Control of a tandem queue with a start-up cost for the second server

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

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 start-up cost for the second server. In our work, we present a rather intuitive, easy to understand, and at the same time very accurate algorithm to approximate the optimal decision policy. Extensive numerical experimentation shows that the approximation works extremely well for a wide range of parameter combinations.

1 Introduction

Networks in which the departures from one server become the arrivals to a downstream server might be modeled as tandem queues. These models have proved to be very challenging to analyze [4,12,15,17], and even despite the decades of research, there are still many open problems without an analytic solution [1, 2, 10, 14]. At the same time, tandem queues abound in applications, and therefore the study of these systems is also important for the 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 [13, 18] . For example, in some practical situations, it is possible to temporary 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 [6], transportation, manufacturing, and many other fields [8, 9, 11, 19]. For instance, one application that can be modeled as a tandem network and uses such control techniques is the caching in computer databases (illustrated in Figure

*A. Hristov (corresponding author), J.W. Bosman, and R.D. van der Mei are with the Center for Mathematics and Computer Science (CWI), Amsterdam 1098 XG, The Netherlands, and also with the Faculty of Science, Vrije Universiteit Amsterdam, Amsterdam 1081 HV, The Netherlands (e-mail: a.v.hristov@cwi.nl; j.w.bosman@cwi.nl; r.d.van.der.mei@cwi.nl). S. Bhulai is with the Faculty of Science, Vrije Universiteit Amsterdam, Amsterdam 1081 HV, The Netherlands (e-mail: s.bhulai@vu.nl).

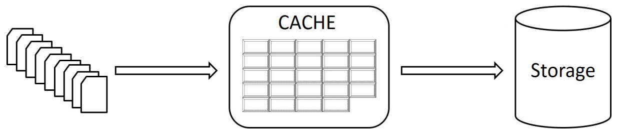


Figure 1: Cache used in data centers

1). By implementing the cache mechanism, the contention at the data storage might be highly reduced by first writing the data on a cache and only afterwards on the database. Furthermore, to avoid overload related problems, 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, we study tandem queueing networks where one can change the state of any of the servers.

On the other hand, switching off a node results in reducing the serving 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 for jobs waiting in the various queues [3,16]. In such a way, one combines both metrics, i.e., the sojourn time spent in the system and the acquired resources, in a single indicator. The problem is, therefore, translated into 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 initialization 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 start-up cost associated with the second node. In such a way, our framework can be considered as a generalization of the one researched in [5,16].

In the following section, we introduce the model that we consider throughout the paper. Furthermore, in Section 3, we illustrate how the optimal decision policy for such systems looks like. Consequently, in Section 4, we propose an approximation algorithm for obtaining the optimal policy. The accuracy of the presented technique is discussed in Section 5. Finally, in Section 6 we conclude with summary and discuss ideas for possible further research.

2 The model

In the following, we present the queueing network that we consider in this paper. The system consists of two servers in a tandem setting. Namely, jobs arrive at the first queue and after receiving service there, they

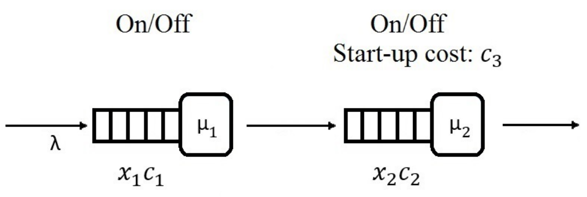


Figure 2: Tandem queue with a start-up cost

are transferred to the second one. Consequently, they are served at the second node and leave the system. We assume a Poisson arrival rate λ and exponential service times with mean $\beta_i = 1/\mu_i$, where $i \in (1, 2)$. Moreover, both servers are serving at maximum one job at a time according to the FIFO policy. For practical matter, the queues are taken to be of a finite size - N and K respectively. As mentioned, there are costs associated with this queueing network. Namely, each job in queue $i \in \{1, 2\}$ generates a linear holding cost $c_i \geq 0$. Furthermore, there is a start-up cost for the second server denoted by $c_3 \geq 0$. To optimize the expenses, one can control the system at any point in time by switching on or off a given server. Nevertheless, we pose two restrictions on the possible decisions: (1) the first server cannot be working if the buffer space at the second node is full (if there are K customers 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 $c_2 > c_1$. For an illustration of the model, the reader is referred to Figure 2.

We formulate the described system as a Markov decision process (MDP) with a state space $S \in (x_1, x_2, s_2)$, where x_i stands for the number of jobs at node i and s_2 is 0 or 1 according to server 2 being *off* or *on*. Due to the memoryless property of the exponentially distributed service time and the no cost or start-up time associated with the first server, one does not have to explicitly keep track of its state. We denote the possible actions as follows:

- action (1) - switch off both servers;
- action (2) - switch on server 1 and switch off server 2;
- action (3) - switch on server 2 and switch off server 1;
- action (4) - switch on both of the servers.

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

$$V(x_1, x_2, s_2) = g + c_1 x_1 + c_2 x_2 + \min\{T_1, T_2, T_3, T_4\},$$

where V denotes the optimal value function and g - the average costs per time unit. Moreover, T_i, for

$1 \leq i \leq 4$, stands for the transition rates and the possible additional costs associated with actions (1) to (4) respectively. Namely:

$$
\begin{aligned}
T_1 =& \lambda V(\max(x_1+1,N),x_2,0) + (1-\lambda)V(x_1,x_2,0)\\
T_2 =& \lambda V(\max(x_1+1,N),x_2,0) + \mu_1 V(x_1-1,x_2+1,0) + \mu_2 V(x_1,x_2,0),\\
T_3 =& c_3(1-s_2)\\
&+ \lambda V(\max(x_1+1,N),x_2,1) + \mu_2 V(x_1,x_2-1,1-\ell) + \mu_1 V(x_1,x_2,1),\\
T_4 =& c_3(1-s_2)\\
&+ \lambda V(\max(x_1+1,N),x_2,1) + \mu_1 V(x_1-1,x_2+1,1) + \mu_2 V(x_1,x_2-1,1-\ell),
\end{aligned}
$$

where the rates are scaled in such a way that $\lambda + \mu_1 + \mu_2 = 1$, and furthermore, $\ell = 1$ if $x_2 = 1$, and $\ell = 0$ otherwise. Moreover, as discussed, some decisions are not possible under certain cases 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

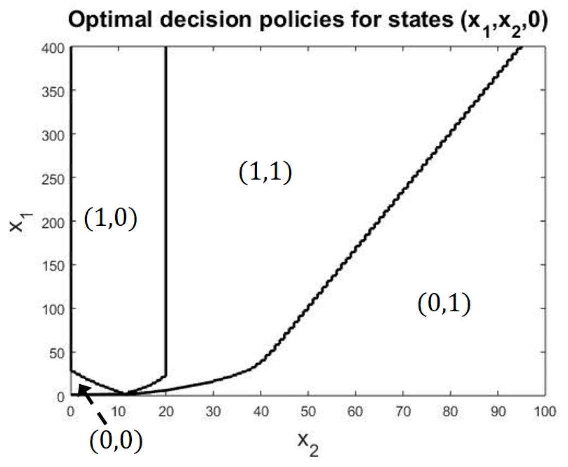
As discussed in Section 1, even the special case of no start-up cost associated with the second server, i.e., the special case of $c_3 = 0$, has withstood an analytic analysis so far. Therefore, as an initial step we will study the system based on results derived by numerically solving the MDP, e.g., by applying the value iteration technique. Note that this numerical approach can be used only for systems of a relatively small size. In practice, both N and K might be a few orders of magnitude larger, which makes the technique computationally unfeasible. Therefore, one needs a different approach to solve such real-world instances, and hence our goal of developing an approximation algorithm capable of dealing with this challenge.

In the following, we use as a non-trivial examples of systems with the following parameters: $N = 750, K = 200, c_1 = 0.1, c_2 = 1, c_3 = 1500, \lambda = 1$. The service rate of the servers are taken to be:

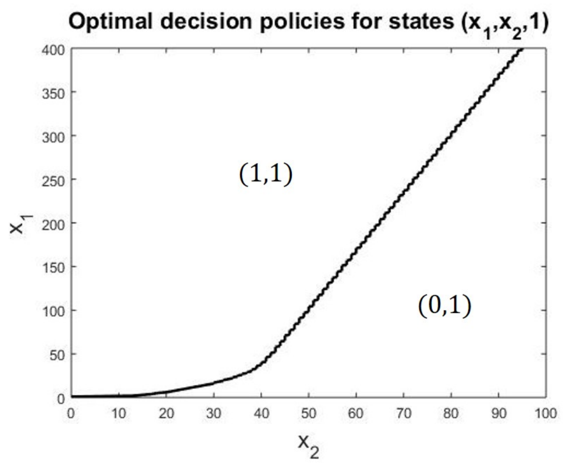
- $\mu_1 = 2.2$ and $\mu_2 = 4$ in the first example,
- $\mu_1 = 4$ and $\mu_2 = 2.2$ in the second example.

The corresponding optimal decision policies are shown in figures 3a, 3b and 4a, 4b. 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 believe that the optimal policy for such a queueing network is of a threshold type. We denote the different regions of states, where a certain decision

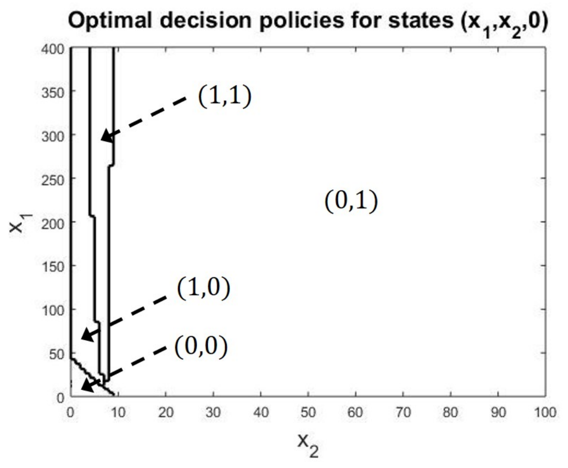


(a) $s_2 = 0$

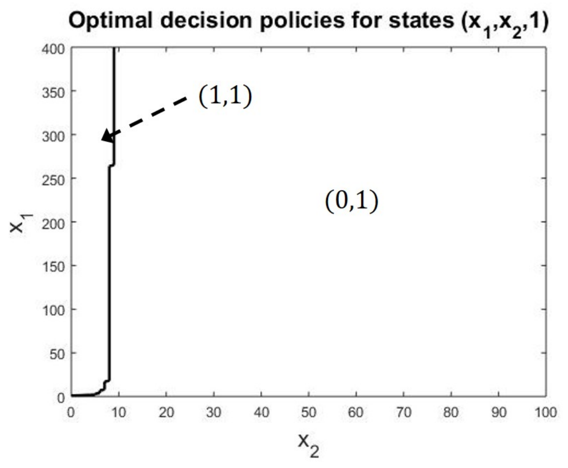


(b) $s_2 = 1$

Figure 3: Optimal decision policy for $\mu_1 = 4$ and $\mu_2 = 2.2$



(a) $s_2 = 0$



(b) $s_2 = 1$

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

is optimal, as follows:

region $(0,0)$: the optimal decision is (1), namely, both of the servers should be off.

region $(1,0)$: the optimal decision is (2), namely, the first server should be on, whereas the second one - off.

region $(0,1)$: the optimal decision is (3), namely, the first server should be off, whereas the second one - on.

region $(1,1)$: the optimal decision is (4), namely, both of the servers should be on.

The goal of our research is to develop an approximation algorithm for the presented MDP. Our approach to obtain the optimal policy is to estimate the above described regions, rather than deriving an approximation of the value function.

4 Approximation technique

In the following, we present our technique for approximating the optimal policy. The main idea behind the algorithm is to decompose the original system into two sub-problems.

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 exponentially distributed variables. There are linear costs c_1 and c_2 for each customer waiting at the corresponding queue. However, in contrast to the main network analyzed in this paper, there is no start-up cost for the second server. The optimal policy for such a system is proven to be defined by a switching curve, see [16]. In other words, for any given number of jobs i at the first server, there is a threshold value T_i for the number of customers at the second node. If the second queue exceeds the corresponding threshold it becomes optimal to switch off the first sever. In the following, we refer to the parameters for this sub-model by appending a superscript (1) to the corresponding parameter in the main model. More precisely, we denote the arrival rate as $\lambda^{(1)}$, the service rates - $\mu_1^{(1)}$ and $\mu_2^{(1)}$, and the holding costs - $c_1^{(1)}$ and $c_2^{(1)}$.

Sub-model 2 consists of an M/M/1 queue with linear holding costs for each waiting customer and a start-up cost. 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 customers. The optimal policy for such a system is proven to be a threshold policy, see [7]. In the following, we refer to the parameters for this sub-model by appending a superscript (2) to the corresponding parameter in the main model. More precisely, we denote the arrival rate for this system as $\lambda^{(2)}$, the service rate $\mu_2^{(2)}$, the holding cost - $c_2^{(2)}$, and the start-up cost - $c_3^{(2)}$.

As discussed in Section 1, an analytic solution for sub-model 1 has proven to be challenging and is still not found. However, there are number of researches that developed efficient numerical algorithms for calculating the optimal threshold levels. Therefore, we assume the optimal policy for *sub-model 1* to be given.

On the other hand, there is a closed form solution for the optimal threshold for *sub-model 2*, given by:

$$\sqrt{2\lambda^{(2)}(1-\lambda^{(2)}/\mu_2^{(2)})c_3^{(2)}/c_2^{(2)}}.$$

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 dependency between the working regime of the first server and the arrival rate at the second one. Therefore, in our algorithm, we try to take into account this dependency.

4.1 The second server is switched off

In this section, we show how to approximate each of the three switching curves in the optimal decision policy 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 of 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 $p_{(1,i)}$, where $0 \leq i \leq N$. 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 $p_{(2,i)}$ that defines this curve. Finally, the *third* set of threshold levels, $p_{(3,i)}$, 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 following, we outline how to estimate those three switching curves.

Switching on the first server

Under certain system parameters, it can be reasoned that keeping both servers off is the optimal decision whenever there are not many jobs in the network. Namely, when $c_2 > c_1$, one would acquire less costs when the customers are 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 *sub-model 2*:

$$\begin{aligned} p_{(1,0)} &= S2(\lambda, \mu_2, c_1 + c_2, c_3), \\ i'_1 &= \frac{S2(\lambda, \mu_2, c_1, c_3) + S2(\mu_1, \mu_2, c_1 + c_2, c_3) + S2(\lambda, \mu_2, c_2, c_3)}{3} \end{aligned}$$

where the operator $S2()$ denotes the threshold value determined by solving *sub-model 2* with the corresponding parameters, i.e.:

$$S2(\lambda^{(2)}, \mu_2^{(2)}, c_2^{(2)}, c_3^{(2)}) = \sqrt{2\lambda^{(2)}(1 - \lambda^{(2)}/\mu_2^{(2)})c_3^{(2)}/c_2^{(2)}}.$$

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. First one being the regime just before taking the decision to switch on the first server and having customers only at the first server. Second regime - server 1 is switched on and there are number of jobs in the queue - therefore the arrival rate to the second node is μ_1. Third regime - the queue at server 1 is empty - hence there are no holding costs acquired there, and furthermore the arrival rate to the second queue is λ.

Note that if $\mu_1 > \mu_2$, *sub-model 2* with arrival rate μ_1 and service rate μ_2 becomes unstable, i.e., the inflow to the system is greater than the outflow. It is clear that in such case the optimal policy is to switch on the server whenever a customer arrives, and therefore we take $S2(\mu_1, \mu_2, c_1 + c_2, c_3) = 1$.

As a second step, we approximate the curve connecting the two endpoints: $p_{(1,0)}$ and i_1' as a straight line. This corresponds to the following characterization of the switching curve:

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

for $0 \leq i \leq N$.

Switching on the second server

Once there are certain number of customers 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:

$$p_{(2,0)} = S2(\lambda, \mu_2, c_2, c_3)$$
$$, p_{(2,N)} = S2(\mu_1, \mu_2, c_1 + c_2, c_3),$$

where again $S2(\mu_1, \mu_2, c_1 + c_2, c_3) = 1$ if $\mu_1 > \mu_2$. Our reason to chose these parameters for *sub-model 2* is that when there are no customers at server 1, there are no holding costs c_1 acquired, and furthermore due to the first queue being an $M/M/1$ 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, led us to the following fit:

$$p_{(2,i)} = p_{(2,N)} + \frac{p_{(2,0)} - p_{(2,N)}}{i^{\frac{1}{2}}},$$

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

Switching off the first server

Having a lower holding costs at the first queue implies that at certain cases it is optimal to switch off the first server so that customers in the system are waiting there instead of 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 $p_{(3,i)}$ for $0 \leq i \leq N$, by the following:

$$p_{(3,i)} = p_{(2,i)} + S1(i; \lambda, \mu_1, \mu_2, c_1, c_2),$$

where the operator $S1(i;)$ denotes the threshold value for x_i determined by solving *sub-model 1* with the corresponding parameters.

4.2 The second server is 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 $p_{(3,i)}$, where $0 \leq i \leq N$. In such a 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.

5 Results

In this section, we evaluate the performance of the approximation algorithm. As explained, our method is based on estimating the switching curves of the optimal policy, and 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 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, E_r, between the acquired cost if one uses the decision policy obtained by our procedure, g^{est}, and the optimal cost, g^{opt}, derived by numerically solving the MDP. More precisely, we calculate:

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

As discussed, various systems across many different fields can be modeled by the network analyzed in this paper. Therefore, we want to evaluate the performance of the approximation algorithm in the full spectrum of system parameters, instead of only those that make sense for the particular motivation this study was inspired by. For that reason, we examined multiple test suites with 729 parameter sets in total, covering 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: For 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 of 1.1. For 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 of 1.1. And for 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 of 1.1.

In all three test suites, we further vary 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 fix $\lambda = 1$. 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 fix one of the costs: $c_2 = 1$. We note that in all tests the start-up costs are considerably higher than the holding costs. We choose such values to ensure that the optimal decision policy is not a trivial one, i.e., switching the second server whenever there are customers in the queue.

To evaluate the accuracy of the algorithm, we compare 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 algorithm 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 $K = 100$.

The results of the conducted tests are shown in Tables 1, 2, and 3. We present the performance of the algorithm in each of the three values for the start-up 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 Tables 1, 2, and 3, we conclude that our algorithm's performance is not influenced by the value of the start-up 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

Test Suite		E_r percentiles		
	Mean	80^{th}	90^{th}	95^{th}
$\mu_1 < \mu_2$	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%

Table 1: Approximation errors for $c_3 = 500$.

Test Suite		E_r percentiles		
	Mean	80^{th}	90^{th}	95^{th}
$\mu_1 < \mu_2$	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%

Table 2: Approximation errors for $c_3 = 1000$.

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 algorithm 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 start-up cost for the second server. We presented a simple, easy to implemtn and efficient algorithm 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 algorithm 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 start-up cost for the first server or a start-up time associated with the nodes.

References

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

[2] A. Brandwajn and Y.-L.L. Jow. An approximation method for tandem queues with blocking. *Operations Research*, 36(1):73–83, 1988.

Test Suite	E_r percentiles			
	Mean	80$^{\text{th}}$	90$^{\text{th}}$	95$^{\text{th}}$
$\mu_1 < \mu_2$	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%

Table 3: Approximation errors for $c_3 = 1500$.

[3] K.-H. Chang and W.-F. Chen. Admission control policies for two-stage tandem queues with no waiting spaces. *Computers & Operations Research*, 30(4):589–601, April 2003.

[4] A.E. Conway and J. Keilson. Analysis of a two-stage finite buffer flow controlled queueing model by a compensation method. *Operations Research Letters*, 18(2):65–74, September 1995.

[5] W.K. Grassmann and S. Drekic. An analytical solution for a tandem queue with blocking. *Queueing Systems*, 36(1):221–235, Nov 2000.

[6] N. Guerouahane, D. Aissani, N. Farhi, and L. Bouallouche-Medjkoune. M/G/c/c state dependent queuing model for a road traffic system of two sections in tandem. *Computers & Operations Research*, 87:98 – 106, 2017.

[7] D.P. Heyman. Optimal operating policies for $M/G/1$ queuing systems. *Operations Research*, 16(2):362–382, 1968.

[8] A. Hordijk and G. Koole. On the shortest queue policy for the tandem parallel queue. *Probability in the Engineering and Informational Sciences*, 6(1):6379, 1992.

[9] A.G. Konheim and M. Reiser. Finite capacity queuing systems with applications in computer modeling. *SIAM Journal on Computing*, 7(2):210–229, 1978.

[10] G. Latouche and M.F. Neuts. Efficient algorithmic solutions to exponential tandem queues with blocking. *SIAM Journal on Algebraic Discrete Methods*, 1(1):93–106, 1980.

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

[12] L. Leskel. Stabilization of an overloaded queueing network using measurement-based admission control. *Journal of Applied Probability*, 43(1):231–244, 2006.

[13] Y. Li, X. Cai, F. Tu, and X. Shao. Optimization of tandem queue systems with finite buffers. *Computers & Operations Research*, 31(6):963 – 984, 2004.

[14] C.-M. Liu and C.-L. Lin. An efficient two-phase approximation method for exponential tandem queueing systems with blocking. *Computers & Operations Research*, 22(7):745 – 762, 1995.

[15] V.F. Nicola and T.S. Zaburnenko. Efficient importance sampling heuristics for the simulation of population overflow in jackson networks. *ACM Transactions on Modeling and Computer Simulation*, 17(2), April 2007.

[16] Z. Rosberg, P. Varaiya, and J. Walrand. Optimal control of service in tandem queues. *IEEE Transactions on Automatic Control*, 27(3):600–610, June 1982.

[17] G.D. Tsiotras and H. Badr. A recursive methodology for the derivation of the blocking probabilities of tandem queues with finite capacity. *Computers & Operations Research*, 17(5):475 – 479, 1990.

[18] N.D. van Foreest, J.C.W. van Ommeren, M.R.H. Mandjes, and W.R.W. Scheinhardt. A tandem queue with server slow-down and blocking. *Stochastic Models*, 21(2-3):695–724, 2005.

[19] B. Zhang and H. Ayhan. Optimal admission control for tandem queues with loss. *IEEE Transactions on Automatic Control*, 58(1):163–167, Jan 2013.