Stochastic Models

ISSN: 1532-6349 (Print) 1532-4214 (Online) Journal homepage: http://www.tandfonline.com/loi/lstm20

Control of a tandem queue with a startup cost for the second server

A. Hristov, S. Bhulai, R. D. van der Mei & J. W. Bosman

To cite this article: A. Hristov, S. Bhulai, R. D. van der Mei & J. W. Bosman (2018): Control of a tandem queue with a startup cost for the second server, Stochastic Models, DOI: 10.1080/15326349.2018.1491314

To link to this article: https://doi.org/10.1080/15326349.2018.1491314

STOCHASTIC MODELS
https://doi.org/10.1080/15326349.2018.1491314

Taylor & Francis
Taylor & Francis Group

Control of a tandem queue with a startup cost for the second server

A. Hristov[a,b], S. Bhulai[b], R. D. van der Mei[a,b], and J. W. Bosman[a,b]

[a]Center for Mathematics and Computer Science (CWI) Amsterdam, The Netherlands; [b]Faculty of Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

ABSTRACT
Various systems across a broad range of applications contain tandem queues. Strong dependence between the servers has proven to make such networks complicated and difficult to study. Exact analysis is rarely computationally tractable and sometimes not even possible. Nevertheless, as it is most often the case in reality, there are costs associated with running such systems, and therefore, optimizing the control of tandem queues is of main interest from both a theoretical and a practical point of view. Motivated by this, the present paper considers a tandem queueing network with linear holding costs and a startup cost for the second server. In our work, we present a rather intuitive, easy to understand, and at the same time very accurate technique to approximate the optimal decision policy. Extensive numerical experimentation shows that the approximation works extremely well for a wide range of parameter combinations.

ARTICLE HISTORY
Received 16 March 2018
Revised 17 June 2018
Accepted 17 June 2018

KEYWORDS
Approximation algorithm; MDP; optimal control; startup costs; tandem queue;

1. Introduction

Queueing systems in which the departures from one server become the arrivals to a downstream server are often modeled as tandem queues. These models have proven to be very challenging to analyze,[1–4] and despite decades of research, there are still many open problems without an analytic solution.[5–8] At the same time, tandem queues abound in applications, and therefore, the study of these systems is also important for practice. Moreover, in cases when one has a certain control over the network, the analysis of the model becomes crucial for optimal management of the application.[9,10] For example, in some practical situations, it is possible to temporarily switch off certain nodes in order to protect servers further down the network from overflow. Systems with this feature can be found in traffic control,[11] transportation, manufacturing, and many other fields.[12–15]

CONTACT A. Hristov a.v.hristov@cwi.nl Center for Mathematics and Computer Science (CWI), Amsterdam, XG 1098, The Netherlands.

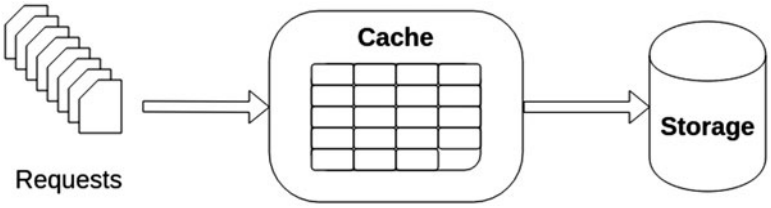

Figure 1. Cache used in data centers.

One particular application that can be modeled as a tandem network and uses such control techniques is the caching in computer databases in data centers (illustrated in Figure 1). By implementing the cache mechanism, the contention at the data storage might be strongly reduced by first accumulating write operations in the cache and only afterwards processing them in the database. Furthermore, to avoid overload, new writes to the cache are blocked whenever there are already a certain number of requests in it. The corresponding threshold value is generally referred to as the *high water mark*. Motivated by this application, in this paper we study two-node tandem queueing networks where one can switch on/off any of the servers.

Note that switching off a server results in reducing the service capacity of the system. Therefore, there is a trade-off for such tandem queues in balancing the requirement for 'good' performance while minimizing the resource usage. One common technique to capture this trade-off is to introduce different penalties (i.e., holding costs) for jobs waiting in the various queues.[16,17] This way, one combines both metrics (e.g., the sojourn time spent in the system and the required resources) in a single indicator. The problem is, therefore, reduced to the following question: "How to control the system in order to minimize the costs associated with it?" However, this challenge is still analytically intractable, in spite of the considerable amount of research and the great importance from a practical point of view.

Next to that, changing the 'state' of a server in such tandem networks (i.e., shutting it down or starting it up) might also require resources on its own. For instance, there are certain costs associated with establishing a data transfer between the cache and the corresponding database. Motivated by this, we study a system with two servers that has a startup cost associated with the second node. Our framework can be considered as a generalization of the ones researched in.[17,18]

In the following section, we introduce the model. In Section 3, we illustrate what the optimal decision policy for such systems looks like. Subsequently, in Section 4, we propose an approximation method for obtaining the optimal control policy. The accuracy of the presented technique is discussed in Section 5. Finally, in Section 6, we conclude with a summary and discuss ideas for possible further research.

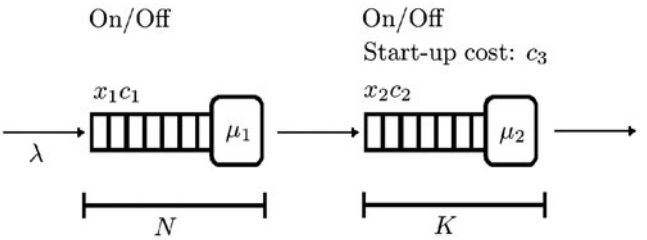

Figure 2. Illustration of the model. (a) $s_2 = 00$ and (b) $s_2 = 1$

2. Model

We consider a two-node tandem queue with single servers at both queues. Jobs arrive at the first node and after receiving service there, they are transferred to the second one. Subsequently, jobs are served at the second node and leave the system. We assume Poisson arrivals with rate λ at queue 1 and exponential service times with mean $\beta_i = 1/\mu_i$ at server $i \in \{1, 2\}$. Moreover, both nodes are serving at maximum one job at a time according to the First In First Out (FIFO) regime. The queues are taken to be of a finite size N and K for queue 1 and queue 2, respectively.

Each job in queue $i \in \{1, 2\}$ generates a holding cost $c_i \geq 0$ per time unit. Moreover, there is a startup cost for the second server, denoted by $c_3 \geq 0$. To optimize the cost, one can control the system at any point in time by switching on or off any of the two servers. We pose two restrictions on the possible decisions: (1) the first server cannot be working when the buffer space at the second node is full (i.e., when there are K jobs in the second queue) and (2) the second server switches off whenever it becomes idle. Furthermore, to avoid a trivial solution of never switching off the first server, we assume that $c_2 > c_1$. This way, for certain system states when the second server is not operating it might be optimal to keep the jobs in the first queue. Next to that, the startup cost of server 2 creates an additional trade-off between serving jobs at the second node as soon as possible and waiting for enough jobs before switching on the server. Therefore, the goal is to identify the decision policy minimizing the average costs per time unit. Figure 2 gives an illustration of the model.

To calculate the optimal policy, we formulate the system as a Markov Decision Process (MDP) with state space $\mathcal{S} := \{0, 1, ..., N\} \times \{0, 1, ..., K\} \times \{0, 1\}$, where state (x_1, x_2, s_2) corresponds to having x_i number of jobs at node $i \in \{1, 2\}$ and server 2 being *off* for $s_2 = 0$ or *on* for $s_2 = 1$. Note that we do not have to explicitly include the state of node 1 in our model due to the following two model properties: (1) the service times are exponentially distributed and exhibit the memoryless property and (2) the second server can be switched on/off instantaneously at no cost.

The action space consists of four possible actions $a \in \mathcal{A} = \{1, 2, 3, 4\}$ defined as follows:

1. switch off both servers;
2. switch on server 1 and switch off server 2;
3. switch on server 2 and switch off server 1;
4. switch on both of the servers.

Now, we can formulate the Bellman equations for this MDP:

$$g + V(x_1, x_2, s_2) = c_1 x_1 + c_2 x_2 + \min_{a \in \mathcal{A}} T_a(x_1, x_2, s_2),$$

where V denotes the value function. Moreover, g denotes the long-term average costs per time unit and $T_a(x_1, x_2, s_2)$ (for $a \in \mathcal{A}$ and $(x_1, x_2, s_2) \in \mathcal{S}$) are given by:

$$\begin{aligned}
T_1(x_1, x_2, s_2) &:= \lambda V(\min\{x_1 + 1, N\}, x_2, 0) + (1-\lambda) V(x_1, x_2, 0);\\
T_2(x_1, x_2, s_2) &:= \lambda V(\min\{x_1 + 1, N\}, x_2, 0) + \mu_1 V(x_1 - 1, x_2 + 1, 0)\\
&\quad + \mu_2 V(x_1, x_2, 0);\\
T_3(x_1, x_2, s_2) &:= c_3(1 - s_2) + \lambda V(\min\{x_1 + 1, N\}, x_2, 1)\\
&\quad + \mu_2 V\big(x_1, x_2 - 1, I(x_2)\big) + \mu_1 V(x_1, x_2, 1);\\
T_4(x_1, x_2, s_2) &:= c_3(1 - s_2) + \lambda V(\min\{x_1 + 1, N\}, x_2, 1)\\
&\quad + \mu_1 V(x_1 - 1, x_2 + 1, 1) + \mu_2 V\big(x_1, x_2 - 1, I(x_2)\big),
\end{aligned}$$

where the rates are scaled in such a way that $\lambda + \mu_1 + \mu_2 = 1$. Furthermore, $I(x_2) = 0$ if $x_2 = 1$, and $I(x_2) = 1$ otherwise. Moreover, recall that some actions are not possible under certain values for x_1 and x_2. More precisely, for $x_1 = 0$ or $x_2 = K$, actions 2 and 4 (i.e., T_2 and T_4) are not permitted; and for $x_2 = 0$, actions 3 and 4 (i.e., T_3 and T_4) are not permitted.

3. Optimal policy

In this section, we study the optimal decision policy based on results derived by numerically solving the MDP by applying the value iteration technique. As an example, we use two systems with the following parameters: $N = 750$, $K = 200$, $c_1 = 0.1$, $c_2 = 1$, $c_3 = 1500$, $\lambda = 1$. The service rates of the servers are taken to be:

- $\mu_1 = 2.2$ and $\mu_2 = 4$ in the first example,
- $\mu_1 = 4$ and $\mu_2 = 2.2$ in the second example.

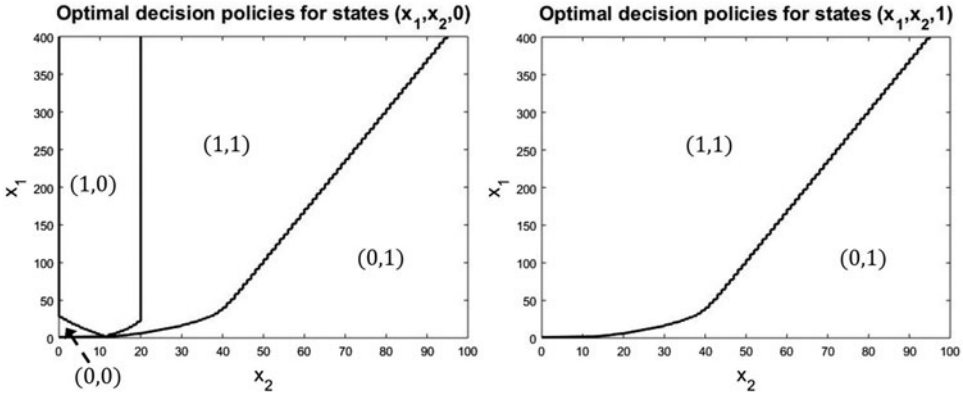

Figure 3. Optimal decision policy for $\mu_1 = 2.2$ and $\mu_2 = 4$. (a) $s_2 = 0$ and (b) $s_2 = 1$

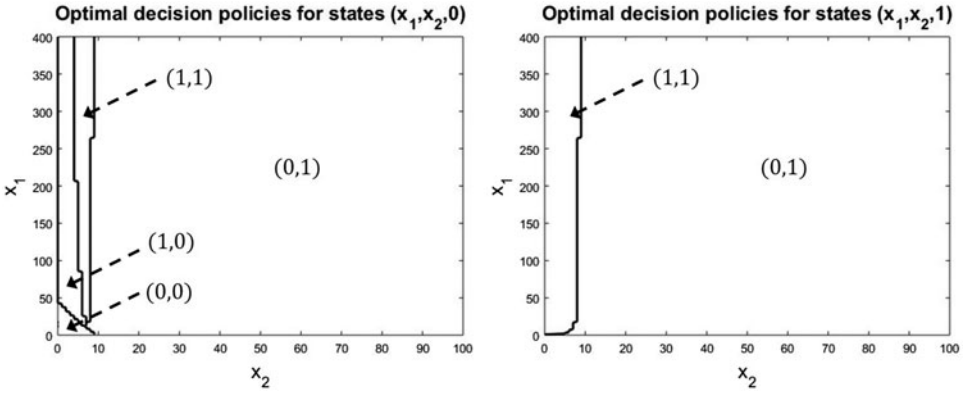

Figure 4. Optimal decision policy for $\mu_1 = 4$ and $\mu_2 = 2.2$.

The corresponding optimal decision policies are shown in Figures 3 and 4. Note that we present only the states for which $0 \leq x_1 \leq 400$ and $0 \leq x_2 \leq 100$ in order to exclude possible boundary effects.

Analyzing numerous cases for various parameter sets, we suspect that the optimal policy for such a queueing network is of a threshold type. We denote the different state-space regions, where a certain action is optimal, as follows:

region (0, 0) – the optimal action is 1, namely, both of the servers should be off;

region (1, 0) – the optimal action is 2, namely, the first server should be on, whereas the second one – off;

region (0, 1) – the optimal action is 3, namely, the first server should be off, whereas the second one – on;

region (1, 1) – the optimal action is 4, namely, both of the servers should be on.

Note that the numerical approach used in the current section can be used only for systems of a relatively small size. In many real-world instances the large buffer sizes, N and K, require a more computationally efficient algorithm. As an example, to optimize the running costs associated with the database caching mechanism, one needs a method different than the value iteration technique, as the latter becomes computationally unfeasible. On the other hand, as discussed in Section 1, even the special case of no startup cost associated with the second server (i.e., the special case of $c_3 = 0$) has withstood an analytic analysis so far. Therefore, the goal of our research is to develop a scalable method with respect to N and K that approximates the optimal policy.

4. Approximation technique

In the following, we present our technique for approximating the optimal policy. Our approach to obtaining the control policy is to estimate the regions described in Section 3, rather than deriving an approximation of the value function. To do this, we decompose the original system into two sub-models. We refer to the parameters for these sub-models by appending a superscript $^{(1)}$ and $^{(2)}$, respectively.

Sub-model 1 consists of two single-server queues in a tandem setting. The input is modeled as a Poisson process and the service times of the jobs are assumed to be independent and exponentially distributed. There are holding costs $c_1^{(1)}$ and $c_2^{(1)}$ per time unit for each job waiting at the corresponding queue. In contrast to the main network analyzed in this paper, there is no startup cost for the second server. The optimal policy for such a system is proven to be defined by a switching curve, see.[17] In other words, for any given number of jobs $x_1^{(1)}$ at the first server, there is a threshold value $S^{(1)}(x_1)$ for the number of jobs at the second node, $x_2^{(1)}$. If $x_2^{(1)}$ exceeds the corresponding threshold, then it becomes optimal to switch off the first server. We denote the arrival rate as $\lambda^{(1)}$ and the service rates as $\mu_1^{(1)}$ and $\mu_2^{(1)}$.

Sub-model 2 consists of an $M/M/1/N$ queue with holding costs $c_2^{(2)}$ per time unit for each waiting job and a startup cost $c_3^{(2)}$. One can control the system by switching on/off the server. As in the main system analyzed in this paper, the server cannot be idle, i.e., it switches off whenever there are no jobs. The optimal policy for such a system is proven to be a threshold policy, see.[19] More specifically, it is optimal to switch on the server when the number of jobs in the system, $x_2^{(2)}$, exceeds a given threshold value, $S^{(2)}$. We denote the arrival rate for this system as $\lambda^{(2)}$ and the service rate as $\mu_2^{(2)}$.

Recall from Section 1 that an analytic solution for sub-model 1 is unavailable. However, there are a number of studies[6,17] that discuss methods which can be used to calculate the optimal threshold levels. Therefore, we assume that the optimal policy for sub-model 1 is given.

On the other hand, there is a closed-form solution for the optimal threshold value $S^{(2)}$ for sub-model 2, given by:

$$S^{(2)}\left(\lambda^{(2)}, \mu_2^{(2)}, c_2^{(2)}, c_3^{(2)}\right) = \sqrt{2\lambda^{(2)}\left(1 - \lambda^{(2)}/\mu_2^{(2)}\right)c_3^{(2)}/c_2^{(2)}}. \tag{1}$$

Therefore, one can derive the optimal policy also for this sub-model.

Intuitively, it might seem that simply combining the solutions of those two sub-models would give a good approximation of the optimal policy for the main model. However, this is not the case, mainly due to the complex dependency between the working regime of the first server and the arrival rate at the second server. Therefore, in our algorithm, we try to take into account this dependency.

4.1. Second server switched off

In this subsection, we show how to approximate each of the three switching curves (see Figures 3(a) and 4(a)) in the optimal decision policy in case the second server is switched off (i.e., for states $(x_1, x_2, 0)$, where $0 \leq x_1 \leq N$ and $0 \leq x_2 \leq K$). Considering the (0, 0, 0) state (i.e., an empty system) as a reference point, we denote the *first* threshold levels to correspond to the decision when to switch on the first server, i.e., the curve separating *region (0, 0)* from *region (1, 0)*. We characterize this curve by the function $p_1(x_1)$, where $0 \leq x_1 \leq N$. The value of the function gives the threshold value of x_2 for the corresponding $0 \leq x_1 \leq N$. Consequently, as the *second* curve, we consider the one describing when to switch on the second server, i.e., the transition between *region (1, 0)* and *region (1, 1)*. Similarly, we introduce the function $p_2(x_1)$ that defines this curve. Finally, the *third* set of threshold levels, $p_3(x_1)$, gives the states for which it is optimal to switch off the first server and have only the second one working – *region (0, 1)*. In the remainder of this subsection we outline how to estimate those three switching curves.

4.1.1. Switching on the first server

Under certain system parameters, the optimal decision is to keep both servers off whenever there are not many jobs in the network. Namely, if $c_2 > c_1$ it would be optimal to keep jobs waiting at the first queue rather than at the second one. Next to that, intuitively, the lower the load of the network is, the bigger this region should be. As a first step of approximating this

region, we determine the endpoints of the corresponding switching curve. Namely, we are interested in the value of $p_1(0)$ and the specific i_1' for which $p_1(i_1') = 0$. To obtain these values, we use the following equations:

$$p_1(0) = S^{(2)}(\lambda, \mu_2, c_1 + c_2, c_3),$$
$$i_1' = \left\lfloor \frac{S^{(2)}(\lambda, \mu_2, c_1, c_3) + S^{(2)}(\mu_1, \mu_2, c_1 + c_2, c_3) + S^{(2)}(\lambda, \mu_2, c_2, c_3)}{3} \right\rfloor,$$

where the operator $S^{(2)}$ denotes the threshold value determined by solving sub-model 2 with the corresponding parameters (see Equation (1)) and $\lfloor x \rfloor$ denotes the floor function that outputs the largest integer less than or equal to x.

The idea to take the average value over three solutions of sub-model 2 for obtaining i_1' is to incorporate three different regimes of the system. The first one being the regime just before taking the decision to switch on the first server and having jobs only at the first server. The second regime is when server 1 is switched on and there is a number of jobs in queue 1. This implies that the arrival rate to the second node is μ_1. In the third regime, the queue at server 1 is empty, and hence, there are no holding costs acquired there and the arrival rate to the second queue is λ.

Note that if $\mu_1 > \mu_2$, then sub-model 2 with arrival rate μ_1 and service rate μ_2 becomes unstable, i.e., the inflow to the system exceeds the outflow. It is clear that in such case the optimal policy is to switch on the server whenever a job arrives, and therefore we take $S^{(2)}(\mu_1, \mu_2, c_1 + c_2, c_3) = 1$.

As a second step, we approximate the curve by a straight line that connects the two endpoints $p_1(0)$ and i_1'. This results in the following switching curve:

$$p_1(x_1) = p_1(0)\left(1 - \frac{x_1}{i_1'}\right)^+,$$

for $0 \leq x_1 \leq N$.

4.1.2. Switching on the second server

Once there is a certain number of jobs at the second queue, it becomes optimal to switch on the corresponding server. Following the same approach as the previous case, we first estimate the endpoints of the switching curve $p_2(0)$ and $p_2(N)$. We use sub-model 2 as follows:

$$p_2(0) = S^{(2)}(\lambda, \mu_2, c_2, c_3),$$
$$p_2(N) = S^{(2)}(\mu_1, \mu_2, c_1 + c_2, c_3),$$

where again $S^{(2)}(\mu_1, \mu_2, c_1 + c_2, c_3) = 1$ if $\mu_1 > \mu_2$. Our reason for choosing these parameters for sub-model 2 is that when there are no jobs at server

1, there are no holding costs c_1 acquired. Furthermore, due to the fact that in sub-model 2 the first queue is an $M/M/1/N$ queue, the inflow to server 2 equals the inflow to the tandem system. On the other hand, when the first queue is full, the arrival rate to the second server becomes μ_1, and furthermore, one should take into account also the holding costs c_1.

As a next step, one has to approximate the shape of the curve. In this case, a straight line proved to be a relatively inaccurate fit for the switching curve. Analysis of the optimal policies, which were derived by numerically solving systems with small buffer sizes N and K, lead us to the following fit:

$$p_2(x_1) = p_2(N) + \frac{p_2(0) - p_2(N)}{\sqrt{x_1}},$$

for $0 < x_1 < N$. We found this to be a good approximation while at the same time has a simple, tractable form.

4.1.3. Switching off the first server

For sufficiently low holding costs at the first queue there will be certain cases where it is optimal to switch off the first server. This will result in jobs waiting at the first queue rather than waiting at the more expensive second queue. We estimate the switching curve separating *region (1, 1)* and *region (0, 1)* by using the results obtained from our approximation algorithm so far, together with the solution for sub-model 1. More precisely, we obtain $p_3(x_1)$ for $0 \leq x_1 \leq N$, by the following equation:

$$p_3(x_1) = p_2(x_1) + S^{(1)}(x_1; \lambda, \mu_1, \mu_2, c_1, c_2),$$

where the operator $S^{(1)}$ denotes the threshold value determined by solving sub-model 1 with the corresponding parameters.

4.2. Second server switched on

It is clear that if the second server is working it is optimal to keep it on. Therefore, the optimal decision when $s_2 = 1$ can be either action 3 or action 4, corresponding to *region (0, 1)* and *region (1, 1)*. Based on studying the conducted numerical examples and evaluating the performance of different approaches, we decided to approximate the switching curve between those two regions in the same manner as in Section 4.1, i.e., when $s_2 = 0$. Namely, the switching curve is given by the points $p_3(x_1)$, where $0 \leq x_1 \leq N$. This way, one can directly use the results derived from the above-described procedure, which implies that this case does not increase the complexity of the algorithm.

Table 1. Approximation errors of the technique.

Start-up cost	Service rates	Median E_r	E_r percentiles		
			80th	90th	95th
$c_3 = 500$	$\mu_1 < \mu_2$	1.22%	3.50%	6.64%	8.34%
	$\mu_1 > \mu_2$	1.93%	5.27%	8.78%	11.28%
	$\mu_1 = \mu_2$	1.96%	5.20%	8.32%	9.24%
$c_3 = 1000$	$\mu_1 < \mu_2$	1.67%	4.23%	8.92%	10.01%
	$\mu_1 > \mu_2$	1.91%	4.28%	7.24%	11.02%
	$\mu_1 = \mu_2$	1.63%	4.31%	8.36%	10.09%
$c_3 = 1500$	$\mu_1 < \mu_2$	1.86%	4.58%	9.34%	10.75%
	$\mu_1 > \mu_2$	1.83%	4.40%	7.90%	10.18%
	$\mu_1 = \mu_2$	1.38%	4.25%	7.24%	9.80%

5. Results

In this section, we evaluate the performance of the approximation algorithm. Recall that our method is based on estimating the switching curves of the optimal policy. Hence, the main idea is to derive a graph as similar as possible to the optimal decision policy graph (see Figures 3 and 4). However, in practice, the goal of 'optimizing' the system is often times to reduce the average costs. Therefore, although our algorithm is approximating the various switching curves, in this section, we will not examine how close are the approximated fitting functions to the optimal switching curves. Instead, we present the relative difference, denoted as E_r, between the acquired long-term average cost if one uses the decision policy obtained by our procedure, g^{est}, and the optimal one, g^{opt}, derived by numerically solving the MDP. More precisely, the relative difference is defined by:

$$E_r = \frac{|g^{est} - g^{opt}|}{g^{opt}} \times 100\%.$$

To cover the full spectrum of parameter values, we created multiple test suites with approximately 750 parameter sets in total. We examine systems with loads in the range $[0.1, 0.9]$ for each of the queues, and ratios between the two holding costs: $c_1/c_2 \in [0.1, 0.9]$. More precisely, we varied the parameters as follows:

(1) systems where $\mu_1 < \mu_2$. We take μ_2 fixed at 10, while varying μ_1 from 1.1 to 9.9 with a step size of 1.1;
(2) systems where $\mu_1 > \mu_2$. We take μ_1 fixed at 10, while varying μ_2 from 1.1 to 9.9 with a step size of 1.1;
(3) systems where $\mu_1 = \mu_2$. Once again we take the same range of values for the service rates – from 1.1 to 9.9 with a step size of 1.1.

In all three test suites, we further varied c_1 between 0.1 and 0.9 with a step of 0.1 and $c_3 = 500, 1000$ or 1500. Next to that, we fixed $\lambda = 1$. The reason is that only the ratio between the various service rates is important

with respect to the approximation error. This comes from the fact that scaling the rates is equivalent to scaling the time, which does not influence the results. Due to the same reasoning, we also fixed one of the costs: $c_2 = 1$. We note that in all tests the startup costs are considerably higher than the holding costs. We chose such values to ensure that the optimal decision policy is not a trivial one, i.e., switching the second server whenever there are jobs in the queue.

To evaluate the accuracy of our method, we compared the average costs acquired by implementing the approximated optimal policy to those derived from numerically solving the MDP. As discussed, due the computational complexity of the numerical approach, one cannot use it for large systems. Therefore, although, our approximation technique is capable of solving such instances of the model, we could create a benchmark only for smaller buffer sizes, e.g., we take $N = 1000$ and $K = 100$. Furthermore, this way we can use the exact solution for sub-model 1, i.e., the threshold values given by the operator $S^{(1)}$, when studying the accuracy of our technique. Since the method is based on the solution of sub-model 1, approximation errors in the latter will influence the results of our technique.

The results of the conducted tests are shown in Table 1. We present the approximation error of the algorithm in each of the three values for the startup cost c_3. Furthermore, motivated by the differences of the optimal policies from Figures 3 and 4, we aggregate the results according to the following three cases for the service rates: $\mu_1 < \mu_2$, $\mu_1 > \mu_2$ and $\mu_1 = \mu_2$. Finally, next to the median of the relative error, we also list the 80th, the 90th, and the 95th percentile for the corresponding nine test suites.

Based on the results from Table 1 we conclude that our algorithm's performance is not influenced by the value of the startup cost, c_3. Next to that, the errors for the three service rate configurations are also comparable, and therefore we believe that these parameters do not affect the accuracy either. In conclusion, our technique performs equally well regardless of the system parameters.

In all nine test suites, the achieved median for the relative error is less than 2%. We believe that this approximation accuracy together with the intuitive and easy to implement nature of our method makes it a suitable choice in practice.

6. Conclusion

In this paper, we analyzed the control of a two-node tandem queuing network with holding costs at both nodes, and a startup cost for the second server. We presented a simple, easy to implement and efficient method to

approximate the optimal decision policy. Extensive numerical evaluation showed that our technique is extremely accurate for a wide range of parameter combinations.

Finally, we address a few topics for further research. First, we find promising the idea of extending the method to tandem queues composed of more than two servers. Second, from a practical point of view it is interesting to study models with other cost structures, for example, models with startup cost for the first server or a startup time associated with the nodes.

References

[1] Conway, A. E.; Keilson, J. Analysis of a two-stage finite buffer flow controlled queueing model by a compensation method. Oper. Res. Lett. **1995**, *18*, 65–74.

[2] Leskel, L. Stabilization of an overloaded queueing network using measurement-based admission control. J. Appl. Probab. **2006**, *43*, 231–244.

[3] Nicola, V. F.; Zaburnenko, T. S. Efficient importance sampling heuristics for the simulation of population overflow in Jackson networks. ACM Trans. Model. Comput. Simul. **2007**, *17*, 10.

[4] Tsiotras, G. D.; Badr, H. A recursive methodology for the derivation of the blocking probabilities of tandem queues with finite capacity. Comput. Oper. Res. **1990**, *17*, 475–479.

[5] Balsamo, S. Closed queueing networks with finite capacity queues: Approximate analysis. In *Proceedings of the 14th European Simulation Multiconference on Simulation and Modelling*, SCS Europe, **2000**; 593–600.

[6] Brandwajn, A.; Jow, Y.-L. L. An approximation method for tandem queues with blocking. Oper. Res. **1988**, *36*, 73–83.

[7] Latouche, G.; Neuts, M. F. Efficient algorithmic solutions to exponential tandem queues with blocking. SIAM J. Algebraic Discrete Method. **1980**, *1*, 93–106.

[8] Liu, C.-M.; Lin, C.-L. An efficient two-phase approximation method for exponential tandem queueing systems with blocking. Comput. Oper. Res. **1995**, *22*, 745–762.

[9] Li, Y.; Cai, X.; Tu, F.; Shao, X. Optimization of tandem queue systems with finite buffers. Comput. Oper. Res. **2004**, *31*, 963–984.

[10] van Foreest, N. D.; van Ommeren, J. C. W.; Mandjes, M. R. H.; Scheinhardt, W. R. W. A tandem queue with server slow-down and blocking. Stochastic Model. **2005**, *21*, 695–724.

[11] Guerouahane, N.; Aissani, D.; Farhi, N.; Bouallouche-Medjkoune, L. M/G/c/c state dependent queuing model for a road traffic system of two sections in tandem. Comput. Oper. Res. **2017**, *87*, 98–106.

[12] Hordijk, A.; Koole, G. On the shortest queue policy for the tandem parallel queue. Prob. Eng. Inf. Sci. **1992**, *6*, 63.

[13] Konheim, A. G.; Reiser, M. Finite capacity queuing systems with applications in computer modeling. SIAM J. Comput. **1978**, *7*, 210–229.

[14] Leskelä, L.; Resing, J. A Tandem Queueing Network with Feedback Admission Control, 129–137. Springer Berlin Heidelberg: Berlin, Heidelberg, **2007**.

[15] Zhang, B.; Ayhan, H. Optimal admission control for tandem queues with loss. IEEE Trans. Automat. Contr. **2013**, *58*, 163–167.

[16] Chang, K.-H.; Chen, W.-F. Admission control policies for two-stage tandem queues with no waiting spaces. Comput. Oper. Res. **2003**, *30*, 589–601.

[17] Rosberg, Z.; Varaiya, P.; Walrand, J. Optimal control of service in tandem queues. IEEE Trans. Automat. Contr. **1982**, *27*, 600–610.

[18] Grassmann, W. K.; Drekic, S. An analytical solution for a tandem queue with blocking. Queue. Syst. **2000**, *36*, 221–235.

[19] Heyman, D. P. Optimal operating policies for $M/G/1$ queuing systems. Oper. Res. **1968**, *16*, 362–382.