Probability in the Engineering and Informational Sciences, page 1 of 31, 2011.
doi:10.1017/S0269964811000155

THE M/G/1-TYPE MARKOV CHAIN WITH RESTRICTED TRANSITIONS AND ITS APPLICATION TO QUEUES WITH BATCH ARRIVALS

JUAN F. PÉREZ AND BENNY VAN HOUDT
Performance Analysis of Telecommunication Systems (PATS)
Department of Mathematics and Computer Science
University of Antwerp – IBBT,
Middelheimlaan 1, B-2020 Antwerp, Belgium
E-mail: juanfernando.perez@ua.ac.be; benny.vanhoudt@ua.ac.be

We consider $M/G/1$-type Markov chains where a transition that decreases the value of the level triggers the phase to a small subset of the phase space. We show how this structure—referred to as restricted downward transitions—can be exploited to speed up the computation of the stationary probability vector of the chain. To this end we define a new $M/G/1$-type Markov chain with a smaller block size, the G matrix of which is used to find the original chain's G matrix. This approach is then used to analyze the $BMAP/PH/1$ queue and the $BMAP[2]/PH[2]/1$ preemptive priority queue, yielding significant reductions in computation time.

1. INTRODUCTION

An $M/G/1$-type Markov chain (MC) [19] is a two-dimensional process $\{(N_t, X_t), t \geq 0\}$ for which the *level* variable N_t takes values on $\mathbb{N}$ and the *phase* variable X_t takes values on a finite set of size m_b or m, depending on whether the level is equal to or greater than zero. In an $M/G/1$-type MC, the transition rates are level independent and the level can only decrease by one during a single transition. Therefore, the generator matrix Q of an $M/G/1$-type MC is of the form

$$Q = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \cdots \\ C_0 & A_1 & A_2 & A_3 & \cdots \\ & A_0 & A_1 & A_2 & \cdots \\ & & A_0 & A_1 & \cdots \\ 0 & & & \ddots & \ddots \end{bmatrix}, \tag{1}$$

 0269-9648/11 $25.00

where $(A_i)_{i\geq 0}$ are $m \times m$ matrixes, B_0 is an $m_b \times m_b$ matrix, C_0 is an $m \times m_b$ matrix, and $(B_i)_{i\geq 1}$ are $m_b \times m$ matrixes. All of these matrixes have nonnegative real entries, with the exception of the diagonal entries of B_0 and A_1, which are negative and such that the matrix Q has zero row sums. There are two main steps to determine the stationary probability vector of this MC. The first step is to find the matrix G—that is, the minimal nonnegative solution of the matrix equation

$$\sum_{i=0}^{\infty} A_i G^i = 0. \tag{2}$$

This equation can be solved using iterative algorithms such as (linearly convergent) functional iterations (FIs) [14,19] or the (quadratically convergent) cyclic reduction (CR) algorithm [3,4]. The second step is to compute the stationary probability vector by means of Ramaswami's formula [22], which relies on the matrix G.

Although it is possible to analyze a broad range of systems using $M/G/1$-type MCs, they suffer from the curse of dimensionality. If m becomes large, the algorithms to find G and the stationary vector might require long computation times. For instance, FI [19] requires $O(Nm^3)$ time per iteration, where N is the smallest integer such that A_i is (numerically) zero for $i > N$. However, there are cases for which the solution of Eq. (2) can be circumvented by exploiting the structure of the blocks $(A_i)_{i\geq 0}$. An example of this is when the matrix A_0 has only one nonzero column. In this case, it is possible to find the matrix G explicitly without resorting to iterative algorithms [14,23]. The main contribution of this article is to analyze and exploit the more general case for which the matrix A_0 has $1 < r \ll m$ nonzero columns. This structure implies that a transition that decreases the value of the level can only trigger the phase to a small subset of the phase space. Therefore, we refer to this class of MCs as $M/G/1$-type MCs with restricted downward transitions. We show how this structure can be exploited to reduce the total computation time to find the stationary probability vector of the chain. For this purpose, we define a new $M/G/1$-type MC by observing the original chain when the phase variable is in one of the r phases corresponding to the nonzero columns of A_0. The blocks of this new MC are of size r and therefore the solution of this chain can be carried out significantly faster. In fact, computing the blocks of the censored chain takes $O(MNrm^2)$ time, where M is the number of blocks of this chain. Then we find the matrix G of the censored chain, which requires $O(Mr^3)$ time *per iteration*. The final step to fully determine the matrix G of the original chain is to solve a linear system; this is done in a *single step* that requires $O((m-r)^3 + Nr^3 + N(m-r)^2r^2)$ time. Our approach therefore offers an advantage compared to general-purpose methods, which require $O(Nm^3)$ time *per iteration*, especially when the ratio m/r is large. In addition, after finding the matrix G, we exploit its structure to speed up the computation of the stationary probability vector.

Previous work on analyzing MCs with restricted transitions has centered mainly on quasi-birth-and-death (QBD) processes [11,18,21]. QBDs can be seen as a particular case of $M/G/1$-type MCs for which the blocks $\{A_i\}_{i\geq 3}$ and $\{B_i\}_{i\geq 2}$ are equal to

zero. To the best of our knowledge, $M/G/1$-type MCs with restricted transitions have only been treated in [5]. There, the authors exploit the referred structure to speed up the computation of G by using two specific iterative algorithms: a functional iteration and a sub-Newton method. However, in [5], an additional restriction is imposed on the chain (see Remark 2 in Section 2), which reduces the applicability of their method. For instance, the MC to model the $BMAP[2]/PH[2]/1$ preemptive priority queue is of the $M/G/1$-type and has the restricted-downward-transitions property. However, this queue cannot be analyzed with the methods introduced in [5] because of the additional restrictions. The approach introduced in this article is closely related to the one presented in [21] for QBDs. In [21] the structure of the matrix A_0 is employed to speed up the computation of the matrix G, from which the stationary probability vector can be easily obtained, as it has the so-called matrix-geometric property. Here, we not only extend [21] to exploit the structure of A_0 to compute the matrix G of an $M/G/1$-type MC, but we also make use of this structure to expedite the computation of the stationary probability vector, for which the matrix-geometric property does not hold. Additionally, whereas our focus is on $M/G/1$-type MCs with restricted downward transitions, the approach presented here can be applied *mutatis mutandis* to compute the matrix R of a $GI/M/1$-type MC with restricted upward transitions. In these MCs, an upward transition, which increases the level by at most one, can only occur if the phase variable is in a small subset of the phase space. After computing the matrix R, the stationary probability vector can be easily obtained, as it also has the matrix-geometric form.

Even though at first sight the form assumed for A_0 appears to be rather restrictive, this structure actually arises or can be induced in several, even well-studied, queuing models. Section 4 will show how the $BMAP/PH/1$ queue and the $BMAP[2]/PH[2]/1$ preemptive priority queue can be modeled as $M/G/1$-type MCs with restricted downward transitions. Moreover, these queues provide some additional structures that can be exploited in combination with our general approach to further reduce the computation time to find the stationary probability vector. These two queuing systems will be used to illustrate numerically the substantial gain in computation time that can be obtained by using the methods introduced in this article. In fact, for the $BMAP[2]/PH[2]/1$ preemptive priority queue, we found that our approach is not only faster than solving the original $M/G/1$-type MC, but it is also faster than other methods previously proposed to find the queue-length distribution of this particular queue [25]. Although we consider these two queues in detail, there are many other systems where the restricted transitions arise, including the meteor burst packet model in [5] and the $PH/MAP/1$ queue with batch services.

In the next section we will show how the computation of the matrix G can be sped up by exploiting the restricted-transitions structure. After finding G, this structure can be used to accelerate the computation of the stationary probability vector, as discussed in Section 3. The performance of our approach is illustrated through the numerical results presented in Section 4. Although this article deals with MCs in continuous time, all of the results can be easily translated and applied to discrete-time MCs.

2. ANALYZING $M/G/1$-TYPE MCS WITH RESTRICTED TRANSITIONS

In this section we focus on the computation of the matrix G—the minimal nonnegative solution of Eq. (2)—for a general $M/G/1$-type MC with restricted downward transitions. The development here follows the lines of [21], for which a similar analysis was carried out for QBD processes with restricted transitions. To start with, let us partition the phase space of the $M/G/1$-type MC (i.e., the set $\{1,\ldots,m\}$) into two subsets: $S^+ = \{1,\ldots,r\}$ and $S^- = \{r+1,\ldots,m\}$. We can partition the matrixes A_i accordingly as

$$A_i = \begin{bmatrix} A_i^{++} & A_i^{+-} \\ A_i^{-+} & A_i^{--} \end{bmatrix}, \quad i = 0,\ldots,N, \tag{3}$$

where N is the smallest integer such that $A_i = 0$ and $B_i = 0$ for $i > N$ and A_i^{++} and A_i^{--} are square matrixes of size r and $m-r$, respectively. We assume that the MC is irreducible and therefore the matrixes A_1^{++} and A_1^{--} are transient generators and their inverses exist. As stated in the previous section, our focus is on the case for which the block A_0 has only a few nonzero columns. Here, we assume, without loss of generality, that those columns are the first $r \ll m$, and therefore the matrix A_0 can be written as

$$A_0 = \begin{bmatrix} A_0^{++} & 0 \\ A_0^{-+} & 0 \end{bmatrix}. \tag{4}$$

In general, to compute the matrix G, one must rely on iterative algorithms like FI or CR [3,14,19]. However, if the matrix A_0 has the structure in Eq. (4) and if $r = 1$ (S^+ is unitary), the matrix G can be computed explicitly [14]. Here, we assume that A_0 has $r > 1$ nonzero columns. The main consequence of this structure is that the matrix G also has r nonzero columns. To see this, recall that the (i,j)th entry of the matrix G holds the probability that if the chain starts in state (k,i), the first visit to level $k-1$ occurs by visiting state $(k-1,j)$, for $k > 1$, $1 \le i,j \le m$ [14]. In addition, in an $M/G/1$-type MC, if the chain starts in level k, the first visit to level $k-1$ must be a downward transition from level k to level $k-1$, which is governed by A_0. If this matrix has the structure in Eq. (4), the first visit to level $k-1$, starting from level k, must be to a state with phase in S^+. Therefore, only the first r columns of G are different from zero and this matrix can be written as

$$G = \begin{bmatrix} G_+ & 0 \\ G_0 & 0 \end{bmatrix},$$

where G_+ (resp. G_0) is an $r \times r$ [resp. $(m-r) \times r$] matrix. In the following subsections we show how the matrixes G_+ and G_0 can be computed separately such that the total time required to compute G reduces significantly when $r \ll m$.

Remark 1: In addition to the structure in Eq. (4), we have found $M/G/1$-type MCs for which the blocks feature an additional structure (e.g., when modeling the batch queues introduced in Section 4). The most relevant feature is that the matrixes $(A_i^{--})_{i=2}^N$ can be written as the product of a scalar and a common matrix (i.e., $A_i^{--} = c_i K$ for $2 \leq i \leq N$). Another observation is that the blocks $(A_i^{+-})_{i=2}^N$ and $(A_i^{-+})_{i=2}^N$ are actually equal to zero. This additional structure will be exploited in those places where it may provide a significant gain in the computation time to find the stationary probability vector. Apart from this, the remainder of the article focuses mostly on the structure in Eq. (4), so that the approach proposed here can be used in other applications for which only this structure arises.

Remark 2: As mentioned in the Introduction, the approach in [5] also deals with $M/G/1$-type MCs with restricted transitions. However, this method imposes an additional restriction on the blocks of the MC. This restriction can be translated into our notation by imposing that

$$A_i = \begin{bmatrix} A^{++} & A^{+-} \\ A^{-+} & A^{--} \end{bmatrix} \begin{bmatrix} a_i^+ I & 0 \\ 0 & a_i^- I \end{bmatrix}, \qquad i \geq 0,$$

where $(a_i^+)_{i\geq 0}$ and $(a_i^-)_{i\geq 0}$ are nonnegative scalars and $a_0^- = 0$. The most restrictive implication of this assumption is that the block A_1 (transitions within the same level) must have the same pattern (up to a multiplication by a scalar) than the blocks $(A_i)_{i\geq 2}$ (transitions to upper levels). This means that, for instance, the $BMAP[2]/PH[2]/1$ priority queue cannot be modeled with the approach in [5], as the transitions within the same level (arrivals and service completions of high-priority customers) and those to upper levels (arrivals of low-priority customers) trigger very different transitions on the phase variable (see the Appendix).

2.1. Computing G_+

The computation of the matrix G is split in two steps: We first compute G_+ by using a censoring argument and, afterward, we obtain G_0 by solving a linear system. To compute G_+, we define a new process by observing the original $M/G/1$-type MC only when the phase variable is in S^+. Since in the original chain any downward transition takes the chain to a state with phase in S^+, the level in the new chain can decrease by at most one in a single transition. Additionally, in the original chain, there can be many upward transitions between two visits to states with phase in S^+. Therefore, the chain that results from observing the original MC when the phase variable is in S^+ is also of the $M/G/1$ type. In this case, however, the size of the blocks is equal to r, which is assumed to be significantly smaller than m. Let the $r \times r$ blocks $(\bar{A}_i)_{i\geq 0}$ characterize the behavior, away from the boundary, of the new $M/G/1$-type MC, and let $\bar{G}$ be the associated minimal nonnegative solution of Eq. (2).

As stated earlier, the (i,j)th entry of $\bar{G}$ holds the probability that, given that the chain starts in state (k,i), the first visit to level $k-1$ occurs by visiting state $(k-1,j)$, for $1 \leq i,j \leq r$. However, this is the same definition of the (i,j)th entry of G_+, since in the original chain, the first visit to level $k-1$, starting from level k, can only occur to a state with phase in S^+. Therefore, to compute G_+, we can compute the blocks $(\bar{A}_i)_{i\geq 0}$ of the censored process and then solve Eq. (2) to obtain $\bar{G}$, which is equal to G_+.

To determine the blocks $(\bar{A}_i)_{i\geq 0}$, we define the $(m-r)\times r$ matrixes $(W_l)_{l\geq -1}$. The (i,j)th entry of W_l holds the probability that, given that the chain starts in level $k>1$ and phase $i \in S^-$, the first passage to a state with phase in S^+ occurs by visiting state $(k+l,j)$, for $j \in S^+$ and $l \geq -1$. Relying on these matrixes, the blocks $(\bar{A}_i)_{i\geq 0}$ can be computed by conditioning on the first transition in the original chain. For instance, to define the matrix $\bar{A}_0$, there are two possible sets of transitions: The chain could move directly from (k,i) to $(k-1,j)$ with $i,j \in S^+$ (according to A_0^{++}) or it could first move within the same level to a state (k,l) with $l \in S^-$ (according to A_1^{+-}), and then make a number of transitions that will take the chain to a state $(k-1,j)$ with $j \in S^+$ while avoiding any state with phase in S^+ (according to W_{-1}). Therefore, the matrix $\bar{A}_0$ can be obtained as $\bar{A}_0 = A_0^{++} + A_1^{+-}W_{-1}$. Other paths from level k to $k-1$ are ruled out because they would involve either a direct transition to a state with phase in S^+ or a transition to a state with phase in S^-, but level greater than k, from which, to return to $k-1$, it would be necessary to visit states in level k with phase in S^+. By proceeding in a similar manner, we find that the blocks $(\bar{A}_i)_{i\geq 0}$ are given by

$$\bar{A}_i = \begin{cases} A_i^{++} + \sum_{j=1}^{i+1} A_j^{+-} W_{i-j}, & 0 \leq i \leq N-1 \\ A_N^{++} + \sum_{j=1}^{N} A_j^{+-} W_{N-j}, & i = N \\ \sum_{j=1}^{N} A_j^{+-} W_{i-j}, & i \geq N+1. \end{cases} \tag{5}$$

Recall that $A_i = 0$ for $i > N$.

We now turn to the computation of the matrixes $(W_l)_{l\geq -1}$. We start by noticing that to go from state (k,i) to $(k-1,j)$, with $i \in S^-$ and $j \in S^+$, while avoiding any states with phase in S^+, the chain must make a downward transition immediately after a sojourn in the set of states (k,l) with $l \in S^-$. This implies that W_{-1} is given by $W_{-1} = (-A_1^{--})^{-1}A_0^{-+}$. In a similar manner, we find that W_0 is given by $W_0 = (-A_1^{--})^{-1}(A_1^{-+} + A_2^{--}W_{-1})$. Here, the chain has the additional option of going from level k to level $k+1$ (according to A_2^{--}) and then returning to level k while avoiding states with phase in S^+ (according to W_{-1}). Following the same argument, we find

that the matrixes $(W_l)_{l\geq -1}$ can be computed recursively as

$$W_i = \begin{cases} (-A_1^{--})^{-1}\left(A_{i+1}^{-+} + \sum_{j=2}^{i+2} A_j^{--} W_{i-j+1}\right), & -1 \leq i \leq N-2 \\ (-A_1^{--})^{-1}\left(A_N^{-+} + \sum_{j=2}^{N} A_j^{--} W_{N-j}\right), & i = N-1 \\ (-A_1^{--})^{-1} \sum_{j=2}^{N} A_j^{--} W_{i-j+1}, & i \geq N \end{cases} \tag{6}$$

From Eq. (5) we observe that to compute $\bar{A}_i$, it is required to keep track of $\{W_{i-N}, \ldots, W_{i-1}\}$ for $i > N$. Therefore, we must store N matrices of size $(m-r)\times r$. After obtaining $\bar{A}_i$, the value of W_i is computed as a function of $\{W_{i-N+1}, \ldots, W_{i-1}\}$, for $i \geq N$. Then the value of W_{i-N} can be discarded, as the set $\{W_{i-N+1}, \ldots, W_i\}$ suffices to determine $\bar{A}_{i+1}$. This procedure continues until the matrix $\bar{A}_M$ is computed, where M is the smallest positive integer such that $\sum_{i=0}^{M} \bar{A}_i \mathbf{1} > -\epsilon \mathbf{1}$, where $\epsilon = 10^{-14}$ and $\mathbf{1}$ is a column vector with all its entries equal to one. For $i > N$, obtaining each block $\bar{A}_i$ requires us to compute one additional matrix W_i, which takes $O(Nr^2(m-r))$ time. After computing W_i, the value of $\bar{A}_i$ can be obtained in $O(Nr(m-r)^2)$ time. Therefore, the computation of *all* of the blocks $(\bar{A}_i)_{i=0}^{M}$ requires $O(M(Nr(m-r)^2) + Nr^2(m-r))$, where, since r is assumed to be much smaller than m, the dominant term is $O(MNrm^2)$. Once the blocks $(\bar{A}_i)_{i=0}^{M}$ are computed, we obtain G_+ by solving Eq. (2) with either FI [19] or CR [4]. As mentioned earlier, FI requires $O(Mr^3)$ time per iteration. The cost of cyclic reduction is $O(d'r^3 + r^2 d' \log d')$ per iteration, where d' is the numerical degree of a matrix power series that is updated at each iteration, starting with a power of 2 greater than or equal to M. Typically, the value of d' decreases rapidly in the positive-recurrent case [3].

2.2. Computing G_0

Once G_+ has been computed, we can obtain G_0 by solving a linear system. This can be understood by considering the partitioning in Eq. (3) and the structure in Eq. (4) to rewrite Eq. (2) as

$$-\begin{bmatrix} A_0^{++} & 0 \\ A_0^{-+} & 0 \end{bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} A_i^{++} & A_i^{+-} \\ A_i^{-+} & A_i^{--} \end{bmatrix} \begin{bmatrix} G_+ & 0 \\ G_0 & 0 \end{bmatrix}^i = \sum_{i=1}^{N} \begin{bmatrix} A_i^{++} & A_i^{+-} \\ A_i^{-+} & A_i^{--} \end{bmatrix} \begin{bmatrix} G_+^i & 0 \\ G_0 G_+^{i-1} & 0 \end{bmatrix}.$$

Now, extracting the lower-left block, we find

$$-\sum_{i=1}^{N} A_i^{--} G_0 G_+^{i-1} = A_0^{-+} + \sum_{i=1}^{N} A_i^{-+} G_+^i, \tag{7}$$

which is a general linear system of the form $\sum_{i=1}^{N} A_i X B_i = C$, where G_0 is the only unknown term. This system has $(m-r)r$ unknowns and equations; therefore, its solution by general procedures has a time complexity of $O((m-r)^3 r^3)$. Hence, this system can be solved directly if r is very small. Another possibility is to use an iterative approach as proposed in [2], although these are not guaranteed to converge to the actual solution. However, the system (7) has a special characteristic: The matrixes that postmultiply the unknown matrix G_0 are all powers of the same matrix G_+ (i.e., $B_i = G_+^{i-1}$). In Section 2.2.1 we devise a way to exploit this fact, reducing the time complexity to $O((m-r)^3 r + Nr^3 + N(m-r)^2 r^2)$. Additionally, as indicated in Section 2.2.2, there are two special cases in which the general system (7) can be reduced to a Sylvester matrix equation [10] and can therefore be solved in $O((m-r)^3)$ time with the Hessenberg–Schur algorithm proposed in [9].

2.2.1. The general case. The key to solve Eq. (7) is to apply a real Schur decomposition [10] to G_+ (i.e., to find an orthogonal matrix $U \in \mathbb{R}^{r\times r}$ such that $U'G_+U = T$, where the prime denotes the transpose operator). Recall that a matrix U is called orthogonal if $U'U = UU' = I$. The matrix $T \in \mathbb{R}^{r\times r}$ is upper quasi-triangular, meaning it is block upper triangular and the diagonal blocks are of size 1 or 2 [10]. We now postmultiply Eq. (7) by U to obtain

$$-\sum_{i=1}^{N} A_i^{--} G_0 U U' G_+^{i-1} U = \sum_{i=0}^{N} A_i^{-+} G_+^i U,$$

which since $U'G_+^j U = T^j$ for any nonnegative integer j can be rewritten as

$$-\sum_{i=1}^{N} A_i^{--} G_0 U T^{i-1} = \sum_{i=0}^{N} A_i^{-+} G_+^i U.$$

Now, let $Y = \sum_{i=0}^{N} A_i^{-+} G_+^i U$, which is a known matrix, and let $X = G_0 U$ to obtain

$$-\sum_{i=1}^{N} A_i^{--} X T^{i-1} = Y. \tag{8}$$

This system can be equivalently written in a columnwise form as

$$-\sum_{i=1}^{N} A_i^{--} \sum_{j=1}^{r} [T^{i-1}]_{j,k} X_j = Y_k \tag{9}$$

for $k = 1, \ldots, r$, where M_k and $[M]_{i,j}$ are the kth column and the (i,j)th entry of a matrix M, respectively.

Notice that in Eq. (8), the matrixes that postmultiply X are all upper quasi-triangular matrixes, and they all have the same block structure, as they are powers of T.

Therefore it is possible to iteratively compute the columns X_k, starting with X_1. Let us assume that we have already found $\{X_1, \ldots, X_{k-1}\}$ and we want to compute X_k for some $1 \leq k \leq r$. Given the upper quasi-triangular nature of T, there are two possibilities. The first is that the entry $[T]_{k+1,k}$ is zero, meaning that Eq. (9) can be rewritten as

$$-\sum_{i=1}^{N} A_i^{--}[T^{i-1}]_{k,k}X_k = Y_k + \sum_{i=1}^{N} A_i^{--} \sum_{j=1}^{k-1} [T^{i-1}]_{j,k}X_j.$$

Therefore, we can obtain the column X_k by solving a linear system of size $m - r$, which requires $O((m-r)^3)$ time. The second case is when $[T]_{k+1,k} \neq 0$, which, due to the upper quasi-triangular structure (the diagonal blocks are at most of size 2), implies that $[T]_{k+2,k+1} = 0$. Therefore, we can find the columns X_k and X_{k+1} simultaneously by solving the system

$$-\begin{bmatrix} \sum_{i=1}^{N} A_i^{--}[T^{i-1}]_{k,k} & \sum_{i=1}^{N} A_i^{--}[T^{i-1}]_{k+1,k} \\ \sum_{i=1}^{N} A_i^{--}[T^{i-1}]_{k,k+1} & \sum_{i=1}^{N} A_i^{--}[T^{i-1}]_{k+1,k+1} \end{bmatrix} \begin{bmatrix} X_k \\ X_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{Y}_k^{k-1} \\ \hat{Y}_{k+1}^{k-1} \end{bmatrix},$$

where $\hat{Y}_k^l = Y_k + \sum_{i=1}^{N} A_i^{--} \sum_{j=1}^{l} [T^{i-1}]_{j,k}X_j$ for $1 \leq l \leq k-1$ and $1 \leq k \leq r$. This is a linear system with $2(m-r)$ unknowns that requires $O((m-r)^3)$ time to be solved. As a result, we can start by finding the first (two) column(s) of X and iteratively compute the others. After computing X, G_0 is obtained from $G_0 = XU'$. Obtaining the Schur decomposition of G_+ and the powers of T requires $O(Nr^3)$. Also, setting up the right-hand side in equations (2.2.1) and (2.2.1) to find all the columns of X takes $O(N(m-r)^2r^2)$ time. As a result, the matrix G_0 can be found in $O((m-r)^3r + Nr^3 + N(m-r)^2r^2)$ time.

2.2.2. Two special cases. There are two special cases for which an $O((m-r)^3)$ algorithm can be used to compute G_0 from G_+. In the first case, it is assumed that only one of the $(A_i^{--})_{i=2}^N$ matrixes is different from zero. If A_k^{--} is the only matrix different from zero, for some $2 \leq k \leq N$, then Eq. (7) can be written as

$$G_0 - (-A_1^{--})^{-1}A_k^{--}G_0G_+^{k-1} = (-A_1^{--})^{-1}\left(A_0^{-+} + \sum_{i=1}^{N} A_i^{-+}G_+^i\right).$$

This is a Sylvester matrix equation [8,9] of the type $AXB + X = C$, which can be solved in $O((m-r)^3)$ time with the Hessenberg–Schur method proposed in [9].

The second case arises when the matrixes $(A_i^{--})_{i=2}^N$ can be written as the product of a scalar and a common matrix (i.e., $A_i^{--} = c_iK$ for $2 \leq i \leq N$). As highlighted in Remark 1, this structure arises, for instance, in the batch queues introduced in

Section 4. Using this assumption, we can rewrite Eq. (7) as

$$G_0 - (-A_1^{--})^{-1}KG_0\sum_{i=2}^{N} c_i G_+^{i-1} = (-A_1^{--})^{-1}\left(A_0^{-+} + \sum_{i=1}^{N} A_i^{-+} G_+^i\right). \quad \textbf{(10)}$$

This is also a Sylvester matrix equation of the type $AXB + X = C$, with $A = (A_1^{--})^{-1}K$ and $B = \sum_{i=2}^{N} c_i G_+^{i-1}$. It can therefore be solved with the Hessenberg–Schur method in [9].

In summary, our methodology to find G has three main steps: (1) compute the blocks $(\bar{A}_i)_{i=0}^{M}$, which takes $O(MNrm^2)$ time, (2) solve Eq. (2) to obtain G_+, which requires $O(Mr^3)$ time per iteration, if computed with FI, and (3) solve Eq. (7) to obtain G_0, which can be done in $O((m-r)^3 r + Nr^3 + N(m-r)^2 r^2)$ time. On the other hand, solving the full-block-size $M/G/1$-type MC with FI requires $O(Nm^3)$ time per iteration (let κ be the number of iterations required by FI for this chain). Therefore, the cost of step (1) is approximately Mr/m times that of a *single* iteration of FI on the full-block-size $M/G/1$-type MC. As a result, for the total time of this step to be similar to that of FI on the full-block-size $M/G/1$-type MC, the number of blocks of the censored process M would need to be on the order of $\kappa m/r$, which is a very large number, since m is already assumed to be large and FI might require a large number of iterations [14]. Additionally, the cost per iteration of step (2) is almost negligible compared to the cost per iteration of FI applied to the full-size $M/G/1$-type MC. Finally, one iteration of FI on the full-size $M/G/1$-type MC has a time complexity comparable to the last step of our method. All in all, we expect that our method should provide a significant reduction in the time to compute G, compared to FI, especially when the ratio m/r is large and the number of blocks M is small compared to $\kappa m/r$. A similar conclusion can be drawn if CR is used to solve the full-block-size $M/G/1$-type MC, although in this case the analysis is more involved, since the time complexity per iteration is $O(d'm^3 + m^2 d' \log d')$, where d' varies at every iteration, as explained earlier [3]. In Section 4 we will show numerically how the use of our methodology leads to significant reductions in the time to compute G, compared to either CR or FI on the full-block-size $M/G/1$-type MC.

3. COMPUTING THE STATIONARY PROBABILITY VECTOR

We have shown how the matrix G can be obtained by first computing G_+ from a censored process and then solving a linear system to determine G_0. Once G has been obtained, the stationary probability vector of the original $M/G/1$-type MC can be computed by means of Ramaswami's formula [22] as follows. Let the matrixes $(\tilde{A}_i)_{i\geq 1}$ and $(\tilde{B}_i)_{i\geq 0}$ be defined as

$$\tilde{A}_i = \sum_{j=i}^{\infty} A_j G^{j-i}, \quad i \geq 1, \quad \tilde{B}_i = \sum_{j=i}^{\infty} B_j G^{j-i}, \quad i \geq 1, \quad \tilde{B}_0 = B_0 + \tilde{B}_1(-\tilde{A}_1)^{-1}C_0. \quad \textbf{(11)}$$

Let π be the stationary probability vector of the MC with generator Q (i.e., the vector such that $\pi Q = 0$ and $\pi \mathbf{1} = 1$). This vector can also be partitioned according to the levels as $\pi = [\pi_0, \pi_1, \dots]$. Then Ramaswami's formula states that the vectors π_i can be found recursively as

$$\pi_i = \left(\pi_0 \tilde{B}_i + \sum_{j=1}^{i-1} \pi_j \tilde{A}_{i-j+1} \right) (-\tilde{A}_1)^{-1}, \quad i \geq 1, \tag{12}$$

and π_0 is such that

$$\pi_0 \tilde{B}_0 = 0, \quad \pi_0 \kappa = 1, \tag{13}$$

where $\kappa = \mathbf{1} + \left(\sum_{j=1}^{\infty} \tilde{B}_j \right) \left(- \sum_{j=1}^{\infty} \tilde{A}_j \right)^{-1} \mathbf{1}$. However, one might wonder whether the structure of the matrix A_0 can be exploited to make the computation of π faster. This is the purpose of this section, in which three main approaches to compute π are described. The first method makes use of the structure of A_0 to accelerate the computation of the matrixes $(\tilde{A}_i)_{i=1}^N$ and $(\tilde{B}_i)_{i=1}^N$. The other two methods compute the vector π from the stationary probability vector of the censored process. Whereas the first two methods apply in general to any $M/G/1$-type MC with restricted transitions, the last one also exploits the additional structure discussed in Remark 1.

3.1. Computing π from the Original Process

In this subsection we consider the problem of exploiting the structure of A_0 to compute the matrixes $(\tilde{A}_i)_{i=1}^N$ and $(\tilde{B}_i)_{i=1}^N$. To compute these matrixes, one typically starts with $\tilde{A}_N = A_N$ and $\tilde{B}_N = B_N$ and computes iteratively $\tilde{A}_i = A_i + \tilde{A}_{i+1} G$ and $\tilde{B}_i = B_i + \tilde{B}_{i+1} G$ for $i = N-1, \dots, 1$. However, if we rewrite Eq. (11) in block form according to the partition in (3), we find that

$$\begin{aligned} \tilde{A}_i = \begin{bmatrix} \tilde{A}_i^{++} & \tilde{A}_i^{+-} \\ \tilde{A}_i^{-+} & \tilde{A}_i^{--} \end{bmatrix} &= \begin{bmatrix} A_i^{++} & A_i^{+-} \\ A_i^{-+} & A_i^{--} \end{bmatrix} + \sum_{j=i+1}^{N} \begin{bmatrix} A_j^{++} & A_j^{+-} \\ A_j^{-+} & A_j^{--} \end{bmatrix} \begin{bmatrix} G_+ & 0 \\ G_0 & 0 \end{bmatrix}^{j-i} \\ &= \begin{bmatrix} A_i^{++} & A_i^{+-} \\ A_i^{-+} & A_i^{--} \end{bmatrix} + \sum_{j=i+1}^{N} \begin{bmatrix} A_j^{++} G_+^{j-i} + A_j^{+-} G_0 G_+^{j-i-1} & 0 \\ A_j^{-+} G_+^{j-i} + A_j^{--} G_0 G_+^{j-i-1} & 0 \end{bmatrix}, \\ &\quad \times 1 \leq i \leq N. \end{aligned}$$

Therefore, the last $m - r$ columns of the matrix $\tilde{A}_i$ are identical to those of the matrix A_i, for $1 \leq i \leq N$. To compute the first r columns of these matrixes we simply start with $\tilde{A}_N^{++} = A_N^{++}$ and $\tilde{A}_N^{-+} = A_N^{-+}$ and then iteratively compute

$$\begin{aligned} \tilde{A}_i^{++} &= A_i^{++} + \tilde{A}_{i+1}^{++} G_+ + A_{i+1}^{+-} G_0, \\ \tilde{A}_i^{-+} &= A_i^{-+} + \tilde{A}_{i+1}^{-+} G_+ + A_{i+1}^{--} G_0 \end{aligned}$$

for $i = N-1, \ldots, 1$. A similar observation can be made for the matrixes $(\tilde{B}_i)_{i=1}^N$, as these, as well as $(B_i)_{i=1}^N$, can be partitioned in two blocks depending on whether the transitions from level 0 take the chain to a state with phase in S^+ or S^-. Therefore, we can write

$$\tilde{B}_i = \begin{bmatrix} \tilde{B}_i^+ & \tilde{B}_i^- \end{bmatrix} = \begin{bmatrix} B_i^+ & B_i^- \end{bmatrix} + \sum_{j=i+1}^{N} \begin{bmatrix} B_j^+ & B_j^- \end{bmatrix} \begin{bmatrix} G_+ & 0 \\ G_0 & 0 \end{bmatrix}^{j-i}.$$

From this equation we find that $\tilde{B}_i^- = B_i^-$ for $1 \le i \le N$. Additionally, $\tilde{B}_i^+$ can be obtained by starting with $\tilde{B}_N^+ = B_N^+$ and sequentially computing $\tilde{B}_i^+ = B_i^+ + \tilde{B}_{i+1}^+ G_+ + B_{i+1}^- G_0$ for $i = N-1, \ldots, 1$. Once we have obtained these matrixes, we can compute the vector π directly using Ramaswami's formula as in Eq. (12). One could also rely on the fast-Fourier-transform-based implementation of Ramaswami's formula proposed by Meini [16]. With either approach, we are computing π by simply using the structure of G to speed up the computation of the matrixes $(\tilde{A}_i)_{i=1}^N$ and $(\tilde{B}_i)_{i=1}^N$. In the next subsection we consider a different approach in which we make use of the censored process to obtain the stationary probability vector π.

3.2. Computing π from the Censored Process

In this subsection we show how the censored process can be used to reduce the time required to compute π by separately computing $(\pi_i^+)_{i\ge1}$ and $(\pi_i^-)_{i\ge1}$, where these vectors arise by partitioning π_i according to S^+ and S^- (i.e., $\pi_i = [\pi_i^+ \, \pi_i^-]$ for $i \ge 1$). As stated earlier, we obtain the matrix G_+ from a censored process of the $M/G/1$ type whose G matrix is actually equal to G_+. To do so, we described the censored process by means of the blocks $(\bar{A}_i)_{i=0}^M$, which only consider the behavior of the process away from the boundary. We now complete the description of this process by adding a boundary level, which is identical to the boundary level of the original process. This means that the new process is built by observing the original process only when it is in the subset of states $\bigcup_{i\ge1}\{(i,j), j \in S^+\} \cup \{(0,j), 1 \le j \le m_b\}$. The boundary behavior of this process is described by the $m_b \times m_b$ matrix $\bar{B}_0$ (transitions within level 0), the $m_b \times r$ matrixes $(\bar{B}_i)_{i=1}^{M_0}$ (transitions from level 0 to level $i > 1$), and the $r \times m_b$ matrix $\bar{C}_0$ (transitions from level 1 to level 0). These blocks, together with the matrices $(\bar{A}_i)_{i=0}^M$, completely characterize the censored process, whose generator $\bar{Q}$ is built from these blocks in a manner similar to the generator Q of the original process in Eq. (1).

To define the blocks $(\bar{B}_i)_{i=0}^{M_0}$ we rely on the matrixes $(W_i)_{i\ge-1}$ as defined in Eq. (6). As stated in the previous subsection, we partition the original blocks $B_i = [B_i^+ \, B_i^-]$, where B_i^+ (resp. B_i^-) holds the transition rates that, from level 0, trigger the chain into level i, and phase in S^+ (resp. S^-). Similarly, the matrix C_0 can be partitioned into C_0^+ and C_0^-, which hold the transition rates to level 0 from states in level 1 and phase in S^+ and S^-, respectively. Using these definitions, we find that $\bar{B}_0 = B_0 + B_1^-(-A_1^{--})^{-1}C_0^-$, which considers the two alternative paths for a transition starting and ending in level 0 in the new process. This might occur directly according to B_0 or by a transition to level 1 and a backward transition, avoiding states with phase in S^+.

Carrying out a similar analysis, we find that the remaining boundary blocks are given by

$$\bar{B}_i = \begin{cases} B_i^+ + \sum_{j=1}^{i+1} B_j^- W_{i-j}, & 1 \le i \le N-1 \\ B_N^+ + \sum_{j=1}^{N} B_j^- W_{N-j}, & i = N \\ \sum_{j=1}^{N} B_j^- W_{i-j}, & i \ge N+1. \end{cases} \tag{14}$$

These matrixes are computed sequentially from $i = 0$ to M_0, where M_0 is the smallest positive integer such that $\sum_{i=0}^{M_0} \bar{B}_i \mathbf{1} > -\epsilon \mathbf{1}$. Finally, we observe that the transitions from level 1 to level 0 in the censored process are governed by $\bar{C}_0 = C_0^+ + A_1^{+-}(-A_1^{--})^{-1} C_0^-$.

With these definitions, we are ready to compute, using Ramaswami's formula, the stationary probability vector of the censored process $\bar{\pi}$, which can be partitioned in blocks as $\bar{\pi} = [\bar{\pi}_0, \bar{\pi}_1, \bar{\pi}_2, \dots]$. The importance of this vector is that it is proportional to the stationary vector π of the original process [13,14] (i.e., $\bar{\pi}_0 \propto \pi_0$ and $\bar{\pi}_i \propto \pi_i^+$ for $i \ge 1$). Actually, we would like to normalize the vector $\bar{\pi}$ by a constant such that $\bar{\pi}_0 = \pi_0$ and $\bar{\pi}_i = \pi_i^+$, for $i \ge 1$. This can be accomplished by computing π_0 as in Eq. (13) and assigning $\bar{\pi}_0 = \pi_0$. This forces $\bar{\pi}_0$ to be normalized by the appropriate constant. The terms $(\bar{\pi}_i)_{i \ge 1}$ can then be obtained, using Ramaswami's formula, after computing the matrixes $(\tilde{\bar{A}}_i)_{i \ge 1}$ and $(\tilde{\bar{B}}_i)_{i \ge 1}$ for the censored process, as in Eq. (11). Notice that to obtain π_0 from Eq. (13), we first need to compute the matrixes $(\tilde{A}_i)_{i=1}^N$ and $(\tilde{B}_i)_{i=1}^N$.

Up to this point, we have computed π_0 and $(\pi_i^+)_{i \ge 1}$, and we can use these to find the vectors $(\pi_i^-)_{i \ge 1}$ in order to completely determine π. This can be done by rewriting Eq. (12) in block form as

$$\begin{bmatrix} \pi_i^+ & \pi_i^- \end{bmatrix} \begin{bmatrix} -\tilde{A}_1^{++} & -\tilde{A}_1^{+-} \\ -\tilde{A}_1^{-+} & -\tilde{A}_1^{--} \end{bmatrix} = \pi_0 \begin{bmatrix} \tilde{B}_i^+ & \tilde{B}_i^- \end{bmatrix} + \sum_{j=1}^{i-1} \begin{bmatrix} \pi_j^+ & \pi_j^- \end{bmatrix} \begin{bmatrix} \tilde{A}_{i-j+1}^{++} & \tilde{A}_{i-j+1}^{+-} \\ \tilde{A}_{i-j+1}^{-+} & \tilde{A}_{i-j+1}^{--} \end{bmatrix},$$

$\times\, i \ge 1.$

From this equation we can express π_i^- as

$$\pi_i^- = \left(\pi_0 B_i^- + \pi_i^+ A_1^{+-} + \sum_{j=1}^{i-1} \left(\pi_j^+ A_{i-j+1}^{+-} + \pi_j^- A_{i-j+1}^{--} \right) \right) (-A_1^{--})^{-1}, \quad i \ge 1. \tag{15}$$

Notice that this equation makes use of the original blocks $\{A_i\}_{i \ge 1}$ and $\{B_i\}_{i \ge 1}$ instead of the blocks $\{\tilde{A}_i\}_{i \ge 1}$ and $\{\tilde{B}_i\}_{i \ge 1}$ because their last $m - r$ columns are identical, as shown in Section 3.1. Using this equation, we can compute π_i^- in terms of π_0, $(\pi_j^+)_{j=1}^i$ and

$(\pi_j^-)_{j=1}^{i-1}$ for $i \geq 1$, thus concluding the computation of π. This approach is summarized in Algorithm 1. One could consider a slight modification to this algorithm by replacing the use of Ramaswami's formula in step 7 with the fast Ramaswami's formula (FRF) proposed in [16], which is based on the fast Fourier transform. This might provide an important computational gain when the number of blocks $(\bar{B}_i)_{i\geq 1}$ and $(\bar{A}_i)_{i\geq 1}$ is large. This approach will also be considered in the numerical experiments in Section 4.

Algorithm 1 Computing π from the censored process

Input: Matrixes $(\bar{A}_i)_{i\geq 1}$ and G

1: Compute the matrixes $(\tilde{A}_i)_{i\geq 1}$ and $(\tilde{B}_i)_{i\geq 0}$ using the iterative scheme in Section 3.1.
2: Find π_0 solving Eq. (13) and set $\bar{\pi}_0 = \pi_0$.
3: Compute the matrixes $(\bar{B}_i)_{i\geq 0}$ according to Eq. (14).
4: Compute the matrixes $(\tilde{\bar{A}}_i)_{i\geq 1}$ and $(\tilde{\bar{B}}_i)_{i\geq 0}$ as in Eq. (11).
5: Set $\sigma = \bar{\pi}_0\mathbf{1}$ and $i = 1$.
6: **while** $\sigma < 1 - \epsilon$ **do**
7: Compute $\bar{\pi}_i$ using (fast) Ramaswami's formula.
8: Set $\pi_i^+ = \bar{\pi}_i$ and compute π_i^- as in Eq. (15).
9: Update $\sigma = \sigma + \pi_i^+\mathbf{1} + \pi_i^-\mathbf{1}$, and $i = i + 1$.
10: **end while**

3.3. Computing π from the Censored Process: A Special Case

In this subsection we consider how to exploit the additional structure described in Remark 1, which arises in the MCs that describe the batch queues to be introduced in Section 4. Recall that this additional structure implies the following. First, the matrixes $(A_i^{--})_{i=2}^N$ can be written as the product of a scalar and a common matrix (i.e., $A_i^{--} = c_i K$ for $2 \leq i \leq N$); second, the blocks $(A_i^{+-})_{i=2}^N$ and $(A_i^{-+})_{i=2}^N$ are equal to zero. We start by noting that due to the additional structure, Eq. (15) can be written as

$$\pi_i^- = \left[\pi_0 B_i^- + \pi_i^+ A_1^{+-} + \left(\sum_{j=1}^{i-1} c_{i-j+1} \pi_j^- \right) K \right] (-A_1^{--})^{-1}, \quad i \geq 1. \tag{16}$$

The advantage of this expression is that each term of the sum is just a vector multiplied by a scalar, whereas in the general case, each term in the sum requires a vector–matrix multiplication. Therefore, one can simply modify step 8 in Algorithm 1 replacing the use of Eq. (15) by Eq. (16). However there is an additional fact that can be exploited if we make use of the FRF to compute $(\pi_i^+)_{i\geq 1}$. The FRF, as introduced in [16], computes not one but many vectors π_i^+ at a time. Specifically, in each iteration, the FRF computes $\bar{M}$ terms, where $\bar{M}$ is a power of 2 such that $\bar{M} \geq M$ and $\bar{M} \geq M_0$. Therefore, after the first iteration, we already have the terms $(\pi_i^+)_{i=1}^{\bar{M}}$ and it suffices

to compute the corresponding terms $(\pi_i^-)_{i=1}^{\bar{M}}$. If the MC has the structure described in Remark 1, we could do this by using Eq. (16) directly. However, we can also use the following observation. Let us assume that we have computed the first $(\pi_i^-)_{i=1}^{N}$ by means of (16). For $i \geq N+1$, we can write

$$\pi_i^- = \left(\pi_i^+ A_1^{+-} + \sum_{j=i-N+1}^{i-1} \left(c_{i-j+1} \pi_j^- \right) K \right) (-A_1^{--})^{-1},$$

as $B_i = 0$ and $A_i = 0$ for $i > N$. Now, if we focus on the computation of $(\pi_i^-)_{i=N+1}^{2N-1}$, we can write the previous expression as

$$\pi_i^- = \pi_i^+ \hat{A}_1^{+-} + \sum_{j=i-N+1}^{N} \left(c_{i-j+1} \pi_j^- \right) \hat{K} + \sum_{j=N+1}^{i-1} \left(c_{i-j+1} \pi_j^- \right) \hat{K},$$
$$\times N+1 \leq i \leq 2N-1, \tag{17}$$

where $\hat{A}_1^{+-} = A_1^{+-}(-A_1^{--})^{-1}$ and $\hat{K} = K(-A_1^{--})^{-1}$. In this equation, the first sum on the right-hand side only involves the terms $(\pi_i^-)_{i=2}^{N}$, whereas the second sum and the left-hand side involve the terms $(\pi_i^-)_{i=N+1}^{2N-1}$. Therefore, we can write this expression as a system of equations. To do so, let $\bar{N} = N-1$ and let $\hat{\pi}_i^+$ and $\hat{\pi}_i^-$ be

$$\hat{\pi}_i^+ = \begin{bmatrix} \pi^+_{1+(i-2)\bar{N}+1} \\ \pi^+_{1+(i-2)\bar{N}+2} \\ \vdots \\ \pi^+_{1+(i-1)\bar{N}} \end{bmatrix} \quad \text{and} \quad \hat{\pi}_i^- = \begin{bmatrix} \pi^-_{1+(i-2)\bar{N}+1} \\ \pi^-_{1+(i-2)\bar{N}+2} \\ \vdots \\ \pi^-_{1+(i-1)\bar{N}} \end{bmatrix}, \quad i \geq 2.$$

Therefore, we can write the set of $\bar{N}$ equations in (17) in matrix form as

$$\hat{\pi}_3^- = \hat{\pi}_3^+ \hat{A}_1^{+-} + R_1 \hat{\pi}_2^- \hat{K} + R_2 \hat{\pi}_3^- \hat{K},$$

where the $\bar{N} \times \bar{N}$ matrixes R_1 and R_2 are given by

$$R_1 = \begin{bmatrix} c_N & c_{N-1} & \dots & c_2 \\ 0 & c_N & \dots & c_3 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_N \end{bmatrix} \quad \text{and} \quad R_2 = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ c_2 & 0 & \dots & 0 & 0 \\ c_3 & c_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-1} & c_{N-2} & \dots & c_2 & 0 \end{bmatrix}.$$

In general, we can find the vector $\hat{\pi}_i^-$ in terms of $\hat{\pi}_{i-1}^-$ and $\hat{\pi}_i^+$ by solving the system

$$\hat{\pi}_i^- - R_2 \hat{\pi}_i^- \hat{K} = \hat{\pi}_i^+ \hat{A}_1^{+-} + R_1 \hat{\pi}_{i-1}^- \hat{K}, \quad i \geq 3. \tag{18}$$

This is also a Sylvester matrix equation of the type $AXB + X = C$, which can be solved using the Hessenberg–Schur method mentioned earlier [9]. Among the two

decompositions that must be carried out to use this method, the Schur decomposition is the most expensive, requiring $10b^3$ operations for a square matrix of size b. On the other hand, the Hessenberg decomposition only requires $\frac{5}{3}b^3$ operations [9]. In this case, the square matrix $A = -R_2$ is of size $\bar{N}$ and the square matrix $B = \hat{K}$ is of size $m - r$. Because we expect a large value for $m - r$, it is actually better to apply this method to the transposed system (i.e., to $B'X'A' + X' = C'$). In this manner we apply the Hessenberg decomposition to the larger matrix (i.e., to $B' = -\hat{K}'$). Additionally, since the matrix $A' = -R_2'$ is already in real Schur form [10] (it is upper triangular), there is no need to use the QR algorithm to obtain the Schur decomposition. Therefore, after finding the Hessenberg decomposition of $B' = -\hat{K}'$, we can find the solution to the transposed Sylvester matrix equation by solving $\bar{N}$ Hessenberg systems of size $m - r$, each one requiring $O((m - r)^2)$ time. Moreover, to find all the vectors $(\hat{\pi}_i^-)_{i\geq 3}$, we need to solve a series of equations as in Eq. (18), but the matrixes R_2 and $\hat{K}$ remain unchanged for $i \geq 3$. Therefore, to solve these equations, we only need to apply the Hessenberg decomposition once and update the right-hand side of Eq. (18). This concludes the computation of π. The next section illustrates the performance of the various methods introduced in this article by means of two queuing examples.

4. EXAMPLES AND NUMERICAL EXPERIMENTS

The purpose of this section is to show how two different queuing systems can be modeled as $M/G/1$-type MCs with restricted downward transitions. The first system is a $BMAP/PH/1$ queue, for which the block A_0 is forced to have a few nonzero columns by using a slightly larger representation of the service-time distribution. The second example is a $BMAP[2]/PH[2]/1$ preemptive priority queue with two types of customers where, by adequately ordering the state space, we obtain the desired structure for the block A_0. These two examples will be used to illustrate the computational gains obtained with our approach. In addition, when the batch sizes are independent and identically distributed (i.i.d.) random variables independent of the arrival process, the MCs used to model these queues show the additional structure described in Remark 1. Therefore, these examples will also be useful to demonstrate the gains obtained by exploiting this additional structure.

We focus on the total time required to compute the vector π, which includes the computation of G. As a benchmark, we compute the matrix G of the original $M/G/1$-type MC using the U-based functional iteration [19] and cyclic reduction [4], for which we use the labels **FI** and **CR**, respectively. To compute π, after finding G, we consider Ramaswami's formula (RF) and its fast implementation (FRF). Therefore, for the total time to compute π, we have four benchmarks: **FI-R**, **FI-F**, **CR-R**, and **CR-F**. For the $BMAP[2]/PH[2]/1$ preemptive priority queue, we also compare with the approach introduced in [25], here labeled **INF**, where both the high- and the low-priority buffers are assumed to be infinite. The methods introduced in this article are labeled **O**, **C**, and **C***. In all of the methods, the censored process is used to exploit the structure of the matrix A_0 to compute G_+. In the methods labeled **O** and **C**, we

only consider this structure, whereas in **C***, we also take into account the additional structure described in Remark 1. Therefore, in methods **O** and **C**, the matrix G_0 is found by solving the linear system (7) for the general case. In method **O**, the vector π is found from the original process as described in Section 3.1. This can be done either with RF or FRF; therefore, we have both **O-R** and **O-F**. On the other hand, for method **C**, the vector π is computed by means of the censored process, as discussed in Section 3.2. In this case, we also have both **C-R** and **C-F**. In the method labeled **C***, we consider the additional structure in Remark 1, and therefore the matrix G_0 is found by solving Eq. (10). Additionally, we can compute π relying on the censored process and applying Eq. (16). As this can be done by using either RF or FRF to compute $(\pi_i^+)_{i\geq 1}$, we have the two alternatives **C*-R** and **C*-F**. Moreover, if FRF is used to compute $(\pi_i^+)_{i\geq 1}$, we can also make use of Eq. (18) to obtain $(\pi_i^-)_{i\geq 1}$. This approach will be labeled **C*-F$_2$**.

Since both queuing systems have similar arrival processes and service-time distributions, we devote the next subsection to describe these. Subsequently, we introduce both models and show how the computation of π can be substantially reduced by exploiting the restricted-transitions structure.

4.1. The Arrival Process and Service-Time Distribution

In Kendall's notation, the two queues under analysis are the $BMAP/PH/1$ queue and the $BMAP[2]/PH[2]/1$ preemptive priority queue. We first define the arrival processes (BMAP and BMAP[2]) and then move on to specify the service-time distributions (PH and PH[2]). The arrival process at both queues can be described by means of a Markovian arrival process (MAP) [15,19]. A MAP(m_a, D_0, D_1) is a point process driven by an underlying continuous-time MC (CTMC) with $m_a \times m_a$ generator matrix $D = D_0 + D_1$. The (i,j)th entry of the matrix D_1 holds the rate at which, when the underlying chain is in state i, a customer arrives and the chain makes a transition to state j for $1 \leq i,j \leq m_a$. The off-diagonal entries of the matrix D_0 hold the rates related to transitions without arrivals, and its (negative) diagonal entries are such that $D\mathbf{1} = \mathbf{0}$, where $\mathbf{0}$ is a vector with all its entries equal to zero. Therefore, this process results from a CTMC whose transitions are marked [12] with either of two labels: the label "0" for the transitions that generate no arrivals and the label "1" for the transitions that trigger an arrival. The matrixes D_0 and D_1 hold the rates associated with the transitions labeled "0" and "1," respectively. These markings can be generalized to include other information about the arriving customers. As we want to model queues with batch arrivals, the markings can be defined to include the number of customers that arrive in a single transition. In this case, the arrival process is characterized by the $m_a \times m_a$ matrixes $\{D_0, D_1, \ldots D_{\bar{L}}\}$, where $\bar{L}$ is the maximum batch size. The matrix D_j holds the rates related to transitions of the underlying chain that trigger a batch arrival of size j for $1 \leq j \leq \bar{L}$. Since in this case the markings are related to the batch size only, this process is called a batch Markovian arrival process (BMAP) [15]. This is the arrival process fed to the $BMAP/PH/1$ queue. In general, this process is able to model correlation between the interarrival times (IATs) and the batch size distribution.

However, when modeling this queue as an $M/G/1$-type MC with restricted transitions, there is a computational gain that can be exploited by assuming that the the batch sizes are i.i.d. random variables and are independent of the IATs (see Remark 1 and Section 2.2). To build a BMAP with these characteristics, let r_j be the probability that a batch is of size j and let the batch IATs be described by MAP(m_a, D_0, D_+). Then the batch arrival process is a BMAP characterized by the $m_a \times m_a$ matrixes $\{D_0, D_1, \ldots D_{\bar{L}}\}$, where $D_j = r_j D_+$ for $1 \leq j \leq \bar{L}$.

The arrivals at the preemptive priority queue also occur in batches, but these are of two types, each one associated with a different priority level. Therefore, the markings must include not only the size of the batch but also the type of the customers in that batch. We assume that each batch is made of customers of only one type, and the maximum size of a batch of high- (resp. low-) priority customers is $\bar{L}_1$ (resp. $\bar{L}_2$). Hence, the arrival process can be modeled as a BMAP[2] characterized by the matrixes $\{D_0, D_1^{j_1}, D_2^{j_2}, 1 \leq j_1 \leq \bar{L}_1, 1 \leq j_2 \leq \bar{L}_2\}$, where the notation BMAP[2] reflects the fact that the markings describe both the batch size and the type of the customers in the batch (two possible types). In this case, the matrix D_1^j (resp. D_2^j) holds the transition rates associated with the arrival of a batch of j high- (resp. low-) priority customers for $1 \leq j \leq \bar{L}_1$ (resp. $1 \leq j \leq \bar{L}_2$). As earlier, we can define a version of this process for which the batch sizes are i.i.d. random variables and are independent of the IATs. For this purpose, let p_i be the probability that a batch of high-priority customers is of size i for $1 \leq i \leq \bar{L}_1$. Accordingly, let q_i be the probability that a batch of low-priority customers is of size i for $1 \leq i \leq \bar{L}_2$. Additionally, let the batch IATs be described by a MAP with markings for each customer type—that is, characterized by the $m_a \times m_a$ matrixes $\{D_0, D_1, D_2\}$, where D_1 (resp. D_2) holds the transition rates at which the underlying chain triggers the arrival of a batch of high- (resp. low-) priority customers. Combining the marked MAP and the batch size distributions, we obtain a BMAP[2] characterized by the $m_a \times m_a$ matrixes $\{D_0, D_1^{j_1}, D_2^{j_2}, 1 \leq j_1 \leq \bar{L}_1, 1 \leq j_2 \leq \bar{L}_2\}$, where $D_1^{j_1} = p_{j_1} D_1$ and $D_2^{j_2} = q_{j_2} D_2$ for $1 \leq j_1 \leq \bar{L}_1$ and $1 \leq j_2 \leq \bar{L}_2$. Notice that although the batch sizes are i.i.d. random variables, the IATs of both types of batch arrivals can be correlated. As with the previous queue, the independence between the IATs and the i.i.d. batch sizes can be exploited when the $BMAP[2]/PH[2]/1$ preemptive priority queue is modeled as an $M/G/1$-type MC with restricted downward transitions.

In the $BMAP/PH/1$ queue, the service times follow a phase-type (PH) distribution [14,18] with parameters (m_s, α, T). A PH distribution describes the time until absorption in a CTMC with m_s transient states and one absorbing state. The initial probability distribution on the transient states is given by the $1 \times m_s$ vector α, and the transitions between the transient states are ruled by the $m_s \times m_s$ transient generator T. The (i, j)th entry of this matrix holds the nonnegative rate at which a transition from state i to state j occurs, with $i \neq j$. The diagonal entries are negative and such that the matrix T has a nonpositive row sums, with at least one row having a strictly negative row sum. The absorption rate at state i is therefore given by the ith entry of the vector $t = -T\mathbf{1}$ for $1 \leq i \leq m_s$. The cumulative distribution function of the time until absorption is given by $F(x) = 1 - \alpha \exp(Tx)\mathbf{1}$ for $x \geq 0$. Similarly, the service times

in the preemptive priority queue for both the high- and the low-priority customers are PH-distributed. For the high- (resp. low-) priority customers, the parameters of the service-time distribution are (m_1, α, T) (resp. (m_2, β, S)). The fact that these parameters might differ for each customer type is made explicit by writing PH[2]. In the remainder of the article, the states of the MCs that underlie the arrival process and the service-time distribution are also referred to as phases.

4.2. The *BMAP/PH/1* Queue

Our first example is a single-server queue for which the customers arrive according to a BMAP characterized by the $m_a \times m_a$ matrixes $\{D_0, D_1, \ldots D_{\bar{L}}\}$, with $\bar{L}$ the maximum batch size. The service times follow a PH distribution with parameters (m_s, α, T). The $BMAP/PH/1$ queue can be modeled as an $M/G/1$-type MC by choosing the number of customers in the queue to be the level. This selection assures that the level decreases by at most one in a single transition, as only one service completion can occur at a time. On the other hand, the level can increase by up to $\bar{L}$, as it is triggered by a batch arrival. Let $N(t)$ be the number of customers in the queue at time t, let $S(t)$ be the phase of the service process at time t if there is a customer in service, and let $J(t)$ be the phase of the arrival process at time t. Then the tuple $\{N(t), S(t), J(t), t \geq 0\}$ forms a CTMC that fully describes the state of the $BMAP/PH/1$ queue. The state space of this MC can be described as follows: Level 0 is the set of states $\Omega_0 = \{(0, j), 1 \leq j \leq m_a\}$, where in state $(0, j)$, the queue is empty and the arrival process is in phase j; level $k \geq 1$ is the set of states $\Omega_k = \{(k, i, j), 1 \leq i \leq m_s, 1 \leq j \leq m_a\}$, where in state (k, i, j) there are k customers in the queue, the service in progress is in phase i, and the arrival process is in phase j. The complete state space is therefore given by $\Omega = \bigcup_{k \geq 0} \Omega_k$. Since this MC is of the $M/G/1$ type, its rate matrix has the structure in Eq. (1) with blocks given by

$$A_0 = t\alpha \otimes I_{m_a}, \quad A_1 = T \oplus D_0, \quad A_{j+1} = I_{m_s} \otimes D_j, \quad j = 1, \ldots, \bar{L}, \qquad \textbf{(19)}$$

where $t = -T\mathbf{1}$ and I_n is the identity matrix of size n. Here $\otimes$ and $\oplus$ stand for Kronecker product and sum [10], respectively. From this definition it is clear that the block size is $m = m_s m_a$ and that the number of nonzero columns in A_0 depends on the number of nonzero elements in the vector α. In fact, if α has only one nonzero element, then A_0 has only $r = m_a$ nonzero columns (i.e., the block size is m_s times larger than the number of nonzero columns in A_0). This is the case if the service times are described by an acyclic PH distribution [6]. This class of distributions (which includes the Erlang and the hyperexponential distributions as special cases) has a canonical form introduced in [6], for which all the mass of the initial probability vector is concentrated in the first phase. Therefore, in this case, the vector α has only one nonzero entry and the matrix A_0 has m_a nonzero columns. In general, the vector α can have any number of nonzero entries, but we can always find a representation of size $m_s + 1$ such that the initial probability vector has only one nonzero entry. In fact, it is not difficult to show [20] that any continuous PH distribution with representation

(m_s, α, T) also has a representation $(m_s + 1, e_1, \bar{T})$, where e_1 and $\bar{T}$ are given by

$$e_1 = \begin{bmatrix} 1 & \mathbf{0}_{m_s} \end{bmatrix} \quad \text{and} \quad \bar{T} = \begin{bmatrix} -\theta & \theta\alpha P \\ 0 & T \end{bmatrix},$$

where θ is the diagonal entry of T of largest absolute value (i.e., $\theta = \max\{|T_{ii}|, 1 \leq i \leq m_s\}$) and P is the uniformized version of the subgenerator matrix T (i.e., $P = (1/\theta)T + I$). Using this result, we can replace α and T by e_1 and $\bar{T}$, respectively, in Eq. (19). As a consequence, the block A_0 has only m_a nonzero columns and the new block size is $(m_s + 1)m_a$. If m_s is large, the structure of the block A_0 can be exploited to speed up the computation of the matrix G and the stationary probability vector of the chain.

To illustrate the behavior of our methods, we choose the following PH distribution for the service time: We assume that the total service time is made up of a random number of i.i.d. elementary operations; each of these operations is described by a continuous PH distribution with parameters $(\bar{m}_s, \gamma, C)$; the number of elementary operations that make up the total service time is described by a discrete PH distribution with parameters (n_s, β, S); and as a result [14], the total service time follows a continuous PH distribution with parameters (m_s, α, T) given by $m_s = n_s\bar{m}_s$, $\alpha = \gamma \otimes \beta$, and $T = C \otimes I + c\gamma \otimes S$, where $c = -Ce$. Notice that we can easily alter the size of this representation by changing n_s, the size of the discrete PH distribution. Additionally, this representation is not acyclic, but we can obtain a representation of size $m_s + 1$ for which the initial probability vector has only one nonzero entry, as mentioned earlier.

The time to complete an elementary operation is assumed to have mean one and squared coefficient of variation (SCV) equal to 2. The moments are matched with a PH distribution of size $\bar{m}_s = 2$ using the method in [24]. Hence, the service-time distribution has a representation of size $2n_s$, which we convert into a representation of size $2n_s + 1$ to induce the block A_0 to have m_a nonzero columns. The number of elementary service operations is assumed to be uniformly distributed between 1 and n_s. Therefore, the mean service time is equal to $(n_s + 1)/2$, which is therefore fixed when n_s is specified. In addition to quantifying the effect of n_s (the block size) on the computation times, we also consider the effect of the load. For this queue, the load is given by $\rho = \lambda E[L]/\mu$, where λ is the mean arrival rate (inverse of the mean IAT), μ is the mean service rate (inverse of the mean service time), and $E[L]$ is the mean batch size. We set the batch size distribution to be uniform between 1 and $\bar{L}$, hence, $E[L] = (\bar{L} + 1)/2$. As μ and $E[L]$ are fixed by specifying the values of n_s and $\bar{L}$, we use the rate λ to match a determined load ρ. The arrival process is thus assumed to have rate $\lambda = \rho\mu/E[L]$, SCV equal to 5, and decay rate of the autocorrelation function of the sequence of IATs equal to 0.5. These characteristics are matched with a MAP(m_a, D_0, D_+) of order $m_a = 2$ with the method introduced in [7]. The matrixes $\{D_1, \ldots, D_{\bar{L}}\}$ are obtained as $D_j = r_j D_+$, for $1 \leq j \leq \bar{L}$, where $r_j = 1/\bar{L}$ is the probability that a batch is of size j. Since $m_a = 2$, the block size of the original chain is $4n_s$ and the number of nonzero columns in A_0 is only two.

In the first scenario, we set both the maximum batch size $\bar{L}$ and the maximum number of elementary service operations n_s equal to 10. The results for different values

TABLE 1. Computation Times (s) for $\bar{L} = 10$, $n_s = 10$

ρ	CR-R	CR-F	FI-R	FI-F	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F_2
0.1	1.6	1.6	0.1	0.1	0.2	0.2	0.3	0.3	0.4	0.3	0.4
0.3	2.6	2.7	0.1	0.1	0.1	0.2	0.2	0.1	0.2	0.1	0.1
0.5	3.5	3.6	0.2	0.3	0.2	0.3	0.3	0.3	0.3	0.2	0.1
0.7	5.3	5.4	0.8	0.9	0.5	0.9	1.4	0.8	0.9	0.3	0.3
0.9	13.7	8.6	9.3	4.1	8.4	2.7	11.6	8.2	10.6	7.2	1.7

of the load are shown in Table 1. In this case, the use of the FRF provides an important gain for all the methods. However, compared to the **CR-R** method, the reduction that can be achieved by using the **O-R** method is significantly larger than the one that can be obtained by using the **CR-F** method. Compared to **CR-R**, we see that **O-R** provides an important gain for all the load values considered, whereas it provides slightly better results than **FI-R**. Additionally, the performance of **C-R** and **C*-R** is actually worse than the simpler **O-R**, with a larger difference for higher loads. In fact, **O-R** is the best method among those that do not consider the structure in Remark 1. We will comment more on this later, after presenting the results for other scenarios. The best method is **C*-F_2**, which outperforms all of the other methods for every value of the load considered, except $\rho = 0.1$. These results show that even for $n_s = 10$ (the ratio m/r is $2n_s$), the methods introduced in this article are able to reduce the times to compute the vector π.

We now consider a scenario in which the computation times become considerably larger due to an increase in the block size, as this is the case in which we expect our methods to provide more important gains. For the results shown in Table 2 we take the previous scenario and simply increase the value of n_s to 50. This makes the block size equal to 200 while the number of nonzero columns in A_0 remains equal to 2. In this case, we notice that the use of the FRF no longer provides a gain for the **CR**, **FI**, **O** methods. This is related to the increase in the block size, since the computational cost of the FRF is more sensitive to the block size than the RF [16]. On the other hand, we see how the **O-R** method shows a dramatical reduction in computation times, especially under low and mid loads. In this case, the **C*-R** method provides further improvements, although this reduction is rather limited for high loads. Even the use of **C*-F** does not provide a substantial reduction under high loads compared to **C-R**.

TABLE 2. Computation Times (s) for $\bar{L} = 10$, $n_s = 50$

ρ	CR-R	CR-F	FI-R	FI-F	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F_2
0.1	129.2	132.6	2.5	5.9	0.5	3.9	0.4	0.4	0.3	0.3	0.4
0.3	145.3	149.6	6.1	10.4	0.5	5.5	0.5	0.5	0.4	0.3	0.4
0.5	201.4	210.8	14.3	23.7	1.0	10.4	1.1	1.0	0.6	0.5	0.6
0.7	240.9	264.3	45.0	68.3	4.3	27.5	4.9	4.3	2.9	2.4	1.5
0.9	388.3	447.4	214.5	273.6	55.3	113.3	58.7	55.5	49.2	46.4	10.8

TABLE 3. Computation Times (s) for $\bar{L} = 20$, $n_s = 50$

ρ	CR-R	CR-F	FI-R	FI-F	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F_2
0.1	196.7	203.8	5.4	12.6	1.8	9.0	1.6	1.9	0.5	0.5	0.6
0.3	251.7	261.7	11.5	21.5	3.1	12.8	3.2	2.8	0.8	0.7	0.8
0.5	291.0	311.1	27.1	47.2	8.2	23.5	7.5	7.1	1.8	1.4	1.3
0.7	393.7	424.4	104.5	135.2	34.3	62.5	35.7	34.5	11.0	9.3	3.6
0.9	740.5	717.5	561.0	538.0	285.0	252.0	210.0	200.6	182.2	172.9	28.2

The only method that is able to show a significant reduction for $\rho = 0.9$, compared to **O-R**, is **C*-F_2**, which again outperforms all the other methods. Here, as well as in the previous case and for most of the load values considered, we find that the **C** approach is not able to reduce the times obtained with the **O** method. In this scenario, we also observe important absolute differences: for a load of 50%, **CR-R** takes more than 3 min, **FI-R** requires around 15 s, and **O-R** performs this computation in just 1 s. Additionally, for a load of 90%, the computation of π with **CR-R** and **FI-R** takes more that 6 and 3 min, respectively, whereas **O-R** requires less than 1 min, and the **C*-F_2** takes around 10 s.

As a final scenario, we consider an increase in the maximum batch size, making $\bar{L}$ equal to 20 while the other parameters are kept as in the previous scenario. The results are shown in Table 3, in which the first obvious observation is that all the methods require longer computation times than in the previous scenario. Although there is a small increase in the time to compute the matrix G, the difference between these scenarios is mostly due to the significantly longer times required to compute π. In this case, the number of blocks $\bar{L}$ and the number of terms π_i are larger, but the block size is the same as in the previous case. Therefore, the FRF-based methods, compared to their RF-based counterparts, show a relative better performance than earlier. However, for the **CR**, **FI**, and **O** approaches, the use of the FRF only provides a computational gain when the load is 0.9. On the other hand, the use of the FRF under the **C** and **C*** approaches results in reductions for almost all of the load values considered. Here again, the **O-R** method provides a significant reduction in computation time compared to either **CR-R** or **FI-R**. Additionally, the **C*-F_2** method is able to significantly reduce the computation times for high loads, outperforming the other approaches once more. We also notice that, in this case, the **C-R** and the **O-R** methods show a similar performance, except when the load is very high (0.9). If we consider their FRF versions, we find that **C-F** is always better than **O-F** and, moreover, **C-F** is the best among the **O** and the **C** approaches. This is in contrast with the previous results, for which **O-R** was typically the best performing among these four methods.

4.3. The *BMAP*[2]/*PH*[2]/1 Preemptive Priority Queue

We now illustrate how the *BMAP*[2]/*PH*[2]/1 preemptive priority queue can be modeled as an *M*/*G*/1-type MC with restricted downward transitions. In this queue, the arrivals also occur in batches and there are two types of customers, each one associated

with a different priority level. We assume that each batch is made of customers of only one type, and the maximum size of a batch of high- (resp. low-) priority customers is $\bar{L}_1$ (resp. $\bar{L}_2$). Hence, the arrival process can be modeled as a BMAP[2] characterized by the matrixes $\{D_0, D_1^{j_1}, D_2^{j_2}, 1 \leq j_1 \leq \bar{L}_1, 1 \leq j_2 \leq \bar{L}_2\}$. As mentioned in Section 4.1, we can define a version of this process for which the batch sizes are i.i.d. random variables and are independent of the IATs. Additionally, the service times for both the high- and the low-priority customers are PH-distributed. For the high- (resp. low-) priority customers, the parameters of the service-time distribution are (m_1, α, T) (resp. (m_2, β, S)).

To model this priority queue as an $M/G/1$-type MC, the level variable must be chosen such that during a single transition, it can decrease by at most 1. Therefore, one may take the level either as the number of high- or low-priority customers, since in both cases the level can only decrease by one (service completion) in a single transition, but it can increase by up to $\bar{L}_1$ or $\bar{L}_2$ (batch arrivals). In previous works [1,17,25], the number of high-priority customers has been used as the level. This choice induces a structure that can be exploited when both priority classes are assumed to have an infinite buffer. However, under BMAP arrivals, this approach requires the determination of $\bar{L}_1$ infinite block matrixes, which is done with a linearly convergent algorithm [25] that might require extremely long computation times. Here, we opt for the second alternative: to take the number of *low-priority* customers as the level. We further assume that the high-priority customers have a finite buffer of size C. This approach induces a completely different structure (restricted downward transitions), which can be exploited to obtain the stationary probability vector of the MC in an efficient manner for moderate values of C (e.g., 50 or 100). The assumption of a finite high-priority queue (of moderate size) does not necessarily limit the applicability of the model since high-priority queues are typically fairly short as opposed to low-priority queues. Hence, there is little difference among having a sufficiently large, finite, or an infinite high-priority buffer.

Since the number of low-priority customers is chosen to be the level, the phase keeps track of the number of high-priority customers and the state of the arrival and the service processes. Let $N_1(t)$ and $N_2(t)$ be the number of high- and low-priority customers in the system at time t, respectively. Additionally, let $J(t)$ be the state of the arrival process at time t, which takes values in the set $\{1, \ldots, m_a\}$. Let $S_1(t)$ (resp. $S_2(t)$) be the phase of the high- (resp. low-) priority service at time t when there is a customer of this class being served. When a low-priority customer is in service, there is no need to keep track of $S_1(t)$. However, when a high-priority customer is being served, $S_2(t)$ keeps track of the phase at which the first low-priority customer in the queue will (re-)start its service. Then when a low-priority customer is preempted by the arrival of a high-priority one, the service phase of the preempted customer is kept in $S_2(t)$ such that its service can be resumed from this phase. The queue is therefore described by the CTMC $X(t) = \{(N_2(t), N_1(t), J(t), S_2(t), S_1(t)), t \geq 0\}$. Its state space is ordered lexicographically and is described as follows. In level 0 there is no need to keep track of $S_2(t)$ since there are no low-priority customers in the system. The states in this level are subdivided in two subsets: The first considers the case for which there are no

high-priority customers either ($N_1(t) = 0$), so it is enough to keep track of the phase of the arrival process. This case is covered by the set of states $\Delta_{00} = \{j, 1 \leq j \leq m_a\}$. The second subset is $\Delta_{0+} = \{(i, j, s_1), 1 \leq i \leq C, 1 \leq j \leq m_a, 1 \leq s_1 \leq m_1\}$, which considers the case for which there are $i > 0$ high-priority customers; that is, the server is busy with a customer of this type and the service phase is s_1. Therefore, the set of states corresponding to level 0 is $\Delta_0 = \Delta_{00} \cup \Delta_{0+}$. For every level $k \geq 1$ there are also two subsets. The first is $\Delta_{+0} = \{(j, s_2), 1 \leq j \leq m_a, 1 \leq s_2 \leq m_2\}$, which describes the case for which there are no high-priority customers and the server is busy with a low-priority one. The second subset of level $k \geq 1$ is $\Delta_{++} = \{(i, j, s_2, s_1), 1 \leq i \leq C, 1 \leq j \leq m_a, 1 \leq s_2 \leq m_2, 1 \leq s_1 \leq m_1\}$, which considers the general case with $k \geq 1$ low-priority and $i \geq 1$ high-priority customers, the arrival process is in phase j, the service of the high-priority customer is in phase s_1, and the next low-priority customer to be served will start (or resume) in phase s_2. The set of states in level $k \geq 1$ is given by $\Delta_k = (k, \Delta_{+0} \cup \Delta_{++})$ and the full state space of the MC is $\Delta = \bigcup_{k \geq 0} \Delta_k$.

A full description of the blocks that define the $M/G/1$-type MC of the $BMAP[2]/PH[2]/1$ priority queue can be found in the Appendix. Of particular importance, however, is the block A_0, which holds the transition rates from level k to level $k - 1$. These transitions are triggered by the completion of a low-priority service, which can only occur in the absence of high-priority customers. Therefore, the block A_0 is given by

$$A_0 = \begin{bmatrix} I_{m_a} \otimes s\beta & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

Only the first $m_a m_2$ columns of this matrix are different from zero, and this number is relatively small compared to the complete block size $m = m_a m_2 + C(m_a m_2 m_1)$. In other words, the block size is $(1 + Cm_1)$ times larger than the number of nonzero columns in A_0. Therefore, by modeling the priority queue in this manner, we induce the restricted-downward-transitions structure. In addition, if the batch sizes are i.i.d. random variables independent of the arrival process, we have that $D_2^i = q_i D_2$. Then, by looking at the blocks $(A_i)_{i \geq 2}$ described in Eq. (A.1), we find that $A_{i+1}^{--} = q_i(I_C \otimes D_2 \otimes I_{m_2 m_1})$, for $1 \leq i \leq \bar{L}_2$. Moreover, from Eq. (A.1), we also find that $(A_i^{+-})_{i=2}^{\bar{L}_2}$ and $(A_i^{-+})_{i=2}^{\bar{L}_2}$ are zero. Therefore, these blocks have the structure defined in Remark 1, which can be exploited together with the restricted-transitions structure as described in Section 3.3.

We now consider some numerical instances of the $BMAP[2]/PH[2]/1$ preemptive priority queue to illustrate the performance of the methods introduced in this article. As earlier, one of the main parameters of this queue is the load ρ, which in this case is given by $\rho = \lambda_1 E[L_1]/\mu_1 + \lambda_2 E[L_2]/\mu_2$. For customers of type i, λ_i is their mean arrival rate, μ_i is their mean service rate, and $E[L_i]$ is their mean batch size, for $1 \leq i \leq 2$. Recall that high- (resp. low-) priority customers are referred to as customers of type 1 (resp. 2). We assume that high- and low-priority customers have the same mean service

rate equal to 1 (i.e., $\mu_1 = \mu_2 = 1$). Additionally, both service-time distributions have SCV equal to 2. We can match these two moments with a PH distribution of order 2 by using the method in [24]. The batch size distribution is assumed to be the same for both customer types. Specifically, we set $\bar{L}_1 = \bar{L}_2 = \bar{L}$, and let the batch size of both types be uniformly distributed between 1 and $\bar{L}$, with mean $E[L]$. Here, we first build a single arrival process with arrival rate $\lambda = \lambda_1 + \lambda_2$. With the assumptions stated earlier this arrival rate is given by $\lambda = \rho/E[L]$. We assume that the SCV of the batch IAT distribution is equal to 5 and the decay rate of the autocorrelation function of the sequence of batch IATs is 0.5. We use the method in [7] to match the first two moments of the IAT distribution and the decay rate of the autocorrelation function with an order-2 MAP. The resulting process is a MAP characterized by the 2×2 matrixes D_0 and D_+ that describe the batch IATs irrespective of their type. The matrixes D_1 and D_2 are obtained as $D_j = v_j D_+$ for $1 \leq j \leq 2$, where $v_1 + v_2 = 1$. In this scenario, we also assume that both customer types have the same arrival rate ($v_1 = v_2 = 0.5$). As the size of the MAP is $m_a = 2$ and the size of the PH representation of the service-time distributions is $m_1 = m_2 = 2$, the number of nonzero columns is $r = 4$ and the block size of the original $M/G/1$-type MC is $m = 4 + 8C$.

For the results in Table 4 we set the maximum batch size equal to 5 and the size of the high priority buffer equal to 20. Even this buffer size causes very few losses, as illustrated by the loss rate (LR) of high-priority customers included in the last column of Table 4. The results for the general techniques with FRF (**CR-F** and **FI-F**) are not included in this or the following tables, because their results were always worse that their RF counterparts due to the large block size. Here, we observe how the use of the censored process to compute π reduces the computation times significantly compared with simply solving the original $M/G/1$-type MC. Moreover, we see that the main gain is obtained due to the fast computation of G, and additional gains can be realized by using the censored process to compute π, although only when the additional structure from Remark 1 is exploited. Recall that only the **C*** methods take advantage of this additional structure. Specifically, the use of **O-R** provides a significant reduction compared to either **CR-R** or **FI-R**. We also observe that **C-R** requires more time than **O-R**, whereas **C-F** shows times similar to **O-R**, and **C*-R** performs worse than **O-R** for high loads. In fact, we have noted that in many cases, as in the previous section, the gain obtained by using **C*-R** instead of **O-R** decreases as the load increases. The reason for this behavior is that to compute π^+ in **C*-R**, we use

TABLE 4. Computation Times (s) for $\bar{L} = 5$, $C = 20$

ρ	CR-R	FI-R	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F$_2$	LR
0.1	51.0	1.1	0.3	1.2	0.5	0.4	0.4	0.4	0.4	2.09E-09
0.3	112.5	2.4	0.4	2.1	0.5	0.6	0.4	0.4	0.4	2.25E-06
0.5	164.8	5.7	0.8	4.2	1.0	0.9	0.7	0.6	0.7	5.92E-05
0.7	222.0	15.3	1.8	10.3	2.9	1.8	2.5	1.4	1.4	4.67E-04
0.9	282.8	59.1	13.2	38.7	20.1	13.5	17.8	9.7	5.0	1.93E-03

Eq (12) with the blocks of the censored process, but the number of blocks increases with the load. For instance, in this scenario for loads 0.1, 0.5, and 0.9, the number of blocks $(\bar{A}_i)_{i\geq 0}$ is 45, 199, and 552, respectively. This effect is reduced when the ratio m/r becomes large (e.g., 50), since in this case the vector–matrix multiplications of the original blocks become very expensive. Therefore, the censored process has the drawback of having many blocks, requiring many vector–matrix multiplications to compute the sum in Eq. (12). However, when the number of blocks and the number of terms in the vector π are large and the block size is small, the FRF is expected to perform significantly better than the customary RF [16]. This is the case for the censored process when the load is high, and we therefore observe important gains when using **C-F** instead of **C-R** and using **C*-F** or **C*-F$_2$** instead of **C*-R**.

We now consider a simple modification on the previous scenario by increasing the buffer size from 20 to 50. This makes the block size equal to 404. For this buffer size, the method **CR** fails to compute the matrix G due to lack of memory. At this point we must mention that all the times shown here were obtained using a personal computer with a 2-GHz processor and 2 GB of RAM. Apart from the methods introduced in this article, Table 5 shows the computation times for the **FI-R** method and for the approach introduced in [25]. The difference between **INF** and our methods is extremely large for moderate to large loads. It must be noted that our methods do not scale easily with the size of the high-priority buffer C. If this parameter is large, our methods would require a large amount of memory, whereas the **INF** method does not suffer from this problem. However, if the buffer is not so large, our methods provide huge savings in computation time. For instance, Table 5 shows that when the load is 0.9, the **INF** approach takes more than 2 h to compute π, **FI-R** takes more than 14 min, **O-R** requires 1.2 min, and **C*-F$_2$** takes only 0.5 min. Additionally, the loss rate assuming a finite buffer is almost negligible, even for high loads. Among the methods that do not use the extra structure introduced in Remark 1, the best choice is **O-R**, as the methods based on the censored process require more time, partly due to the computation of the blocks of the censored process.

In the next scenario we make the maximum batch size $\bar{L}$ equal to 15, such that both the original and the censored $M/G/1$-type MCs have more blocks. The computation times are shown in Table 6. Comparing with the results in Table 5, we see that all of the methods require more time to compute π, but the increase is not proportional. Actually, the **INF** method takes more than 10 times longer in the new scenario, whereas

Table 5. Computation Times (s) for $\bar{L} = 5$, $C = 50$

ρ	INF	FI-R	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F$_2$	LR
0.1	2	17.4	1.9	11.7	2.6	3.3	1.9	1.9	2.5	2.14E-18
0.3	343	37.4	2.7	17.2	4.3	5.6	2.7	2.8	3.8	8.10E-12
0.5	1010	84.9	5.0	32.2	7.7	10.4	4.3	4.5	6.1	1.25E-08
0.7	2427	223.1	12.7	79.2	17.9	21.7	9.5	9.7	11.3	1.41E-06
0.9	6832	846.2	71.1	308.9	95.3	81.4	59.9	45.2	31.7	3.70E-05

TABLE 6. Computation Times (s) for $\bar{L} = 15$, $C = 50$

ρ	INF	FI-R	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F_2	LR
0.1	55	31.0	5.0	28.0	9.3	13.4	4.7	4.8	5.7	2.77E-10
0.3	4149	69.1	9.8	39.6	16.0	24.6	6.6	7.0	8.2	2.33E-07
0.5	10661	171.3	24.0	76.7	30.3	29.5	10.9	10.0	11.2	6.58E-06
0.7	31033	497.2	76.1	195.8	93.6	87.0	32.6	25.8	22.8	5.90E-05
0.9	82545	1832.9	398.3	699.4	475.7	426.4	240.3	187.2	64.9	2.78E-04

TABLE 7. Computation Times (s) for $\bar{L} = 15$, $C = 50$, $v_1 = 0.1$

ρ	INF	FI-R	O-R	O-F	C-R	C-F	C*-R	C*-F	C*-F_2	LR
0.1	44	36.7	5.5	28.2	11.5	14.8	6.1	6.1	7.0	3.87E-14
0.3	1854	81.6	10.3	41.1	18.0	26.5	8.5	8.9	10.2	1.44E-11
0.5	2730	201.7	24.6	78.1	32.6	31.8	12.2	11.5	12.8	2.58E-10
0.7	4247	575.3	79.3	202.8	92.9	89.1	28.7	24.6	21.0	1.82E-09
0.9	6312	2570.4	700.8	1035.2	740.0	628.8	390.8	306.0	71.6	8.06E-09

the **FI-R** method doubles its computation times. Among our methods, **C*-F_2** shows the best behavior for high loads, roughly doubling the computation time compared to the previous scenario. We also observe that for mid loads, the difference among the **C*** methods is negligible, but for high loads, the advantage of using **C*-F_2** is large, even compared to **C*-F**. Here, again, **O-R** is the best method among those that do not exploit the additional structure from Remark 1. Additionally, we notice that, in this and the previous scenarios, the use of the FRF provides no gain for the **O** method, as is to be expected given the large block size. For the other methods, the use of the FRF provides a significant gain for mid-to-high loads, as, in this case, the FRF is applied on the censored process, the size of which is by construction very small.

We conclude this section by looking at a scenario in which the number of high-priority customers is a small fraction of the total number of customers. This scenario might arise, for instance, in a communication system for which the high priority is assigned to a subset of the customers such that this subset will experience a better quality of service. One way to accomplish this is to assign high priority only to a small proportion of the customers. To consider this, we simply set $v_1 = 0.1$, meaning that the high-priority customers represent only 10% of the customers. In Table 7 we see that under this configuration, the **INF** approach has a better performance than in the previous scenario, for which the proportions of high- and low-priority customers were equal. This gain is due to the fact that, in this case, the most expensive operation in all the methods is the actual computation of π, after obtaining π_0 and the set of R matrixes (in the **INF** approach) or the matrix G (in our approach). Since now there are relatively fewer high-priority customers in the system and the **INF** method uses the number of these customers as the level, this method needs to compute fewer terms of the vector π. On the other hand, our methods use the number of low-priority customers as the level and, therefore, they now need to compute many more terms of

π. In spite of this gain, under high loads, the **O-R** and **C*-F$_2$** methods are still around 10 and 100 times faster than the **INF** approach, respectively. Compared to **FI-R**, **O-R** is between 4 and 10 times faster, whereas **C*-F$_2$** is between 5 and 400 times faster. We also observe that for loads up to 0.7, the **O-R** method has the best performance among those that do not use the structure in Remark 1. However, for $\rho = 0.9$, the **C-F** method is able to outperform the **O-R** approach. As, in this case, the proportion of high-priority customers is smaller than in the previous scenario, the loss rate is also smaller and becomes negligible.

In addition to the computation times, it is relevant to consider the behavior of the approach introduced in this article in terms of the residual error. Let the infinity norm of an $n \times m$ matrix K be given by $||K||_\infty = \max_{i=1}^{n} \sum_{j=1}^{m} K_{ij}$. Let $\hat{G}$ be the matrix that solves Eq. (2) obtained with FI and let $\tilde{G}$ be the matrix found with the methods introduced in this article. Additionally, let $\hat{\pi}$ be the vector obtained by applying RF on $\hat{G}$ and let $\tilde{\pi}$ be the vector obtained with any of the methods introduced here. We have computed the residual error for $\tilde{G}$, defined as

$$\left\| \sum_{i=0}^{N} A_i \tilde{G}^i \right\|_\infty ,$$

which gives a measure of the goodness of $\tilde{G}$ as a solution for Eq. (2). For all the instances considered in this article, the residual error was always below $\epsilon = 10^{-14}$. We also computed the differences $||\hat{G} - \tilde{G}||_\infty$ and $||\hat{\pi} - \tilde{\pi}||_\infty$, obtaining a similar result. This reveals the good behavior of the approach proposed here, which is to be expected since the algorithms on which our method relies (cyclic reduction, the Schur decomposition, and the Hessenberg–Schur method for the Sylvester equation) are numerically stable.

5. CONCLUSION

From the results in the previous section we can conclude that the methods proposed in this article provide an important tool to evaluate the performance of the *BMAP*/*PH*/1 queue and the *BMAP*[2]/*PH*[2]/1 preemptive priority queue, since they are able to analyze rather large systems in a fraction of the time required by other methods. In general, we see that by exploiting the restricted-downward-transitions property, we are able to reduce the total time to compute π. The gains depend not only on the ratio m/r but also on the parameters of the system and how these affect the expected sojourn times in S^-, compared to those in S^+. However, we have seen in the previous examples that our methods are able to provide significant gains in computation time even for a rather small ratio m/r. Additionally, among the methods that only exploit the restricted-transitions structure, we have observed that typically the **O-R** method provides the best results, whereas the **O-F** is badly affected by the large block size. The use of the censored process, either **C-R** or **C-F**, can in some cases provide some additional gains. However, given the overhead required by the **C** methods, compared

to the simpler **O-R**, the latter appears as a better option when the additional structure of Remark 1 is absent. Notwithstanding, when this additional structure is present, the use of the censored process leads to very important gains. In this case, the **C*-R** method has the best performance for low loads, whereas for high loads the **C*-F$_2$** approach is the best alternative to compute π.

References

1. Alfa, A.S. (1998) Matrix-geometric solution of discrete time *MAP*/*PH*/1 priority queue. *Naval Research Logistics* 45: 23–50.
2. Bini, D., Latouche, G., & Meini, B. (2003) Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra and Its Applications* 366: 39–64.
3. Bini, D., Latouche, G., & Meini, B. (2005) *Numerical methods for structured Markov chains*. Oxford: Oxford University Press.
4. Bini, D. & Meini, B. (1996) On the solution of a nonlinear matrix equation arising in queueing problems. *SIAM Journal of Matrix Analysis and Applications* 17: 906–926.
5. Chandramouli, Y., Neuts, M., & Ramaswami, V. (1989) A queueing model for meteor burst packet communication systems. *IEEE Transactions on Communications* 37: 1024–1030.
6. Cumani, A. (1982) On the canonical representation of homogeneous Markov processes modeling failure-time distributions. *Microeconomics and Reliability* 22: 583–602.
7. Diamond, J.E. & Alfa, A.S. (2000) On approximating higher order MAPs with MAPs of order two. *Queueing Systems* 34: 269–288.
8. Gardiner, J.D., Laub, A.J., Amato, J.J., & Moler, C.B. (1992) Solution of the Sylvester matrix equation $AXBT + CXDT = E$. *ACM Transactions on Mathematical Software* 18: 223–231.
9. Golub, G.H., Nash, S., & Van Loan, C. (1979) A Hessenberg–Schur method for the problem $AX + XB = C$, *IEEE Transactions on Automatic Control* 24: 909–913.
10. Golub, G.H. & Van Loan, C. (1996) *Matrix computations*. Baltimore, MD: The Johns Hopkins University Press.
11. Grassmann, W.K. & Tavakoli, J. (2008) Solving QBD processes when levels can increase only in certain phases. Presented at the MAM6 conference, Beijing.
12. He, Q. & Neuts, M.F. (1998) Markov chains with marked transitions. *Stochastic Processes and their Applications* 74: 37–52.
13. Kemeny, J.G., Snell, J.L., & Knapp, A.W. (1976) *Denumerable Markov chains*. New York: Springer-Verlag.
14. Latouche, G. & Ramaswami, V. (1999) *Introduction to matrix analytic methods in stochastic modeling*. Philadelphia: SIAM.
15. Lucantoni, D. (1991) New results on the single server queue with a batch markovian arrival process. *Stochastic Models* 7: 1–46.
16. Meini, B. (1997) An improved FFT-based version of Ramaswami's formula. *Stochastic Models* 13(2): 223–238.
17. Miller, D.R. (1981) Computation of steady-state probabilities for *M*/*M*/1 priority queues. *Operations Research* 29: 945–958.
18. Neuts, M.F. (1981) Matrix-geometric solutions in stochastic models. Baltimore, MD: The John Hopkins University Press.
19. Neuts, M.F. (1989) Structured stochastic matrices of *M*/*G*/1 type and their applications. New York: Marcel Dekker.
20. Pérez, J.F. & Van Houdt, B. (2009) Quasi-birth-and-death processes with restricted transitions and its application. *Performance Evaluation* 68: 126–141.
21. Pérez, J.F. & Van Houdt, B. (2009) Exploiting restricted transitions in quasi-birth-and-death processes. Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST).

22. Ramaswami, V. (1988) A stable recursion for the steady state vector in Markov chains of *M*/*G*/1 type. *Stochastic Models* 4: 183–188.
23. Ramaswami, V. & Latouche, G. (1982) A general class of Markov processes with explicit matrix-geometric solutions. *OR Spectrum* 8(4): 209–218.
24. Whitt, W. (1982) Approximating a point process by a renewal process, I: Two basic methods. *Operations Research* 30: 125–147.
25. Zhao, J.A., Li, B., Cao, X.-R., & Ahmad, I. (2006) A matrix-analytic solution for the *DBMAP*/*PH*/1 priority queue. *Queueing Systems* 53(3): 127–145.

APPENDIX

The *BMAP*[2]/*PH*[2]/1 Preemptive Priority Queue

The purpose of this appendix is to provide the definition of the blocks that make up the generator matrix Q (Eq. (1)) of the MC that describes the $BMAP[2]/PH[2]/1$ priority queue. First, let the matrix $\bar{D}_1^j$ be defined as $\bar{D}_1^j = \sum_{k=j}^{\bar{L}_1} D_1^k$ for $1 \le j \le \bar{L}_1$ and let $\bar{D}_1^0 = D_0 + \bar{D}_1^1$. The $m_b \times m_b$ block B_0 is given by

$$B_0 = \begin{bmatrix}
D_0 & D_1^1 \otimes \alpha & D_1^2 \otimes \alpha & \cdots & D_1^{\bar{L}_1} \otimes \alpha & & & & 0 \\
I \otimes t & D_0 \oplus T & D_1^1 \otimes I & \cdots & D_1^{\bar{L}_1-1} \otimes I & \ddots & & & \\
 & I \otimes t\alpha & D_0 \oplus T & \cdots & D_1^{\bar{L}_1-2} \otimes I & \ddots & & & \\
 & & \ddots & \ddots & \ddots & \ddots & & & \\
 & & & I \otimes t\alpha & D_0 \oplus T & \cdots & D_1^{\bar{L}_1-2} \otimes I & D_1^{\bar{L}_1-1} \otimes I & \bar{D}_1^{\bar{L}_1} \otimes I \\
 & & & & \ddots & \ddots & \vdots & \vdots & \vdots \\
 & & & & & I \otimes t\alpha & D_0 \oplus T & D_1^1 \otimes I & \bar{D}_1^2 \otimes I \\
 & & & & & & I \otimes t\alpha & D_0 \oplus T & \bar{D}_1^1 \otimes I \\
0 & & & & & & & I \otimes t\alpha & \bar{D}_1^0 \oplus T
\end{bmatrix},$$

where $m_b = m_a + Cm_am_1$ and I is the identity matrix of appropriate dimension. Since in level 0 there are no low-priority customers, all of the transitions in B_0 are associated with arrivals and service completions of high-priority customers. In addition, the finiteness of the high-priority buffer is reflected in the last block column of B_0, that is, if there are $C-2$ high-priority customers in the queue and two or more of these customers arrive in a batch ($\bar{D}_1^2$), the queue will only accept two and the others will be dropped. The transitions from level 0 to upper levels are governed by the $m_b \times m$ blocks B_j, defined as

$$B_j = \begin{bmatrix}
D_2^j \otimes \beta & 0 & \cdots & 0 \\
0 & D_2^j \otimes \beta \otimes I & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & D_2^j \otimes \beta \otimes I
\end{bmatrix}, \quad j = 1, \ldots, \bar{L}_1,$$

where $m = m_am_2 + Cm_am_2m_1$. Since in level 0 there are no low-priority customers, if one of these arrives and finds the server idle, it starts service immediately, selecting an initial service

phase according to β. If there is a high-priority customer in service, the incoming customer selects the phase in which it will eventually start service. The $m \times m_b$ block C_0 holds the transitions from level 1 to level 0 and is given by

$$C_0 = \begin{bmatrix} I_{m_a} \otimes s & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix},$$

which reflects the fact that the completion of a low-priority service can only occur if there are no high-priority customers in the system.

Before determining the remaining blocks, we define the operator $R(\cdot)$ applied on a matrix M as $R(M) = I_{m_2} \otimes M$, which will help to make the notation simpler. The $m \times m$ block A_1 is defined as

$$A_1 = \begin{bmatrix} D_0 \oplus S & D_1^1 \otimes R(\alpha) & D_1^2 \otimes R(\alpha) & \cdots & D_1^{\bar{L}_1} \otimes R(\alpha) & & & 0 \\ I \otimes t & D_0 \oplus R(T) & D_1^1 \otimes I & \cdots & D_1^{\bar{L}_1-1} \otimes I & \ddots & & \\ & I \otimes t\alpha & D_0 \oplus R(T) & \cdots & D_1^{\bar{L}_1-2} \otimes I & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & I \otimes t\alpha & D_0 \oplus R(T) & \cdots & D_1^{\bar{L}_1-1} \otimes I & \bar{D}_1^{\bar{L}_1} \otimes I \\ & & & & \ddots & \ddots & \vdots & \vdots \\ & & & & & I \otimes t\alpha & D_0 \oplus R(T) & \bar{D}_1^1 \otimes I \\ 0 & & & & & & I \otimes t\alpha & \bar{D}_1^0 \oplus R(T) \end{bmatrix}.$$

The first block row of this matrix reflects how a low-priority customer that is being attended might be preempted by the arrival of a batch of high-priority customers, and the first customer in the batch starts service in a phase selected according to α. The use of $R(\cdot)$ reflects that the system "remembers" the phase in which the first low-priority customer in the queue will eventually restart its service. The transitions to upper levels are driven by the blocks

$$A_{j+1} = \begin{bmatrix} D_2^j \otimes I & 0 & \cdots & 0 \\ 0 & D_2^j \otimes I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_2^j \otimes I \end{bmatrix}, \quad j = 1, \dots, \bar{L}_1, \tag{A.1}$$

which are related only to arrivals of low-priority customers. This concludes the description of the blocks, since the block A_0, which holds the transitions from level k to level $k-1$, was already described in Section 4.3.