The impact of dampening demand variability in a production/inventory system with multiple retailers

B. Van Houdt and J.F. Pérez

Abstract We study a supply chain consisting of a single manufacturer and two retailers. The manufacturer produces goods on a make-to-order basis, while both retailers maintain an inventory and use a periodic replenishment rule. As opposed to the traditional (r,S) policy, where a retailer at the end of each period orders the demand seen during the previous period, we assume that the retailers dampen their demand variability by smoothing the order size. More specifically, the order placed at the end of a period is equal to β times the demand seen during the last period plus $(1-\beta)$ times the previous order size, with $\beta \in (0,1]$ the smoothing parameter. We develop a GI/M/1-type Markov chain with only two nonzero blocks A_0 and A_d to analyze this supply chain. The dimension of these blocks prohibits us from computing its rate matrix R in order to obtain the steady state probabilities. Instead we rely on fast numerical methods that exploit the structure of the matrices A_0 and A_d, i.e., the power method, the Gauss-Seidel iteration and GMRES, to approximate the steady state probabilities. Finally, we provide various numerical examples that indicate that the smoothing parameters can be set in such a manner that all the involved parties benefit from smoothing. We consider both homogeneous and heterogeneous settings for the smoothing parameters.

Key words: Structured Markov chains; Supply chain; Inventory; MSC: Primary 60J22; Secondary 90B30, 90B05

B. Van Houdt
Department of Mathematics and Computer Science, University of Antwerp - IBBT,
Middelheimlaan 1, B-2020 Antwerpen, Belgium,
e-mail: benny.vanhoudt@ua.ac.be

J.F. Pérez
Department of Electrical and Electronics Engineering, Universidad de los Andes,
Cra 1 No. 18A-12, Bogotá, Colombia,
e-mail: jf.perez33@uniandes.edu.co

1 Introduction

Consider a two-echelon supply chain consisting of a single retailer and a single manufacturer, where the retailer places an order for a batch of items with the manufacturer at regular time instants, i.e., the time between two orders is fixed and denoted as r. The manufacturer may be regarded as a single server queue that produces these items and delivers them to the retailer as soon as a complete order is finished. The retailer sells these items and maintains an inventory on hand to meet customer demands. When the customer demand exceeds the current inventory on hand, only part of the demand is immediately fulfilled and the remaining items are delivered as soon as new items become available at the retailer. Hence, items are backlogged instead of being lost (i.e., there are no *lost sales*). We assume that the manufacturer does not maintain an inventory, but simply produces items whenever an order arrives, i.e., it operates on a *make-to-order* basis.

A key performance measure in such a system is the *fill-rate*, which is a measure for the proportion of customer demands that can be met without any delay. In order to guarantee a certain fill-rate it is important to determine the size of the orders placed at the regular time instants. This size will depend on the current *inventory position*, defined as the inventory on hand plus the number of items on order minus the number of backlogged items. The rule that determines the order size is termed the *replenishment* rule. A well-studied replenishment rule exists in ordering an amount such that the inventory position is raised after each order to some fixed position S, called the *base-stock level*. This basically means that at the regular time instants, you simply order the amount of items sold since the last order instant. As a result, the order policy of the retailer is called an (r,S) policy.

A common approach in the analysis of such a policy is to assume an *exogenous* lead time, which means that the time required to *deliver* an order is independent of the size of the current order and independent of the lead time of previous orders. In [3] the (R,S) policy was studied with *endogenous* lead times, meaning the lead times depend on the order size and consecutive lead times are correlated. The results in [3] indicate that exogenous lead times result in a severe underestimation of the required inventory on hand, as expected.

When the lead times are endogenous, it is clear that a high variability in the order sizes comes at a cost, as this increases the variability of the arrival process at the manufacturer and therefore increases the lead times. As a result, replenishment rules that smooth the order pattern at the retailer were studied in [4] and it was shown that the retailer can reduce the upstream demand variability without having to increase his safety stock (much) to maintain customer service at the same target level. Moreover, on many occasions the retailer can even decrease his safety stock somewhat when he smooths his orders. This is clearly advantageous for both the retailer and the manufacturer. The manufacturer receives a less variable order pattern and the retailer can decrease his safety stock while maintaining the same fill rate, so that a cooperative surplus is realized.

In this paper we analyze the same set of replenishment rules as in [4], but now we look at a two-echelon supply chain consisting of one manufacturer and two retailers, where either both, one or neither of the retailers uses a smoothing rule. The main question that we wish to address therefore exists in studying whether all parties can still benefit when the orders are smoothed and moreover who benefits most.

As in [4], one of the key steps in the analysis of this supply chain system will exist in setting up a GI/M/1-type Markov chain [8], that has only two non-zero blocks, denoted as A_0 and A_d. However, as opposed to [4], the size of these blocks often prohibits us from storing them into main (or secondary) memory. This implies that iteratively computing the dense R matrix, used to express the matrix geometric steady state vector of the GI/M/1-type Markov chain, by one of the existing methods such as functional iterations or cyclic reduction [1], is no longer possible/efficient. Instead, we will rely on the specific structure of the matrices A_0 and A_d and will make use of numerical methods typically used to solve large finite Markov chains, such as the shuffling algorithm [5], Kronecker products, the power method, the Gauss-Seidel iteration and GMRES [10].

2 Model Description

We consider a two-echelon supply chain with two retailers and a single manufacturer, where both retailers maintain their own inventory. Every period, both retailers observe their customer demand. If there is enough on-hand inventory available at a retailer, the demand is immediately satisfied. If not, the shortage is backlogged. To maintain an appropriate amount of inventory on hand, both retailers place a replenishment order with the manufacturer at the end of every period. The manufacturer does not hold a finished goods inventory but produces the orders on a make-to-order basis. The manufacturers production system is characterized by a single server queueing model that sequentially processes the orders, which require stochastic processing times. Once the complete replenishment order of both retailers is produced, the manufacturer replenishes both inventories. Hence, the order in which the two orders are produced is irrelevant, as shipping only occurs when both orders are ready.

The time from the moment an order is placed to the moment that it replenishes the retailers inventory, is the *replenishment* lead time T_r. The queueing process at the manufacturer clearly implies that the retailers replenishment lead times are stochastic and correlated with the order quantity. The sequence of events in a period is as follows. The retailer first receives goods from the manufacturer, then he observes and satisfies customer demand and finally, he places a replenishment order with the manufacturer. The following additional assumptions are made:

1. Customer demand during a period for retailer i is independently and identically distributed (i.i.d.) over time according to an arbitrary, finite, discrete distribution $D^{(i)}$ with a maximum of $m_D^{(i)}$, for $i = 1$ and 2. The demand at the retailers is also assumed to be independent of each other. For further use, denote $m_D = m_D^{(1)} + m_D^{(2)}$.

2. The order quantity $O_t^{(i)}$ of retailer i during period t is determined by the retailers replenishment rule and influences the variability in the orders placed on the manufacturer. Possible replenishment rules are discussed in the next section.
3. The replenishment orders are processed by a single FIFO server. This excludes the possibility of order crossovers. When the server is busy, new orders join a queue of unprocessed orders.
4. The orders placed during period t are delivered when both orders have been produced.
5. Orders consist of multiple items and the production time of a single item is i.i.d. according to a discrete-time phase type (PH) distribution with representation (α, U). For further use, we define $u^* = e - Ue$, with e a column vector of ones.

The PH distribution is determined using the matching procedure presented in [4], that matches the first two moments of the production time using an order 2 representation, meaning the matrix U is a 2×2 matrix and α a size 2 row vector, even if the squared coefficient of variation is small by exploiting the scaling factor as in [2]. This implies that the length of a time slot is chosen as half of the mean production time of an item. In other words, the mean production time of an item is two time slots, while the length of a period is denoted as d time slots, where d is assumed to be an integer.

The time from the moment the order arrives at the production queue to the point that the production of the entire batch is finished, is the *production* lead time or response time, denoted by T_p. Note that the production lead time is not necessarily an integer number of periods. Since in our inventory model events occur on a discrete time basis with a time unit equal to one period, the replenishment lead time T_r is expressed in terms of an integer number of periods. For instance, suppose that the retailer places an order at the *end* of period t, and it turns out that the production lead time is 1.4 periods. This order quantity will be added to the inventory in period $t+2$, and due to our sequence of events, can be used to satisfy demand in period $t+2$. As such, we state that the replenishment lead time T_r is $\lfloor T_p \rfloor$ periods, i.e., 1 period in our example.

3 Replenishment Rules

The retailers considered in this paper apply an (r,S) policy with or without smoothing, meaning amongst others they place an order at the end of each period. Without smoothing, the order size is such that the inventory position IP, defined as the on-hand inventory plus the number of items on order minus the backlogged items, equals some fixed S after the order is placed. In other words, the size of the order O_t at the end of period t simply equals the demand D_t observed during period t.

If smoothing is applied with parameter $0 < \beta < 1$, we do not order the difference between S and IP, but instead only order β times $S - IP$. As will become clear below, this does not imply that fewer items are ordered in the long run, it simply means that some items will be ordered at a later time. As shown in [4], this rule is equivalent to stating that the size of the order at the end of period t, denoted O_t, is given by

$$O_t = (1-\beta)O_{t-1} + \beta D_t,$$

where D_t is the demand observed by a retailer in period t. Hence, setting $\beta = 1$ implies that we do not smooth. This equation also shows that the mean order size is still equal to the mean demand size $E[D]$. It is also easy to show [4] that the variance of the order size $Var[O]$ equals

$$\frac{\beta}{(2-\beta)} Var[D],$$

meaning the variance decreases to zero as β approaches zero, where $Var[D]$ is the variance in the demand. It is also possible to consider β values between 1 and 2, but this would amplify the variability instead of dampening it.

The key question that our analytical model will answer is how to select the base-stock level S such that the fill-rate, a measure for the proportion of demands that can be immediately delivered from the inventory on hand, defined as

$$1 - \frac{\text{expected number of backlogged items}}{\text{expected demand}},$$

is sufficiently high. The level S is typically expressed using the *safety stock SS*, defined as the average net stock just before a replenishment arrives (where the net stock equals the inventory on hand minus the number of backlogged items). For a retailer that smooths with parameter β, S and SS are related as follows [4]

$$S = SS + (E[T_r]+1)E[D] + \frac{1-\beta}{\beta}E[D],$$

where $E[T_r]$ is the mean replenishment lead time. Thus, a good policy will result in a smaller safety stock SS, which implies a lower average storage cost for the retailer.

4 The Markov chain

Both Markov chains developed in this section are a generalization of the Markov chain introduced in [4], for the system with a single retailer. The numerical method to attain their stationary probability vector, discussed in Section 5, is however very different.

From now on we will express all our variables in time slots, where the length of a single slot equals half of the mean production time, i.e., $\alpha(I-U)^{-1}e/2$, and orders are placed by both retailers every d time slots. Hence, the order size of retailer i at the end of period t is now written as $O_{td}^{(i)}$ and

$$O^{(i)}_{td} = (1-\beta_i)O^{(i)}_{(t-1)d} + \beta_i D^{(i)},$$

where β_i is the smoothing parameter of retailer i, for $i = 1,2$. As the order size must be an integer, the integer amount ordered $O^{(i*)}_{td}$ will equal $\lceil O^{(i)}_{td} \rceil$ with probability $O^{(i)}_{td} - \lfloor O^{(i)}_{td} \rfloor$ and $\lfloor O^{(i)}_{td} \rfloor$ with probability $\lceil O^{(i)}_{td} \rceil - O^{(i)}_{td}$ in case $O^{(i)}_{td}$ is not an integer. This guarantees that $E[O^{(i*)}_{td}] = E[O^{(i)}_{td}] = E[D^{(i)}]$.

The joint order O^*_{td} of both retailers placed at time td equals $O^{(1*)}_{td} + O^{(2*)}_{td}$. Recall, both these orders are only delivered by the manufacturer when the joint order has been produced. Next, define the following random variables:

- t_n: the time of the n-th observation point, which we define as the n-th time slot during which the server is busy,
- $a(n)$: the arrival time of the joint order in service at time t_n,
- B_n: the age of the joint order in service at time t_n, expressed in time slots, i.e., $B_n = t_n - a(n)$,
- C_n: the number of items part of the joint order in service that still need to start or complete service at time t_n,
- S_n: the service phase at time t_n.

All events, such as arrivals, transfers from the waiting line to the server, and service completions are assumed to occur at instants immediately after the discrete time epochs. This implies that the age of an order in service at some time epoch t_n is at least 1. We start by introducing the Markov chain for the case where both retailers smooth.

4.1 Both retailers smooth

It is clear that the stochastic process $(B_n, C_n, O^{(1)}_{a(n)}, O^{(2)}_{a(n)}, S_n)_{n\geq 0}$ forms a discrete time Markov process on the state space $\mathbb{N}_0 \times \{(c,x_1,x_2) | c \in \{1,\ldots,m_D\}, 1 \leq x_i \leq m^{(i)}_D, i \in \{1,2\}\} \times \{1,2\}$, as the PH service requires only two phases. Note that the process makes use of the order quantities $O^{(i)}_{a(n)}$ instead of the integer values $O^{(i*)}_{a(n)}$. Given that these order quantities are real numbers, the Markov process $(B_n, C_n, O^{(1)}_{a(n)}, O^{(2)}_{a(n)}, S_n)_{n\geq 0}$ has a continuous state space which makes it very hard to find its steady state vector.

Therefore, instead of keeping track of $O^{(i)}_{a(n)}$ in an exact manner, we will round it in a probabilistic way to the nearest multiple of $1/g$, where $g \geq 1$ is an integer termed the *granularity* of the system. Clearly, the larger g, the better the approximation. Hence, we approximate the Markov process above by the Markov chain $(B_n, C_n, O^{g,(1)}_{a(n)}, O^{g,(2)}_{a(n)}, S_n)_{n\geq 0}$ on the discrete state space $\mathbb{N}_0 \times \{(c,x_1,x_2) | c \in \{1,\ldots,m_D\}, x_i \in \mathbb{S}^{(i)}_g, i \in \{1,2\}\} \times \{1,2\}$, where $\mathbb{S}^{(i)}_g = \{1, 1+1/g, 1+2/g, \ldots, m^{(i)}_D\}$

and the quantity $O_{td}^{g,(i)}$ evolves as follows. Let

$$x = (1-\beta_i)O_{(t-1)d}^{g,(i)} + \beta_i D^{(i)},$$

then $O_{td}^{g,(i)} = x$ if $x \in \mathbb{S}^{(i)}$, otherwise it equals $\lceil x \rceil_g$ with probability $g(x - \lfloor x \rfloor_g)$, or $\lfloor x \rfloor_g$ with probability $g(\lceil x \rceil_g - x)$, where $\lceil x \rceil_g$ ($\lfloor x \rfloor_g$) rounds up (down) to the nearest element in $\mathbb{S}_g^{(i)}$. Notice, by induction, we have $E[O_{td}^{g,(i)}] = E[D^{(i)}]$. Using this probabilistic rounding, we can easily compute the conditional probabilities $P[O_{td}^{g,(i)} = q' | O_{(t-1)d}^{g,(i)} = q]$, which we denote as $p_g^{(i)}(q,q')$, from $D^{(i)}$ (see [4, Eqn. (12)] for details).

The transition matrix P_g of the Markov chain $(B_n, C_n, O_{a(n)}^{(1)}, O_{a(n)}^{(2)}, S_n)_{n\geq 0}$ is a GI/M/1-type Markov chain [8] with the following structure,

$$P_g = \begin{bmatrix} A_d & A_0 & & & \\ \vdots & & \ddots & & \\ A_d & & & A_0 & \\ & A_d & & & A_0 \\ & & \ddots & & & \ddots \end{bmatrix},$$

as B_n either increases by one if the same joint order remains in service, or decreases by $d-1$ if a joint order is completed. Hence, there are d occurrences of A_d on the first block column. The size m of the square matrices A_0 and A_d is $2m_D m_g$, with $m_g = \prod_{i=1}^{2}(m_D^{(i)} g - g + 1)$, which is typically such that we cannot store the matrices A_0 and A_d in memory. Although we can eliminate close to 50% of the states by removing the transient states with $C_n > \lceil O_{a(n)}^{(1)} \rceil + \lceil O_{a(n)}^{(2)} \rceil$, the size m remains problematic and this would slow down the numerical solution method presented in Section 5. A more detailed discussion of the structure of A_0 and A_d is given in Section 5.1.

4.2 One retailer smooths

Assume without loss of generality that retailer one smooths, while retailer two does not, i.e., $\beta_1 < 1$ and $\beta_2 = 1$. In this case we can also rely on the Markov chain defined above, but now there is no longer a need to keep track of $O_{a(n)}^{g,(2)}$, as the orders of retailer two are distributed according to $D^{(2)}$. This not only simplifies the transition probabilities, but also considerably reduces the time and memory requirements of the numerical solution method introduced in Section 5. Although storing the matrices A_0 and A_d in memory may no longer be problematic, a numerical approach as presented in the next section outperforms the more traditional approach that relies on computing the rate matrix R [8] by a considerable margin.

5 Numerical solution

The objective of this section is to introduce a numerical method to compute the steady state distribution of the Markov chain introduced in Section 4.1 by avoiding the need to store the matrices A_0 and A_d.

5.1 Fast multiplication

In order to multiply the vector $x = (x_0, x_1, \ldots)$ with P_g, where x_i is a length $m = 2m_D m_g$ vector, without storing the matrices A_0 or A_d, we will write P_g as the sum of $P_g^{(0)} + P_g^{(d)} =$

$$\begin{bmatrix} A_0 & & & \\ & \ddots & & \\ & & A_0 & \\ & & & \ddots \end{bmatrix} + \begin{bmatrix} A_d & & \\ \vdots & & \\ A_d & & \\ & \ddots & \end{bmatrix},$$

and compute xP_g as $xP_g^{(0)} + xP_g^{(d)}$. To express the time complexity of these multiplications, assume $x_i = 0$ for $i \geq n$ for some n (as will be the case in the next subsection).

The matrix A_0 corresponds to the case where the same joint order remains in service, meaning C_n either remains the same or decreases by one. Due to the order of the random variables, the matrix A_0 is a bi-diagonal block Toeplitz matrix, with blocks of size $2m_g$. The block appearing on the main diagonal equals $I \otimes U$, as the production of the same item continues in this case. The block below the main diagonal is $I \otimes u^*\alpha$, as the item is finished, but at least one item of the joint order still needs to be produced. Hence,

$$A_0 = \begin{bmatrix} I \otimes U & & & \\ I \otimes u^*\alpha & I \otimes U & & \\ & \ddots & \ddots & \\ & & I \otimes u^*\alpha & I \otimes U \end{bmatrix},$$

where I is the size m_g unity matrix and we have m_D blocks $I \otimes U$ on the main diagonal. As the PH representation is of order 2 (even in case of low variability), we can multiply x with $P_g^{(0)}$ in $O(mn)$ time.

When multiplying with A_d, we first argue that A_d can be written as

$$A_d = (e_1 \otimes (I \otimes u^*))(W_1 \otimes W_2)(Y \otimes \alpha),$$

where e_1 is a size m_D column vector which equals one in its first entry and zero elsewhere, W_i is a square matrix of size $m_D^{(i)} g - g + 1$ and Y is a $m_g \times m_g m_D$ matrix.

To understand this decomposition we split the transition in four steps. First, a service completion of an order must occur, meaning C_n must equal one and the item in service must be completed. Thus, the matrix $(e_1 \otimes (I \otimes u^*)$ describes this step. Next, in step 2, we determine the new order size for each retailer based on the previous order size (using the granularity g). Let the (q,q')-th entry of W_i equal $p_g^{(i)}(q,q')$ (as defined in Section 4.1), for $i = 1,2$. As each retailer determines its next order size independently, $W_1 \otimes W_2$ captures step 2. To complete the transition we need to determine the joint *integer* order size given the individual *granularity g* order sizes of both retailers (in step 3) and the initial service phase of the first item part of the joint order (in step 4). Step 4 is clearly determined by α, while step 3 corresponds to the matrix Y. A row of the matrix Y contains either 1, 2 or 4 non-zero entries (depending on whether the row corresponds to a case where both, one or none of the granularity g orders are integers).

Thus, when multiplying $x = (x_0, x_1, \ldots)$ with $P_g^{(d)}$, each of the vectors x_i is first reduced to a length m_g vector in $O(nm_g)$ time, because of $(e_1 \otimes (I \otimes u^*))$. A multiplication with $W_1 \otimes W_2$ is done in two steps. First we multiply with $(I \otimes W_2)$, which can be trivially done in $O((m_D^{(2)}g)^2 m_D^{(1)}g) = O(m_g m_D^{(2)}g)$ for each vector, followed by the multiplication with $(W_1 \otimes I)$. This latter multiplication can be rewritten as a multiplication with $(I \otimes W_1)$ using the shuffle algorithm[5]. Hence, it can also be done in $O(m_g m_D^{(1)}g)$. Due to its sparse structure, a multiplication with Y can be implemented in $O(m_g)$ time. In conclusion, the overall time required to multiply x with $P_g^{(d)}$ can be written as $O(nm_g(m_D^{(1)} + m_D^{(2)})g) = O(nmg)$ and the time needed to multiply x with P_g is therefore also $O(nmg)$. In practice, for g small, the multiplication with $P_g^{(0)}$ is more time demanding than the multiplication with $P_g^{(d)}$ and a considerable percentage of the time is also spent on allocating memory.

5.2 The power method, the Gauss-Seidel iteration and GMRES

To determine the steady state probability vector of the transition matrix P_g we rely on the fast matrix multiplication between a vector x and P_g introduced above.

When combined with the power method, we basically start with some initial vector $x(0)$ and define $x(k+1) = x(k)P_g$ until the infinity norm of $x(k+1) - x(k)$ is smaller than some predefined ε_1 (e.g., $\varepsilon_1 = 10^{-8}$). If we start from an empty system, $x(0)$ has only one nonzero component $x_0(0)$ of length m and $x(k)$ has $k+1$ nonzero components $x_0(k)$ to $x_k(k)$. Whenever some of the last components are smaller than some predefined ε_2, we reduce the length of $x(k)$ (by adding these components to the last component larger than ε_2). Notice, introducing ε_2 is not exactly equivalent to a truncation of the Markov chain at some predefined level N. Instead we dynamically truncate the vector x during the computation and its length may still vary over time. The impact of both ε_1 used by the stopping criteria and ε_2 used by the dynamic truncation will be examined in Section 7.1. Both these parameters will be used in a similar manner for the other iterative schemes as well.

When applying the *forward* Gauss-Seidel iteration [9], we compute $x(k+1)$ from $x(k)$ by solving the linear system

$$x(k+1)(I - P_g^{(0)}) = x(k)P_g^{(d)},$$

which can be done efficiently using forward substitution as $(I - P_g^{(0)})$ is upper triangular. If x is an arbitrary stochastic vector, we initialize $x(0)$ such that it solves $x(0)(I - P_g^{(0)}) = x$. As indicated in [9], this Gauss-Seidel iteration is equivalent to a preconditioned power method if we use $(I - P_g^{(0)})$ as the preconditioning matrix M. Notice, we can benefit from the fast multiplications discussed in the previous section when computing $x(k)P_g^{(d)}$ as well as during the forward substitution phase.

The GMRES method [10] computes an approximate solution of the linear system $(I - P'_g)x = 0$, by finding a vector $x(1)$ that minimizes $\left\|(I - P'_g)x\right\|_2$ over the set $x(0) + \mathscr{K}(I - P'_g, r_0, n)$. Here r_0 is the residual of an initial solution $x(0)$: $r_0 = -(I - P'_g)x(0)$; $\mathscr{K}(I - P'_g, r_0, n)$ is the Krylov subspace, i.e., the subspace spanned by the vectors $\{r_0, (I - P'_g)r_0, \ldots, (I - P'_g)^{n-1}r_0\}$; and n is the dimension of the Krylov subspace [6]. To do this GMRES relies on the Arnoldi iteration to find an orthonormal basis V_n for the Krylov subspace, such that $V'_n(I - P'_g)V_n = H_n$, where H_n is an upper Hessenberg matrix of size n. Once V_n and H_n have been obtained, a vector y_n is found such that $J(y) = \left\|\beta e_1 - \tilde{H}_n y\right\|_2$ is minimized. Here β is the 2-norm of r_0, e_1 is the first column of the identity matrix, and $\tilde{H}_n$ is an $(n+1) \times n$ matrix whose first n rows are identical to H_n, and its last row has one nonzero element that also results from the Arnoldi iteration. A new approximate solution $x(1)$ is computed as $x(1) = x(0) + V_n y_n$. The process is then repeated with $x(1)$ as $x(0)$ until the difference between two consecutive solutions is less than some predefined ε. Although this algorithm is defined to solve linear systems of the type $Ax = b$, with A nonsingular, it can also be used to solve homogeneous systems with A singular, as is the case with Markov chains [11].

The GMRES algorithm also benefits from the fast multiplication discussed in the previous section. To find the residual r_0 at each iteration we need to compute the product $(I - P'_g)x(0) = x(0) - P'_g x(0)$. Also, for the Arnoldi process we need to determine the vectors $v_j = (I - P'_g)^{j-1}r_0$, which are computed iteratively, and require $n-1$ products of the type $(I - P'_g)v_{j-1} = v_{j-1} - P'_g v_{j-1}$. As with the power method, when analyzing several scenarios we can use the final approximate solution of one scenario as the starting solution for the next one to speed up convergence.

6 The safety stock

The required safety stock SS_i for each retailer to guarantee a certain fill rate is one of the main performance measures of this supply chain problem. The derivation for the case where both retailers smooth is nearly identical to the one presented in [4] and is mainly included for reasons of completeness. As indicated in Section 3, computing

SS_i is equivalent to determining the base-stock S_i provided that we know the mean replenishment lead time $E[T_r]$ (which equals the floor of the production lead time T_p). The production lead time distribution T_p is easy to obtain from the steady state probability vector π of P_g as follows. First define the length $2m_g$ vectors $\pi_b(c)$ as the steady state probabilities of being in a state with $B_n = b$ and $C_n = c$. Then, the probability of having a production lead time of b slots equals

$$P[T_p = b] = \rho \pi_b(1)(e \otimes u^*)/(1/d)$$

for $b > 0$, where $\rho = 2(E[D^{(1)}] + E[D^{(2)}])/d$ is the load at the manufacturer and $1/d$ is the arrival rate of the joint orders.

The fill rate is defined as $1 - E[(-NS)^+]/E[D]$, where NS is the net stock (i.e., inventory on hand minus backlog) and $x^+ = \max\{0, x\}$. Hence, $E[(-NS)^+]$ is the expected number of backlogged items. Similar to [4, Section 5.1], we can show that

$$NS_i = S_i + \sum_{j=1}^{k} D^{(i)} + O_k^{(i)}/\beta, \tag{1}$$

where k is the age, expressed in periods, of the joint order in production at the manufacturer at the end of a period and this joint order contains $O_k^{(i)}$ items for retailer i, for $i = 1,2$. If $k = 0$, meaning the last order left the queue before the end of the period, $O_k^{(i)}$ is the number of items ordered by retailer i in the next joint order. Thus, the key step in determining the required base-stock value S_i, exists in computing the joint probabilities $p_{k,q}^{(i)}$ of having an order of age kd in service when a period ends and the order in service contains q items for retailer i, for $i = 1,2$, $k \geq 0$ and $q \in \{1, \ldots, m_D^{(i)}\}$.

These joint probabilities can be readily obtained from the steady state of the Markov chain introduced in Section 4.1 as

$$p_{k,q}^{(i)} = \rho d \pi_{kd}^{(i)}(q) e,$$

for $k > 0$, where $\pi_b^{(i)}(q)$ is the steady state vector for the states with $B_n = b$ and $O_{a(n)}^{g,(i)} = q$. For $k = 0$, we note that an order finds the queue empty upon arrival if the previous order had a lead time of at most $d - 1$, yielding

$$p_{0,q}^{(i)} = \rho d \sum_{b=1}^{d-1} \sum_{q_1, q_2, s} \pi_b(1, q_1, q_2, s) u_s^* p_g(q_i, q),$$

where $\pi_b(c, q_1, q_2, s)$ is the steady state probability of state (b, c, q_1, q_2, s).

If we wish to compute the joint probabilities $p_{k,q}^{(2)}$ from the Markov chain $(B_n, C_n, O_{a(n)}^{g,(1)}, S_n)_{n \geq 0}$ in case only the first retailer smooths, things are somewhat more involved when $k > 0$. For $k = 0$, we clearly have

Table 1 Accuracy and computation times of the power and Gauss-Seidel method for $\varepsilon_2 = 10^{-9}$

	Power			Gauss-Seidel		
ε_1	10^{-6}	10^{-7}	10^{-8}	10^{-6}	10^{-7}	10^{-8}
res. error	1.76E-5	1.68E-6	1.85E-7	2.14E-6	2.38E-7	2.56E-8
SS	0.64%	0.03%	0.01%	1.10%	0.17%	0.02%
$E[T_r]$	0.11%	0.02%	0.00%	0.63%	0.09%	0.01%
time (sec)	31	54	79	1.7	3.0	4.4
iter	804	1207	1636	21	34	49

$$p^{(2)}_{0,q} = P[T_p < d]P[D^{(2)} = q].$$

For $k > 0$, we start by computing $p_w(q_1,x)$, the probability that an order consisting of q_1 items for retailer 1 has a waiting time of $x > 0$ slots. As the waiting time x of an order with $x > 0$ equals the lead time of the previous order minus the inter-arrival time d, we find

$$p_w(q_1,x) = \frac{\rho d}{\pi(q)} \sum_{q,s} \pi_{x+d}(1,q,s)u^*_s p_g(q,q_1),$$

where $\pi_b(c,q,s)$ is the steady state probability of state (b,c,q,s) and $\pi(q)$ is the probability that an arbitrary order contains q items for retailer 1.

Next, we determine the probabilities $p_o(q_1,q_2,y)$ that an arbitrary joint order consists of q_i items for retailer i and its production time equals y time slots. These probabilities are readily obtained from $p_g(q,q')$ and (α,U). Then,

$$p_a(q_1,q_2,x) = \sum_{y\geq x} \frac{p_o(q_1,q_2,y)}{2(E[D^{(1)}]+E[D^{(2)}])},$$

is the probability that we find a joint order consisting of q_i items for retailer i in service at an arbitrary moment when the server is busy, while the service of this joint order started x time slots ago. Taking the convolution over x between $p_w(q_1,x)$ and $p_a(q_1,q_2,x)$ and summing over q_1, gives us the probability that the order in service has an age of x time slots and consists of q_2 items for retailer 2, given that we observe the system when the server is busy. From these probabilities the joint probabilities $p^{(2)}_{k,q}$ are readily found.

We can also compute the probabilities $p^{(2)}_{k,q}$ from the Markov chain in Section 4.1 by setting $\beta_2 = 1$, but this approach requires more time and considerably more memory. As required, the numerical experiments indicated a perfect agreement between both approaches.

Table 2 Accuracy and computation times of GMRES for $\varepsilon_2 = 10^{-9}$

	GMRES - n=1			GMRES - n=3			GMRES - n=5		
ε_1	10^{-6}	10^{-7}	10^{-8}	10^{-6}	10^{-7}	10^{-8}	10^{-6}	10^{-7}	10^{-8}
res. error	4.18E-5	5.31E-6	1.16E-7	2.79E-6	2.47E-7	2.48E-8	1.42E-6	1.33E-7	1.46E-8
SS	15.40%	8.01%	0.92%	9.70%	1.95%	0.29%	6.80%	1.23%	0.19%
$E[T_r]$	10.63%	4.70%	0.48%	6.86%	1.07%	0.14%	4.44%	0.64%	0.09%
time (sec)	15	34	105	21	64	290	38	170	446
iter	186	261	797	89	341	301	61	120	190

7 Numerical examples

In this section we illustrate the effect of smoothing on the performance of the production/inventory system. We focus on the safety stock as the main measures of performance, and consider various scenarios for the demand distribution, the load and the smoothing parameters β_1 and β_2. The required safety stock in all the numerical examples guarantees a fill rate of 0.98.

For the demand we consider three different distributions, let us call the three associated random variables X, Y and Z, respectively. X is defined as $X = 1 + \hat{X}$, where $\hat{X}$ is a Binomial distribution with parameters $N-1$ and $p = 1/2$. Thus, X takes vales on the set $\{1,\ldots,N\}$. The expected value and variance of X are $E[X] = (N+1)/2$ and $\mathrm{Var}(X) = (N-1)/4$. The second random variable Y is uniformly distributed between 1 and N, and its expected value and variance are $E[Y] = (N+1)/2$ and $\mathrm{Var}(Y) = (N^2-1)/12$. The last random variable is defined as $P(Z = k) = (1+\alpha)P(Y = k) - \alpha P(X = k)$, for $k = 1,\ldots,N$. As a result Z has a U-shaped probability mass function, with $E[Z] = (N+1)/2$ and $\mathrm{Var}(Z) = (N^2 - 1 + \alpha(N^2 - 3N + 2))/12$. Clearly, for Z to be a proper random variable, the value of α has to be such that $P(Z = k) \geq 0$ for all k. In our experiments we set $N = 10$, for which α can take values up to roughly 0.68. We choose $\alpha = 0.6$ to make Z highly variable. With this setup, $\mathrm{Var}(X) = 2.25$, $\mathrm{Var}(Y) = 8.25$ and $\mathrm{Var}(Z) = 8.25 + 6\alpha = 11.85$. Also, setting the maximum demand size to $N = 10$, the size of the square matrices A_0 and A_D ranges from 4000 (for $g = 1$) to 84640 (for $g = 5$).

As mentioned before, the mean production time is set equal to 2, and for the experiments in this section the standard deviation is also set to 2. The load is set by adjusting d, the number of slots between two orders placed by the retailers. In our setup we choose d from the set $\{40, 34, 29, 26\}$, which generate loads of roughly $\{0.55, 0.65, 0.76, 0.85\}$, respectively. We will start by looking at the case where both retailers use the same value of the smoothing parameters β_1 and β_2. Afterward we consider the case where these parameters may differ. However, before we generate any numerical results let us first evaluate the impact of discretizing the state space (that is, the impact of the granularity g) as well as the parameters ε_1 and ε_2 used in the stopping criteria and dynamic truncation of the state space, respectively.

7.1 Computation times and accuracy

We start by looking at the accuracy and computation times required to obtain the results in the paper with the power, Gauss-Seidel and GMRES methods when $g = 1$ (even though larger g values are needed for small β values as indicated below). Table 1 shows the residual-error of the steady state vector, that is, the norm of $xP_g - x$, as well as the accuracy of SS and $E[T_r]$ for both the power and Gauss-Seidel methods when compared against a solution obtained with $\varepsilon_1 = \varepsilon_2 = 10^{-14}$ (by the power method). The table also lists the computation times and the required number of iterations. Table 2 provides the same data for the GMRES method, where the size of the Krylov subspace was set equal to 1, 3 and 5. These results correspond to the example where the demand follows a Binomial distribution, the load $\rho = 0.85$ (which is the most demanding among the 4 loads considered), and both retailers smooth with $\beta_1 = \beta_2 = 0.8$. All the experiments were run on a PC with 4 cores at 2.93GHz and 4GB of RAM. We observe that, for the same ε_1, the Gauss-Seidel method is far superior to both the power method and GMRES, as it requires substantially less time and has a similar accuracy than the power method. This can be explained by the fact that the Markov chain characterized by P_g typically makes many consecutive upward transitions according to A_0 followed by an occasional downward jump using A_d.

The accuracy of GMRES is quite poor for larger ε_1 values and is far worse than the power or Gauss-Seidel method. As ε_1 decreases the difference in accuracy between GMRES and the other methods becomes smaller (and eventually negligible). GMRES is faster than the power method for $\varepsilon_1 = 10^{-6}$ and when n is one or three, but as indicated above, the accuracy of GMRES is poor in these cases. As stated in Section 5.2 the Gauss-Seidel method may be regarded as a preconditioned power method where the preconditioning matrix M is equal to $(I - P_g^{(0)})$. In principle we can use the same preconditioning for GMRES, which should improve the performance of GMRES significantly. However, as GMRES is typically inferior to the power method, it seems unlikely that we can do better than the Gauss-Seidel method using $(I - P_g^{(0)})$ as a preconditioning matrix.

Next, let us have a look at the impact of the granularity of g on the results for the Gauss-Seidel method only, as the other methods are too time consuming for larger g values. We let g vary from 1 to 5 for a load $\rho = 0.85$, while $\varepsilon_1 = 10^{-8}$ and $\varepsilon_2 = 10^{-9}$. Figure 1 depicts the required safety stock SS as a function of β for the three demand distributions discussed before. These results indicate that for β close to one, letting $g = 1$ suffices, however, for smaller β values setting $g = 1$ may lead to a serious overestimation of the required safety stock. Thus, in order to guarantee an acceptable accuracy for smaller β values, we have generated all the subsequent results with $g = 5$ (and $\varepsilon_1 = 10^{-8}$ and $\varepsilon_2 = 10^{-9}$).

Finally, we would like to mention that a significant amount of the computation time is devoted to allocating memory, due to the large sizes of the vectors, e.g., the size of the final vector x in Figure 1, for $\beta = 0.8$ and the binomial distribution, is 732000 (for $g = 1$) and 15065920 (for $g = 5$). Since GMRES computes n large

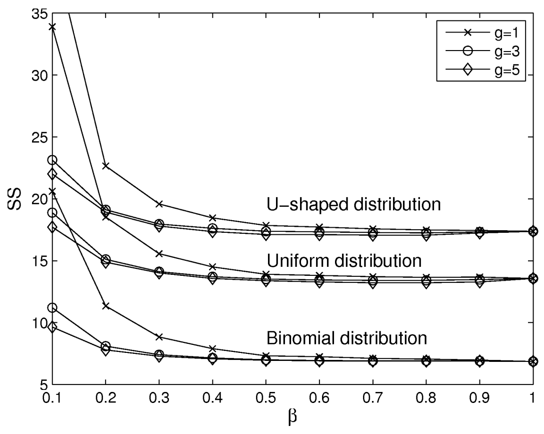

Fig. 1 Safety stock as a function of β

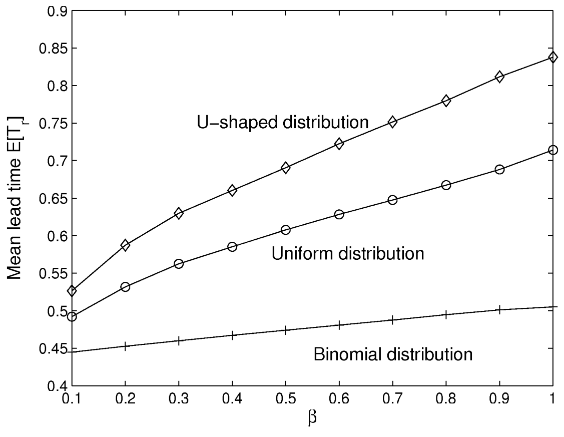

Fig. 2 Mean Lead Time $E[T_r]$ vs. $\beta = \beta_1 = \beta_2$ for $\rho = 0.85$

vectors, it is more significantly affected by the memory allocation delay. Also, the computation times of all the methods are highly influenced by the system parameters, especially by the load ρ and the variance of the demand and processing times. Larger values for these parameters imply longer computation times and larger memory requirements.

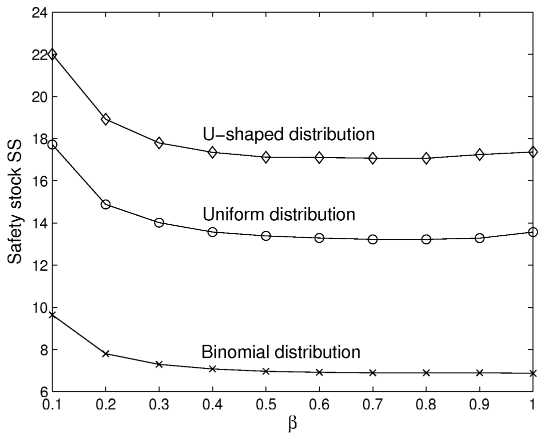

Fig. 3 Safety Stock SS vs. $\beta = \beta_1 = \beta_2$ for $\rho = 0.85$

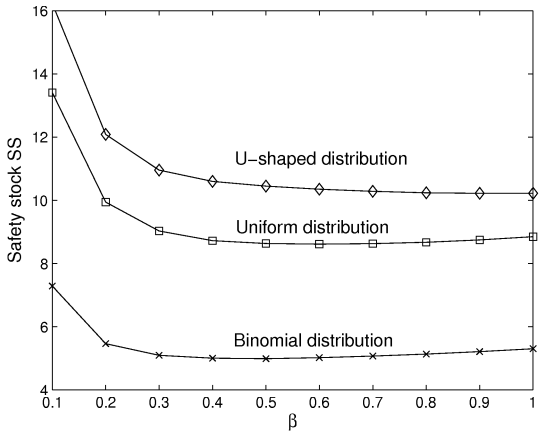

Fig. 4 Safety Stock SS vs. $\beta = \beta_1 = \beta_2$ for $\rho = 0.65$

7.2 Homogeneous smoothing

We start by looking at a system facing a load of $\rho = 0.85$, and we consider values of $\beta = \beta_1 = \beta_2$ in the set $\{0.1, 0.2, \ldots, 1\}$, and the three different demands described above. The results are included in Figure 2, where we observe that the mean replenishment lead time increases as a function of β, meaning both retailers benefit from smoothing with respect to the replenishment time. As expected, the lead time reduction increases with the variability of the demand distribution. This reduction in the lead time is key in understanding the effect of β on the safety stock.

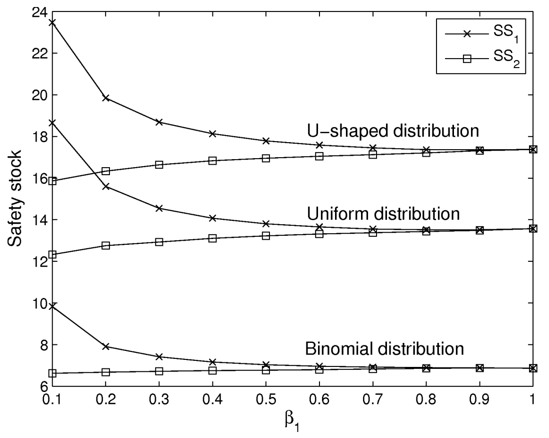

Fig. 5 Safety stock SS vs. β_1 - $\beta_2 = 1$ and $\rho = 0.85$

Figure 3 depicts the corresponding safety stock to guarantee a fill rate of 0.98. The results indicate that unless β is small, the required safety stock does not increase much as β decreases, meaning both retailers can perform a considerable amount of smoothing without the need to increase their SS much. Note, as β decreases the response of the retailer to a sudden increase in the demand tends to become slower, which intuitively should result in an increased SS. However, the decrease in the lead time (partially) compensates the slower response. When β becomes too small, the reduction in the lead time is insufficient to avoid a significant increase in the SS. Actually, when decreasing β, starting in $\beta = 1$, the SS initially even decreases a little in case of a more variable demand.

Similar results were obtained for lower load scenarios as well, the corresponding plot for an approximate load of $\rho = 0.65$ (i.e., for $d = 29$) is given in Figure 4. These results and insights are similar in nature to the single retailer case (see [4]).

7.3 Heterogeneous smoothing

We start by considering the scenario where only one retailer smooths, say retailer one. Thus, we assume that β_2 is fixed and equal to one, while β_1 changes. As expected, the mean lead time can be shown to decrease as β_1 decreases. Figure 5 depicts the safety stock of both retailers as a function of β_1 for $\rho = 0.85$. The results indicate that the safety stock SS_1 of retailer one behaves very similar as in the homogeneous case (it is a fraction larger to be precise). Thus, the retailer can still smooth his demand considerably without affecting his safety stock too much. The safety stock of the second retailer SS_2 on the other hand decreases slightly as β_1 decreases. This can be understood by noting that the second retailer also benefits

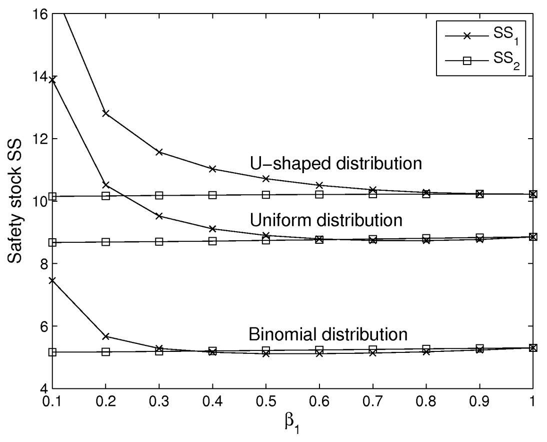

Fig. 6 Safety stock SS vs. β_1 - $\beta_2 = 1$ and $\rho = 0.65$

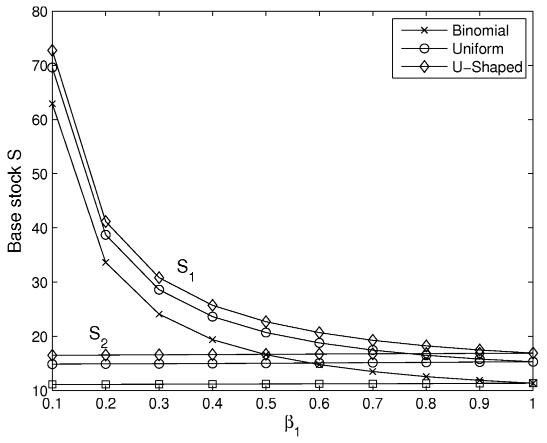

Fig. 7 Base stock S vs. β_1 - $\beta_2 = 1$ and $\rho = 0.65$

from the reduced lead time, while he is more reactive to a sudden increase in the demand than retailer one (as $\beta_2 = 1$).

In Figure 6 we consider the same example, but with a reduced load $\rho = 0.65$. In this case we observe a more remarkable result: the safety stock SS_1 of retailer one first decreases and is even below the safety stock SS_2 of retailer two for some β_1 values. This may seem counterintuitive at first as both retailers benefit from the reduction in lead time, while the second is still more reactive. To understand this, consider Eqn. (1) for the net stock distribution NS of retailer i. The last term $O_k^{(i)}/\beta$ is clearly larger on average for retailer one, but $O_k^{(1)}$ is less variable than $O_k^{(2)}$ as the orders of retailer one are smoothed. Thus, if S_1 is chosen larger than S_2 to compen-

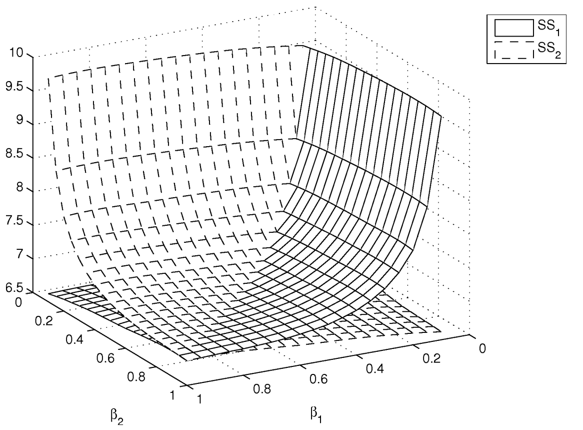

Fig. 8 Safety stock SS_1 and SS_2 vs. β_1 and $\beta_2 = 1$ - $\rho = 0.85$, Binomial demand distribution

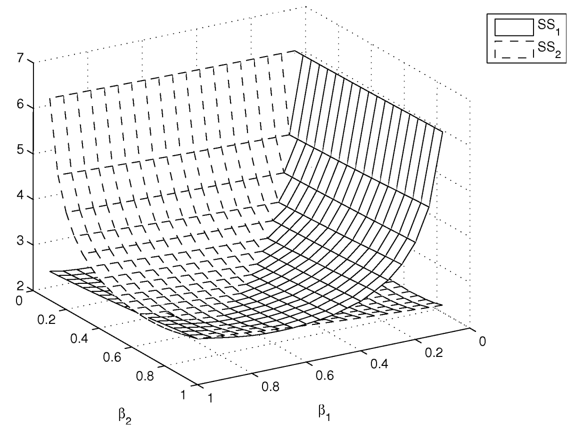

Fig. 9 Safety stock SS_1 and SS_2 vs. β_1 and $\beta_2 = 1$ - $\rho = 0.55$, Binomial demand distribution

sate for the larger average of $O_k^{(i)}/\beta$, the lower variability of $O_k^{(i)}$ might indeed result in a less variable net stock (for β sufficiently close to one) and therefore in a smaller safety stock as well. Figure 7 shows that this is exactly what happens: S_1 decreases, while S_2 increases as a function of β_1.

If we consider the selection of β_1 and β_2 in a game theoretic setting, where the objective of retailer i exists in minimizing SS_i, it is already clear from Figure 6 that $(\beta_1, \beta_2) = (1,1)$ is not always a Nash equilibrium[1], as retailer one can decrease his safety stock SS_1 by selecting a β_1 less than one. Figures 8 and 9 depict the safety

[1] A common strategy is called a Nash equilibrium if neither player can improve his objective by deviating from the common strategy.

stock of both retailers for $\beta_1, \beta_2 \in \{0.1, 0.15, 0.2 \ldots, 1\}$ when the demand follows a Binomial distribution and the load equals 0.85 and 0.55, respectively. These results indicate that there exists a unique Nash equilibrium (β_1, β_2) in these scenarios. More specifically, for $\rho = 0.85$ and 0.55 the Nash equilibrium is located in $(\beta_1, \beta_2) = (1, 1)$ and $(0.75, 0.75)$, respectively. Numerical experiments not depicted here indicated that there is also a unique Nash equilibrium (β_1, β_2) when the load equals 0.65 and 0.76 (being $(\beta_1, \beta_2) = (0.5, 0.5)$ and $(0.85, 0.85)$, respectively).

8 Further discussion

The main focus of this paper has been the analysis of a supply chain with a single manufacturer and two retailers. We model this system as a GI/M/1-type Markov chain with blocks whose size is large enough to make the computation of the (dense) matrix R, with traditional algorithms, infeasible. To overcome this issue, we propose to use numerical methods, such as the power method, Gauss-Seidel and GMRES, to compute the stationary probability vector of the chain. As these methods rely heavily on vector-matrix multiplications, we exploit the structure of the transition-matrix blocks to performs these multiplications efficiently. Clearly, the same approach can be used to analyze other systems modeled as a structured Markov chain, the blocks of which are large and possess an inner structure that can be exploited to perform the vector-matrix multiplications. In this section, we conclude the paper with two fairly arbitrary examples of other systems that can be analyzed with the approach used in the paper.

8.1 An edge router

Edge routers provide access to core networks from service providers as well as carrier networks. For instance, edge routers are located at the edge of an Internet Service Provider (ISP) network, connecting multiple users to the ISP's core network. Therefore, the edge router typically has multiple low-speed interfaces (connected to the users) and one (or a few) high-speed interfaces (connected to the core network). Given the difference in transmission rates, the router may collect multiple, say b, packets arriving from the low-speed interfaces into a single packet to forward through the high-speed channel. Assuming the router always collects b user-generated packets into one packet for high-speed transmission, we can model the number of user-generated packets in the system (buffered and in transmission) as a (continuous-time) GI/M/1-type Markov chain with only three nonzero blocks.

Let $N(t)$, $S(t)$ and $J(t)$ be the number of packets, the service phase of the packet in transmission and the phase of the arrival process at time t, respectively. The service time distribution is a continuous PH distribution with parameters (α, T) and the packet arrival process is a markovian arrival process (MAP) with parameters (D_0,

D_1) [7]. The process $X(t) = \{(N(t),\, S(t),\, J(t))\,,\, t \geq 0\}$ is thus a continuous-time Markov chain with generator matrix

$$Q = \begin{bmatrix} B_1 & B_0 & & & & & & \\ & A_1 & A_0 & & & & & \\ & & A_1 & A_0 & & & & \\ & & & \ddots & \ddots & & & \\ B_{b+1} & & & & A_1 & A_0 & & \\ & A_{b+1} & & & & A_1 & A_0 & \\ & & A_{b+1} & & & & A_1 & A_0 \\ & & & \ddots & & & \ddots & \ddots \end{bmatrix},$$

where $A_0 = D_1 \otimes I, A_1 = D_0 \oplus T, A_{b+1} = I \otimes t\alpha, t = -Te$ and $\oplus$ stands for Kronecker sum [7]. Notice that a packet in service is actually a bundle of b user-generated packets. Assuming the transmission time of the latter packets follows a PH distribution with parameters (β, S), the parameters α and T are given by

$$\alpha = \begin{bmatrix} \beta \; 0 \ldots \; 0 \end{bmatrix}, \text{ and } T = \begin{bmatrix} S & s\beta & & & \\ & S & s\beta & & \\ & & \ddots & \ddots & \\ & & & S & s\beta \\ & & & & S \end{bmatrix},$$

where $s = -Se$. Letting m_s and m_a be the size of the matrices S and D_0, respectively, the block size is $m = bm_s m_a$.

As mentioned above, the edge router receives packets from many, say n, low-speed interfaces. If the traffic incoming through interface j is modeled as a MAP with parameters $(C_0^j,\, C_1^j)$, the total incoming traffic is the superposition of these n MAPs. Thus, D_0 and D_1 are given by

$$D_0 = \oplus_{j=1}^n C_0^j \text{ and } D_1 = \oplus_{j=1}^n C_1^j.$$

If the size of each of the matrices C_0^j is m_u, the block size is $m = bm_s m_u^n$. As a result, the block size grows linearly with b, the number of user-generated packets per forwarded packet, and exponentially with n, the number the sources. It is clear then that the block size can be very large for rather limited values of b and, particularly, n. For instance, with $m_s = m_u = 2$, $b = 10$ and $n = 16$, the block size is over a million. This model is therefore well-suited to be analyzed with the approach proposed in this paper since, in addition to a large block size, the number of nonzero blocks is small and the blocks have a structure that can be exploited to perform the vector-matrix multiplications efficiently.

8.2 FS-ALOHA++

The FS-ALOHA++ algorithm is a contention resolution algorithm used for dynamic bandwidth allocation [12]. This algorithm operates on a time-division multiple access (TDMA) channel that consists of fixed length frames. Each frame contains, among others, $T = S + N$ minislots used to support the contention channel. When a user wants to transmit new data, it will send a request packet on the contention channel by selecting one of the first S minislots at random. If none of the other users transmit in the same minislot, the user is successful. All the users that were involved in a collision (in one of the first S minislots) on the other hand form a transmission set (TS). Hence, in each frame either one or zero TSs are formed. If a TS is formed it joins the back of a (distributed) FIFO queue and is called a backlogged TS.

The backlogged TSs are served, in groups of $K \geq 1$, using ALOHA on the last N minislots, that is, all the users part of the first K backlogged TSs select one of the last N minislots at random. Users that are successful leave the contention channel, the ones involved in a collision retransmit in the next frame in one of the last N minislots. This procedure is repeated until the last N minislots are collision free. As soon as this occurs the next set of K TSs can make use of the last N minislots (if the queue contains $i < K$ TSs, only i TSs are served simultaneously).

Under the assumption that new requests form a Poisson process, one can analyze the FS-ALOHA++ algorithm (with parameters S, N and K) by means of a GI/M/1-type Markov chain with a generalized boundary condition. This is achieved by keeping track of the number of backlogged TSs $N(t)$ at the start of frame t and the number of users $S(t)$ that will make use of the last N minislots in frame t (see [12] for more details). As at most one TS can be added to the back of the queue during a frame and K TSs may start service, one finds that the number of TSs may either increase by one, remain fixed, decrease by $K-1$ or decrease by K (provided that at least K TSs are backlogged). Thus, if $N(t)$ represents the level of the Markov chain and $S(t)$ the phase, one finds that only the matrices A_0, A_1, A_K and A_{K+1} differ from zero (we do not discuss the boundary matrices here). Further, as the probability of having i users in a TS decreases quickly with i (due to the Poisson arrivals), we can easily truncate the value of $S(t)$, the number of users that are part of K TSs, by some s_{max}.

It is not hard to see that the time needed to serve a group of K TSs can be represented by a phase-type distribution with an order $s_{max} - 1$ representation (α_N, T_N). Further, if we denote p_S as the probability that a TS is formed in the first S minislots of a frame, we find that

$$A_0 = p_S T_N, \;\; A_1 = (1 - p_S) T_N, \;\; A_K = p_S T_N^* \alpha_N \;\; A_{K+1} = (1 - p_S) T_N^* \alpha_N,$$

where $T_N^* = e - T_N e$. For details on how to compute p_S, α_N and T_N we refer to [12]. As values for s_{max} equal to 20 typically guarantee a very small truncation error, one can easily compute the R matrix of this chain using traditional methods.

However, suppose we which to modify FS-ALOHA such that the last N minislots are partitioned into M subsets of each N' slots (with $N = MN'$), such that up to M

groups of K TSs can be served simultaneously. More specifically, for each of the M subsets that becomes collision free during a frame, we take a group of K TSs from the queue and serve this group using ALOHA on the N' slots of the subset. In other words, we replace the single server queue with batch service and service time (α_N, T_N) by M batch servers with service time distribution $(\alpha_{N'}, T_{N'})$, where the order of the representation $(\alpha_{N'}, T_{N'})$ is also $s_{max} - 1$.

In this case, we can still obtain a GI/M/1-type Markov chain in a similar manner, but the phase has to maintain the state of *each* of the M servers, which implies that the block size grows very quickly with M and exceeds a few thousand even for $M =$ 3 or 4. Further, as several TSs may become collision free in a frame, up to M groups of K TSs may be removed from the queue. This implies that the block matrices A_{iK} and A_{iK+1} will defer from zero, for $i = 0, \ldots, M$. Hence, in this case the traditional approach of computing R to obtain the steady state is no longer feasible, but the approach taken in this paper still applies as the block matrices have a Kronecker product form.

Acknowledgements The first author was supported by the FWO G.0333.10N project entitled "The study of a two-echelon integrated production/inventory system solved by means of matrix analytic models".

References

1. Bini, D.A., Meini, B., Steffé, S., Van Houdt, B.: Structured Markov chains solver: algorithms. In: SMCtools Workshop. ACM Press, Pisa, Italy (2006)
2. Bobbio, A., Horváth, A., Telek, M.: The scale factor: a new degree of freedom in phase type approximation. Performance Evaluation **56**, 121–144 (2004)
3. Boute, R., Lambrecht, M., Van Houdt, B.: Performance evaluation of a production/inventory system with periodic review and endogeneous lead times. Naval Research Logistics **54**, 462–473 (2007)
4. Boute, R.N., Disney, S.M., Lambrecht, M.R., Van Houdt, B.: An integrated production and inventory model to dampen upstream demand variability in the supply chain. European Journal of Operational Research **178**, 121–142 (2007)
5. Fernandes, P., Plateau, B., Stewart, W.: Efficient descriptor-vector multiplications in stochastic automata networks. J. ACM **45**, 381–414 (1998)
6. Golub, G.H., Van Loan, C.: Matrix Computations. The Johns Hopkins University Press (1996)
7. Latouche, G., Ramaswami, V.: Introduction to Matrix Analytic Methods in Stochastic Modeling. ASA-SIAM Series on Statistics and Applied Probability. SIAM, Philadelphia, PA (1999)
8. Neuts, M.F.: Matrix-Geometric Solutions in Stochastic Models. The John Hopkins University Press, Baltimore (1981)
9. Philippe, B., Saad, Y., Stewart, W.: Numerical methods in Markov chain modeling. Operations Research **40**, 1156–1179 (1992)
10. Saad, Y., Schultz, M.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
11. Stewart, W.: Introduction to the numerical solution of Markov chains. Princeton University Press (1994)
12. Vázquez Cortizo, D., García, J., Blondia, C.: FS-ALOHA++, a collision resolution algorithm with QoS support for the contention channel in multiservice wireless LANs. In: Proc. of IEEE Globecom (1999)