We relax some of the assumptions made in [16] so as to be able to analyze a larger class of models. Observe in particular that we do not require product form transition rates here (cf. Definition 1 in [16]).

Definition 1. The kth transition class is a pair $(\alpha_k, v^{(k)})$, where $\alpha_k \in \mathbb{R}_{\geq 0}$ and $v^{(k)} \in \mathbb{Z}^{1\times n}$ are respectively its transition rate and state change vector for $k = 1, \ldots, K$. The first element of the pair, $\alpha_k(x)$, is a function of state $x \in \mathcal{S}$ and specifies the transition rate from state x to state $(x + v^{(k)}) \in \mathcal{S}$. The second element of the pair, $v^{(k)}$, specifies the successor state of the transition, where $v_h^{(k)}$ denotes the value change in variable $x_h \in \mathcal{S}_h$ due to the kth transition class.

Example 1 (*N-Model*)**.** Consider the parallel service system in Fig. 1 known as the N-model under the threshold routing control policy proposed in [34]. In this model, there are two types of customers, two types of infinite queues, and two types of server pools. Customers of type 1 can be in queue 1, server pool 1, or server pool 2, but type 2 customers can only be in queue 2 or server pool 2. Hence, $x = (x_1, x_2, x_3, x_4, x_5)$ is a possible state representation, where x_1, x_3, and x_4 denote the number of type 1 customers in queue 1, pool 1, and pool 2, respectively, whereas, x_2 and x_5 denote the number of type 2 customers in queue 2 and pool 2, respectively. Then, $n = 5$, $n_I = 2$, and $n_F = 3$. Type i customers arrive to the system according to a Poisson process with rate λ_i and server pool i has N_i servers with exponentially distributed service times for $i = 1, 2$. Hence, $\mathcal{S}_1 = \mathcal{S}_2 = \mathbb{Z}_+$, $\mathcal{S}_3 = \{0, \ldots, N_1\}$, and $\mathcal{S}_4 = \mathcal{S}_5 = \{0, \ldots, N_2\}$. Upon arrival, a type 1 customer joins pool 1

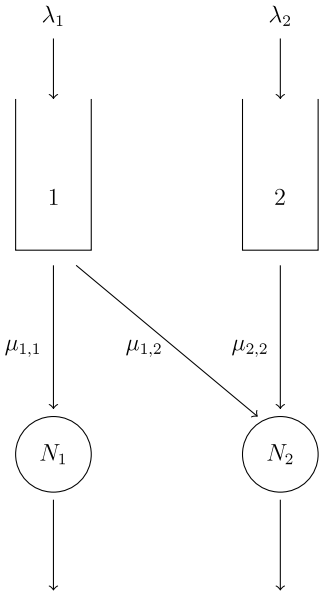


Fig. 1. N-model.

Table 1
Transition classes of the N-model.

k	$\alpha_k(x)$	$v^{(k)}$
1	$\lambda_1 \mathbb{1}_{x_3=N_1} \mathbb{1}_{x_4+x_5=N_2}$	e_1^T
2	$\lambda_1 \mathbb{1}_{x_3=N_1} \mathbb{1}_{x_4+x_5<N_2}$	e_4^T
3	$\lambda_1 \mathbb{1}_{x_3<N_1}$	e_3^T
4	$\lambda_2 \mathbb{1}_{x_4+x_5=N_2}$	e_2^T
5	$\lambda_2 \mathbb{1}_{x_4+x_5<N_2}$	e_5^T
6	$\mu_{1,1} x_3 \mathbb{1}_{x_1=0}$	$-e_3^T$
7	$\mu_{1,1} x_3 \mathbb{1}_{x_1>0}$	$-e_1^T$
8	$\mu_{1,2} x_4 \mathbb{1}_{x_1=0} \mathbb{1}_{x_2=0}$	$-e_4^T$
9	$\mu_{2,2} x_5 \mathbb{1}_{x_1=0} \mathbb{1}_{x_2=0}$	$-e_5^T$
10	$\mu_{1,2} x_4 \mathbb{1}_{x_1>0} \mathbb{1}_{x_2=0}$	$-e_1^T$
11	$\mu_{2,2} x_5 \mathbb{1}_{x_1>0} \mathbb{1}_{x_2=0}$	$(-e_1+e_4-e_5)^T$
12	$\mu_{1,2} x_4 \mathbb{1}_{x_1\leq M} \mathbb{1}_{x_2>0}$	$(-e_2-e_4+e_5)^T$
13	$\mu_{2,2} x_5 \mathbb{1}_{x_1\leq M} \mathbb{1}_{x_2>0}$	$-e_2^T$
14	$\mu_{1,2} x_4 \mathbb{1}_{x_1>M} \mathbb{1}_{x_2>0}$	$-e_1^T$
15	$\mu_{2,2} x_5 \mathbb{1}_{x_1>M} \mathbb{1}_{x_2>0}$	$(-e_1+e_4-e_5)^T$

if it has idle servers (i.e., $x_3 < N_1$) and receives service at a rate of $\mu_{1,1}$. If there are no idle servers in pool 1 (i.e., $x_3 = N_1$), an arriving type 1 customer joins pool 2 if it has idle servers (i.e., $x_4 + x_5 < N_2$) and receives service at a rate of $\mu_{1,2}$. If there are no idle servers in either pool (i.e., $x_3 = N_1, x_4 + x_5 = N_2$), an arriving type 1 customer enters queue 1. On the other hand, an arriving type 2 customer joins pool 2 if it has idle servers (i.e., $x_4 + x_5 < N_2$) and receives service at a rate of $\mu_{2,2}$; otherwise (i.e., $x_4 + x_5 = N_2$) it enters queue 2. Upon departure of a customer from server pool 2, the first customer in queue 1 joins it if the number of type 1 customers in queue 1 exceeds a given threshold M (i.e., $x_1 > M$); otherwise (i.e., $x_1 \leq M$) the first customer in queue 2 joins it. Hence, type 1 customers take nonpreemptive priority over type 2 customers in server pool 2 when the number of type 1 customers in queue 1 exceeds M.

The transition classes of this model are given in Table 1. Note that the number of transition classes is $K = 15$, N_1 and N_2 are positive integers, $M \in \mathbb{Z}_+$, and $\lambda_1, \lambda_2, \mu_{1,1}, \mu_{1,2}, \mu_{2,2} \in \mathbb{R}_{>0}$.

The following definition associates transition matrices with each transition class in Definition 1. Being motivated by generalized functional transitions in SANs [28–30], we let the form of the dependency of the transition rate on the values of the variables be more general. In order to achieve this, the local state dependent transition rates are moved out of the transition matrices and replaced with 1's in their respective places (cf. Definition 2 in [16]). As we shall see, the use of such transition rates, which are functions of (global) state $x \in \mathcal{S}$ rather than product of functions of local states $x_h \in \mathcal{S}_h$ for $h = 1, \ldots, n$, does not pose a computational efficiency problem in this context, since a direct method in which each

nonzero block is processed once is employed in the matrix analytic solution. In other words, each function is evaluated once for each global state in the truncated state space $\bigcup_{l=Low}^{High} \mathcal{S}^{(l)}$ during analysis. Note that, here we do not differentiate between countably infinite variables and finite variables in defining the transition matrices (cf. Definition 2 in [16]).

Definition 2. The transition matrix of variable $x_h \in \mathcal{S}_h$ for the kth transition class, denoted by $Z^{(k,h)} \in \mathbb{R}_{\geq 0}^{|\mathcal{S}_h|\times|\mathcal{S}_h|}$, for $h = 1, \ldots, n$ and $k = 1, \ldots, K$ is given entrywise as

$$Z^{(k,h)}(x_h, y_h) = \begin{cases} 1 & \text{if } y_h = x_h + v_h^{(k)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } x_h, y_h \in \mathcal{S}_h.$$

Here we refrain from providing transition matrices of models in Kronecker form due to the space limitations and because they all follow in a straightforward manner from Definition 2 and the respective transition classes as was shown in [16]. Our goal is to obtain a Kronecker representation for the nonzero blocks of Q from the transition rates and the transition matrices of Definitions 1 and 2, respectively. To this end, we use the same level definition in [16] since the maximum valued countably infinite variable $x_h \in \mathcal{S}_h$ for $h = 1, \ldots, n_l$ in any state $x \in \mathcal{S}$ changes by at most one through any transition due to the particular form of the state change vectors $v^{(k)}$ in the transition classes of models we consider.

Definition 3. The subset of states corresponding to level $l \in \mathbb{Z}_+$ is given by

$$\mathcal{S}^{(l)} = \{x \in \mathcal{S} \mid l = \max(x_1, \ldots, x_{n_l})\},$$

so that $\mathcal{S} = \bigcup_{l=0}^{\infty} \mathcal{S}^{(l)}$.

Note that the set operation underlying the Kronecker product is the Cartesian product, and therefore the subset of states at each level needs to be expressed as a union of Cartesian products of state space partitions of the countably infinite variables. Hence, level definitions, such as $l = \sum_{h=1}^{n_l} x_h$, which have arithmetic dependencies among countably infinite variables seem to be less suitable in trying to come up with a Kronecker representation.

For each level l, the values a variable can take depend on the values of other variables. Therefore, as in [16] first we define the partitions of state spaces of countably infinite variables where there is no such dependency in a way similar to HMMs in [22]. Then we introduce a partition of $\mathcal{S}^{(l)}$ in Definition 3 based on the partitions of state spaces of countably infinite variables.

Definition 4. Let

$$\mathcal{S}_h^{(l,u)} = \begin{cases} \{x_h \mid 0 \leq x_h \leq l-1\} & \text{if } h < u \\ \{l\} & \text{if } h = u \\ \{x_h \mid 0 \leq x_h \leq l\} & \text{if } h > u \end{cases} \quad \text{for } h, u = 1, \ldots, n_l,$$

and $\mathcal{S}_h^{(l,u)} = \mathcal{S}_h$ for $h = n_l + 1, \ldots, n$, and $u = 1, \ldots, n_l$. Then the uth partition of $\mathcal{S}^{(l)}$, denoted by $\mathcal{S}^{(l,u)}$, for $u = 1, \ldots, n_l$ satisfies

$$\mathcal{S}^{(l,u)} = \{x \in \mathcal{S}^{(l)} \mid (x_1, \ldots, x_n) \in \times_{h=1}^{n} \mathcal{S}_h^{(l,u)}\},$$

so that $\mathcal{S}^{(l)} = \bigcup_{u=1}^{n_l} \mathcal{S}^{(l,u)}$. Without loss of generality, the partitions $\mathcal{S}^{(l,u)}$ are assumed to be ordered within $\mathcal{S}^{(l)}$ according to increasing partition index, u.

Example 1 (*N-Model (cntd.)*)**.** The partitions of $\mathcal{S}_h^{(l)}$ for $h = 1, \ldots, n$ and $l \geq 0$ are computed from Definition 4 as

$$\begin{aligned}
&\mathcal{S}_1^{(0,1)} = \mathcal{S}_2^{(0,1)} = \{0\}, \quad \mathcal{S}_3^{(0,1)} = \{0, \ldots, N_1\}, \quad \mathcal{S}_4^{(0,1)} = \mathcal{S}_5^{(0,1)} = \{0, \ldots, N_2\},\\
&\mathcal{S}_1^{(l,1)} = \{l\}, \quad \mathcal{S}_2^{(l,1)} = \{0, \ldots, l\}, \quad \mathcal{S}_3^{(l,1)} = \{0, \ldots, N_1\},\\
&\mathcal{S}_4^{(l,1)} = \mathcal{S}_5^{(l,1)} = \{0, \ldots, N_2\},\\
&\mathcal{S}_1^{(l,2)} = \{0, \ldots, l-1\}, \quad \mathcal{S}_2^{(l,2)} = \{l\}, \quad \mathcal{S}_3^{(l,2)} = \{0, \ldots, N_1\},\\
&\mathcal{S}_4^{(l,2)} = \mathcal{S}_5^{(l,2)} = \{0, \ldots, N_2\} \quad \text{for } l > 0.
\end{aligned}$$

Note that partition $\mathcal{S}^{(l,u)}$ consists only of reachable states, and is a subset of the Cartesian product of state space partitions of all variables since certain values of finite variables may yield unreachable states (cf. Definition 5 in [16]). Therefore, we define partitions of $\mathcal{S}^{(l,u)}$ and $\mathcal{S}_h^{(l,u)}$ in Definition 4 to eliminate any unreachable states due to finite variables.

Definition 5. The ith partition of $\mathcal{S}^{(l,u)}$, denoted by $\mathcal{S}^{(l,u,i)}$, for $i = 1, \ldots, I^{(l,u)}$ is given by

$$\mathcal{S}^{(l,u,i)} = \{x \in \mathcal{S}^{(l,u)} \mid (x_1, \ldots, x_n) \in \times_{h=1}^{n} \mathcal{S}_h^{(l,u,i_h)}\},$$

so that $\mathcal{S}^{(l,u)} = \bigcup_{i=1}^{I^{(l,u)}} \mathcal{S}^{(l,u,i)}$, where $\mathcal{S}_h^{(l,u,i_h)}$ is the i_hth partition of $\mathcal{S}_h^{(l,u)}$ for $h = 1, \ldots, n$, $u = 1, \ldots, n_l$, and $i_h = 1, \ldots, I_h^{(l,u)}$. Without loss of generality, the partitions $\mathcal{S}^{(l,u,i)}$ are assumed to be ordered within $\mathcal{S}^{(l,u)}$ lexicographically according to the corresponding partition indices, $(i_1, \ldots, i_n)$.

Note that, partition $\mathcal{S}^{(l,u,i)}$ consists only of reachable states, and is a subset of the Cartesian product of the state space subpartitions of all variables since certain values of finite variables may yield unreachable states. As we shall see, the values of $I^{(l,u)}$ and $I_h^{(l,u)}$ for $h = 1, \ldots, n$ depend on the model, and although $I^{(l,u)} = \prod_{h=1}^{n} I_h^{(l,u)}$ was true for all the models considered in [16] (due to the fact that $I^{(l,u)}$ and $I_h^{(l,u)}$ for $h = 1, \ldots, n$ were all 1's), it need not be true in general.

Example 1 (*N-Model (cntd.)*)**.** The total number of type 1 and type 2 customers cannot exceed the number of servers in pool 2, so $x_4 + x_5 \leq N_2$. Besides, a queue can be nonempty only if all servers capable of serving that queue are busy. Then, $x_3 = N_1$ if $x_1 > 0$, and $x_4 + x_5 = N_2$ if $x_1 > 0$ or $x_2 > 0$. Due to these dependencies, partitions of $\mathcal{S}_h^{(0,1)}$ for $h = 1, \ldots, n$ and $\mathcal{S}^{(0,1)}$ can be written from Definition 5 as

$$
\begin{aligned}
&I_1^{(0,1)} = I_2^{(0,1)} = I_3^{(0,1)} = 1, \qquad I_4^{(0,1)} = I_5^{(0,1)} = N_2 + 1,\\
&\mathcal{S}_1^{(0,1,1)} = \mathcal{S}_2^{(0,1,1)} = \{0\}, \qquad \mathcal{S}_3^{(0,1,1)} = \{0, \ldots, N_1\},\\
&\mathcal{S}_4^{(0,1,i)} = \{i-1\}, \qquad \mathcal{S}_5^{(0,1,i)} = \{N_2 + 1 - i\} \quad \text{for } i = 1, \ldots, N_2 + 1,
\end{aligned}
$$

so that $I^{(0,1)} = (N_2 + 1)(N_2 + 2)/2$ and

$$
\mathcal{S}^{(0,1,i)} = \{0\} \times \{0\} \times \{0, \ldots, N_1\} \times \{x_4\} \times \{x_5\},
$$

where $i = \sum_{j=1}^{x_4} (N_2 + 2 - j) + x_5 + 1$ for $x_5 = 0, \ldots, N_2 - x_4$ and $x_4 = 0, \ldots, N_2$.

Partitions of $\mathcal{S}_h^{(l,1)}$ for $h = 1, \ldots, n$ and $\mathcal{S}^{(l,1)}$ and $l > 0$ can be written from Definition 5 as

$$
\begin{aligned}
&I_1^{(l,1)} = I_2^{(l,1)} = 1, \qquad I_3^{(l,1)} = 2, \qquad I_4^{(l,1)} = I_5^{(l,1)} = N_2 + 1,\\
&\mathcal{S}_1^{(l,1,1)} = \{l\}, \qquad \mathcal{S}_2^{(l,1,1)} = \{0, \ldots, l\}, \qquad \mathcal{S}_3^{(l,1,1)} = \{0, \ldots, N_1 - 1\}, \qquad \mathcal{S}_3^{(l,1,2)} = \{N_1\},\\
&\mathcal{S}_4^{(l,1,i)} = \{i-1\}, \qquad \mathcal{S}_5^{(l,1,i)} = \{N_2 + 1 - i\} \quad \text{for } i = 1, \ldots, N_2 + 1,
\end{aligned}
$$

so that $I^{(l,1)} = N_2 + 1$ and

$$
\mathcal{S}^{(l,1,i)} = \{l\} \times \{0, \ldots, l\} \times \{N_1\} \times \{x_4\} \times \{x_5\},
$$

where $i = x_4 + 1$ and $x_5 = N_2 - x_4$ for $x_4 = 0, \ldots, N_2$.

Partitions of $\mathcal{S}_h^{(1,2)}$ for $h = 1, \ldots, n$ and $\mathcal{S}^{(1,2)}$ can be written from Definition 5 as

$$
\begin{aligned}
&I_1^{(1,2)} = I_2^{(1,2)} = 1, \qquad I_3^{(1,2)} = 1, \qquad I_4^{(1,2)} = I_5^{(1,2)} = N_2 + 1,\\
&\mathcal{S}_1^{(1,2,1)} = \{0\}, \qquad \mathcal{S}_2^{(1,2,1)} = \{1\}, \qquad \mathcal{S}_3^{(1,2,1)} = \{0, \ldots, N_1\},\\
&\mathcal{S}_4^{(1,2,i)} = \{i-1\}, \qquad \mathcal{S}_5^{(1,2,i)} = \{N_2 + 1 - i\} \quad \text{for } i = 1, \ldots, N_2 + 1,
\end{aligned}
$$

so that $I^{(1,2)} = N_2 + 1$ and

$$
\mathcal{S}^{(1,2,i)} = \{0\} \times \{1\} \times \{0, \ldots, N_1\} \times \{x_4\} \times \{x_5\},
$$

where $i = x_4 + 1$ and $x_5 = N_2 - x_4$ for $x_4 = 0, \ldots, N_2$. Furthermore, partitions of $\mathcal{S}_h^{(l,2)}$ and $\mathcal{S}^{(l,2)}$ for $h = 1, \ldots, n$ and $l > 1$ can be written from Definition 5 as

$$
\begin{aligned}
&I_1^{(l,2)} = 2, \qquad I_2^{(l,2)} = 1, \qquad I_3^{(l,2)} = 2, \qquad I_4^{(l,2)} = I_5^{(l,2)} = N_2 + 1,\\
&\mathcal{S}_1^{(l,2,1)} = \{0\}, \qquad \mathcal{S}_1^{(l,2,2)} = \{1, \ldots, l-1\}, \qquad \mathcal{S}_2^{(l,2,1)} = \{l\},\\
&\mathcal{S}_3^{(l,2,1)} = \{0, \ldots, N_1 - 1\}, \qquad \mathcal{S}_3^{(l,2,2)} = \{N_1\},\\
&\mathcal{S}_4^{(l,2,i)} = \{i-1\}, \qquad \mathcal{S}_5^{(l,2,i)} = \{N_2 + 1 - i\} \quad \text{for } i = 1, \ldots, N_2 + 1,
\end{aligned}
$$

so that $I^{(l,2)} = 3(N_2 + 1)$ and

$$
\begin{aligned}
&\mathcal{S}^{(l,2,i_1)} = \{0\} \times \{l\} \times \{0, \ldots, N_1 - 1\} \times \{x_4\} \times \{x_5\},\\
&\mathcal{S}^{(l,2,i_2)} = \{0\} \times \{l\} \times \{N_1\} \times \{x_4\} \times \{x_5\},\\
&\mathcal{S}^{(l,2,i_3)} = \{1, \ldots, l-1\} \times \{l\} \times \{N_1\} \times \{x_4\} \times \{x_5\},
\end{aligned}
$$

where $i_1 = x_4 + 1$, $i_2 = (N_2 + 1) + x_4 + 1$, $i_3 = 2(N_2 + 1) + x_4 + 1$, $x_5 = N_2 - x_4$ for $x_4 = 0, \ldots, N_2$.

Now, we are in a position to introduce the Kronecker representation of nonzero blocks in Q (cf. Definition 6 in [16]) following the partitions of subsets of states at each level given by Definition 5.

Definition 6. The nonzero blocks $Q_{0,0}$, $Q_{0,1}$, $Q_{1,0}$, and $Q_{l,m}$ for $l > 0$, $m = l-1, l, l+1$ are respectively (1×1), $(1 \times n_l)$, $(n_l \times 1)$, and $(n_l \times n_l)$ block matrices as in

$$Q_{0,0} = \left(Q_{0,0}^{(1,1)}\right), \qquad Q_{0,1} = \left(Q_{0,1}^{(1,1)} \quad \cdots \quad Q_{0,1}^{(1,n_l)}\right),$$
$$Q_{1,0} = \begin{pmatrix} Q_{1,0}^{(1,1)} \\ \vdots \\ Q_{1,0}^{(n_l,1)} \end{pmatrix}, \qquad Q_{l,m} = \begin{pmatrix} Q_{l,m}^{(1,1)} & \cdots & Q_{l,m}^{(1,n_l)} \\ \vdots & \ddots & \vdots \\ Q_{l,m}^{(n_l,1)} & \cdots & Q_{l,m}^{(n_l,n_l)} \end{pmatrix},$$

where $Q_{l,m}^{(u,w)}$ is an $(I^{(l,u)} \times I^{(m,w)})$ block matrix given by

$$Q_{l,m}^{(u,w)} = \begin{pmatrix} Q_{l,m}^{((u,1),(w,1))} & \cdots & Q_{l,m}^{((u,1),(w,I^{(m,w)}))} \\ \vdots & \ddots & \vdots \\ Q_{l,m}^{((u,I^{(l,u)}),(w,1))} & \cdots & Q_{l,m}^{((u,I^{(l,u)}),(w,I^{(m,w)}))} \end{pmatrix}.$$

Furthermore, the blocks of $Q_{l,m}^{(u,w)}$ can be written in terms of transition rates and transition matrices as in

$$Q_{l,m}^{((u,i),(w,j))} = \begin{cases} \tilde{Q}_{l,m}^{((u,i),(w,j))} - D_{l,m}^{((u,i),(w,j))} & \text{if } u = w \text{ and } l = m \\ \tilde{Q}_{l,m}^{((u,i),(w,j))} & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, I^{(l,u)}$, $j = 1, \ldots, I^{(m,w)}$, $u, w = 1, \ldots, n_l$, and $l, m \geq 0$, where

$$D_{l,m}^{((u,i),(w,j))} = \operatorname{diag}\left(\sum_{m'=l-1}^{l+1} \sum_{w'=1}^{n_l} \sum_{j'=1}^{I^{(m,w)}} \tilde{Q}_{l,m'}^{((u,i),(w',j'))} e\right),$$
$$\tilde{Q}_{l,m}^{((u,i),(w,j))} = \sum_{k=1}^{K} \alpha_k(x) \left(\bigotimes_{h=1}^{n} Z^{(k,h)}(\mathcal{s}_h^{(l,u,i_h)}, \mathcal{s}_h^{(m,w,j_h)})\right),$$

$\alpha_k(x)$ is the transition rate computed at state $x \in \times_{h=1}^{n} \mathcal{s}_h^{(l,u,i_h)}$, and $Z^{(k,h)}(\mathcal{s}_h^{(l,u,i_h)}, \mathcal{s}_h^{(m,w,j_h)})$ denotes the submatrix of $Z^{(k,h)}$ incident on row indices in $\mathcal{s}_h^{(l,u,i_h)}$ and column indices in $\mathcal{s}_h^{(m,w,j_h)}$. The first summation in diag should have a starting index of 0 rather than -1 for the equation of the blocks $Q_{0,0}^{((1,1),(1,1))}, \ldots, Q_{0,0}^{((1,I^{(0,1)}),(1,I^{(0,1)}))}$, and the second summation in diag should have an ending index of 1 rather than n_l for the equation of the blocks $Q_{1,1}^{((1,1),(1,1))}, \ldots, Q_{1,1}^{((n_l,I^{(1,nl)}),(n_l,I^{(1,n_l)}))}$ when $m' = l - 1$.

Here we refrain from providing the Kronecker representation of nonzero blocks in Q due to space limitations, but refer to [4,16] for two such example representations. Nevertheless, we indicate for our running example how the nonzero blocks in Q will look like if they were represented as flat matrices.

Example 1 (*N-Model (cntd.)*)**.** The first few nonzero blocks of Q for the N-model with $N_1 = 1$ and $N_2 = 2$ as flat sparse matrices are given by

$$Q_{0,0} = \begin{array}{c|cccccc|cccccc} & (0,0,0,0,0) & (0,0,0,0,1) & (0,0,0,0,2) & (0,0,0,1,0) & (0,0,0,1,1) & (0,0,0,2,0) & (0,0,1,0,0) & (0,0,1,0,1) & (0,0,1,0,2) & (0,0,1,1,0) & (0,0,1,1,1) & (0,0,1,2,0) \\ \hline (0,0,0,0,0) & * & \lambda_2 & & & & & \lambda_1 & & & & & \\ (0,0,0,0,1) & \mu_{2,2} & * & \lambda_2 & & & & & \lambda_1 & & & & \\ (0,0,0,0,2) & & 2\mu_{2,2} & * & & & & & & \lambda_1 & & & \\ (0,0,0,1,0) & \mu_{1,2} & & & * & \lambda_2 & & & & & \lambda_1 & & \\ (0,0,0,1,1) & & \mu_{1,2} & & \mu_{2,2} & * & & & & & & \lambda_1 & \\ (0,0,0,2,0) & & & & 2\mu_{1,2} & & * & & & & & & \lambda_1 \\ \hline (0,0,1,0,0) & \mu_{1,1} & & & & & & * & \lambda_2 & & \lambda_1 & & \\ (0,0,1,0,1) & & \mu_{1,1} & & & & & \mu_{2,2} & * & \lambda_2 & & \lambda_1 & \\ (0,0,1,0,2) & & & \mu_{1,1} & & & & & 2\mu_{2,2} & * & & & \\ (0,0,1,1,0) & & & & \mu_{1,1} & & & \mu_{1,2} & & & * & \lambda_2 & \lambda_1 \\ (0,0,1,1,1) & & & & & \mu_{1,1} & & & \mu_{1,2} & & \mu_{2,2} & * & \\ (0,0,1,2,0) & & & & & & \mu_{1,1} & & & & 2\mu_{1,2} & & * \end{array},$$

$$
Q_{0,1} = \begin{array}{c}
\\ (0,0,0,0,0)\\ (0,0,0,0,1)\\ (0,0,0,0,2)\\ (0,0,0,1,0)\\ (0,0,0,1,1)\\ (0,0,0,2,0)\\ (0,0,1,0,0)\\ (0,0,1,0,1)\\ (0,0,1,0,2)\\ (0,0,1,1,0)\\ (0,0,1,1,1)\\ (0,0,1,2,0)
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} (1,0,1,0,2) & (1,0,1,1,1) & (1,0,1,2,0) & (1,1,1,0,2) & (1,1,1,1,1) & (1,1,1,2,0) & (0,1,0,0,2) & (0,1,0,1,1) & (0,1,0,2,0) & (0,1,1,0,2) & (0,1,1,1,1) & (0,1,1,2,0) \end{array}\\
\left(\begin{array}{cccccc|cccccc}
 & & & & & & \lambda_2 & & & & & \\
 & & & & & & & \lambda_2 & & & & \\
 & & & & & & & & \lambda_2 & & & \\
\hline
\lambda_1 & & & & & & & & & \lambda_2 & & \\
 & \lambda_1 & & & & & & & & & \lambda_2 & \\
 & & \lambda_1 & & & & & & & & & \lambda_2
\end{array}\right)
\end{array},
$$

$$
Q_{1,0} = \begin{array}{c}
\\ (1,0,1,0,2)\\ (1,0,1,1,1)\\ (1,0,1,2,0)\\ (1,1,1,0,2)\\ (1,1,1,1,1)\\ (1,1,1,2,0)\\ (0,1,0,0,2)\\ (0,1,0,1,1)\\ (0,1,0,2,0)\\ (0,1,1,0,2)\\ (0,1,1,1,1)\\ (0,1,1,2,0)
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} (0,0,0,0,0) & (0,0,0,0,1) & (0,0,0,0,2) & (0,0,0,1,0) & (0,0,0,1,1) & (0,0,0,2,0) & (0,0,1,0,0) & (0,0,1,0,1) & (0,0,1,0,2) & (0,0,1,1,0) & (0,0,1,1,1) & (0,0,1,2,0) \end{array}\\
\left(\begin{array}{cccccc|cccccc}
 & & & & & & & & \mu_{1,1} & 2\mu_{2,2} & & \\
 & & & & & & & & \mu_{1,1}+\mu_{1,2} & \mu_{2,2} & & \\
 & & & & & & & & & \mu_{1,1}+2\mu_{1,2} & & \\
\hline
 & 2\mu_{2,2} & & & & & & & & & & \\
 & \mu_{1,2} & & \mu_{2,2} & & & & & & & & \\
 & & & 2\mu_{1,2} & & & & & & & & \\
 & & & & & & & 2\mu_{2,2} & & & & \\
 & & & & & & & \mu_{1,2} & & \mu_{2,2} & & \\
 & & & & & & & & & 2\mu_{1,2} & &
\end{array}\right)
\end{array},
$$

$$
Q_{1,1} = \begin{array}{c}
\\ (1,0,1,0,2)\\ (1,0,1,1,1)\\ (1,0,1,2,0)\\ (1,1,1,0,2)\\ (1,1,1,1,1)\\ (1,1,1,2,0)\\ (0,1,0,0,2)\\ (0,1,0,1,1)\\ (0,1,0,2,0)\\ (0,1,1,0,2)\\ (0,1,1,1,1)\\ (0,1,1,2,0)
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} (1,0,1,0,2) & & (1,0,1,1,1) & (1,0,1,2,0) & (1,1,1,0,2) & (1,1,1,1,1) & (1,1,1,2,0) & (0,1,0,0,2) & (0,1,0,1,1) & (0,1,0,2,0) & (0,1,1,0,2) & (0,1,1,1,1) & (0,1,1,2,0) \end{array}\\
\left(\begin{array}{cccccc|cccccc}
* & & & \lambda_2 & & & & & & & & \\
 & * & & & \lambda_2 & & & & & & & \\
 & & * & & & \lambda_2 & & & & & & \\
\hline
2\mu_{2,2}\mathbb{1}_{1\le M} & & & * & & & & & \mu_{1,1} & 2\mu_{2,2}\mathbb{1}_{1>M} & & \\
\mu_{1,2}\mathbb{1}_{1\le M} & \mu_{2,2}\mathbb{1}_{1\le M} & & & * & & & & & \mu_{1,1}+\mu_{1,2}\mathbb{1}_{1>M} & \mu_{2,2}\mathbb{1}_{1>M} & \\
 & 2\mu_{1,2}\mathbb{1}_{1\le M} & & & & * & & & & & & \mu_{1,1}+2\mu_{1,2}\mathbb{1}_{1>M} \\
\hline
 & & & & & & * & & \lambda_1 & & & \\
 & & & & & & & * & & \lambda_1 & & \\
 & & & & & & & & * & & & \lambda_1 \\
 & & & \lambda_1 & & & \mu_{1,1} & & * & & & \\
 & & & & \lambda_1 & & & \mu_{1,1} & & * & & \\
 & & & & & \lambda_1 & & & \mu_{1,1} & & & *
\end{array}\right)
\end{array}.
$$

The nonzero blocks are generally very sparse and have nonzero entries that may depend on the level number. The off-diagonal entries of Q are nonnegative, whereas its diagonal entries are negative. Clearly, the ordering of states within a level is only fixed up to a permutation. Observe that transitions are possible only between adjacent levels and the number of states within each level increases with increasing level number. The latter is due to the increase in the number of different possibilities for the n_l countably infinite variables according to the level definition being used.

Other than the N-model in Example 1, we also consider two other models of call centers, namely the W-model under the fixed queue ratio routing control policy proposed in [35] and the V-model with $T = 2, 3, 4$ types of customers under the

static priority control policy proposed in [36]. Due to space limitations, here we provide only their brief summaries. Details of these models can be found in [33].

Example 2 (*W-model*)**.** In the W-model, there are three types of customers, three types of infinite queues, and two types of server pools. Customers of type 1 can be in queue 1 or server pool 1, customers of type 3 can be in queue 3 or server pool 2, and customers of type 2 can be in queue 2 or either server pool. Since the server pools do not differentiate between the two types of customers they serve, $x = (x_1, x_2, x_3, x_4, x_5)$ is a possible state representation, where x_1, x_2, and x_3 denote the number of customers in queues 1, 2, and 3, respectively, whereas x_4 and x_5 denote the number of busy servers in pools 1 and 2, respectively. Then $n = 5$, $n_l = 3$, $n_F = 2$ and the respective state spaces are given by $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \mathbb{Z}_+$, $\mathcal{S}_4 = \{0, \ldots, N_1\}$, $\mathcal{S}_5 = \{0, \ldots, N_2\}$, where N_i is the number of servers in pool $i = 1, 2$. For this model, $K = 19$.

Example 3 (*V-Model*)**.** In the V-model, there are as many types of infinite queues as there are types of customers, T, and one pool of N servers serving the queues. Since the servers in the pool do not differentiate among the types of customers they serve, $x = (x_1, \ldots, x_T, x_{T+1})$ is a possible state representation, where x_h denotes the number of customers of type i in queue i for $i = 1, \ldots, T$ and x_{T+1} denotes the number of busy servers. Then $n = T + 1$, $n_l = T$, $n_F = 1$, and the respective state spaces are given by $\mathcal{S}_1 = \cdots = \mathcal{S}_T = \mathbb{Z}_+$, $\mathcal{S}_{T+1} = \{0, \ldots, N\}$. For this model, $K = 1 + 3T$.

3. Implementation issues

In C, we implemented an LDQBD solver [37] built on the Nsolve package [38] of the APNN toolbox [32], which includes data structures and functions for sparse and Kronecker structured matrices. In this solver, we set R_{High} to 0 as suggested in [27] and as observed to be the best overall choice in [17].

In the next two subsections, we first discuss how *Low* and *High* are computed with the help of Lyapunov functions from stability theory. Then we explain in detail how memory usage is monitored and reported during the experiments.

3.1. Handling infiniteness

By judiciously choosing a Lyapunov function $g(x) : \mathcal{S} \to \mathbb{R}_{\geq 0}$ and therefore, $g \in \mathbb{R}_{\geq 0}^{1\times|\mathcal{S}|}$ as its corresponding row vector ordered conformally with the state indices in Q, we are able to compute the drift of the LDQBD process, $d(x) : \mathcal{S} \to \mathbb{R}$, or its corresponding row vector $d \in \mathbb{R}^{1\times|\mathcal{S}|}$ as $d^T = Qg^T$ again ordered conformally, to prove that there exists a finite set $\mathcal{C} \subset \mathcal{S}$ with positive scalars $\gamma, c \in \mathbb{R}_{\geq 0}$ for which $c = \sup_{x\in\mathcal{S}} d(x) < \infty, -\gamma \geq \sup_{x\in\mathcal{S}\setminus\mathcal{C}} d(x)$, and the set of states $g(x)$ attains a finite value is always finite if and only if the LDQBD process is ergodic. When the LDQBD process is ergodic, $\sum_{x\in\mathcal{C}} \pi(x) \geq 1 - \epsilon$, where $\epsilon = c/(c + \gamma) \in (0, 1)$. Since c is the supremum of the drift over $\mathcal{S}$, we compute the drift at the states in the neighborhood of all extrema and choose its maximum value. In doing this, the resulting nonlinear equation systems are solved using the HOM4PS2-2.0 package [39], an implementation of the polyhedral homotopy continuation method as in [15–17]. Hence, we set $Low = \min\{l \in \mathbb{Z}_+ \mid \mathcal{S}^{(l)} \cap \mathcal{C} \neq \emptyset\}$ and $High = \max\{l \in \mathbb{Z}_+ \mid \mathcal{S}^{(l)} \cap \mathcal{C} \neq \emptyset\}$, so that the finite set $\bigcup_{l=Low}^{High} \mathcal{S}^{(l)}$ contains at least $(1 - \epsilon)$ of the steady-state probability.

In [16], $g(x) = \sum_{h=1}^{n} x_h^2$ is chosen as the Lyapunov function for models of stochastic chemical kinetics systems. Unfortunately, the drift associated with this Lyapunov function in call center models we consider has a countably infinite variable whose coefficient is positive for the states where a customer from the corresponding queue cannot join the respective server pool. Although the coefficients of other countably infinite variables may be negative, there are values of $x \in \mathcal{S}$ for which the countably infinite variable with positive coefficient dominates the drift and causes it to grow unboundedly.

Example 1 (*N-Model (cntd.)*)**.** x_2 has this property when $M < x_1 < \infty$.

Hence, Lyapunov functions for call centers should be carefully chosen by considering transition classes of their models since dependencies among their variables are more intricate than those in models of stochastic chemical kinetics systems. Here, we use Lyapunov functions of the form

$$g(x) = \sum_{s=1}^{S} \left(\sum_{h\in H_s} x_h + r_s \right)^2 \tag{1}$$

so that the kth transition class affects x_h for $h \in H_s$ if $\sum_{h\in H_s} v_h^{(k)} \neq 0$, where $S \in \mathbb{Z}_+$, $r_s \in \mathbb{R}$, $H_s \subseteq \{1, \ldots, n\}$, and $\{1, \ldots, n_l\} \subseteq \bigcup_{s=1}^{S} H_s$. The corresponding drift is given by

$$d(x) = \sum_{h=1}^{n_l} d^{(h)}(x) x_h + D(x), \tag{2}$$

where $d^{(h)}(x) = 2\sum_{s=1}^{S}\sum_{k=1}^{K}\mathbb{1}_{h\in H_s}\left(\sum_{h'\in H_s}v_{h'}^{(k)}\right)\alpha_k(x)$ and

$$D(x) = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(\sum_{h\in H_s}v_h^{(k)}\right)\left(\sum_{h=n_l+1}^{n}\mathbb{1}_{h\in H_s}2x_h + 2r_s + \sum_{h\in H_s}v_h^{(k)}\right)\alpha_k(x).$$

Note that, $\sum_{h\in H_s}v_h^{(k)}$ is the total change in the value of the variables that are in H_s for the kth transition class.

In the Appendix, we first prove that the set of states $g(x)$ attains a finite value is always finite (see Lemma 1). The form of $d(x)$ suggests that $c = \sup_{x\in\mathcal{S}} d(x) < \infty$ and $\mathcal{C}$ is finite if $\sup_{x\in\mathcal{S}} D(x) < \infty$, $\sup_{x\in\mathcal{S}} d^{(h)}(x) < \infty$, and there exists a finite $U_h \in \mathbb{Z}_+$ such that $\sup_{x\in\mathcal{S},x_h>U_h} d^{(h)}(x) < 0$ for $h = 1,\ldots,n_l$ (see Lemmas 2 and 3 in the Appendix). In models of call centers we consider, the first two conditions hold since $\sup_{x\in\mathcal{S}} \alpha_k(x) < \infty$ for $k = 1,\ldots,K$ (see Table 1 and those in [33]). Therefore, Lyapunov functions should be chosen to satisfy the third condition. To that end, we let S be the number of server pools and H_s be the union of indices of variables denoting the number of customers in queues from which departing customers can join server pool s and indices of variables used in defining the number of busy servers in pool s for $s = 1,\ldots,S$. Note that the value of r_s has no effect on the finiteness of the set of states $g(x)$ attains a finite value, the boundedness of the drift, and the finiteness of the set $\mathcal{C}$.

Example 1 (*N-Model (cntd.)*)**.** For this model, $S = 2$, $H_1 = \{1, 3\}$, and $H_2 = \{1, 2, 4, 5\}$. Furthermore, $U_1 = M$ and $U_2 = 0$. These choices yield the condition

$$\lambda_1 + \lambda_2 < N_2 \ \min\left\{\mu_{1,2}, \mu_{2,2}\right\},$$

which requires that the maximum arrival rate to server pool 2 be smaller than the minimum service rate of that pool when all servers are busy.

Along with S and H_s, the value of r_s needs to be determined for a smaller *High* value. Hence, we let r_s be the negated approximation to the average number of busy servers in pool s for $s = 1,\ldots,S$, in some sense similar to what was done in [17] to find a smaller value of *High*. In the following, $H(Low, High)$ represents the number of states within levels *Low* through *High*.

Example 1 (*N-Model (cntd.)*)**.** We let $\lambda_1 = 40$, $\lambda_2 = 32$, $\mu_{1,1} = 7$, $\mu_{1,2} = 8$, $\mu_{2,2} = 9$, $M = 4$, $N_1 = 4$, and $N_2 = 11$ be the parameters. Then the drift is given by

$$\begin{aligned} d(x_1,x_2,x_3,x_4,x_5) = {} & (2(x_1+x_3-4)+1)\left(40(\mathbb{1}_{x_3=4}\,\mathbb{1}_{x_4+x_5=11}+\mathbb{1}_{x_3<4})\right)\\ & + (2(x_1+x_2+x_4+x_5-5.06)+1)\left(40\,\mathbb{1}_{x_3=4}+32\right)\\ & + (-2(x_1+x_3-4)+1)\left(7x_3+(8x_4+9x_5)(\mathbb{1}_{x_1>0}\,\mathbb{1}_{x_2=0}+\mathbb{1}_{x_1>4}\,\mathbb{1}_{x_2>0})\right)\\ & + (-2(x_1+x_2+x_4+x_5-5.06)+1)\left(7x_3\,\mathbb{1}_{x_1>0}+(8x_4+9x_5)\right)\end{aligned}$$

for the Lyapunov function

$$g(x) = (x_1+x_3-r_1)^2 + (x_1+x_2+x_4+x_5-r_2)^2,$$

where $r_1 = N_1$ and $r_2 = (\lambda_1 - N_1\mu_{1,1})/\mu_{1,2} + \lambda_2/\mu_{2,2}$ are used to obtain a smaller value of *High*. Note that r_1 is the number of servers in pool 1. Hence, $(x_1+x_3-r_1)$ is the number of customers in queue 1 if $x_3 = r_1$ and $x_4+x_5 = N_2$. It is the negated number of idle servers in pool 1 if $x_3 < r_1$ or $x_4+x_5 < N_2$ (i.e., $x_1 = 0$). On the other hand, r_2 is the average number of busy servers in pool 2 when there are no idle servers in pool 1 (i.e., $x_3 = N_1$). In this case, $(x_1+x_2+x_4+x_5-r_2)$ is the total number of customers in queues 1 and 2 plus the average number of idle servers in pool 2 if $x_4+x_5 = N_2$. It is the difference between the number of busy servers and the average number of busy servers in pool 2 if $x_4+x_5 < N_2$ (i.e., $x_1 = 0, x_2 = 0$). In this way, we have a Lyapunov function that depends on the real parameters of the model. We obtain the global maximum drift as $c = 218.22$. For $\epsilon = 0.1$, we compute $\gamma = 1963.98$, $(Low, High) = (0, 62)$, and $H(0, 62) = 50,982$.

Unless otherwise specified, in all models we set $\epsilon = 0.1$. The particular Lyapunov functions used with the W- and V-models, can be found in [33]. Note that, none of the drifts used for the models of call centers we consider are nonlinear functions of the countably infinite variables. Hence, there is no need to resort to the HOM4PS2-2.0 package for these models. We remark that *Low* turns out to be 0 in all models for the chosen parameters. If this were not the case, it would still be possible to work with R_{High} set to 0 as shown in [17]. In passing, we remark that the R_l matrices and the $\pi^{(l)}$ solution subvectors become approximations once R_{High} is set to 0, and how good approximations they are depends on the value of ϵ [4,17].

3.2. Reporting memory usage

We consider the solution of the linear system of equations

$$\tilde{R}_l A_l = B_l \quad \text{starting from } l = High - 1 \text{ down to } Low$$

for the rectangular matrix $\tilde{R}_l$ of unknowns, where

$$A_l = Q_{l+1,l+1} + \tilde{R}_{l+1}Q_{l+2,l+1}$$

is the square coefficient matrix and

$$B_l = -Q_{l,l+1}$$

is the rectangular matrix of multiple right-hand sides such that $\tilde{R}_l \in \mathbb{R}_{\geq 0}^{|\mathcal{S}^{(l)}|\times|\mathcal{S}^{(l+1)}|}$, $A_l \in \mathbb{R}^{|\mathcal{S}^{(l+1)}|\times|\mathcal{S}^{(l+1)}|}$, and $B_l \in \mathbb{R}^{|\mathcal{S}^{(l)}|\times|\mathcal{S}^{(l+1)}|}$. Clearly, $\tilde{R}_{l+1}$ must be readily available at step l for this to be possible. The fact that R_l and $\pi^{(l)}$ become approximations once R_{High} is set to 0 is indicated by using tilde over them. Then the algorithm is as follows.

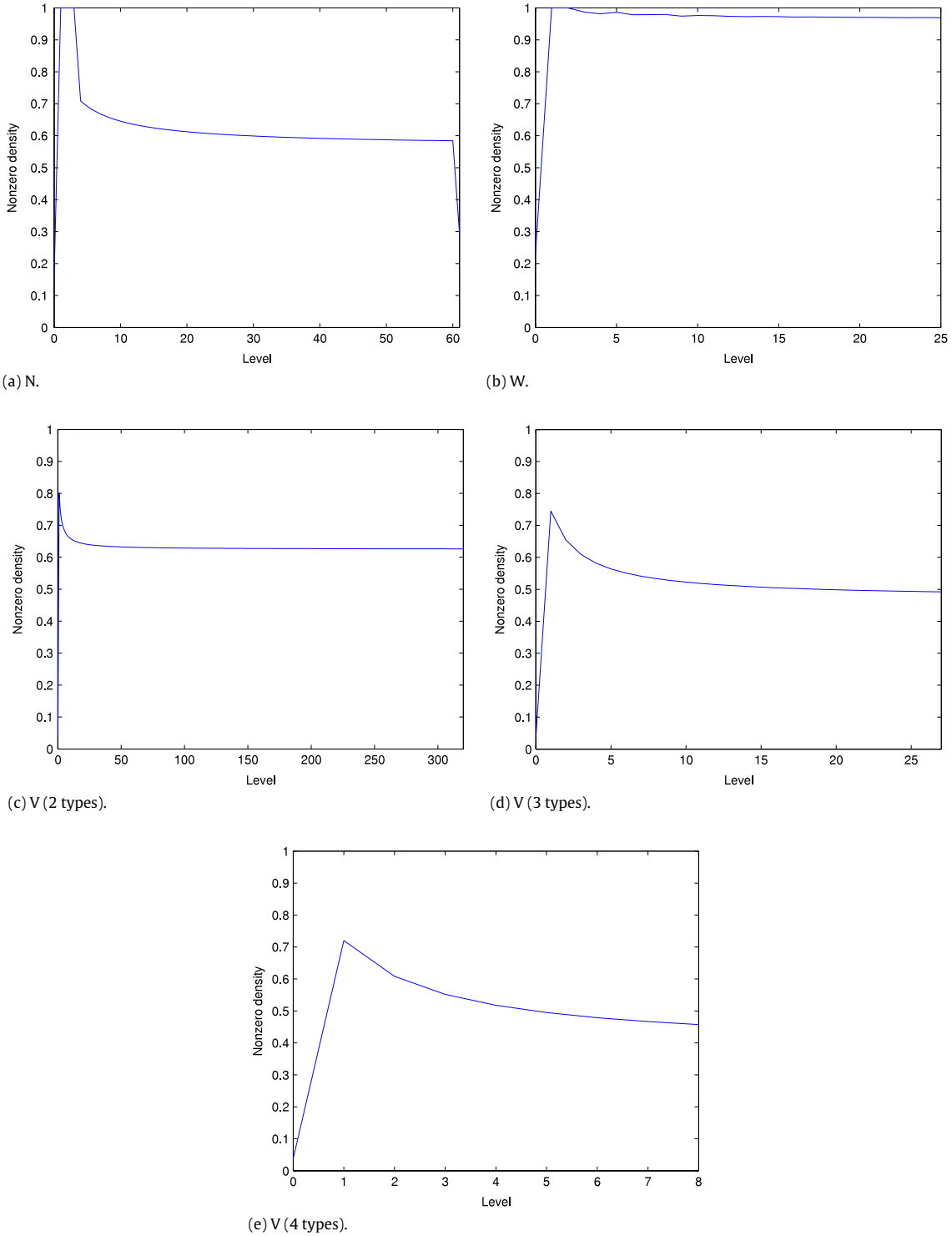
Algorithm 1. Computation of steady-state vector of an LDQBD process

Choose an appropriate Lyapunov function $g(x)$ proving ergodicity:
  Choose $g(x)$ such that the set of states for which $g(x) < \infty$ is finite
  Obtain the drift $d(x)$ and show that it is bounded
  $c \leftarrow \sup_{x\in\mathcal{S}} d(x)$ (using HOM4PS2-2.0 if necessary)
  $\gamma \leftarrow c(1/\epsilon - 1)$ for given ϵ
  Show that $\mathcal{C} = \{x \in \mathcal{S} \mid d(x) > -\gamma\}$ is finite
Compute (*Low*, *High*)
$\tilde{R}_{High} \leftarrow 0$
for $l = High - 1, \ldots, Low$ **do**
  $A_l \leftarrow Q_{l+1,l+1} + \tilde{R}_{l+1}Q_{l+2,l+1}$
  $B_l \leftarrow -Q_{l,l+1}$
  Solve $\tilde{R}_l A_l = B_l$ for $\tilde{R}_l$
endfor
$A_{Low-1} \leftarrow Q_{Low,Low} + \tilde{R}_{Low}Q_{Low+1,Low}$
Solve $\tilde{\pi}^{(Low)}A_{Low-1} = 0$ for $\tilde{\pi}^{(Low)}$ subject to $\|\tilde{\pi}^{(Low)}\|_1 = 1$
for $l = Low, \ldots, High - 1$ **do**
  $\tilde{\pi}^{(l+1)} \leftarrow \tilde{\pi}^{(l)}\tilde{R}_l$
  Normalize $\tilde{\pi}$ subject to $\|(\tilde{\pi}^{(Low)}, \ldots, \tilde{\pi}^{(l+1)})\|_1 = 1$
endfor

Having observed in our previous Matlab implementation [17] that the $\tilde{R}_l$ matrices are not necessarily very sparse (see Fig. 2), we considered their full and sparse storages. If we exclude the change in the value of the nonzero density at the boundary level when going from 1 to 0 and observe the nonzero density's value for $l \leq High - 1$, we see that it is at least 45% and attained by the V-model with 4 type of customers (see Fig. 2(e)).

In order to compute $\tilde{R}_l$, we need to obtain the nonzero blocks $Q_{l+1,l+1}$ and $Q_{l+2,l+1}$ of Q, form A_l by multiplying $\tilde{R}_{l+1}$ with $Q_{l+2,l+1}$, and then add $Q_{l+1,l+1}$ to the product. At the end, A_l should be LU factorized and the linear system solved for each right-hand side vector in B_l. The matrix–matrix multiplication $\tilde{R}_{l+1}Q_{l+2,l+1}$ can proceed in two different ways. First, $Q_{l+2,l+1}$ may be generated from the Kronecker representation as a sparse matrix and the pre-multiplication with $\tilde{R}_{l+1}$ performed. Second, the efficient vector-Kronecker product algorithm [4,28] can be used to multiply rows of $\tilde{R}_{l+1}$ with $Q_{l+2,l+1}$ while the latter operand is being held in Kronecker form.

When the $\tilde{R}_l$ matrices are chosen to be stored as full matrices, a temporary matrix in full storage needs to be kept to form A_l and then compute its LU factorization in place. Since $\tilde{R}_{High} = 0$, the sparsity pattern of A_{High-1} is equal to that of $Q_{High,High}$. Therefore, it makes sense to obtain $\tilde{R}_{High-1}$ using sparse LU factorization even though it will be stored as a full matrix. Now, although all the $\tilde{R}_l$ matrices should be kept until the end of the computation to obtain the steady-state solution and the sizes of the matrices are known at the outset (i.e., $\tilde{R}_l$ is $(|\mathcal{S}^{(l)}| \times |\mathcal{S}^{(l+1)}|)$ and A_l is $(|\mathcal{S}^{(l+1)}| \times |\mathcal{S}^{(l+1)}|)$), it is still possible to use two different memory allocation–deallocation schemes for the $\tilde{R}_l$ matrices and the temporary matrix. First, memory to store all $\tilde{R}_l$ matrices from $l = High - 1$ down to *Low* and the largest temporary matrix, A_{High-1}, can be allocated at the outset and deallocated at the end of the computation. In this scheme, successive A_l matrices will overwrite the same temporary matrix. Second, it is possible to deallocate the memory of A_l at the end of step l and allocate the memory for $\tilde{R}_{l-1}$ and A_{l-1} at the beginning of step $(l-1)$. When the temporary matrix is allocated once at the beginning of the program, peak total memory usage is higher especially when n_l is larger. See Table 2 in which peak total memory usage values are computed analytically in Megabytes (MB) using the sizes of the matrices across all models for the full storage of $\tilde{R}_l$ matrices, $\tilde{R}_l$ matrices plus temporary matrix A_{High-1} at the outset, and $\tilde{R}_l$ matrices plus A_l at each level. In the next section, these analytically obtained values will be used to verify the memory usage measurements in the numerical experiments. On the other hand, when the



(a) N.

(b) W.

(c) V (2 types).

(d) V (3 types).

(e) V (4 types).

Fig. 2. Nonzero densities of $\tilde{R}_l$ matrices at levels $l = 0 \ldots, High - 1$ for the call center models.

$\tilde{R}_l$ matrices are chosen to be stored as sparse matrices, the temporary matrix to form A_l is also stored as a sparse matrix and memory is allocated at each level. This implies that extra memory needs to be allocated for the sparse LU factorization of A_l since there is expected to be fill-in.

We monitor the memory allocation of the LDQBD solver with the `pidstat` command of the `sysstat` package under Linux. This command returns the memory allocated by the program at run time in one second intervals. When the $\tilde{R}_l$ matrices

Table 2
Peak total memory usage of full storage $\tilde{R}_l$ matrices, $\tilde{R}_l$ matrices plus A_{High-1} at outset, and $\tilde{R}_l$ matrices plus A_l matrices at each level for the call center models.

Model	*High*	*Peak*	$\tilde{R}_l$ (MB)	$\tilde{R}_l$, A_{High-1} at outset (MB)	$\tilde{R}_l$, A_l at each level (MB)
N	62	0	412	431	412
W	26	5	192	228	192
V (2 types)	321	0	354	358	354
V (3 types)	28	6	270	318	270
V (4 types)	9	7	125	219	150

and the temporary matrix are allocated at each level, the peak total memory usage is attained at a particular level, say *Peak*, between 0 and $High - 1$ (see Table 2). If allocated memory is monitored at *Peak* (meaning granularity of the monitoring period is small enough), peak total memory usage is obtained correctly. When *Peak* is close to 0 and the $\tilde{R}_l$ matrices around level 0 are computed relatively fast, peak total memory usage may not be reported correctly. In order to circumvent this measurement problem, we repeated the numerical experiments three times for each model, reported the maximum of the peak total memory usages, and indicated the level at which the maximum is attained as *Peak*.

In the proposed Kronecker representation of Definition 6, nonzero entries of transition matrices of variables should be available during computation when required. In many cases, transition matrices have specific nonzero structures such as being subdiagonal, diagonal, or superdiagonal. If two transition matrices belonging to the same transition class have the same nonzero structure, then it is possible for the two matrices to share the storage space of one vector. The vector will be allocated as long as the larger state space size of two variables in this case. Besides, when $I_h^{(0,1)} = 1$ and $I_h^{(l,u)} = 1$ for $h = 1, \ldots, n, u = 1, \ldots, n_l$, and $l > Low$, memory required to store the nonzero entries of submatrices of transition matrices at a given level is smaller than that of a higher level. In this case, it is feasible to allocate memory to store submatrices once at the highest level and keep reusing it when moving from level *High* down to *Low*. Otherwise, memory necessary to store nonzero entries may be allocated and deallocated on the fly. Furthermore, vector-Kronecker product multiplication requires an additional vector over vector–matrix multiplication. When $\tilde{R}_l$ and A_l are stored as sparse matrices, an additional temporary vector is used to compute, compact, and store the rows of A_l. Besides, adding a row of a matrix in Kronecker form to a vector requires two additional vectors. We also store the values of the transition rate functions for states in levels $l-1, l, l+1$ when processing level l in order not to evaluate the functions more than once. We allocate all the additional vectors at the beginning of the program and deallocate them at the end. Amount of memory allocated for all these vectors is negligible compared to the total amount of memory allocated for the $\tilde{R}_l$ matrices. Hence, we do not report them separately.

The more recent approach in [31] that reduces memory requirements by not having to store all $\tilde{R}_l$ matrices for $l = High-1$ down to *Low* simultaneously is also implemented. The approach is based on a Horner-like computational scheme. In order to evaluate K different functions of the steady-state distribution, $(K+1)$ temporary vectors as long as the number of states within levels *Low* through *High* must be used. For instance, if the mean is to be computed for $K = n_l$ countably infinite variables, $(n_l + 1)$ temporary vectors of length $H(Low, High)$ must be allocated. The additional vector is employed for normalization purposes. At step l, $\tilde{R}_l$ is computed as usual and stored. This implies that $\tilde{R}_{l+1}$ from the previous step need not be in memory any longer, and hence, $\tilde{R}_l$ is the only conditional expected sojourn time matrix in memory at step l in this approach. Then $\tilde{R}_l$ is multiplied with the subvectors corresponding to level $(l+1)$ of the $(K+1)$ temporary vectors. The product is added to the running sum of subvectors corresponding to level l of the $(K+1)$ temporary vectors in order to keep on accumulating steady-state expectations. This alternative approach is not able to compute the steady-state distribution and does not introduce any time savings, but yields significant memory savings at the expense of loss of the accuracy measure as we shall see in the next section.

4. Numerical results

We performed experiments on a PC with an Intel Core2 Duo 2.4 GHz processor and 4 Gigabytes (GB) of main memory. The $\tilde{R}_l$ and A_l matrices are allocated at each level having observed that it is the more memory efficient implementation in Table 2. We considered the eight LDQBD solvers listed in Table 3, where π indicates that the solver computes the steady-state distribution and E_π indicates that the solver uses the alternative memory efficient approach [31], thus computing steady-state expectations but not the distribution. Timing results are provided in seconds (s) of CPU time.

In the first set of results in Table 4 we present, solver E_π emerges as the better one in terms of memory usage. This is in line with our expectations. Memory savings can be substantial as in the N-model and V-model with 2 types of customers. In between full and sparse storages of $\tilde{R}_l$ matrices, full storage is better in the W-model. This is a model having very high nonzero densities in the $\tilde{R}_l$ matrices for $l = 0, \ldots, High - 1$ (see Fig. 2). In the call center models except the W-model, sparse storage yields better results than full storage in terms of memory usage. When there are memory savings with sparse storage of $\tilde{R}_l$ matrices, the respective time savings are even more substantial. In this case, the LU factorization of A_l seems to be benefiting considerably from sparsity. On the other hand, there is no significant difference between using sparse versus Kronecker representations of the $Q_{l+1,l}$ matrices. We believe this to be the case because each subdiagonal nonzero block is

Table 3
LDQBD solvers used.

Solver	Type	$\tilde{R}_l$	$Q_{l+1,l}$
1	π	Full	Kronecker
2			Sparse
3		Sparse	Kronecker
4			Sparse
5	E_π	Full	Kronecker
6			Sparse
7		Sparse	Kronecker
8			Sparse

Table 4
Time in s (T_i) and memory requirements in MB (M_i) for LDQBD solver i, $i = 1, \ldots, 8$, in the call center models.

Model	*High*	T_1 M_1	T_2 M_2	T_3 M_3	T_4 M_4	T_5 M_5	T_6 M_6	T_7 M_7	T_8 M_8
N	62	149	146	51	48	150	146	51	47
		417	416	363	364	43	43	28	28
W	26	79	78	106	105	79	78	106	105
		196	194	284	284	71	71	93	93
V (2 types)	321	44	43	16	15	44	42	15	15
		358	359	341	340	11	10	9	7
V (3 types)	28	127	126	21	20	127	127	21	20
		273	273	205	202	93	93	44	44
V (4 types)	9	46	46	8	8	45	46	8	8
		154	154	88	89	148	148	57	57

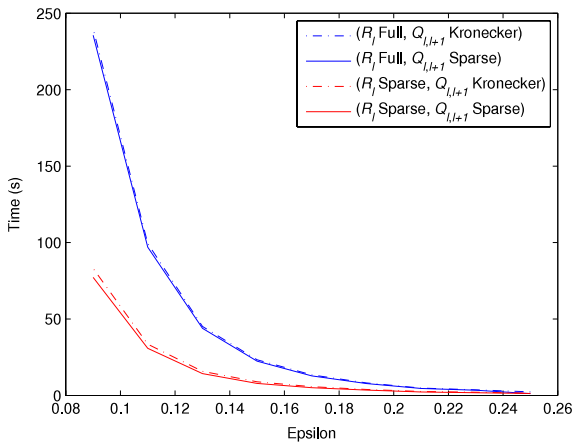
Table 5
Mean values of variables, their errors and relative errors for solver E_π with respect to the mean values obtained with solver π in the call center models.

Model	*High*	Variable	Mean	Error	Rel. err
N	62	X_1	0.03944	2e−15	5e−14
		X_2	0.03051	1e−15	3e−14
		X_3	3.15592	1e−13	4e−14
		X_4	2.23857	2e−13	7e−14
		X_5	3.55556	1e−13	3e−14
W	26	X_1	0.00831	2e−16	3e−14
		X_2	0.00351	1e−16	3e−14
		X_3	0.06069	2e−15	3e−14
		X_4	3.74418	9e−14	3e−14
		X_5	3.61130	1e−13	3e−14
V (2 types)	321	X_1	0.83072	2e−14	3e−14
		X_2	25.96154	5e−13	2e−14
		X_3	26.54969	4e−13	2e−14
V (3 types)	28	X_1	0.07139	3e−15	4e−14
		X_2	0.12167	3e−15	2e−14
		X_3	0.31274	1e−14	4e−14
		X_4	19.24928	8e−13	4e−14
V (4 types)	9	X_1	0.00570	3e−17	5e−15
		X_2	0.00743	2e−17	3e−15
		X_3	0.01001	5e−17	5e−15
		X_4	0.01299	4e−17	3e−15
		X_5	15.74957	8e−14	5e−15

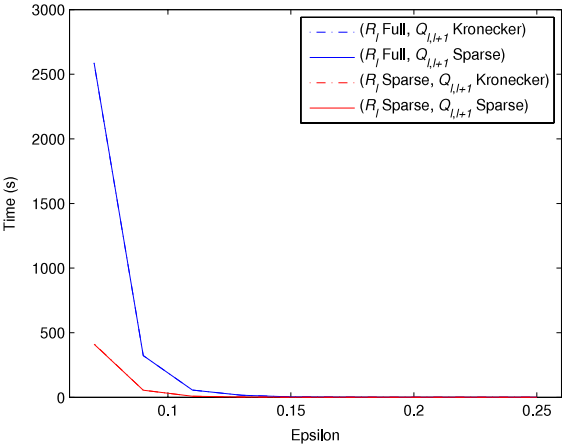
used once and the sparse generation procedure associated with it and the pre-multiplication with $\tilde{R}_{l+1}$ amount to performing the same number of floating-point operations as would be done by the vector-Kronecker product multiplication algorithm between the rows of $\tilde{R}_{l+1}$ and the subdiagonal nonzero block when the latter is kept in Kronecker form.

Table 5 indicates that solver E_π is able to compute mean values of variables stably as solver π. The reported relative errors of the mean values obtained with solver E_π with respect to those obtained with solver π are close to machine precision in all cases.

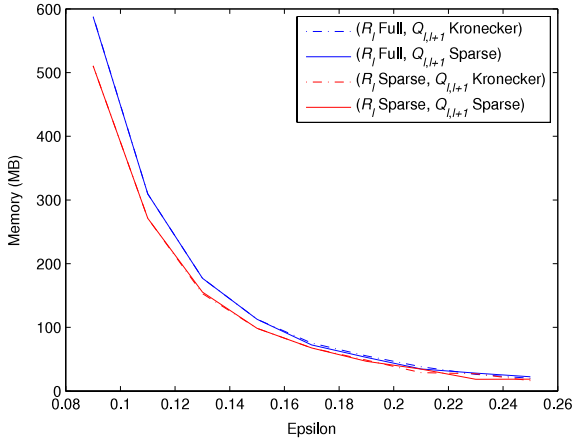
We have investigated the scalability of the eight LDQBD solvers for increasing values of ϵ. In Fig. 3, we plot the results for the N-model and V-model with 3 types of customers. In doing this, we provide timing results only with solver π since timings of solvers π and E_π are almost identical as can be seen in Table 4. Note that we have two models with $n_l = 2$ (N-model and V-model with 2 types of customers), two models with $n_l = 3$ (W-model and V-model with 3 types of customers), and one model with $n_l = 4$ (V-model with 4 types of customers). Results of numerical experiments on these models for increasing values of *High* can be found in [33]. Since time complexity of A_l's LU factorization at level l is cubic in the order



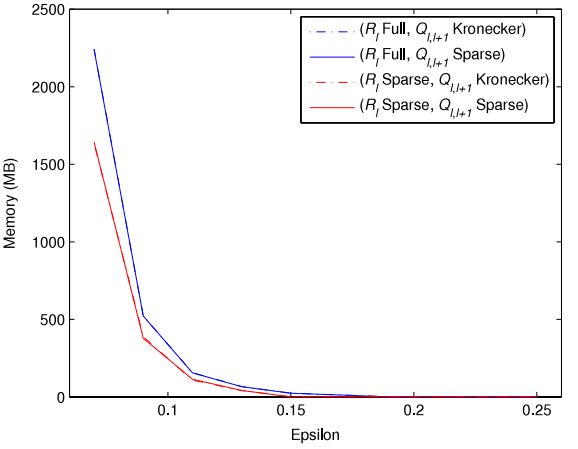
(a) N time.



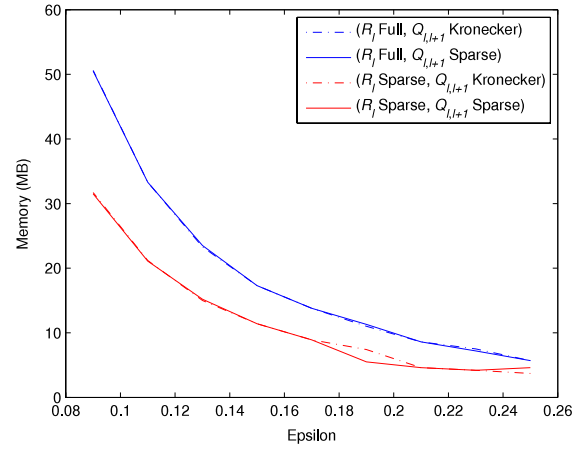
(b) V (3 types) time.



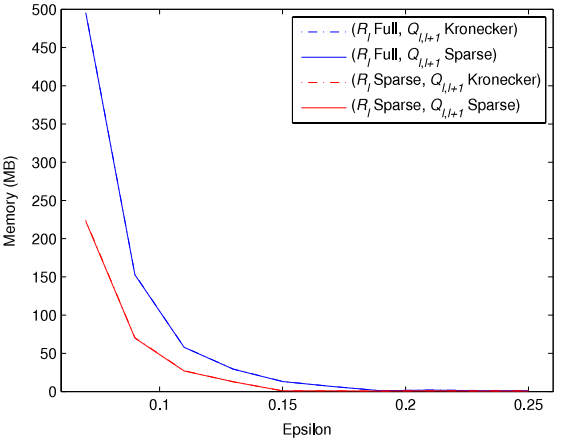
(c) N memory for solver π.



(d) V (3 types) memory for solver π.



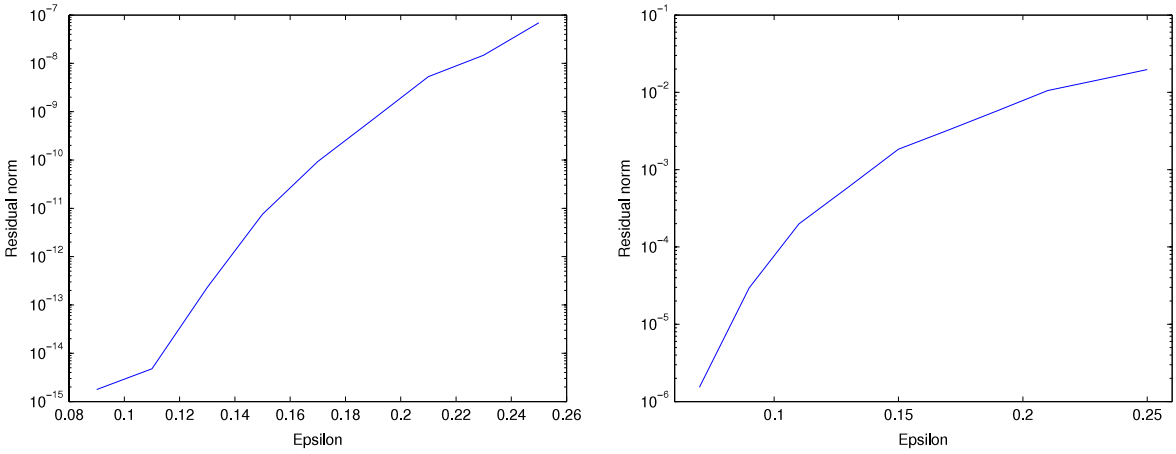
(e) N memory for solver E_π.



(f) V (3 types) memory for solver E_π.

Fig. 3. Time and memory requirements of LDQBD solvers for increasing ϵ values of the call center N-model and V-model with 3 types of customers.

of $|\mathcal{S}^{(l)}|$ for dense $\tilde{R}_l$ matrices and $|\mathcal{S}^{(l)}|$ is a polynomial with degree $(l-1)$, time requirements become more pronounced for models with higher n_l values. The situation regarding memory is better since the requirements at level l is quadratic in $|\mathcal{S}^{(l)}|$ for dense $\tilde{R}_l$ matrices. Clearly, the time and memory requirements are much better when the $\tilde{R}_l$ matrices are sparser.



(a) N. (b) V (3 types).

Fig. 4. Logarithmic residual norms of LDQBD solver for increasing ϵ values of the call center N-model and V-model with 3 types of customers.

In Fig. 4, we provide the logarithmic residual norms obtained with solver π for the steady-state distributions $\tilde{\pi}$ of the N-model and V-model with 3 types of customers for increasing values of ϵ. The results confirm our recent findings in [17] that it is possible to improve the accuracy of the solution in the residual norm (toward machine precision) by considering smaller values of ϵ in each model.

5. Conclusion

We have considered the Kronecker representation of nonzero blocks of transition matrices underlying level-dependent QBD processes. A representation proposed recently for models from systems of stochastic chemical kinetics has been extended so that queueing network models of call centers can also be analyzed for their steady-state. The improved representation requires the partitions of state spaces of countably infinite state variables obtained recently and the state spaces of the finite state variables to be partitioned to eliminate unreachable states from the product state space of the model.

An analysis regarding peak memory usage has suggested that it is better to allocate memory for the matrix of conditional expected sojourn times and the coefficient matrix of the linear system to be solved at each level before the computation at that level commences rather than at the outset. Having decided on when to allocate memory for the matrices used in the matrix analytic solution, an extensive numerical study is undertaken in which full versus sparse storages of the matrices of conditional expected sojourn times and sparse versus Kronecker representations of the subdiagonal nonzero blocks are considered for implementation. Results indicate that it is better to opt for full storage of the matrices of conditional expected sojourn times when they are relatively dense, whereas either of sparse and Kronecker representations of the subdiagonal nonzero blocks can be used. Besides, a more recent approach that enables the computation of steady-state expectations without computing the steady-state distribution has been shown to perform stably and can be utilized to improve memory usage at the expense of losing the accuracy measure associated with the solution. Ongoing work is concerned with obtaining accuracy measures for this approach as well, but they are not yet available. Both the more recent approach and the original matrix analytic solution have room for improvement regarding their scalability for models with a large number of countably infinite state variables.

Acknowledgments

The work of M.C. Orhan is supported by The Scientific and Technological Research Council of Turkey. We thank the anonymous referees whose comments led to an improved manuscript.

Appendix

Lemma 1. *The set of states the Lyapunov function $g(x)$ in (1) attains a finite value is always finite.*

Proof. Consider the Lyapunov function $g(x)$ with $\{1, \ldots, n_I\} \subseteq \bigcup_{s=1}^{S} H_s$

$$g(x) = \sum_{s=1}^{S} \left(\sum_{h \in H_s} x_h + r_s \right)^2 = \sum_{s=1}^{S} \left[\left(\sum_{h \in H_s} x_h \right)^2 + 2 \left(\sum_{h \in H_s} x_h \right) r_s + r_s^2 \right]$$

$$\geq \sum_{s=1}^{S}\sum_{h=1}^{n}\left(\mathbb{1}_{h\in H_s}x_h^2+\mathbb{1}_{h\in H_s}2r_sx_h\right)+\sum_{s=1}^{S}r_s^2$$
$$=\sum_{s=1}^{S}\sum_{h=1}^{n_I}\left(\mathbb{1}_{h\in H_s}x_h^2+\mathbb{1}_{h\in H_s}2r_sx_h\right)+G(x),$$

where $G(x)=\sum_{s=1}^{S}\sum_{h=n_I+1}^{n}\mathbb{1}_{h\in H_s}\left(x_h^2+2r_sx_h\right)+\sum_{s=1}^{S}r_s^2$ is finite. Then

$$g(x)\geq\sum_{h=1}^{n_I}\left[\left(\sum_{s=1}^{S}\mathbb{1}_{h\in H_s}\right)x_h^2+2\left(\sum_{s=1}^{S}\mathbb{1}_{h\in H_s}r_s\right)x_h\right]+G(x).$$

Let $a_{h,1}$ and $a_{h,2}$ denote $\sum_{s=1}^{S}\mathbb{1}_{h\in H_s}$ and $\left(\sum_{s=1}^{S}\mathbb{1}_{h\in H_s}r_s\right)/\left(\sum_{s=1}^{S}\mathbb{1}_{h\in H_s}\right)$, respectively. $\bigcup_{s=1}^{S}H_s$ includes all countably infinite variables by our assumption. So, $a_{h,1}>0$ holds and $a_{h,2}$ is well-defined for $h=1,\ldots,n_I$. Therefore,

$$g(x)\geq\sum_{h=1}^{n_I}a_{h,1}\left(x_h^2+2a_{h,2}x_h\right)+G(x)$$
$$=\sum_{h=1}^{n_I}a_{h,1}\left[\left(x_h+a_{h,2}\right)^2-a_{h,2}^2\right]+G(x)$$
$$=\sum_{h=1}^{n_I}a_{h,1}\left(x_h+a_{h,2}\right)^2-\sum_{h=1}^{n_I}a_{h,1}a_{h,2}^2+G(x).$$

Now, $\sum_{h=1}^{n_I}a_{h,1}\left(x_h+a_{h,2}\right)^2$ attains finite values only in a finite number of states since it is a weighted sum of squares of all countably infinite variables each shifted by a scalar and $-\sum_{h=1}^{n_I}a_{h,1}a_{h,2}^2+G(x)$ is finite. The set of states that $g(x)$ is finite is a subset of the states of a function that is smaller than $g(x)$ in all states. Hence, the set of states $g(x)$ attains a finite value is always finite if $\{1,\ldots,n_I\}\subseteq\bigcup_{s=1}^{S}H_s$. □

Lemma 2. *The drift $d(x)$ in (2) satisfies $c=\sup_{x\in\mathcal{S}}d(x)<\infty$ if $\sup_{x\in\mathcal{S}}D(x)<\infty$, $\sup_{x\in\mathcal{S}}d^{(h)}(x)<\infty$, and there exists a finite $U_h\in\mathbb{Z}_+$ such that $\sup_{x\in\mathcal{S},x_h>U_h}d^{(h)}(x)<0$ for $h=1,\ldots,n_I$.*

Proof. The result follows from

$$\sup_{x\in\mathcal{S}}d(x)\leq\sum_{h=1}^{n_I}\sup_{x\in\mathcal{S}}(d^{(h)}(x)x_h)+\sup_{x\in\mathcal{S}}D(x)$$
$$\leq\sum_{h=1}^{n_I}\max(\sup_{x\in\mathcal{S},x_h\leq U_h}(d^{(h)}(x)x_h),\sup_{x\in\mathcal{S},x_h>U_h}(d^{(h)}(x)x_h))+\sup_{x\in\mathcal{S}}D(x)$$
$$\leq\sum_{h=1}^{n_I}\max(\sup_{x\in\mathcal{S},x_h\leq U_h}(d^{(h)}(x)x_h),0)+\sup_{x\in\mathcal{S}}D(x)$$
$$\leq\sum_{h=1}^{n_I}U_h\max(\sup_{x\in\mathcal{S}}d^{(h)}(x),0)+\sup_{x\in\mathcal{S}}D(x)<\infty.\quad\square$$

Lemma 3. *The set $\mathcal{C}=\{x\in\mathcal{S}\mid d(x)>-\gamma\}$ is finite for the drift $d(x)$ in (2) if $\sup_{x\in\mathcal{S}}D(x)<\infty$, $\sup_{x\in\mathcal{S}}d^{(h)}(x)<\infty$, and there exists a finite $U_h\in\mathbb{Z}_+$ such that $\sup_{x\in\mathcal{S},x_h>U_h}d^{(h)}(x)<0$ for $h=1,\ldots,n_I$, where $\gamma=c(1/\epsilon-1)\in\mathbb{R}_{\geq0}$, $c=\sup_{x\in\mathcal{S}}d(x)<\infty$, and $\epsilon\in(0,1)$.*

Proof. Let us define

$$m_h=-\frac{\gamma+\sum_{h'=1}^{n_I}\mathbb{1}_{h'\neq h}\sup_{x\in\mathcal{S}}(d^{(h')}(x)x_{h'})+\sup_{x\in\mathcal{S}}D(x)}{\sup_{x\in\mathcal{S}}d^{(h)}(x)}$$

and

$$\mathcal{D}_h=\{x\in\mathcal{S}\mid x_h>\max(U_h,m_h)\}$$

for $h = 1, \ldots, n_l$. Observing that $m_h \geq -\gamma - d(x) + d^{(h)}(x)x_h / \sup_{x\in\mathcal{S}} d^{(h)}(x)$ holds for $x \in \mathcal{S}$, $x_h > U_h$, and $h = 1, \ldots, n_l$, we have

$$\begin{aligned}\mathcal{D}_h &\subseteq \{x \in \mathcal{S} | x_h > U_h, \sup_{x\in\mathcal{S}}(d^{(h)}(x))x_h < -\gamma - d(x) + d^{(h)}(x)x_h\}\\ &\subseteq \{x \in \mathcal{S} | x_h > U_h, d(x) + (\sup_{x\in\mathcal{S}} d^{(h)}(x) - d^{(h)}(x))x_h < -\gamma\}\\ &\subseteq \{x \in \mathcal{S} | x_h > U_h, d(x) < -\gamma\}.\end{aligned}$$

Then $\mathcal{C} \cap \mathcal{D}_h = \emptyset$ since $\mathcal{C} \cap \{x \in \mathcal{S} | x_h > U_h, d(x) < -\gamma\} = \emptyset$ for $h = 1, \ldots, n_l$, and

$$\mathcal{C} \subseteq \mathcal{S} \setminus \left(\bigcup_{h=1}^{n_l} \mathcal{D}_h\right) = \{x \in \mathcal{S} | x_h \leq \max(U_h, m_h) \text{ for } h = 1, \ldots, n_l\}$$

holds. The set $\mathcal{C}$ is a subset of the finite set

$$\{x \in \mathcal{S} | x_h \leq \max(U_h, m_h) \text{ for } h = 1, \ldots, n_l\}.$$

Hence, $\mathcal{C}$ is also finite. □

References

[1] L. Bright, P.G. Taylor, Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes, Stochastic Models 11 (1995) 497–525.
[2] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, SIAM Press, Philadelphia, PA, 1999.
[3] V. Ramaswami, P.G. Taylor, Some properties of the rate operators in level dependent quasi-birth-and-death processes with a countable number of phases, Stochastic Models 12 (1996) 143–164.
[4] T. Dayar, Analyzing Markov Chains Using Kronecker Products: Theory and Applications, Springer, New York, 2012.
[5] W.K. Grassman (Ed.), Computational Probability, Kluwer, Norwell, MA, 2000.
[6] W.J. Stewart, Introduction to the Numerical Solution of Markov Chains, Princeton University Press, Princeton, NJ, 1994.
[7] M.F. Neuts, Matrix-Geometric Solutions in Stochastic Models, The Johns Hopkins University Press, Baltimore, MD, 1981.
[8] M.F. Neuts, Structured Stochastic Matrices of M/G/1 Type and Their Applications, Marcel Dekker, New York, 1989.
[9] D.A. Bini, G. Latouche, B. Meini, Numerical Methods for Structured Markov Chains, Oxford University Press, Oxford, 2005.
[10] G. Latouche, V. Ramaswami, A logarithmic reduction algorithm for quasi-birth-and-death processes, Journal of Applied Probability 30 (1993) 650–674.
[11] D.A. Bini, B. Meini, On the solution of a nonlinear matrix equation arising in queueing problems, SIAM Journal on Matrix Analysis and Applications 17 (1996) 906–926.
[12] P.K. Pollett, P.G. Taylor, On the problem of establishing the existence of stationary distributions for continuous-time Markov chains, Probability in the Engineering and Informational Sciences 7 (1993) 529–543.
[13] P. Glynn, A. Zeevi, Bounding stationary expectations of Markov processes, in: S.N. Ethier, J. Feng, R.H. Stockbridge (Eds.), Markov Processes and Related Topics: A Festschrift for Thomas G. Kurtz, IMS Collections, vol. 4, IMS, Beachwood, Ohio, 2008, pp. 195–214.
[14] R.L. Tweedie, Sufficient conditions for regularity, recurrence and ergodicity of Markov processes, Mathematical Proceedings of the Cambridge Philosophical Society 78 (1975) 125–136.
[15] T. Dayar, H. Hermanns, D. Spieler, V. Wolf, Bounding the equilibrium distribution of Markov population models, Numerical Linear Algebra with Applications 18 (2011) 931–946.
[16] T. Dayar, M.C. Orhan, Kronecker-based infinite level-dependent QBDs, Journal of Applied Probability 49 (2012) 1166–1187.
[17] T. Dayar, W. Sandmann, D. Spieler, V. Wolf, Infinite level-dependent QBD processes and matrix analytic solutions for stochastic chemical kinetics, Advances in Applied Probability 38 (2011) 1005–1026.
[18] T.G. Kurtz, The relationship between stochastic and deterministic models for chemical reactions, Journal of Chemical Physics 57 (1972) 2976–2978.
[19] D.A. McQuarrie, Stochastic approach to chemical kinetics, Journal of Applied Probability 4 (1967) 413–478.
[20] K. Singer, Application of the theory of stochastic processes to the study of irreproducible chemical reactions and nucleation processes, Journal of the Royal Statistical Society: Series B 15 (1953) 92–106.
[21] N.G. van Kampen, Stochastic Processes in Physics and Chemistry, Elsevier, Amsterdam, The Netherlands, 1992.
[22] P. Buchholz, A class of hierarchical queueing networks and their analysis, Queueing Systems 15 (1994) 59–80.
[23] T. Hanschke, A matrix continued fraction algorithm for the multiserver repeated order queue, Mathematical and Computer Modelling 30 (1999) 159–170.
[24] W.J. Stewart, Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling, Princeton University Press, Princeton, NJ, 2009.
[25] D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions, The Journal of Physical Chemistry 81 (1977) 2340–2361.
[26] H. Li, Y. Cao, L.R. Petzold, D. Gillespie, Algorithms and software for stochastic simulation of biochemical reacting systems, Biotechnology Progress 24 (2008) 56–62.
[27] H. Baumann, W. Sandmann, Numerical solution of level dependent quasi-birth-and-death processes, Procedia Computer Science 1 (2010) 1555–1563.
[28] P. Fernandes, B. Plateau, W.J. Stewart, Efficient descriptor–vector multiplications in stochastic automata networks, Journal of the ACM 45 (1998) 381–414.
[29] B. Plateau, On the stochastic structure of parallelism and synchronization models for distributed algorithms, Performance Evaluation Review 13 (2) (1985) 147–154.
[30] B. Plateau, W.J. Stewart, Stochastic automata networks, in: W.K. Grassmann (Ed.), Computational Probability, Kluwer, Norwell, MA, 2000, pp. 113–152.
[31] H. Baumann, W. Sandmann, Computing stationary expectations in level-dependent QBD processes, Journal of Applied Probability 50 (2013) 151–165.
[32] F. Bause, P. Buchholz, P. Kemper, A toolbox for functional and quantitative analysis of DEDS, in: R. Puigjaner, N.N. Savino, B. Serra (Eds.), Quantitative Evaluation of Computing and Communication Systems, in: Lecture Notes in Computer Science, vol. 1469, Springer, Berlin, 1998, pp. 356–359.
[33] H. Baumann, T. Dayar, M.C. Orhan, W. Sandmann, On the numerical solution of Kronecker-based infinite level-dependent QBD processes, Technical Report BU-CE-1206, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2012.
[34] S.L. Bell, R.J. Williams, Dynamic scheduling of a parallel server system in heavy traffic with complete resource pooling: asymptotic optimality of a threshold policy, The Annals of Applied Probability 11 (2001) 608–649.
[35] I. Gurvich, W. Whitt, Service-level differentiation in many-server service systems: a solution based on fixed-queue-ratio routing, Operations Research 29 (2007) 567–588.
[36] I. Gurvich, M. Armony, A. Mandelbaum, Service-level differentiation in call centers with fully flexible servers, Management Science 54 (2008) 279–294.

[37] T. Dayar, M.C. Orhan, LDQBD Solver version 3, http://www.cs.bilkent.edu.tr/~tugrul/software.html (accessed: 21.01.13).
[38] P. Buchholz, The Nsolve Package, http://ls4-www.cs.tu-dortmund.de/download/buchholz/struct-matrix-market.html (accessed: 21.01.13).
[39] T.L. Lee, T.Y. Li, C.H. Tsai, HOM4PS-2.0, a software package for solving polynomial systems by the polyhedral homotopy continuation method, Computing 83 (2008) 109–133.

H. Baumann is a postdoc working at Clausthal University of Technology, Germany. He graduated in mathematics in 2008 and completed his Ph.D. in 2010 with the thesis titled "Continued fractions in Banach algebras—convergence criteria and applications" at the same university. His research interests include numerical methods for Markov chains, particularly matrix-analytic methods, and (generalized) matrix continued fractions and their applications to difference equations.

T. Dayar received his B.S. degree in Computer Engineering from Middle East Technical University, Ankara, Turkey, in 1989, and the M.S. and Ph.D. degrees in Computer Science from North Carolina State University, Raleigh, NC, in 1991 and 1994, respectively. Since 1995, he has been with the Department of Computer Engineering at Bilkent University, Ankara, Turkey, where he is now a full professor. His research interests are in the areas of performance modeling and analysis, numerical linear algebra for stochastic matrices, scientific computing, bioinformatics, and computer networks. He is a member of Upsilon Pi Epsilon, IEEE Computer Society, ACM Special Interest Group on Measurement and Evaluation, SIAM Activity Group on Linear Algebra, and AMS.

M.C. Orhan received his B.S. and M.S. degrees in Computer Engineering from Bilkent University, Ankara, Turkey, in 2009 and 2011, respectively. He is currently a Ph.D. student in the same department. His research interests include performance modeling and analysis, bioinformatics, numerical linear algebra, and scientific computing.

W. Sandmann received his Ph.D. in Computer Science and Applied Mathematics from the University of Bonn, Germany, in 2004. From 2004 to 2009 he was an Assistant Professor of Computer Science at the University of Bamberg, and from 2009 to 2012 an Associate Professor of Mathematics and Substitute Chair of Stochastic Models in the Engineering Sciences at Clausthal University of Technology. Currently, he is a senior researcher in the Group of Modeling and Simulation at Saarland University. His research interests are in applied and computational stochastics, in particular stochastic modeling and system analysis with applications to performance evaluation of computer and communication networks, operations research, and systems biology.