EURANDOM PREPRINT SERIES
2010-014

APPROXIMATE ANALYSIS OF SINGLE-SERVER TANDEM QUEUES WITH FINITE BUFFERS

R. Bierbooms, I. Adan, M. van Vuuren
ISSN 1389-2355

APPROXIMATE ANALYSIS OF SINGLE-SERVER TANDEM QUEUES WITH FINITE BUFFERS

Remco Bierbooms, Ivo J.B.F. Adan, and Marcel van Vuuren
Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
E-mail: r.bierbooms@tue.nl, i.j.b.f.adan@tue.nl, vanvuuren@cqm.nl

Abstract: *In this paper we study single-server tandem queues with general service times and finite buffers. Jobs are served according to the Blocking-After-Service protocol. To approximately determine the throughput and mean sojourn time, we decompose the tandem queue into single-buffer subsystems, the service times of which include starvation and blocking, and then iteratively estimate the unknown parameters of the service times of each subsystem. The crucial feature of this approach is that in each subsystem successive service times are no longer assumed to be independent, but a successful attempt is made to include dependencies due to blocking by employing the concept of Markovian service processes. An extensive numerical study shows that this approach produces very accurate estimates for the throughput and mean sojourn time, outperforming existing methods, especially for longer tandem queues and for tandem queues with service times with a high variability.*

Keywords: *blocking, decomposition, finite buffers, flow lines, Markovian arrival process, matrix-analytic.*

1 Introduction

This subject of this paper is the approximative analysis of single-server tandem queues with general service times and finite buffers. The blocking protocol is Blocking-After-Service (BAS): if the downstream buffer is full upon service completion the server is blocked and has to wait until space becomes available before starting to serve the next job (if there is any). Networks of queues (and in particular, tandem queues) with blocking, have been extensively investigated in the literature; see e.g. [1, 7, 8, 9]. In most cases, however, queueing networks with finite buffers are analytically intractable and therefore the majority of the literature is devoted to approximate analytical investigations. The approximation developed in this paper is based on decomposition, following the pioneering work of [2]: the tandem queue is decomposed into single-buffer subsystems, the parameters of which are determined iteratively. In each subsystem, the "actual" service time, starvation and blocking are aggregated in a single service time, and these aggregate service times are typically assumed to be independent. However, these aggregate service times are clearly not independent, and especially in longer tandem queues with small buffers and in tandem queues with highly variable service times, these dependencies may have a strong impact on the performance. In this paper an approach is proposed to include such dependencies in the aggregate service times.

The model considered in the current paper is a tandem queue L consisting of N servers and $N-1$ buffers in between. The servers (or machines) are labeled M_i, $i = 0, 1, ..., N$. The first server M_0 acts as a source for the tandem queue, i.e., there is always a new job available for servicing. The service times of server M_i are independent and identically distributed, and they are also independent of the service times of the other servers; S_i denotes the generic service time of server M_i, with rate μ_i and squared coefficient of variation $c^2_{S_i}$. The buffers are labeled B_i and the size of buffer B_i is b_i (i.e., b_i jobs can be stored in B_i). We assume that each server employs the BAS blocking protocol. An example of a tandem queue with 4 machines is illustrated in Figure 1.

The approximation is based on decomposition of the tandem queue into subsystems, each one consisting of a single buffer. To take into account the relation of buffer B_i with the upstream and downstream

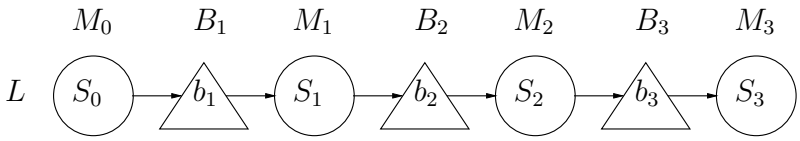

Figure 1: A tandem queue with 4 servers.

part of the tandem queue, the service times of server M_{i-1} in front of buffer B_i and server M_i after buffer B_i are adapted by aggregating the "real" service times S_{i-1} and possible starvation of M_{i-1} before service, and S_i and possible blocking of M_i after service. The aggregate service processes of M_{i-1} and M_i are described by employing the concept of Markovian service processes, the parameters of which are determined iteratively. It is important to note that Markovian service processes can be used to describe dependencies between *successive* service times. Although decomposition techniques for single-server queueing networks have also been widely used in the literature, see e.g. [2, 4, 8, 14, 3, 13], the distinguishing feature of the current approximation is the inclusion of dependencies between successive (aggregate) service times by employing Markovian service processes.

The paper is organized as follows. In Section 2 we describe the decomposition of the tandem queue in subsystems; the service processes of each subsystem are explained in detail in Sections 3 and 4. Section 4 presents the iterative algorithm. Numerical results can be found in Section 5 and they are compared to simulation and other approximation methods. Finally, Section 6 contains some concluding remarks and gives suggestions for further research.

2 Decomposition

The original tandem queue L is decomposed into $N-1$ subsystems $L_1, L_2, \ldots, L_{N-1}$. Subsystem L_i consists of buffer B_i of size b_i , an arrival server in front of the buffer, and a departure server after the buffer. Figure 2 displays the decomposition of line L of Figure 1.

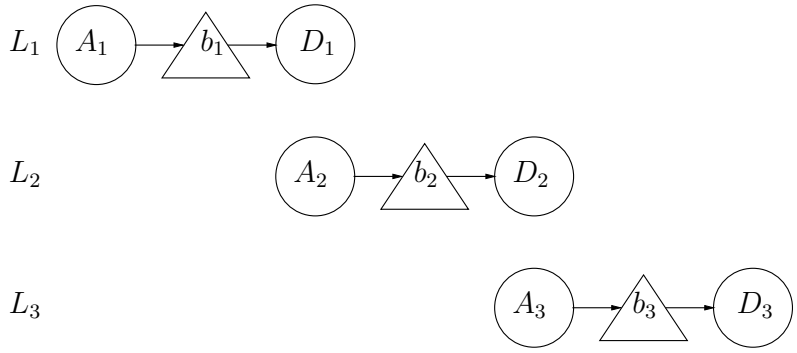

Figure 2: Decomposition of the tandem queue of Figure 1 into 3 subsystems.

The arrival server of subsystem L_i is, of course, server M_{i-1}. To account for the connection with the upstream part of L, its service time, however, is different from S_{i-1}. The random variable A_i denotes the service time of the arrival server in subsystem L_i. This random variable aggregates S_{i-1} and possible starvation of M_{i-1} because of an empty upstream buffer B_{i-1}. Accordingly, the random variable D_i represents the service time of the departure server in subsystem L_i; it aggregates S_i and possible blocking of M_i after service completion, because the downstream buffer B_{i+1} is full. Note that successive service times A_i and D_i are *not independent*: a long A_i indicating starvation is more likely to be followed by again a long one, and the same holds for a long D_i indicating blocking. In the next section we will explain in more detail the modeling of the service processes of the arrival and departure server of subsystem L_i.

3 Service process of the arrival server

In this section we model the service process of arrival server M_{i-1} of subsystem L_i. Note that an arrival in buffer B_i, i.e., a job being served by M_{i-1} moves to buffer B_i when space becomes available, corresponds to a departure from the upstream subsystem L_{i-1}. Just after this departure, two situations may occur: subsystem L_{i-1} is empty with probability q^e_{i-1}, or it is not empty with probability $1 - q^e_{i-1}$ (where, by convention, we do not count the job at M_{i-2} as being in L_{i-1}, if there is any). In the former situation, M_{i-1} has to wait for a residual service time of arrival server M_{i-2} of subsystem L_{i-1}, denoted as RA_{i-1}, before the actual service S_{i-1} can start. In the latter situation, the actual service S_{i-1} can start immediately. Hence, we have

$$A_i = \begin{cases} RA_{i-1} + S_{i-1} & \text{with probability } q^e_{i-1}, \\ S_{i-1} & \text{otherwise.} \end{cases} \tag{1}$$

The determination of RA_{i-1} and q^e_{i-1} is deferred to Section 5. As an approximation, we will act as if the service times A_i are independent and identically distributed, thus ignoring dependencies between successive service times A_i.

4 Service process of the departure server

This section is devoted to a detailed description of the service process of departure server M_i. To describe D_i we take into account the occupation of the last position in buffer B_{i+1} (or server M_{i+1} if $b_{i+1} = 0$). A job served by M_i may encounter three situations in downstream subsystem L_{i+1} on departure from L_i, or equivalently, *on arrival at* L_{i+1}; see Figure 3.

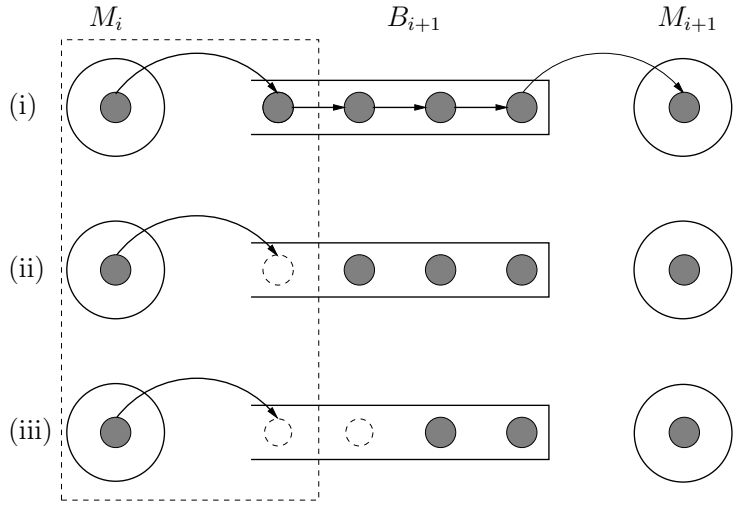

Figure 3: Possible situations in downstream subsystem L_{i+1} encountered at a departure from L_i.

(i) The arrival is triggered by a service completion of departure server M_{i+1} of L_{i+1}, i.e., server M_i was *blocked* because the last position in B_{i+1} was occupied, and waiting for M_{i+1} to complete service. Then the next service of M_i (if there is one) and M_{i+1} start simultaneously and buffer B_{i+1} is full. We denote the time elapsing till the next service completion of departure server M_{i+1} by D^b_{i+1}, which is, of course, equal to the time the last position in B_{i+1} will be occupied before it becomes available again. Hence, in this situation, D_i is equal to the maximum of S_i and D^b_{i+1}, if M_i can immediately start with the next service. Otherwise, if M_i is starved just after the departure, D_i is equal to the maximum of S_i and the residual time of D^b_{i+1} at the service start of M_i.

(ii) Just before the arrival there is only one position left in buffer B_{i+1}. So, right after this arrival, B_{i+1} is full. Now we denote the time elapsing till the next service completion of departure server M_{i+1} by D_{i+1}^f, which is again the time the last position in B_{i+1} will stay occupied. Thus D_i is equal to the maximum of S_i and the residual time D_{i+1}^f at the start of M_i.

(iii) Finally, when neither of the above situations occurs, the arrival does not fill up buffer B_{i+1}, because there are at least two positions available in B_{i+1}. Hence, the last position in B_{i+1} stays empty and D_i is equal to S_i.

Now we are not going to act as if the probability that the service of M_i starts in either of the three situations is independent of the past. This would imply that the service times D_i are independent. Instead, we are going to introduce transition probabilities between the above three situations, i.e., the service process of departure server M_i will be described by a Markov chain.

If service of M_i starts in situation (i), and M_i finishes before M_{i+1} (i.e., $S_i < D_{i+1}^b$), then for sure, the next service of M_i starts again in (i). However, if M_{i+1} finishes first, then on service completion of M_i, both (ii) or (iii) may be encountered. We denote by $p_{i+1}^{b,nf}$ the probability that departure server M_{i+1} completes at least two services before the next arrival at L_{i+1} (given that M_{i+1} completes at least one service before the next arrival). So, if M_{i+1} finishes first, then the next service of M_i starts in (iii) with probability $p_{i+1}^{b,nf}$, and in (ii) otherwise. Similarly, if service of M_i starts in situation (ii), and M_i finishes before M_{i+1} (i.e., $S_i < D_{i+1}^f$), then the next service of M_i certainly starts in (i). If $S_i > D_{i+1}^f$, then M_i will start in (iii) with probability $p_{i+1}^{f,nf}$ and in (ii) otherwise. Finally, in situation (iii), the next service of M_{i+1} can never start in (i); it will start in (ii) with probability $p_{i+1}^{nf,f}$ and in (iii) otherwise, where $p_{i+1}^{nf,f}$ is defined as the probability that, on an arrival at L_{i+1}, the buffer of L_{i+1} fills up.

This completes the description of the service processes of the arrival and departure servers of L_i. In the next section we will translate subsystem L_i to a Quasi-Birth-Death (QBD) process; see [5].

5 Subsystem

In this section we describe the analysis of a subsystem L_i; for ease of notation we drop the subscript i in the sequel of this section. In order to translate L to a Markov process, we will describe the random variables introduced in the foregoing sections in terms of exponential phases according to the following recipe; see e.g. [12]. Consider a random variable X with mean $\mathbb{E}(X)$ and squared coefficient of variation c_X^2. If $1/k \leq c_X^2 \leq 1/(k-1)$ for some $k = 2, 3, \ldots$, then the mean and squared coefficient of variation of the Erlang$_{k-1,k}$ distribution with density

$$f(x) = p\mu^{k-1}\frac{x^{k-2}}{(k-2)!}e^{-\mu t} + (1-p)\mu^k\frac{x^{k-1}}{(k-1)!}e^{-\mu x}, \qquad x \geq 0, \tag{2}$$

match $\mathbb{E}(X)$ and c_X^2, provided the parameters p and μ are chosen as

$$p = \frac{1}{1+c_X^2}(kc_X^2 - (k(1+c_X^2) - k^2c_X^2)^{1/2}), \qquad \mu = \frac{k-p}{\mathbb{E}(X)}.$$

Hence, in this case we may describe X in terms a random sum of $k-1$ or k independent exponential phases, each with rate μ. Alternatively, if $c_X^2 > 1$, then the Hyper-Exponential$_2$ distribution with density

$$f(x) = p\mu_1 e^{-\mu_1 x} + (1-p)\mu_2 e^{-\mu_2 x}, \qquad x \geq 0, \tag{3}$$

matches $E(X)$ and c_X^2, provided the parameters p, μ_1 and μ_2 are chosen as

$$p = \frac{1}{2}(1 + \sqrt{\frac{c_X^2-1}{c_X^2+1}}), \qquad \mu_1 = \frac{2p}{\mathbb{E}(X)}, \qquad \mu_2 = \frac{2(1-p)}{\mathbb{E}(X)}.$$

This means that X can be represented in terms of a probabilistic mixture of two exponential phases with rates μ_1 and μ_2, respectively. Obviously, there exist many other phase-type distributions matching the first two (or more) moments; see e.g. [6]. In our experience, however, use of other distributions does not essentially affect the quality of the approximation.

Now we apply this recipe to represent each of the random variables A, S, D^b and D^f in terms of exponential phases. The status of the service process of the arrival server can be easily described by the service phase of A. The description of the service process of the departure server is more complicated. Here we need to keep track of the phase of S and the phase of D^b or D^f, depending on situation (i), (ii) or (iii). The description of this service process is illustrated in the following example.

Example: Suppose that S can be represented by two successive exponential phases, D^b by three phases and D^f by a single phase, where each phase possibly has a different rate. Then the phase-diagram for each situation (i), (ii) and (iii) is sketched in Figure 4. States a, b and c are the initial states for each situation. The grey states indicate that either S, D^b or D^f has completed all phases. A transition from one of the states d, e, f, g and h corresponds to a service completion of departure server M (i.e., a departure from subsystem L); the other transitions correspond to a phase completion, and do not trigger a departure. The probability that a transition from state e is directed to initial state a is equal to 1; the probability that a transition from state d is directed to initial state a, b and c is equal to 0, $1 - p^{b,nf}$ and $p^{b,nf}$, respectively. The transition probabilities from the other states f, g and h can be found similarly.

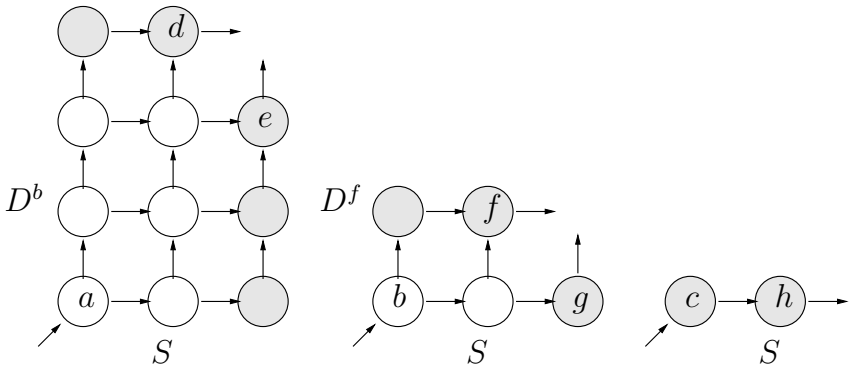

Figure 4: Phase diagram for the service process of the departure server.

In Figure 4 it is assumed that M can immediately start with the next service S after a departure. However, if M is starved, then S will not immediately start but has to wait for the next arrival at L (i.e., service completion of the arrival server of L). However, D^b or D^f will immediately start completing their phases, and may even have completed all their phases at the start of S.

From the example above it will be clear that the service process of the departure server can be described by a Markovian Service Process: it is a finite-state Markov process, the generator Q_d of which can be decomposed as $Q_d = Q_{d0} + Q_{d1}$, where the transitions of Q_{d1} correspond to service completions (i.e., departures from L) and the ones of Q_{d0} correspond to transitions not leading to departures. The dimension n_d of Q_d can be large, depending on the number of phases required for S, D^b and D^f. Similarly, the service process of the arrival service can also be described by a (somewhat simpler) Markovian Arrival Process, with generator $Q_a = Q_{a0} + Q_{a1}$ of dimension n_a.

Subsystem L can be described by a QBD with states (i, j, l), where i denotes the number of jobs in subsystem L, excluding the one at the arrival server. Clearly, $i = 0, ..., b + 2$, where $i = b + 2$ indicates that the arrival server is blocked because buffer B is full. The state variables j and l denote the state of the arrival and departure process, respectively. To specify the generator $\mathbf{Q}$ of the QBD we use the

Kronecker product: If A is an $n_1 \times n_2$ matrix and B is an $n_3 \times n_4$ matrix, the Kronecker product $A \otimes B$ is defined as

$$A \otimes B = \begin{pmatrix} A(1,1)B & \cdots & A(1,n_2)B \\ \vdots & & \vdots \\ A(n_1,1)B & \cdots & A(n_1,n_2)B \end{pmatrix}.$$

We order the states lexicographically and partition the state space into *levels*, where level $i = 0, 1, \ldots, b+2$ is the set of all states with i jobs in the system. Then $\mathbf{Q}$ takes the form:

$$\mathbf{Q} = \begin{pmatrix} B_{00} & B_{01} & & & & \\ B_{10} & A_1 & A_0 & & & \\ & A_2 & \ddots & \ddots & & \\ & & \ddots & \ddots & A_0 & \\ & & & A_2 & A_1 & C_{10} \\ & & & & C_{01} & C_{00} \end{pmatrix}.$$

Below we specify the submatrices in $\mathbf{Q}$. The transition rates from levels $1 \leq i \leq b$ are given by

$$\begin{aligned} A_0 &= Q_{a1} \otimes I_{n_d}, \\ A_1 &= Q_{a0} \otimes I_{n_d} + I_{n_a} \otimes Q_{d0}, \\ A_2 &= I_{n_a} \otimes Q_{d1}, \end{aligned}$$

where I_n is the identity matrix of size n. The transition rates are different for the levels $i = 0$ and $i = b+2$. At level $b+2$ the arrival server is blocked, so

$$\begin{aligned} C_{10} &= Q_{a1}(:,1) \otimes I_d, \\ C_{00} &= Q_{d0}, \\ C_{01} &= I_{n_a}(1,:) \otimes Q_{d1}, \end{aligned}$$

where $P(x,:)$ is the x-th row of matrix P and $P(:,y)$ is the yth column of P. To specify the transition rates to level 0, we introduce that transition rate matrix Q_s of dimension n_s, describing the progress of the phases of D^b or D^f while the departure server is starved. Further, the $n_d \times n_s$ matrix $\bar{Q}_{d1}$ contains the transition rates from states in Q_d, that correspond to a departure, to the initial states in Q_s. Finally, $\bar{I}_{n_s,n_d}$ is the 0-1 matrix of size $n_s \times n_d$ that preserves the phase of Q_s (i.e., the phase of D^b or D^f) when the departure server starts serving the next job after having been starved. Then we obtain

$$\begin{aligned} B_{10} &= I_{n_a} \otimes \bar{Q}_{d1}, \\ B_{00} &= Q_{a0} \otimes I_{n_s} + I_{n_a} \otimes Q_s, \\ B_{01} &= Q_{a1} \otimes \bar{I}_{n_s,n_d}. \end{aligned}$$

This concludes the specification of $\mathbf{Q}$.

The steady-state distribution of the QBD can be determined by the matrix-geometric method; see e.g [5, 10, 13]. Denoting the equilibrium probability vector of level i by π_i, then π_i has the matrix-geometric form

$$\pi_i = x_1 R^{i-1} + x_{b+1} \hat{R}^{b+1-i}, \qquad i = 1, \ldots, b+1, \tag{4}$$

where R is the minimal nonnegative solution of the matrix-quadratic equation

$$A_0 + RA_1 + R^2 A_2 = 0,$$

and $\hat{R}$ is the minimal nonnegative solution of

$$A_2 + \hat{R}A_1 + \hat{R}^2 A_0 = 0.$$

The matrices R and $\hat{R}$ can be efficiently determined by using an iterative algorithm developed in [10]. The vectors π_0, x_1, x_{b+1} and π_{b+2} follow from the balance equations at the boundary levels $0, 1, b+1$ and $b+2$,

$$\begin{aligned} 0 &= \pi_0 B_{00} + \pi_1 B_{10}, \\ 0 &= \pi_0 B_{01} + \pi_1 A_1 + \pi_2 A_2, \\ 0 &= \pi_b A_0 + \pi_{b+1} A_1 + \pi_{b+2} C_{01}, \\ 0 &= \pi_b C_{10} + \pi_{b+1} C_{00}. \end{aligned}$$

Substitution of (4) for π_1 and π_{b+1} in the above equations yields a set of linear equations for π_0, x_1, x_{b+1} and π_{b+2}, which together with the normalization equation, has a unique solution. This completes the determination of the equilibrium probabilities vectors π_i. Once these probability vectors are known, we can easily derive performance measures and quantities required to describe the service times of the arrival and departure server.

Throughput:
The throughput T satisfies

$$\begin{aligned} T &= \pi_1 B_{10} e + \sum_{i=2}^{b+1} \pi_i A_2 e + \pi_{b+2} C_{01} e \\ &= \pi_0 B_{01} e + \sum_{i=1}^{b} \pi_i A_0 e + \pi_{b+1} C_{10} e, \end{aligned} \tag{5}$$

where e is the all-one vector.

Service process of the arrival server:
To specify the service time of the arrival server we need the probability q^e that the system is empty just after a departure and the first two moments of the residual service time RA of the arrival server at the time of such an event. The probability q^e is equal to the mean number of departures per time unit leaving behind an empty system divided by the mean total number of departures per time unit. So

$$q^e = \pi_1 B_{10} e / T.$$

The moments of RA can be easily obtained, once the distribution of the phase of the service time of the arrival server, just after a departure leaving behind an empty system, is known. Note that component (j,k) of the vector $\pi_1 B_{10}$ is the mean number of transitions per time unit from level 1 entering state (j,k) at level 0. By adding all components with $j = l$ and dividing by $\pi_1 B_{10} e$, i.e., the mean total number of transitions per time unit from level 1 to 0, we obtain the probability that the arrival server is in phase l just after a departure leaving behind an empty system.

Service process of the departure server:
We need to calculate the first two moments of D^b and D^f and the transition probabilities $p^{b,nf}$, $p^{f,nf}$ and $p^{nf,f}$. This requires the distribution of the initial phase upon entering level $b+1$ due to a departure (or arrival). Clearly, component (j,k) of $\pi_{b+2} C_{01}$ is equal to the number of transitions per time unit from level $b+2$ entering state (j,k) at level $b+1$. Hence, $\pi_{b+2} C_{01} / \pi_{b+2} C_{01} e$ yields the distribution of the initial phase upon entering level $b+1$ due to a departure. Defining $D^b(1)$ and $D^b(2)$ as the time till the first, respectively second, departure and $A^b(1)$ as the time till the first arrival, from the moment of entering level $b+1$, it is straightforward to calculate the moments of $D^b(1) \equiv D^b$ and the probabilities $\Pr[D^b(1) < A^b(1)]$ and $\Pr[D^b(2) < A^b(1)]$. Transition probability $p^{b,nf}$ now follows from

$$p^{b,nf} = \Pr[D^b(2) < A^b(1) | D^b(1) < A^b(1)] = \frac{\Pr[D^b(2) < A^b(1)]}{\Pr[D^b(1) < A^b(1)]}.$$

Calculation of the moments of D^f and transition probability $p^{f,nf}$ proceeds along the same lines, where the distribution of the initial phase upon entering level $b+1$ due to an arrival is given by $\pi_b A_0 / \pi_b A_0 e$.

Finally, $p^{nf,f}$ satisfies

$$p^{nf,f} = \frac{\pi_b A_0 e}{\pi_0 B_{01} e + \sum_{i=1}^{b} \pi_i A_0 e}.$$

6 Iterative method

This section is devoted to the description of the iterative algorithm to approximate the performance of tandem queue L. The algorithm is based on decomposition of L in $N-1$ subsystems $L_1, L_2, \ldots, L_{N-1}$.

Step 0: Initialization

The first step of the algorithm is to initially assume that there is no blocking. This means that the random variables D_i^b and D_i^f are initially set to 0.

Step 1: Evaluation of subsystems

We subsequently evaluate each subsystem, starting from L_1 and up to L_{N-1}. First we determine new estimates for the first two moments of A_i, before calculating the equilibrium distribution of L_i.

(a) Service process of the arrival server

For the first subsystem L_1, the service time A_1 is equal to S_0, because server M_0 cannot be starved. For the other subsystems we proceed as follows in order to determine the first two moments of A_i. By Little's law we have for the throughput T_i of subsystem L_i,

$$T_i = \frac{1 - p_{i,b_i+2}}{\mathbb{E}(A_i)}, \tag{6}$$

where p_{i,b_i+2} denotes the long-run fraction of time the arrival server of subsystem L_i is blocked. By substituting in (6) the estimate $T_{i-1}^{(k)}$ for T_i, which is the principle of conservation of flow, and $p_{i,b_i+2}^{(k-1)}$ for p_{i,b_i+2} we get as new estimate for $\mathbb{E}(A_i)$,

$$\mathbb{E}(A_i^{(k)}) = \frac{1 - p_{i,b_i+2}^{(k-1)}}{T_{i-1}^{(k)}},$$

where the superscripts indicate in which iteration the quantities have been calculated. The second moment of A_i cannot be obtained by using Little's law. Instead we calculate q_{i-1}^e and the first two moments of RA_{i-1} from the equilibrium distribution of L_{i-1} as described in Section 5. Then the squared coefficient of variation $c_{A_i}^2$ can be determined from (1), after which the second moment of A_i readily follows from $\mathbb{E}(A_i^2) = (1 + c_{A_i}^2)/\mu_{a,i}^2$.

(b) Analysis of subsystem L_i

Based on the new estimates for the first two moments of A_i, we calculate the equilibrium probability vectors $\pi_0, \pi_1, \ldots, \pi_{b_i+2}$ for subsystem L_i as described in Section 5.

(c) Determination of the throughput of L_i

Once the equilibrium distribution is known, we determine the new throughput $T_i^{(k)}$ according to (5).

Step 2: Service process of the departure server

From subsystem L_{N-2} down to L_1, we calculate new estimates for the first two moments of D_i^b and D_i^f and the transition probabilities $p_i^{b,nf}$, $p_i^{f,nf}$ and $p_i^{nf,f}$, as explained in Section 5. Note that D_{N-1}^b and D_{N-1}^f are 0, because server M_{N-1} can never be blocked.

Step 3: Convergence

After Step 1 and 2 we verify whether the iterative algorithm has converged or not by comparing the throughputs in the $(k-1)$-th and k-th iteration. When

$$\sum_{i=1}^{N-1} |T_i^{(k)} - T_i^{(k-1)}| < \varepsilon,$$

we stop and otherwise repeat Step 1 and 2.

7 Numerical Results

In order to investigate the quality of the current method we evaluate a large set of examples and compare the results with discrete-event simulation. We also compare the results with the approximation of [13]. In each example we assume that only mean and squared coefficient of variation of the service times at each server are known, and we match, both in the approximation and discrete-event simulation, mixed Erlang or Hyper-exponential distributions to the first two moments of the service times, depending on whether the coefficient of variation is less or greater than 1; see (2) and (3) in Section 5. Then we compare the throughput and the mean sojourn time (i.e., the mean time that elapses from the service start at server M_0 until service completion at server M_{N-1}) produced by the current approximation and the one in [13] with the ones produced by discrete-event simulation. Each simulation run is sufficiently long such that the widths of the 95% confidence intervals of the throughput and mean sojourn time are smaller than 1%.

We use the following set of parameters for the tests. The mean service times of the servers are all set to 1. We vary the number of servers in the tandem queue between 4, 8, 16, 24 and 32. The squared coefficient of variation (SCV) of the service times of each server is the same and is varied between 0.5, 1, 2, 3 and 5. The buffer sizes between the servers are the same and varied between 0, 1, 3 and 5. We will also test three kinds of *imbalance* in the tandem queue. We test imbalance in the mean service times by increasing the average service time of the 'even' servers from 1 to 1.2. The effect of imbalance in the SCV is tested by increasing the SCV of the service times of the 'even' servers by 0.5. Finally, imbalance in the buffer sizes is tested by increasing the buffers size of the 'even' buffers by 2. This leads to a total of 800 test cases.

The results for each category are summarized in Tables 1 up to 4. Each table lists the average error in the throughput and the mean sojourn time compared with simulation results. Each table also gives for three error-ranges the percentage of the cases that fall in that range, and the average error of the approximation of van Vuuren and Adan [13], denoted by VA.

From the tables we can conclude that the current method performs well and better than [13]. The overall average error in the throughput is 2.56% and the overall average error in the mean sojourn time is 2.54%, while the corresponding percentages for [13] are 4.40% and 5.82%, respectively.

In table 1 it is striking that in case of zero buffers the current method produces the most accurate estimates, while the method of [13] produces the least accurate results. A possible explanation is that for each subsystem the current method keeps track of the status of the downstream server while its departure server is starved; this is not done in [13]. Both methods seem to be robust to variations in buffer

Buffer sizes	Error (%) in the throughput					Error (%) in mean sojourn time				
	Avg.	0-2	2-4	> 4	VA	Avg.	0-2	2-4	> 4	VA
$0, 0, \ldots$	1.41	88	10	2	7.22	3.94	53	19	28	11.66
$1, 1, \ldots$	3.99	46	36	18	4.6	2.89	59	30	11	7.14
$3, 3, \ldots$	3.32	56	28	16	3.85	2.03	75	25	0	4.61
$5, 5, \ldots$	2.23	75	20	5	3.69	2.25	66	32	2	3.89
$0, 2, \ldots$	1.56	89	11	0	4.70	2.38	75	23	2	6.76
$1, 3, \ldots$	3.36	58	27	15	3.95	2.41	69	31	0	4.94
$3, 5, \ldots$	2.71	66	21	13	3.63	1.88	77	22	1	3.88
$5, 7, \ldots$	1.88	79	21	0	3.53	2.50	66	30	4	3.70

Table 1: Overall results for tandem queues with different buffer sizes.

SCVs	Error (%) in the throughput					Error (%) in mean sojourn time				
	Avg.	0-2	2-4	> 4	VA	Avg.	0-2	2-4	> 4	VA
$0.5, 0.5, \ldots$	0.77	100	0	0	1.03	2.37	70	21	9	2.67
$1, 1, \ldots$	1.22	100	0	0	1.27	2.18	75	19	6	2.83
$2, 2, \ldots$	1.85	90	10	0	3.09	2.24	76	20	4	4.95
$3, 3, \ldots$	2.90	51	49	0	5.53	2.40	75	23	2	7.20
$5, 5, \ldots$	5.58	15	45	40	9.64	3.29	48	45	7	10.31
$0.5, 1, \ldots$	0.88	100	0	0	1.60	2.53	70	23	7	3.08
$1, 1.5, \ldots$	1.22	100	0	0	1.85	2.26	73	21	6	3.20
$2, 2.5, \ldots$	2.00	85	15	0	3.74	2.23	76	20	4	5.60
$3, 3.5, \ldots$	3.19	41	59	0	6.18	2.40	75	23	2	7.75
$5, 5.5, \ldots$	5.96	14	40	46	10.06	3.43	38	51	11	10.63

Table 2: Overall results for tandem queues with different SCVs of the service times.

sizes along the line. Table 2 convincingly demonstrates that especially in case of service times with high variability the current approximation performs much better than [13]. Remarkably, Table 4 shows that the average error in the throughput does not seem to increase for longer lines, a feature not shared by the approximation of [13].

8 Conclusions

In this paper we developed an approximate analysis of single-server tandem queues with finite buffers, based on decomposition into single-buffer subsystems. The distinguishing feature of the analysis is that dependencies between successive aggregate service times (including starvation and blocking) are taken into account. Numerical results convincingly demonstrated that it pays to include such dependencies, especially in case of longer tandem queues and service times with a high variability. The price to be paid, of course, is that the resulting subsystems are more complex and computationally more demanding.

We conclude with a remark on the subsystems. There seems to be an asymmetry in the modeling of the service processes of the arrival and departure server; the service times of the arrival server are simply assumed to be independent and identically distributed, whereas the service times of the departure server are modeled by a Markovian service process, carefully taking into account dependencies between successive service times. Investigating whether a similar Makovian description of the service process of the arrival server is also feasible (and rewarding) seems to be an interesting direction for future research.

Mean service times	Error (%) in the throughput					Error (%) in mean sojourn time				
	Avg.	0-2	2-4	> 4	VA	Avg.	0-2	2-4	> 4	VA
$1, 1, \ldots$	2.65	68	23	9	4.23	2.50	69	26	5	5.71
$1, 1.2, \ldots$	2.46	71	21	8	4.57	2.57	67	27	6	5.93

Table 3: Overall results for tandem queues with different mean service times.

Servers in line	Error (%) in the throughput					Error (%) in mean sojourn time				
	Avg.	0-2	2-4	> 4	VA	Avg.	0-2	2-4	> 4	VA
4	2.26	69	29	2	0.57	1.77	83	17	0	0.95
8	2.68	66	27	7	2.87	1.82	81	18	1	2.80
16	2.68	68	21	11	5.30	1.63	88	9	3	5.39
24	2.55	72	17	11	6.41	2.65	66	26	8	8.38
32	2.61	73	16	11	6.84	4.80	19	62	19	11.59

Table 4: Overall results for tandem queues of different length.

References

[1] Y. Dallery and B. Gershwin (1992) Manufacturing flow line systems: a review of models and analytical results. *Queueing Systems* 12, 3-94.

[2] S.B. Gershwin (1987) An efficient decomposition algorithm for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research* 35, 291-305.

[3] S. Helber (2005) Analysis of flow lines with Cox-2-distributed processing times and limited buffer capacity *OR Spectrum* 27, 221-242.

[4] L. Kerbache and J. MacGregor Smith (1987) The generalized expansion method for open finite queueing networks. *The European Journal of Operations Research* 32, 448-461.

[5] G. Latouche and V. Ramaswami (1999) *Introduction to matrix analytic methods in stochastic modeling*. ASA-SIAM Series on Statistics and Applied Probability 5.

[6] T. Osogami and M. Harchol-Balter (2003) Closed form solutions for mapping general distributions to minimal PH distributions. *Performance Evaluation* 63, 524-552.

[7] H.G. Perros (1989) A bibliography of papers on queueing networks with finite capacity queues. *Performance Evaluation* 10, 255-260.

[8] H.G. Perros (1994) *Queueing networks with blocking*. Oxford University Press.

[9] H.G. Perros and T. Altiok (1989) *Queueing networks with blocking*. North-Holland, Amsterdam.

[10] V. Naoumov, U.R. Krieger and D. Wagner (1997) Analysis of a multiserver delay-loss system with a general Markovian arrival process. *Matrix-analytic methods in stochastic models (eds. A.S. Alfa, S.R. Chakravarthy), Lecture notes in pure and applied mathematics*, 183, Marcel Dekker, New York, 1996.

[11] M.F. Neuts (1989) *Structured stochastic matrices of M/G/1-type and their applications*. Marcel Dekker, New York.

[12] H.C. Tijms (1994) *Stochastic models: an algorithmic approach*. John Wiley & Sons, Chichester.

[13] M. van Vuuren and I.J.B.F. Adan (2009) Performance analysis of tandem queues with small buffers. *IIE Transactions* 41, 882-892.

[14] M. van Vuuren, I.J.B.F. Adan and S.A. Resing-Sassen (2005) Performance analysis of multi-server tandem queues with finite buffers and blocking. *OR Spectrum* 27, 315-339.