After-Service Blocking in Tandem Queues

Tim Rens de Boer[1], René Bekker[2], and Rob D. van der Mei[1,2]

[1] Stochastics, Centrum Wiskunde & Informatica, Science Park 123, 1098XG Amsterdam

[2] Department of Mathematics, Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam

Abstract. Tandem queues with zero buffers find many applications in systems where jobs are processed via a predefined number of sequential processing steps. From an analytic point of view, the analysis of such systems is highly complicated, caused by the phenomenon of *after-service blocking* (ASB). ASB occurs when a completed job occupies a server while waiting for the next phase. This challenge is addressed by introducing and comparing heuristics to quantify the impact of ASB on the performance of the tandem queue. Our proposed heuristics offer significant advantages over existing methods, delivering more insightful and accurate performance estimations. This allows for faster evaluations and their use in optimization, making it ideal for real-time decision making and capacity allocation. This research presents a valuable tool for queueing system designers and managers to make informed decisions regarding capacity allocation, resource management, and service level optimization, particularly in scenarios with limited budgets.

Keywords: Tandem Queues · After-Service Blocking · Queueing Theory · Heuristics · Queue Performance Optimization Strategies.

1 Introduction

Tandem queues with zero buffers are frequently employed to model and analyze the performance of real-world systems across various application domains, particularly in cases where tasks must be processed sequentially through multiple stages. These systems are often represented by M/M/c/c queues. Typical applications include areas such as healthcare [6], manufacturing [22], and telecommunications [20]. These realistic situations often have no (or limited) buffer space, meaning that jobs or patients cannot wait, or only a few are able to wait. Inefficiencies in queueing systems with finite buffers often arise from the limited buffer capacity, particularly due to the phenomenon known as *after-service blocking* (ASB) [1], in which a server becomes blocked while waiting for available downstream capacity.

An example of ASB can be observed in healthcare systems, where a patient remains in the Emergency Department or a hospital due to a lack of beds available in a downstream care facility, such as a nursing home [5]. This situation not only incurs higher costs due to more expensive acute care, but also contributes to system-wide congestion, reducing overall patient throughput, and potentially exacerbating patient conditions due to delayed care [12]. More generally, ASB occurs when jobs, after completing service, are directed to another queue but are forced to remain at their current server due to capacity constraints. This results in the server being occupied by completed jobs, preventing it from serving new ones [9]. This scenario not only impedes the efficiency of the server, but can also lead to cascading delays, as subsequent jobs are held up by those that are blocked. As a result, costs can increase, especially when the next stage of service is more appropriate and cost-effective.

Motivated by this, our aim in this paper is to approximate the impact of ASB on the throughput in tandem queues. Previous research has indicated that it is possible to approximate the effects of ASB in specific scenarios [6], or through machine learning techniques [8]. However, given the anticipated increase in the demand for healthcare and the limited availability of healthcare professionals, there is a pressing need for an approximate method that can handle a wide range of scenarios. This is especially critical in scenarios where the demand is close to the maximum capacity that the servers can manage and where budget constraints necessitate optimal allocation of capacity.

The contribution of this paper is two-fold. First, we introduce a heuristic that provides more precise estimates compared to existing methods in approximating the effects of ASB in tandem queues with zero buffers across a wide range of scenarios. The approximations provide insight in how ASB is affected by the system parameters. As a byproduct of developing the heuristic, we also present continuous extensions of M/M/c/$c+k$ performance formulas for non-integer values of c. Second, our heuristic allows for fast evaluations of system performance, making it an ideal tool for optimization contexts. Unlike simulation-based performance evaluations, which are often more computationally intensive, the heuristic significantly accelerates the optimization process. We demonstrate this using a capacity allocation setting.

The organisation of the paper is as follows. Section 2 provides a detailed description of the model. Section 3 reviews the related literature. Section 4 discusses the approximations employed in this research and indicates how these approximations can be applied to optimization problems. Section 5 shows the results of the approximation, as well as the optimization. Finally, Section 6 presents the conclusion and discussion.

2 Model description

We consider a tandem of n zero-buffer multi-server nodes, as illustrated in Figure 1. Arrivals to the first node occur according to a Poisson process with rate λ. The service times are exponentially distributed, whereas the service rates are denoted by $(\mu_1, \mu_2, \ldots, \mu_n)$. The number of servers at each node is given by the vector $(c_1, c_2, \ldots, c_n)$. We denote the buffer size at node i by k_i, but the buffer size is set to zero in all experiments. Jobs that arrive at the first node when all servers are busy (and the buffer is full) are rejected. Upon completing the service at node i, the job moves to the next node $i+1$. If node $i+1$ lacks available server space, the job must wait in the current node i, thus blocking other jobs from entering service. We denote by P_i the probability that all servers are busy (and the buffer is full) at node i, for $i = 1, \ldots, n$. Note that P_1 is also the fraction of jobs denied access to the first node. The ASB mechanism is visualized in Figure 2. In this figure, we can see that a new job, say job A, arrives at the first queue, enters service, is blocked after completing service, and is finally unblocked and can start service at the second node. This job can only move to the second node if server capacity is available. The jobs are served and unblocked on a First-Come-First-Served (FCFS) basis.

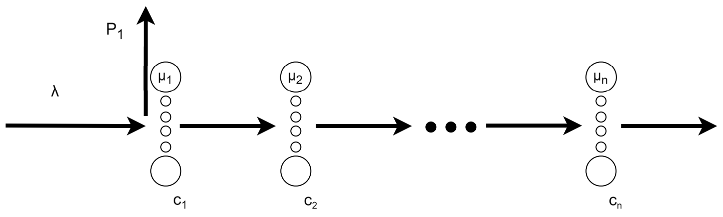

Fig. 1. Visual representation of a tandem of n M/M/c/c queues

3 Literature

In this section, we consider previous studies that address ASB in tandem queues. The exact analysis of ASB in tandem systems turns out to be hardly available in the literature. An exception is Akyildiz and von Brand [1]. They show that exact solutions are possible, but only for a small number of scenarios, such as two-node single-server systems and closed networks. However, the computation time of these exact solutions increases dramatically when more servers are considered or when the system size increases, as this directly affects the state space. Other papers [1–3, 11, 13, 17] focusing on exact analyses encountered similar challenges, involving the extensive computation time required and the limited set of scenarios in which they are applicable. This underscores the need for more efficient approximation methods.

Next, we discuss various approximation methods from the existing literature. Koizumi et al. [15] introduced a simple approximation, which adjusts the service time of queue i by incorporating the estimated waiting time to access queue $i+1$. Bretthauer et al. [6], motivated by bed-blocking problems in healthcare

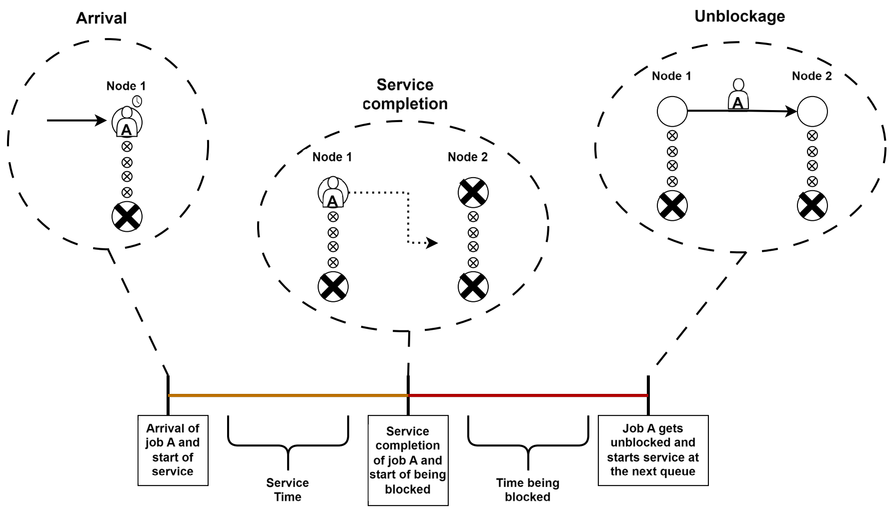
Arrival
Node 1
A
Service completion
Node 1
Node 2
A
Unblockage
Node 1
Node 2
A
Arrival of job A and start of service
Service Time
Service completion of job A and start of being blocked
Time being blocked
Job A gets unblocked and starts service at the next queue

Fig. 2. Visual representation of ASB for job A in a two-queue tandem model

settings, improved the heuristic of Koizumi et al. by developing an approximation for throughput rates in tandem queues. Their approach extends the work of [15] by adjusting not only the service rate, but also the number of servers and the flow rate. However, their study does not address issues such as how the calculations of the heuristic are adapted to accommodate non-integer modified numbers of server, and is limited to tandem queues with a load smaller than one.

Dallery and Frein [7] introduced a decomposition method customized for single-server tandem queues. Their method entails dissecting the tandem network into individual subsystems, with buffers positioned between a pair of servers. Their approach is not applicable to networks with multiple servers. Similarly, Van Vuuren et al. [23] introduced a decomposition method for tandem server pools and buffers. Their iterative method approximates throughput and sojourn time for various buffer sizes under the assumption that the system is never starved. This assumption implies that there is always a new arrival at the first queue, ensuring that any available service capacity is immediately utilized. The assumption that the system is never starved renders this approach impractical for realistic scenarios.

Osorio and Bierlaire [18] presented a heuristic that distinguishes between arrival, service, blocking, and unblocking rates. This method approximates the blocking probabilities for different networks, but its application is limited to specific networks and assumes immediate resolution of *deadlocks*. A deadlock occurs when jobs are blocked directly or indirectly by themselves. Alternatively, Dieleman et al. [8] used neural networks (NN) to predict throughput for stable tandem queues. Their research shows that NNs informed by queueing-theoretic knowledge outperform those based solely on input parameters, although the complexity of NNs complicates the interpretation of their methods.

Apart from approximating the effect of ASB, there are also other papers considering ASB in queueing networks. The impact of ASB on *open* queueing networks and the modeling of deadlocks within these systems has been explored by Palmer et al. [19]. This study focuses on the time until a deadlock occurs, noting that once a system is in a deadlock state, it remains so without external intervention. El-Taha and Wolff [9] developed a simulation model to analyze the effects of ASB in healthcare care, to understand the capacity requirements in such settings, highlighting the complexity of ASB.

From the literature, we observe that most ASB-related studies focus on approximating the ASB effect, but they are typically limited in the number of scenarios in which the approximations provide an accurate estimation. This paper addresses a research gap by introducing a new heuristic for rapidly quantifying the impact of ASB on the performance of tandem queues. This heuristic is required to accommodate a wider range of tandem queues. Moreover, the heuristic should be both explainable and transparent to illustrate how different parameters interact.

4 Methodology

In this section, we discuss the methodology employed to address ASB in tandem queues. Section 4.1 presents four approximation techniques: two established methods from the literature and two newly developed approaches. In Section 4.2, we apply these methods within an optimization framework, focusing on optimizing capacity allocation.

4.1 Approximations

This section presents a detailed analysis of the four approximation techniques. The key idea of each method is to decompose the network into a series of individual queues, where the impact of ASB is incorporated by a modified service capacity in terms of service rate and number of servers. The strengths of such approximations is that they are fast, whereas they also provide insight in how subsequent queues affect each others behavior. It should be noted that although the buffer size is zero ($k_i = 0$) for all nodes in our scenarios, the approximations are formulated without imposing this restriction on k_i to enhance their generalizability.

M/M/c/$c + k$ based approximation (Loss)
The first approximation is the most straightforward and is based on the M/M/c/c+k queue, i.e., the classical Poisson-driven finite-buffer c-server queue with exponential service times. We refer to the M/M/c/$c + k$ approximation as 'Loss'

throughout the remainder of this paper. For such a queue, arriving jobs finding $c+k$ jobs present are lost. For this approximation, the fraction of lost jobs is approximated by only considering the first queue. This implies that additional blocking that occurs when jobs are delayed at a server due to congestion in downstream queues is not taken into account. Consequently, the blocking probability derived from this approximation represents a lower bound since ASB will increase the blocking probability. Let $a = \frac{\lambda}{\mu}$ denote the *offered load* and let $\rho = \frac{\lambda}{c\mu}$ be the *load per server*. The stationary blocking probability, denoted as π_{c+k}, follows directly from the steady-state distribution of the number of customers in the system (cf. [16])

$$\pi_{c+k}(\lambda, \mu, c, k) = \frac{\rho^{c+k} c^c}{\Gamma(c+1)} \pi_0. \tag{1}$$

The above is a reformulation of the well-known steady-state probabilities, as presented in Appendix 6, where c is no longer required to be integer. This expression is essential to improve the speed of calculations as well as for the other heuristics where the number of servers may be non-integer. The probability of an empty system, π_0, is given by:

$$\pi_0 = \begin{cases} \left[e^a \frac{\Gamma(c,a)}{\Gamma(c)} + \frac{a^c}{\Gamma(c+1)}(k+1) \right]^{-1}, & \text{if } \rho = 1, \\ \left[e^a \frac{\Gamma(c,a)}{\Gamma(c)} + \frac{a^c}{\Gamma(c+1)} \frac{1-\rho^{k+1}}{1-\rho} \right]^{-1}, & \text{otherwise,} \end{cases} \tag{2}$$

with $\Gamma(c+1)$ and $\Gamma(c,a)$ the Gamma function and the upper incomplete Gamma function, respectively, see also Appendix 6. The first approximation of the blocking probability is then given by $\pi_{c_1}(\lambda, \mu_1, c_1, k_1)$.

The heuristic by Bretthauer et al. [6] (BR)
We use the heuristic proposed in [6], from now on referred to as 'BR', as a benchmark because it is closely aligned with our method and is tested on scenarios that are particularly relevant. BR introduces a flow rate $F_{1,2}$, which represents the flow from queue 1 to queue 2. Since no jobs are lost after queue 1, the flow rate is equal for each pair of subsequent queues i and $i+1$, and is denoted by F. The flow rate is calculated using:

$$F = F_{i,i+1} = F_{1,2} = \lambda \cdot (1 - P_1). \tag{3}$$

In addition to the flow rate, the heuristic modifies the number of servers, denoted by c_i^*, and the service rate, denoted by μ_i^*, to account for the blocking effects of subsequent queues. The modified number of servers captures how many of the servers are not blocked due to ASB and the modified service rate can be used to capture an extra time a server is blocked due to ASB. Specifically, the expected number of customers waiting for service at a stage with c servers, service rate μ and flow rate F, is approximated using the M/M/c queue, cf. [14] for the case of any c,

$$L(F,\mu,c)=\frac{\left(\frac{F}{\mu}\right)^{c}F\mu}{\Gamma(c)(c\mu-F)^{2}}\left[\frac{e^{\frac{F}{\mu}}\Gamma(c,\frac{F}{\mu})}{\Gamma(c)}+\frac{\left(\frac{F}{\mu}\right)^{c}c\mu}{\Gamma(c+1)(c\mu-F)}\right]^{-1}. \tag{4}$$

Observe that when $F \geq c\mu$ the M/M/c queue becomes unstable, such that the expected number of jobs waiting for service explodes and $L(F,\mu,c)$ grows without bound.

The modified number of servers at station i is then calculated by the number of servers that are not blocked:

$$c_i^* = [c_i - L_{i+1}]^+, \tag{5}$$

and the modified service rate is determined by

$$\mu_i^* = \left[\frac{c_i^*}{c_i}\frac{1}{\mu_i}+\frac{c_i-c_i^*}{c_i}\frac{1}{c_{i+1}\mu_{i+1}}\right]^{-1}. \tag{6}$$

The rationale behind this adaptation is that blocked jobs must wait for the service completion at the next node before they can be unblocked (see [6] for a more elaborate discussion). The heuristic BR can be found in Algorithm 1.

Algorithm 1 Heuristic Algorithm by Bretthauer et al.

```
 1: Initialize $m = 0$, $P_1^0 = 0$, $\mu_i^0 = \mu_i$, $c_i^0 = c_i$ for $i = 1, \dots, n$
 2: while $|P_1^m - P_1^{m-1}| \geq \delta$ do
 3:     $m = m + 1$
 4:     Calculate the flow rate $F^m = \lambda(1 - P_1^{m-1})$
 5:     for $i = 1, \dots, n$ do
 6:         Calculate the modified number of servers $c_i^m$ using equation (5)
 7:         Calculate the modified service rate $\mu_i^m$ using equation (6).
 8:     end for
 9:     Update the blocking probability $P_1^m = \pi_i(F^m, \mu_1^m, c_1^m)$
10: end while
```

The modified service rate heuristic (MS)
One primary limitation of BR lies in its dependence on calculations derived from M/M/c formulas. For example, these formulas are only valid when the system workload remains below 1, that is, when $\rho < 1$. As a result, BR cannot provide accurate approximations in scenarios where the arrival rate leads to a workload larger than or equal to 1. This constraint significantly narrows the range of scenarios in which the heuristic can be applied effectively, limiting its usefulness in higher workload situations. Furthermore, we also observed a significant decrease in the performance of the heuristic in [6] when the number of servers becomes prohibitively small.

We address the limitations by focusing on a different adaption of the service rate, while we do not change the number of servers. To this end, we first notice that while nodes $\{2, 3, ..., n\}$ concern queues without buffer, jobs cannot be directly denied access to the system as these jobs wait at another node. We incorporate the ASB impact by including the waiting time in a suitable finite-buffer multi-server system. This differs from [6], which uses the expected number of waiting jobs. The adaptations are further explained below.

The service rate is modified to reflect the effect of ASB, as jobs remain longer at the nodes. The modified service rate is determined using the sojourn time, where we define the expected sojourn time as:

$$E[S_i] = \frac{1}{\mu_i} + E[B_{i+1}], \tag{7}$$

where $E[B_{i+1}]$ is the expected time being blocked by the next queue. This is straightforward for the last queue, yielding $E[S_n] = 1/\mu_n$. For the other queues, the time blocked by the next queue is determined by modeling queue $i + 1$ as an M/M/$c/c + k$ queue where a fraction of the servers of queue i are used as a buffer for queue $i+1$. This is done since the servers at node i function as a buffer for node $i + 1$, due to the ASB effect. We approximate the time blocked by the expected waiting time of an M/M/$c/c + k$ model with the altered parameters. The arrival rate is the original arrival rate λ and the service rate is the modified service rate μ^*_{i+1}. The number of servers remains c_{i+1}, and the number of buffer spaces is the number of buffer spaces at queue $i + 1$ plus the average number of servers at queue i that do not serve a job, yielding $k_{i+1} + \left[c_i - \frac{\lambda(1-P_1)}{\mu_i}\right]^+$. Combining the above, the modified service rate for queue i is determined by

$$\mu^*_i = \begin{cases} \left[\frac{1}{\mu_i} + W_q\left(\lambda, \mu^*_{i+1}, c_{i+1}, k_{i+1} + \left[c_i - \frac{\lambda(1-P_1)}{\mu_i}\right]^+\right)\right]^{-1}, & \text{for } i = 1, \dots, n-1, \\ \mu_n, & \text{for } i = n, \end{cases} \tag{8}$$

with $W_q(\lambda, \mu, c, k)$ the expected waiting time in the M/M/$c/c + k$ queue with arrival rate λ, service rate μ, c servers and buffer size k, see Appendix 6.

Next, we determine the probability that node 1 is fully occupied, thus blocking newly arriving jobs and denying them access to the system. The probability is based on the M/M/$c/c + k$ queue, with the service rate corresponding to a modified service rate. The probability of queue 1 being fully occupied is then computed as follows:

$$P_1 = \pi_{c_1+k_1}(\lambda, \mu^*_1, c_1, k_1). \tag{9}$$

This yields the 'modified service rate' heuristic, shortened to 'MS', as presented in Algorithm 2.

Algorithm 2 The modified service rate heuristic

```
1: Initialize $m = 0$, $\mu_i^m = \mu_i$, $P_1^m = 1.0$ for $i = 1, \dots, n$
2: while $|P_1^m - P_1^{m-1}| \geq \delta$ do
3:     $m = m + 1$
4:     $P_1^m = \pi_{c_1+k_1}(\lambda, \mu_1^m, c_1, k_1)$
5:     for $i = 1$ to $n$ do
6:         Update the modified service rate using equation (8)
7:     end for
8: end while
```

The modified service rate and number of servers heuristic (MS&C) Recall that a key limitation of the heuristic in [6] is its inability to handle systems with a load exceeding one, primarily due to its reliance on the M/M/c model. We propose a modified heuristic that also relies on adapting the service rate and the number of servers, named the modified service rate and number of servers heuristic ('MS&C'). However, instead of relying on the M/M/c model, we use the M/M/c/$c + k$ variant, which is stable for any finite load. Moreover, our adjustments enhance the accuracy of the heuristic. The specific modifications are detailed below.

We modify the service rate, μ_i^*, by conditioning on whether a job experiences blockage. If a job is not blocked, which occurs with probability $1 - P_{i+1}$, the job proceeds with its expected service time of $\frac{1}{\mu_i}$. However, if a job is blocked, with probability P_{i+1}, it must wait until all previously blocked jobs, which is equal to $c_i - c_i^*$, are unblocked. The expected unblocking time for these jobs is given by $1/(c_{i+1}^*\mu_{i+1}^*)$. Since there is no blocking at node n, we have $\mu_n^* = \mu_n$. This leads to the following recursive relation:

$$\mu_i^* = \begin{cases} \left[(1 - P_{i+1})\frac{1}{\mu_i} + P_{i+1}(c_i - c_i^*)\frac{1}{c_{i+1}^*\mu_{i+1}^*}\right]^{-1}, & \text{for } i = 1, \dots, n-1, \\ \mu_n, & \text{for } i = n. \end{cases} \tag{10}$$

We adopt a straightforward modification for the number of servers. The underlying intuition is that only a portion of the servers are actively serving jobs, while the remaining servers are occupied by blocked jobs. The fraction of blocked servers is represented by P_{i+1}. Consequently, the modified number of servers is denoted by:

$$c_i^* = \begin{cases} c_i(1 - P_{i+1}), & \text{for } i = 1, \dots, n-1, \\ c_n, & \text{for } i = n. \end{cases} \tag{11}$$

The probability that queue i is fully occupied is again estimated using the M/M/c/$c + k$ system, as shown in Equation (1). It is important to note that while the first queue experiences an arrival rate of λ, subsequent queues receive only a fraction of these arrivals, $\lambda(1 - P_1)$. The blocking probabilities for the queues are then computed as follows:

$$P_i = \begin{cases} \pi_{c_1^*+k_1}(\lambda, \mu_1^*, c_1^*, k_1), & \text{for } i = 1, \\ \pi_{c_i^*+k_i}(F, \mu_i^*, c_i^*, k_i), & \text{for } i = 2, \ldots, n. \end{cases} \tag{12}$$

Here, F is the number of jobs that complete service at queue 1 and is calculated, similar to [6], by $F = \lambda(1 - P_1)$.

For computational purposes, we have chosen to update the blocking probability using averaging with decreasing weights, such that the blocking probabilities are updated as

$$P_i^m = \begin{cases} P_i^{m-1}\frac{m-1}{m} + \pi_{c_i^*+k_i}(F, \mu_i^*, c_i^*, k_i)\frac{1}{m}, & \text{if } i > 1, \\ P_1^{m-1}\frac{m-1}{m} + \pi_{c_1^*+k_1}(\lambda, \mu_1^*, c_1^*, k_1)\frac{1}{m}, & \text{if } i = 1. \end{cases} \tag{13}$$

Combining the above yields Algorithm 3 as a second new heuristic.

Algorithm 3 The modified service rate and number of servers heuristic

```
 1: Initialize $m = 1$, $\mu_i^m = \mu_i$, $c_i^m = c_i$ for $i = 1, \ldots, n$
 2: $P_1^m = \pi_{c_1+k_1}(\lambda, \mu_1^m, c_1^m, k_1)$, $F^m = \lambda(1 - P_1^m)$
 3: $P_i^m = \pi_{c_i+k_i}(F^m, \mu_i^m, c_i^m, k_i)$ for $i = 2, \ldots, n$
 4: while $|P_1^m - P_1^{m-1}| \geq \delta$ do
 5:     $m = m + 1$
 6:     for $i = n, \ldots, 1$ do
 7:         Update the modified service rate using equation (10)
 8:         Update the modified number of servers using equation (11)
 9:         Update the blocking probability of queue $i$ using equation (13).
10:     end for
11:     Update $F^m = \lambda(1 - P_1^m)$
12: end while
```

4.2 Optimization

There is a vast increase in the number of scenarios for capacity allocation as the number of queues grows. Therefore, a fast performance evaluation is crucial when optimizing the capacity over the different nodes. Previously, these evaluations had to be done by simulation or a flawed approximation. The heuristics in Section 4.1 enable us to speed up these evaluations, reducing the time of an evaluation from minutes to seconds. For optimization, various methods can be employed, such as (i) grid search [10], (ii) Evolutionary Algorithm [4], and (iii) marginal allocation [21]. This section illustrates the optimization method that combines our heuristic with grid search in the case of *budget-constrained server allocation.*

In the optimization problem, each server at node i has a cost of b_i. The problem is to determine a server allocation that minimizes the rejection probability at the first node, $P_1(c_1,\ldots,c_n)$, where the total cost should not exceed a budget B. The mathematical formulation of the optimization problem is then given by:

$$\begin{aligned} \min_{(c_1,\ldots,c_n)} \quad & P_1(c_1,\ldots,c_n) \\ \text{S.t.} \quad & \sum_{i=1}^{n} b_i c_i \leq B \\ & c_i \in \mathbb{Z}^+ \quad \text{for } i=1,\ldots,n. \end{aligned}$$

The algorithm to approximate the optimal allocation of servers can be found in Algorithm 4 and involves three main steps. First, in line 3 of the pseudocode, all possible combinations of server allocations are generated, where a server allocation is represented by $(c_1,c_2,\ldots,c_n)$, with c_i the number of servers allocated to node i. Each combination must satisfy the budget constraint, ensuring that the total cost does not exceed B, that is, $\sum_{i=1}^{n} b_i c_i \leq B$. In line 4 we exclude server allocations where the remaining budget allows for adding at least one more server. To further limit the number of evaluations needed, we exclude server allocations that are unlikely to be optimal as a result of large speed differences, that is, $\frac{\mu_i c_i}{\mu_1 c_1} < 0.5$ or $\frac{\mu_i c_i}{\mu_1 c_1} > 1.5$. This step helps to eliminate suboptimal allocations.

For each of the remaining server allocations, we evaluate the blocking probability P_1 using a heuristic. Finally, we compare the blocking probabilities for all evaluated allocations and select the one with the lowest P_1. This allocation is optimal according to the heuristic, as it minimizes the likelihood of job blocking at the first node, thereby enhancing the overall performance of the queueing system.

Other optimization problems, such as the *cost-minimizing server allocation*, can also be addressed using new heuristic evaluations. This is a similar optimization problem, but instead of minimizing the blocking probability under a budget constraint, we aim to minimize the total cost while adhering to a maximum allowable blocking probability. This yields the following formulation:

$$\begin{aligned} \min_{(c_1,\ldots,c_n)} \quad & \sum_{i=1}^{n} c_i b_i \\ \text{S.t.} \quad & P_1(c_1,\ldots,c_n) \leq P_1^{max} \\ & c_i \in \mathbb{Z}^+ \quad \text{for } i=1,\ldots,n \end{aligned}$$

The optimal server allocation for this problem can be determined using a method similar to the algorithm described above.

Algorithm 4 Optimal budget constrained server allocation based on heuristic

```
 1: Input: Budget $B$, cost of a server at node $i$ as $b_i$ for $i = 1, \ldots, n$
 2: Output: Optimal server allocation $(c_1^{\#}, \ldots, c_n^{\#})$ with the lowest $P_1$
 3: Generate all possible server allocations $(c_1, \ldots, c_n)$ such that $\sum_{i=1}^{n} b_i c_i \leq B$
 4: Remove all server allocations where $\sum_{i=1}^{n} b_i c_i < B - \min_i(b_i)$
 5: Remove all server allocations where $\frac{\mu_i c_i}{\mu_1 c_1} < 0.5$ or $\frac{\mu_i c_i}{\mu_1 c_1} > 1.5$
 6: Initialize $(c_1^{\#}, \ldots, c_n^{\#}) = \emptyset$ and $P_1^{\#} = \infty$
 7: for each server allocation $(c_1, \ldots, c_n)$ do
 8:     Evaluate $P_1$ using a heuristic for allocation $(c_1, \ldots, c_n)$
 9:     if $P_1 < P_1^{\#}$ then
10:         $P_1^{\#} = P_1$
11:         $(c_1^{\#}, \ldots, c_n^{\#}) = (c_1, \ldots, c_n)$
12:     end if
13: end for
14: return $(c_1^{\#}, \ldots, c_n^{\#})$
```

5 Results

In this section, we present an evaluation of the four approximation methods for ASB within tandem queues. Our analysis focuses on identifying which approximation provides the most accurate representation of blocking probabilities when jobs are held up by subsequent nodes. In Section 5.1, we compare these approximations by evaluating their performance in various scenarios. In Section 5.2 we provide preliminary results that demonstrate the application of our proposed heuristic to the budget-constrained server allocation problem.

5.1 Accuracy approximations

The heuristics are tested on various scenarios of n nodes in tandem with $n \in \{2, \ldots, 5\}$. The arrival process is Poisson with rate λ to the first node. Each node i has a number of servers c_i and a service rate μ_i. The aggregate service rate of the queue with the smallest service capacity is $\min_i\{c_i\mu_i\}$. Our primary interest is in scenarios where roughly $\lambda \approx \min_i\{c_i\mu_i\}$, as these correspond to realistic settings that are also challenging to approximate. The generated scenarios are presented in Table 1. In this context, P_1 is determined by simulation, denoted P_1^{sim}, and the mean value of P_1 is obtained by averaging across all scenarios of the respective row in Table 1. The number of possible scenarios quickly increases with more nodes; therefore, we have chosen to limit the number of parameter settings when increasing the number of queues. This explains why the mean P_1 decreases when moving from the $n = 3$ scenarios to the $n = 4$ scenarios, despite an expected increase caused by the additional strain of after-service blocking from an extra queue.

Each of the $23,121$ scenarios has a warm-up period of $50,000$ service completions. The simulation ends when P_1 has reached a stable average, which we have chosen

Table 1. Overview of simulation scenarios

n	c_i	μ_1	$\frac{c_i\mu_i}{c_1\mu_1}$	$\frac{\lambda}{min(c_i\mu_i)}$	# scenarios	Mean ${P_1}^{\text{sim}}$, (min; max)
2	$\{1, 5, 10, 20\}$	1.0	$\{0.7, 0.8, \ldots, 1.3\}$	$\{0.7, 0.8, \ldots, 1.3\}$	784	$0.29, (0.00; 0.63)$
3	$\{1, 5, 10, 20\}$	1.0	$\{0.8, 0.9, \ldots, 1.2\}$	$\{0.8, 0.9, \ldots, 1.2\}$	8000	$0.29, (0.02; 0.63)$
4	$\{5, 10, 20\}$	1.0	$\{0.9, 1.0, 1.1\}$	$\{0.9, 1.0, 1.1\}$	6561	$0.22, (0.07; 0.38)$
5	$\{5, 20\}$	1.0	$\{0.9, 1.0, 1.1\}$	$\{0.9, 1.0, 1.1\}$	7776	$0.23, (0.07; 0.39)$

such that the confidence interval (CI) of the blocking probability is less than 5% of the mean value of P_1. If this condition is not met, then the simulation is continued for another $10,000$ service completions, and repeated until the stability condition is met. For the convergence of the heuristic algorithms, we take $\delta = 0.000001$. The error of heuristic H is then given by

$$\Delta^{(H)} = |P_1^{(H)} - {P_1}^{\text{sim}}|.$$

The errors of the heuristics can be found in Table 2 and Figure 3. Figure 3 shows only the three best-performing heuristics to allow a clearer and more accurate comparison of these approximations. As can be seen, MS&C performs the best over all scenarios. This heuristic has both the smallest mean error and the smallest maximum error. From Figure 3, we observe that MS&C is able to maintain small errors, also when the number of queues, and thus the complexity of the system, increases. The Loss approximation turns out to perform quite well. This approximation naturally breaks down when the service rate imbalance between queues increases, for instance, when the speed of other queues is significantly lower than that of the first queue. Consequently, this approximation is not suitable for optimization as it is based on only the first queue and ignores the effect of ASB. The effectiveness of BR notably lags behind that of the other heuristics. As the benchmark of [6] is constructed for cases with $\rho < 1$, we split the scenarios between $\rho^{\#} < 1$ and $\rho^{\#} \geq 1$, with $\rho^{\#} = \min_{i=1,\ldots,n}\{\frac{\lambda}{c_i\mu_i}\}$, see Figures 4 and 5. The boxplots for $\rho^{\#} < 1$, in Figure 4, reveal that the new heuristics clearly outperform the Loss approximation and BR. Moreover, for $\rho^{\#} \geq 1$ it also becomes clear that MS&C overall has the best performance.

The impact of the load on the blocking probability, both for MS&C and the simulation, is visualized in Figures 6 and 7, for a two- and three-node tandem system, respectively. The parameters for the scenario of Figure 6 are $c_1 = 10$, $c_2 = 10$, $\mu_1 = 1.0$, with μ_2 equal to 1.0 and 0.8 for the left- and right-hand sides of the figure, respectively. The parameters of Figure 7 are $c_1 = 10$, $c_2 = 10$, $c_3 = 10$, $\mu_1 = 1.0$, with μ_2 and μ_3 equal to 1.0 or 0.8 for the different scenarios.

Table 2. Mean error for different scenarios

n	**Loss**	**BR**	**MS**	**MS&C**
	(min; max)	(min; max)	(min; max)	(min; max)
2	0.03, (0.00; 0.16)	0.10, (0.00; 0.22)	0.02, (0.00; 0.15)	0.02, (0.00; 0.12)
3	0.04, (0.00; 0.30)	0.39, (0.00; 0.83)	0.03, (0.00; 0.31)	0.02, (0.00; 0.13)
4	0.04, (0.00; 0.15)	0.42, (0.02; 0.82)	0.03, (0.00; 0.15)	0.01, (0.00; 0.08)
5	0.04, (0.00; 0.18)	0.43, (0.02; 0.82)	0.03, (0.00; 0.18)	0.02, (0.00; 0.12)

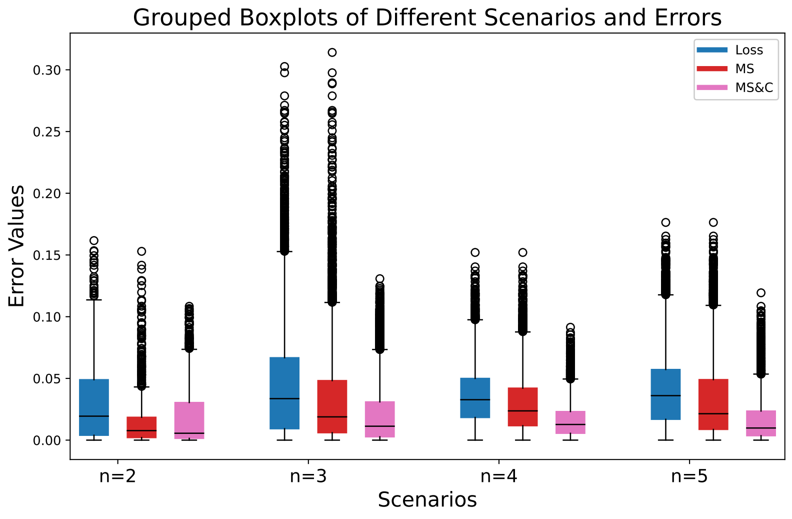

Fig. 3. Box plots of the errors of the three best performing approximations

The load varies due to λ ranging from 0 to 40. The results show that the impact of the load is similar for both the heuristic and simulation. When queue 2, queue 3, or both become a bottleneck (that is, for $\mu_2 = 0.8$ and/or $\mu_3 = 0.8$), we see that the error increases somewhat with the arrival rate, although the functional impact of the load on the heuristic is similar to the simulation.

Runtime To illustrate the runtime efficiency of the heuristics, we tested four simple scenarios, with $n = 2, 3, 4$ or 5. Each queue has the same number of servers and service rate with $c_i = 10$ and $\mu_i = 1.0$, while fixing the arrival rate at $\lambda = 10.0$. The resulting runtimes and iteration counts required for convergence are presented in Table 3. As observed, all approximations significantly reduce

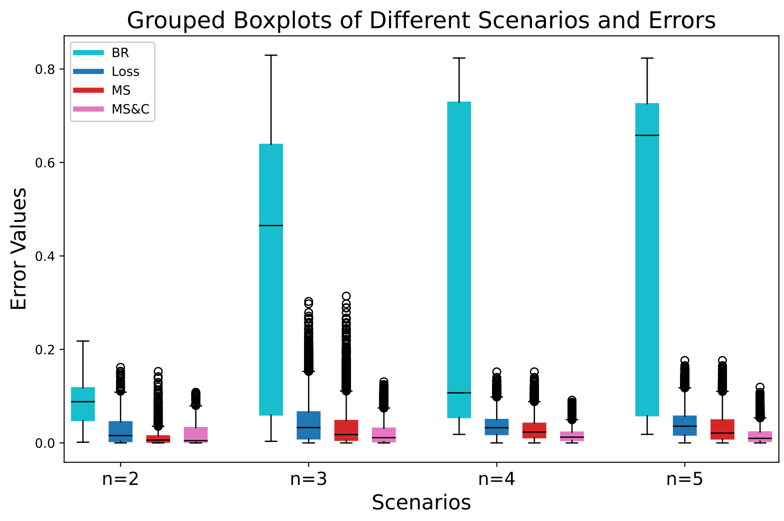

Fig. 4. Box plots of the errors of approximations for $\rho^{\#} < 1.0$

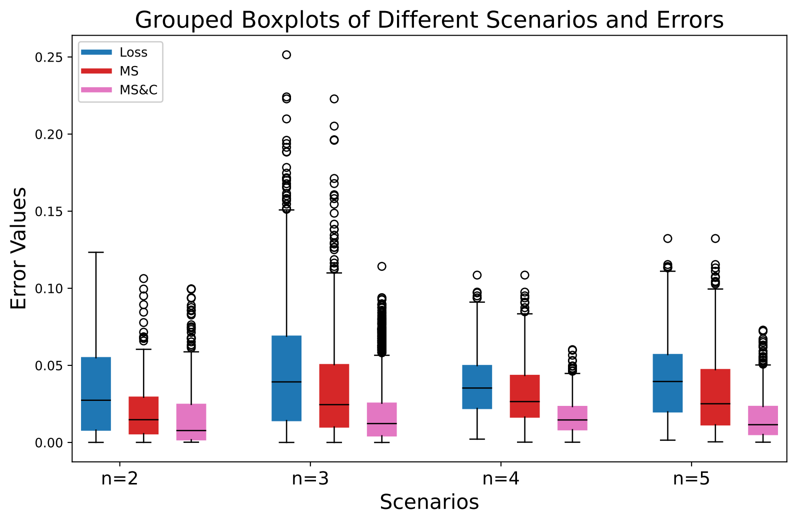

Fig. 5. Box plots of the errors of the three best performing approximations for $\rho^{\#} \geq 1.0$

runtime compared to the simulation approach for the same scenario. Among the heuristics, MS achieves the shortest runtime, followed by MS&C, with both providing a clear runtime advantage over the heuristic BR. The iteration counts needed for convergence, also shown in Table 3, further support this result. While

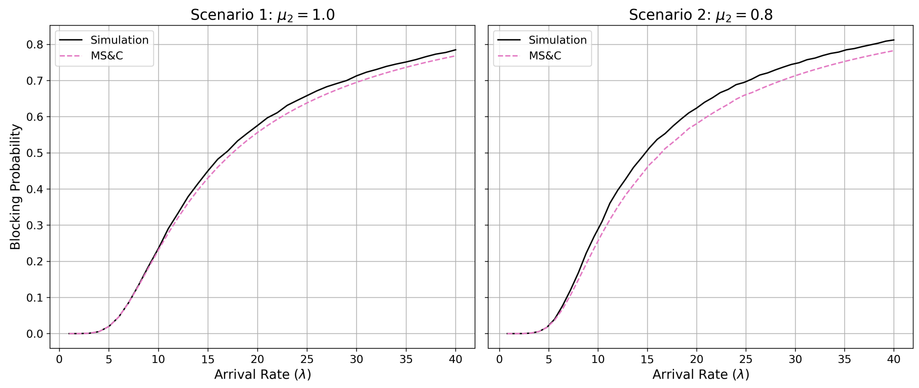

Fig. 6. Comparison of the simulation and the heuristic for a system with two nodes with a varying arrival rate

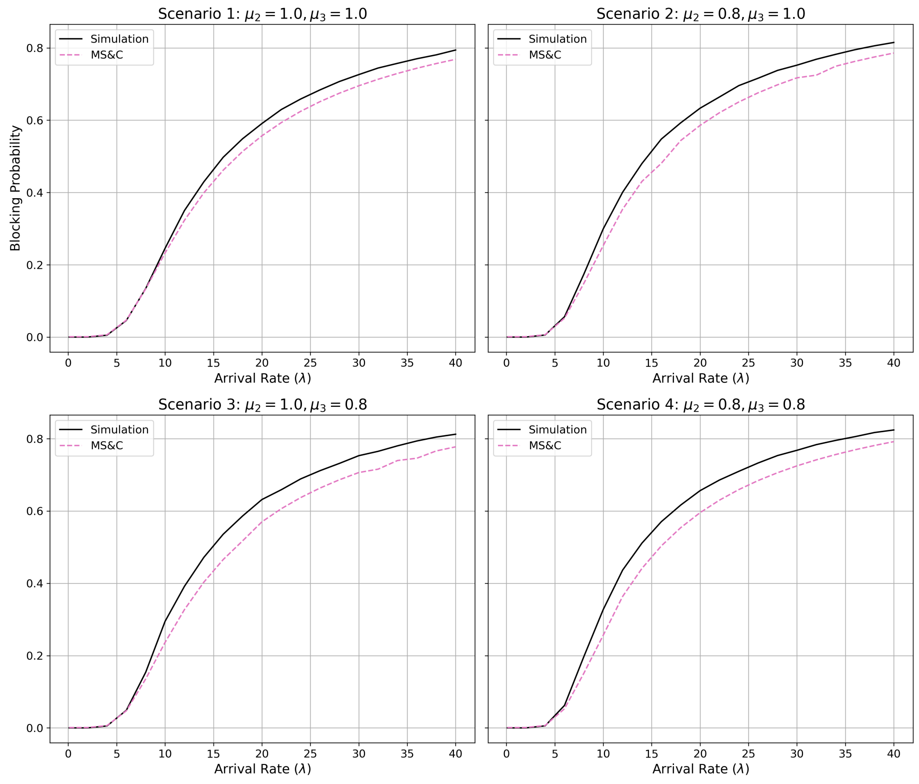

Fig. 7. Comparison of the simulation and the heuristic for a system with three nodes with a varying arrival rate

MS&C is not the fastest overall, it still offers a substantial runtime reduction, making it a viable option for optimization, especially with respect to simulation times.

Table 3. Runtime and number of iterations for different methods

n	**Sim.** (sec; iter)	**M/M/c/$c+k$** (sec; iter)	**BR** (sec; iter)	**MS** (sec; iter)	**MS&C** (sec; iter)
2	54.470 (-)	0.000 (-)	0.099 (767)	0.002 (9)	0.097 (369)
3	56.587 (-)	0.000 (-)	1.679 (8590)	0.003 (10)	0.136 (250)
4	78.707 (-)	0.000 (-)	1.780 (8600)	0.004 (12)	0.212 (231)
5	97.804 (-)	0.000 (-)	1.776 (8551)	0.005 (13)	0.245 (228)

5.2 Optimization

We illustrate the optimization based on a relatively small tandem system of three queues. For such a system, we are able to evaluate the different server configurations using simulation. For larger systems, this approach becomes impractical; however, the heuristic remains feasible, as it significantly accelerates the evaluation process. Specifically, we chose the following parameters: $\lambda = 10, \mu_1 = 2.0, \mu_2 = 1.0, \mu_3 = 0.5$, each server having a cost of 1 ($b_1 = b_2 = b_3 = 1.0$), and a budget B ranging from 10 to 50. For example, $B = 10$ represents that there is a budget for 10 servers in total. Table 4 shows the optimization results when the evaluations are based on both the simulation and MS&C. The optimal allocation according to the heuristic is close to that of the optimal allocation according to the simulation; the blocking probabilities typically differ only by a few percentage points. Overall, the optimization results indicate that the simulation leads to somewhat more server capacity further downstream of the tandem. Moreover, $c_1\mu_1 \geq c_2\mu_2 \geq c_3\mu_3$, which implies that it is desirable to have the largest service capacity upstream.

6 Conclusion & discussion

This paper investigated the impact of ASB in tandem queues and introduced two new heuristics to approximate its effects. The best performing heuristic is MS&C and offers several advantages over the existing methods. First, it delivers more accurate estimations compared to previous approaches. Second, it enables rapid evaluations and adjustments, making it suitable for real-time decision-making and capacity allocation. Finally, the heuristic is transparent, providing insight in the impact of different parameters on system performance.

Table 4. Server allocations based on optimization using simulation and heuristic

B	$(c_1^{*sim}, c_2^{*sim}, c_3^{*sim})$	$(c_1^{*heur}, c_2^{*heur}, c_3^{*heur})$	P_1^{*sim}	P_1^{*heur}	$P_1^{*heur} - P_1^{*sim}$
10	(2,3,5)	(2,4,4)	0.79	0.81	0.02
20	(4,6,10)	(4,7,9)	0.55	0.57	0.02
30	(6,9,15)	(7,9,14)	0.32	0.34	0.02
40	(8,12,20)	(10,10,20)	0.11	0.14	0.03
50	(12,14,24)	(12,14,24)	0.02	0.02	0.00

With the improved heuristic, there is clear potential for practical applications. For instance, the heuristic can be easily applied in a Decision Support System for real-world queueing systems such as manufacturing lines, telecommunications networks, and service industries. It enhances the ability to optimize server allocations, thereby improving operational efficiency and without being reliant on time-consuming simulations, or black-box prediction models. This makes it a valuable tool for professionals who want to effectively balance cost and service quality.

Future work While our heuristic shows promising performance, there are some limitations. For example, the model assumes a specific structure of tandem queues and may not be directly applicable to networks with more complex topologies. Extending the heuristic to handle more complex queueing networks is an interesting direction for future research. We envisage that the first extension should focus on feedforward networks. In networks with feedback loops, the phenomenon of *deadlocks* will be crucial, which occur when blocked jobs are directly or indirectly blocked by themselves. These deadlocks need to be resolved, adding another layer of complexity to the system. Furthermore, the heuristic relies on assumptions about the arrival (Poisson process) and service processes (exponentially distributed), which may not hold in all real-world scenarios. Albeit the impact of the service time distribution is usually small for systems with a considerable number of servers, it is of interest to determine how the heuristic needs to be adapted to handle general arrival and service processes.

Apart from the approximation, future research could also focus on the optimization part of this research. During experiments it seemed that the following rule seems to hold for optimal allocations: $c_1\mu_1 \geq c_2\mu_2 \geq \ldots \geq c_n\mu_n$. This observation is as expected; upstream queues require more capacity due to the cumulative effect of strain and stochasticity introduced by ASB in downstream queues. As jobs progress through the system, any delays or blockages in later queues propagate backward, increasing the workload and variability experienced by earlier queues. This requires additional capacity in upstream queues to manage the compounded effects of downstream inefficiencies. In contrast, downstream queues require comparatively less capacity, since each subsequent queue only needs to handle the additional workload generated by the ASB of the preceding upstream queues. Therefore, the capacity of the i^{th} queue must be sufficient to process the increased workload resulting from blockages in the $n-i$ queues after it. This could be further researched to determine whether this optimization process can be sped up without the need for evaluations.

Appendix: Queues with fractional number of servers

Here we present the performance analysis for the M/M/c and M/M/c/$c+k$ systems in case c is not necessarily integer. The results rely on the Gamma function, given by,

$$\Gamma(c) = \int_0^\infty t^{c-1}e^{-t}\mathrm{d}t, \tag{14}$$

and the upper incomplete Gamma function,

$$\Gamma(c,x) = \int_x^\infty t^{c-1}e^{-t}\mathrm{d}t. \tag{15}$$

M/M/c

The M/M/c queue is a queue with Poisson arrivals, with rate λ, and exponential service times, with rate μ, and c servers. While not resembling our ASB situation due to the fact that all jobs are accepted in this case, we can still use the knowledge from these formulas in approximations. The steady-state probabilities of the number of customers in the system are given by [16], with $\rho = \frac{\lambda}{\mu c}$,

$$\theta_i = \begin{cases} \theta_0 \frac{(c\rho)^i}{i!}, & \text{for } 0 < i < c, \\ \theta_0 \frac{(c\rho)^i c^{c-i}}{c!}, & \text{for } i \geq c, \\ \left[\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!} + \frac{(c\rho)^c}{c!} \cdot \frac{1}{1-\rho}\right]^{-1}, & \text{for } i = 0. \end{cases} \tag{16}$$

The probability of waiting $\theta_{i\geq c}$ is then given by:

$$\theta_{i\geq c} = \sum_{i=c}^{\infty} \theta_i. \tag{17}$$

To improve the speed of the approximations, these calculations are rewritten, using [14], such that the value c does not have to be integer. It is rewritten to:

$$\theta_0 = \left[\frac{e^{c\rho}}{\Gamma(c)}\Gamma(c,c\rho) + \frac{(c\rho)^c}{\Gamma(c+1)}\frac{1}{(1-\rho)}\right]^{-1} \tag{18}$$

and

$$\theta_{i\geq c} = 1 - \theta_{i<c} = 1 - \theta_0 \frac{e^{c\rho}}{\Gamma(c)}\Gamma\left(c, c\rho\right). \tag{19}$$

M/M/c/$c+k$

The steady-state probabilities of the number of customers in the system in the M/M/c/$c+k$ queue are given by, with $a = \frac{\lambda}{\mu}$:

$$\pi_i = \begin{cases} \pi_0 \frac{a^i}{i!}, & \text{for } 0 < i < c, \\ \pi_0 \frac{a^i}{c^{i-c} c!}, & \text{for } c \leq i \leq c+k, \\ \left[\sum_{j=0}^{c} \frac{a^j}{j!} + \frac{a^c}{c!} \sum_{j=c+1}^{c+k} \left(\frac{a}{c}\right)^{j-c}\right]^{-1} & \text{for } i = 0. \end{cases} \quad (20)$$

For the heuristics we also require the expected waiting time $E[W_q]$ in the M/M/c/$c+k$ model. This is computed using

$$E[W_q] = \frac{E[L_q]}{\lambda \cdot (1 - \pi_{c+k})}, \quad (21)$$

where the expected number of customers in the queue $E[L_q]$ is given by

$$E[L_q] = \sum_{i=c}^{c+k} (i-c)\pi_i. \quad (22)$$

To handle cases where c is not required to be an integer, Equation (22) can be rewritten as:

$$E[L_q] = \begin{cases} \frac{c^c \pi_0 (k+1)}{\Gamma(c+1)}, & \text{for } \rho = 1, \\ \frac{\pi_0}{\Gamma(c+1)} (c\rho)^c \left[\frac{\rho - \rho^{k+1}}{(1-\rho)^2} - \frac{k\rho^{k+1}}{(1-\rho)}\right], & \text{otherwise.} \end{cases} \quad (23)$$

Moreover, π_0 can be rewritten as (2) using a similar argument as for the M/M/c queue.

References

1. Akyildiz, I.F., von Brand, H.: Exact solutions for networks of queues with blocking-after-service. Theoretical Computer Science **125**(1), 111–130 (1994)
2. Avi-Itzhak, B., Yadin, M.: A sequence of two servers with no intermediate queue. Management Science **11**(5), 553–564 (1965)
3. Baber, J.M.A.: Queues in series with blocking. Cardiff University (United Kingdom) (2008)
4. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation **1**(1), 1–23 (1993)
5. Bamezai, A., Melnick, G., Nawathe, A.: The cost of an emergency department visit and its relationship to emergency department volume. Annals of Emergency Medicine **45**(5), 483–490 (2005)

6. Bretthauer, K.M., Heese, H.S., Pun, H., Coe, E.: Blocking in healthcare operations: A new heuristic and an application. Production and Operations Management **20**(3), 375–391 (2011)
7. Dallery, Y., Frein, Y.: On decomposition methods for tandem queueing networks with blocking. Operations Research **41**(2), 386–399 (1993)
8. Dieleman, N., Berkhout, J., Heidergott, B.: A neural network approach to performance analysis of tandem lines: The value of analytical knowledge. Computers & Operations Research **152**, 106124 (2023)
9. El-Darzi, E., Vasilakis, C., Chaussalet, T., Millard, P.: A simulation modelling approach to evaluating length of stay, occupancy, emptiness and bed blocking in a hospital geriatric department. Health Care Management Science **1**, 143–149 (1998)
10. Ensor, K.B., Glynn, P.W.: Stochastic optimization via grid search. Lectures in Applied Mathematics-American Mathematical Society **33**, 89–100 (1997)
11. Gordon, W.J., Newell, G.F.: Cyclic queuing systems with restricted length queues. Operations Research **15**(2), 266–277 (1967)
12. Haraden, C., Resar, R.: Patient flow in hospitals: Understanding and controlling it better. Frontiers of Health Services Management **20**(4), 3 (2004)
13. Hunt, G.C.: Sequential arrays of waiting lines. Operations Research **4**(6), 674–683 (1956)
14. Jagers, A., Van Doorn, E.A.: On the continued Erlang loss function. Operations Research Letters **5**(1), 43–46 (1986)
15. Koizumi, N., Kuno, E., Smith, T.E.: Modeling patient flows using a queuing network with blocking. Health Care Management Science **8**, 49–60 (2005)
16. Kulkarni, V.G.: Introduction to modeling and analysis of stochastic systems, vol. 1. Springer (2011)
17. Latouche, G., Neuts, M.F.: Efficient algorithmic solutions to exponential tandem queues with blocking. SIAM Journal on Algebraic Discrete Methods **1**(1), 93–106 (1980)
18. Osorio, C., Bierlaire, M.: An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. European Journal of Operational Research **196**(3), 996–1007 (2009)
19. Palmer, G.I., Harper, P.R., Knight, V.A.: Modelling deadlock in open restricted queueing networks. European Journal of Operational Research **266**(2), 609–621 (2018)
20. Pourbabai, B.: Tandem behavior of a telecommunication system with repeated calls: II, a general case without buffers. European Journal of Operational Research **65**(2), 247–258 (1993)
21. Rolfe, A.J.: A note on marginal allocation in multiple-server service systems. Management Science **17**(9), 656–658 (1971)
22. Seo, D.W., Lee, H.: Stationary waiting times in m-node tandem queues with production blocking. IEEE Transactions on Automatic Control **56**(4), 958–961 (2011)
23. Van Vuuren, M., Adan, I.J., Resing-Sassen, S.A.: Performance analysis of multi-server tandem queues with finite buffers and blocking. OR Spectrum **27**, 315–338 (2005)