Optimal Scheduling in a General Single-Server System with Heterogeneous Queues and Switching Costs Using Simulation and Neural Network Paradigms

Dmitry Efrosinin *, Vladimir Vishnevsky, Natalia Stepanova

Posted Date: 2 May 2023

doi: 10.20944/preprints202305.0080.v1

Keywords: Optimal scheduling; heterogeneous queues; Markov decision problem; queue simulation; simulated annealing; neural network

Article

Optimal Scheduling in a General Single-Server System with Heterogeneous Queues and Switching Costs Using Simulation and Neural Network Paradigms

Dmitry Efrosinin [1,2,*], Vladimir Vishnevsky [3], Natalia Stepanova [4]

[1] Johannes Kepler University Linz, Austria; dmitry.efrosinin@jku.at
[2] Peoples' Friendship University of Russia (RUDN University)
[3] Institute of Control Sciences of Russian Academy of Sciences
[4] AO NPF INSET, Moscow, Russia
* Correspondence: dmitry.efrosinin@jku.at

Abstract: The problem of optimal scheduling in a system with parallel queues and a single server has been extensively studied in the queueing theory. However, such systems have mostly been studied by assuming homogeneous attributes of arrival and service processes, or in heterogeneous case a Markov random process was usually assumed. In this paper, we explore the possibility of combining simulation and neural network paradigms for optimal scheduling in a heterogeneous system with an arbitrary inter-arrival and service time distributions. The service of any queue by the server is exhaustive. Further routing of the server after the queue has been emptied is as instructed by the trained neural network. Any change in the queue that currently has a server is accompanied by a switching cost. The simulated annealing optimization algorithm which is used to optimize the parameters of the neural network was adapted for numerically evaluated average cost function. A Markov decision problem was formulated for the corresponding markovian queueing system to verify the results of solving the optimization problem. In addition to comparing the actual values of the average cost, the parameter values of the neural network trained on the optimal control policy of the Markov model are also used to determine the domains of parameters required for the simulated annealing defined on a finite discrete domain. Numerical analysis shows the effectiveness of this approach. Moreover, a comparison of the results for different distributions illustrates the insensitivity of the optimal control policy to the shape of the distributions for the same two first moments. Thus, the optimal control policy for an exponential model can be used as a suboptimal one for the general system.

Keywords: optimal scheduling; heterogeneous queues; Markov decision problem; queue simulation; simulated annealing; neural network

1. Introduction

Machine learning algorithms have been used in the last ten years in almost all fields where the problems associated with the data classification, pattern recognition, non-linear regression, etc., have to be solved. Application of such algorithms has also intensified in the field of queueing theory. While the first steps in the successful application of machine learning to evaluate the performance characteristics of simple and complex queueing systems have already been taken, the total number of works on this topic still remains modest. As for reviews, we can only refer to recent paper by Vishnevskiy and Gorbunova [32] which proposes a systematic introduction to the use of machine learning in the study of queueing systems and networks. Thus, we would also like to make a small contribution to the popularisation of the topic by briefly describing existing works. In Stintzing and Norrman [29], the artificial neural network was used for predicting the number of busy servers in the $M/M/s$ queueing system. The papers of Nii et al. [23] and Sherzer et al. [26] have answered positively the question,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

whether the machines could be useful for solving the problems in general queueing systems. They have used neural network to estimate the mean performance measures of the multi-server queues $GI/G/s$ based on the first two moments of the inter-arrival and service time distributions. A machine learning approach was used in Kyritsis and Deriaz [20] to predict the waiting time in queueing scenarios. The combination of a simulation and a machine learning techniques for assessing the performance characteristics have been illustrated in Vishnevsky et al. [31] on a queueing system $MMAP/PH/M/N$ with K priority classes. Markovian queues were simulated using artificial neural networks in Sivakami et al. [27]. The neural networks were used also in Efrosinin and Stepanova [9] to estimate the optimal threshold policy in a heterogeneous $M/M/K$ queueing system. The combination of the Markov decision problem and the neural network for the heterogeneous queueing model with a process sharing was studied by Efrosinin et al. [10]. The performance parameters of the closed queueing network by means of a neural network were evaluated in Gorbunova and Vishnevskiy [13]. The main conclusion to be drawn from the results already obtained by application of the machine learning to models of the queueing theory is that the neural networks cannot be treated as a replacement for classical methods for system performance analysis, but rather complement the capabilities of such analysis.

This paper proposes a fairly universal method for solving the problem of optimal dynamic scheduling or allocation in queueing systems of the general type, i.e. where the times between events are arbitrarily distributed, and in queueing systems with correlated inter-arrival and service times. It can provide also the performance analysis of complex controlled systems described by multidimensional random processes, for which finding analytical, approximate or heuristic solutions is a difficult task. The method is exemplified by some version of a well-known queueing model consisting of several parallel queues and one server which serve the queues according to some control policy. The system is assumed to have heterogeneous arrival and service attributes, i.e. unequal arrival and service rates, as well as holding and switching costs. Such systems are known also as polling systems which have found wide application in various fields such as computer networks, telecommunications systems, control in manufacturing and road traffic. For analytic and numerical results in various types of polling systems with applications to the broadband wireless Wi-Fi and Wi-MAX networks, we refer interested readers to the textbook by Vishnevsky and Semenova [33] and the references therein. The same authors in [34] developed their research on polling systems to systems with correlated arrival flows such as MAP, $BMAP$, and the group Poisson arrivals. In Vishnevskiy et al. [35] it was shown that the results obtained by a neural network are close enough to the results of analytical or simulation calculations for the $M/M/1$ and $MAP/M/1$-type polling systems with cyclic polling.

A Markovian analog of such a model has been investigated by a number of authors. The two queue homogeneous model with equal service rates and holding costs has been studied in Horfi and Ross [15], where it was shown that the queues must be serviced exhaustively according to the optimal policy. In Liu et al. [21] it was shown that the scheduling policy that routes the server with respect to the LQF (Longest Queue First) policy is optimal when all queue lengths are known and that the cyclic scheduling policy is optimal in the case that the only information available is the previous decisions. The systems with multiple heterogeneous queues, also known as asymmetric polling systems, in different settings have been studied intensively for the case of no switching costs in Buyukkoc et al. [4], Cox and Smith [5], where the optimality of the static $c\mu$-rule was proved. This policy schedules a server first to the queue i with a maximum weight $c_i\mu_i$ consisting of the holding cost and service rate. In Koole [19], the problem of optimal control in a two-queue system was analysed by means of the continuous-time Markov decision process and dynamic programming approach. The author has found numerically that the optimal policy which minimizes the average cost per unit of time could be quite complex if there are both holding and switching costs. The threshold-based policy for such a queueing system was applied by Avram and Gómez-Corral [3], where the expressions for the long-run expected average cost of holding units and switching actions of the server were given. The queueing system with general service times and set-up costs which effect on instantaneous switch

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

from one queue to another was studied in Duenyas and Van Oyen [6]. The authors proposed a simple heuristic scheduling policy for the system with multiple queues. A rather similar model is described in Matsumoto [22], where the optimal scheduling problem is solved in a system with arbitrary time distributions. Here, instead of switching costs, the corresponding set-up time intervals required for switching are used. The system is controlled by the Learning Vector Quantization (LVQ) network, see for details Kohonen [18], which classifies the system state by the closest codebook vector of a certain class in terms of the Euclidean metric. The problem with this approach is the large number of parameters associated with the codebook vectors, normally it is required several vectors per class, which must be estimated for a given control policy using computationally quite expensive recurrent algorithm.

It is assumed in our model that the queue currently being served by the server is serviced exhaustively. The next queue to be served by the server is selected according to a dynamic scheduling policy based on the queue state information, i.e. on the number of customers waiting in each of parallel queues. It is expected that the changing of the serviced queue involves the switching costs. The holding of a customer in the system is also linked to the corresponding cost. Obviously, even with some fixed scheduling control policy, calculating any characteristics of the proposed queueing system with arbitrary inter-arrival and service time distributions in explicit form is not a trivial task. It is also difficult to fix the dynamic control policy defining the scheduling in large systems in a standard way, e.g. through a control matrix that would contain for all possible states of the system the corresponding control action. Therefore, in such a case we consider it justified to solve the problem of finding the optimal scheduling policy with the aim to minimize the average cost per unit of time by combining the simulation as a tool to calculate the performance characteristics of the system with a machine learning paradigm, where the neural network will be responsible for the dynamic control. By training a neural network for some initial control policy, we obtain characteristics of the network in form of a matrix of weights and a vector of biases. Then the process of solving the optimal scheduling problem is reduced to a discrete parametric optimization. The parameters of the neural network must be optimized in such a way that this network by generating control actions at decision epochs can guarantee the minimal values of the average cost functional. For this purpose we have chosen one of the random search methods, such as simulated annealing, see e.g. in Aarts and Korst [1], Ahmed [2]. It is a heuristic method based on a concept of heating and controlled cooling in metallurgy and it is normally used for global optimization problems in a large search space without any assumption on the form of the objective function. Specially for the probabilistic scheduling problem this algorithm was implemented by Gallo and Capozzi [12]. The algorithm will be adapted for a non-explicitly defined parametric function with a large number of variables defined on a discrete domain. To verify the quality of the calculated optimal parameters of the neural network, the values of the average cost functional for the markovian version of the queueing system are compared with the results obtained by solving the Markov decision problem (MDP). The general theory on MDP models is discussed in Puterman [25] and Tijms [30]. The details on application of MDP to controlled queueing systems with heterogeneous servers can be found in Efrosinin [8]. The optimal control policy and the corresponding objective function are calculated in the paper by a policy-iteration algorithm proposed in Howard [16] for an arbitrary finite-state Markov decision process. According to the MDP, the router in our system has to find an optimal control action in the state visited at a decision epoch with the aim to minimize the long-run average cost. Note that for our queueing model under general assumptions the semi-markov decision problem (SMDP) can be formulated. The SMDP is more powerful model than the MDP since by calculating the objective function the time spent by the system in each state before a transition is taken into account. The objective function must be calculated here also by means of a simulation. In this case the reinforcement learning algorithm, e.g. Q-P-Learning, can be applied. The main problem of this approach consists in the fact that for deterministic control policy many pairs of state and action can remain non-observable and as a result the control actions in such states can not be optimized. However, in our opinion, neural networks could also be used to solve this problem which is a potential

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

task for further research. The SMDP topic is outside the scope of this article but we refer the readers to book by Gosavi [14], where one can find very interesting overview on reinforcement learning and a well-designed classification of simulated-based optimization algorithms.

Summarising our research in this paper we can highlight the following main contributions: (a) We proposed a new controlled single-server system with parallel queues where the router uses a trained multi-level neural network to perform a scheduling control; (b) a simulated annealing method was adapted to optimize the weights and the biases of the neural network with the aim to minimize the average cost function which can be calculated only by a simulation; (c) the quality of the resulting optimal scheduling policy was verified solving a Markov decision problem for the markovian analog of the queueing system; (d) we provide detailed numerical analysis of the optimal scheduling policy and discuss its sensitivity to the shape of the inter-arrival and service time distributions; (e) the distinctive feature of our paper is the presence of algorithms used in the paper in form of pseudocodes with detailed descriptions of relevant steps.

The rest of the paper is organized as follows. Section 2 presents the formal description of the queueing system and optimization problem. Section 3 describes the Markov decision problem and the policy-iteration algorithm used to calculate optimal scheduling policy. In Section 4, the event based simulation procedure of the proposed queueing system is discussed. The neural network architecture, parametrization and training algorithm are summarized in Section 5. Section 6 presents simulated annealing optimization algorithm. Numerical analysis is shown in Section 7 and we conclude the paper in Section 8.

The following notations are introduced for use in sequel. Let $\mathbf{e}_j$ denote the vector of appropriate dimension with 1 in the jth position beginning from 0th and 0 elsewhere, $1_{\{A\}}$ denote the indicator function which takes the value 1 if the event A occurs and 0 otherwise. The notations $\min\limits_i\{a_i\}$ and $\max\limits_i\{a_i\}$ mean the minimum and maximum of the values that a can assume, and $\arg\min\limits_i\{a_i\}$, $\arg\max\limits_i\{a_i\}$ denote the element index associated respectively with the minimum and maximum value.

2. Single-server system with parallel queues

Consider a single-server system with N parallel heterogeneous queues of the type $GI/G/1$ and router for scheduling of one server between the queues. Heterogeneity here refers to unequal attributes associated with the arrival and service of customers, as well as unequal holding and switching costs. If a queue starts to be serviced, it is serviced exhaustively. Denote by $I = \{1, 2, \ldots, N\}$ the queue index set. The proposed queueing system is shown schematically in Figure 1.

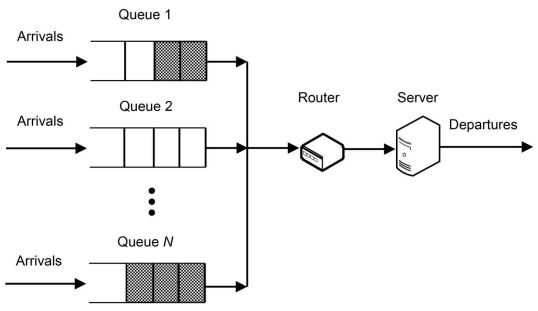

Figure 1. Controlled single-server queueing system with parallel queues

Denote by $\tau_{n,i}, n \geq 1$, the time instants of arrivals to queue i and by $\nu_i := \nu_{n,i} = \tau_{n,i} - \tau_{n-1,i}, n \geq 1$, the sequence of mutually independent and identically distributed inter-arrival times with a CDF $A_i(t), i \in I$. Further denote by $\zeta_i := \zeta_{n,i}, n \geq 1$, the service time of the nth customer in the ith queue.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

These random variables are also assumed to be mutually independent and generally distributed with CDF $B_i(t), i \in I$. We assume that introduced random variables have at least two first finite moments

$$a_{k,i} = k \int_0^\infty x^{k-1}(1 - A_i(t))dt,\ b_{k,i} = k \int_0^\infty x^{k-1}(1 - B_i(t))dt,\ k = 1, 2.$$

The squared coefficients of variation are defined then respectively by

$$CV_{\nu_i}^2 = \frac{a_{2,i}}{a_{1,i}^2} - 1, \qquad CV_{\zeta_i}^2 = \frac{b_{2,i}}{b_{1,i}^2} - 1.$$

The last characteristic will be required to provide a comparison analysis of the optimal scheduling policy for different types of inter-arrival and service time distributions. From now it is assumed that the ergodicity condition is fulfilled, i.e. the traffic load $\rho = \sum_{i=1}^N \rho_i = \sum_{i=1}^N \frac{b_{1,i}}{a_{1,i}} < 1$.

Let $D(t)$ indicates the sequence number of the queue currently being serviced by the server at time t and $Q_i(t)$ denote the number of customers in the ith queue at time t, where $i \in I$. The states of the system at time t are then given by a multidimensional random process

$$\{X(t)\}_{t\geq 0} = \{D(t), Q_1(t), \ldots, Q_N(t)\}_{t\geq 0} \tag{1}$$

with a state space

$$E = \{x = (d, q_1, \ldots, q_N) : d \in I,\ q_i \in \mathbb{N}_0,\ i \in I\}. \tag{2}$$

Further in this section, the notations $d(x)$ and $q_i(x)$ will be used to identify the corresponding components of the vector state $x \in E$. The cost structure consists of the holding cost c_i per unit of time the customer spends in queue i and the switching cost $c_{i,j}$ to switch the server from queue i to queue j.

It is assumed that the system states $X(t)$ are constantly monitored by the router which defines which queue must be serviced next after a current queue becomes empty. In initial state, when the total system is empty, a server is randomly scheduled to some queue. If the ith queue to be served becomes empty, such a moment we call a decision epoch, the router decides by means of the neural network, whether it must leave the server at the current queue or dispatch it to another queue according to a specified scheduling control policy. The routing to an idle queue is also possible. We remind that the server allocated by the router to a certain queue serves it exhaustively, i.e. it is only possible to change the queue if it becomes empty. Denote by $A = I$ an action space with elements $a \in A$, where a indicates the sequence number of the queue to be served next after the current queue has been emptied. The subsets $A(x)$ of control actions in state $x \in \hat{E} \subset E$ with

$$\hat{E} = \{x = E : q_d(x) = 0\}$$

coincide with the action space A. In all other states x from $E \setminus \hat{E}$ the subsets $A(x) = \{0\}$ includes only a fictitious control action 0 which has no influence on the system's behavior.

The router can operate according to some heuristic control policies. It could be for example a Longest Queue First (LQF) policy which is a dynamic one and it prescribes at decision epochs to serve the next queue with the highest number of customers. If there are more than one queue with the same maximal number of customers, the queue number is selected randomly. Alternatively, the static $c\mu$-rule, which needs only the information if a certain queue in non-empty, can be used as an initial policy. According to this policy the queue i with the highest factor $c_i\mu_i$, which is the product of the holding cost and the service intensity, must be serviced next. In the system with totally symmetric queues the former policy is according to [21] optimal. The latter control policy is optimal due to [4] if

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

there is no switching costs, i.e. $c_{i,j} = 0$. Otherwise, in case of positive switching costs and asymmetric queues such policies are not optimal with respect to minimization of the average cost per unit of time.

The main idea of an optimal scheduling in a such general model is as follows. We will equip the router with a trained neural network which will inform it on the sequence number of the next queue to which the server should be routed with the aim to reach formulated optimization aims. Obviously, we can only train the neural network based on the available data, i.e. on some heuristic control policy, and then we will need to optimize the network parameters to solve the problem of finding the optimal scheduling policy. In the average cost criterion the limit of the expected average cost over finite time intervals is minimized in a set of admissible policies. The control policy $f : \hat{E} \to A(x)$ is a stationary policy which prescribes the usage of a control action $f(x) \in A(x)$ whenever at a decision epoch the system state is $x \in E$. The decision epochs arise whenever after serving any queue that queue becomes empty. For studied controllable queueing system operating under a control policy f the average cost per unit of time for the ergodic system is of the form

$$g^f = \lim_{t\to\infty} \frac{1}{t}\mathbb{E}^f\Big[\int_0^t \sum_{i=1}^N c_i Q_i(u)du + \sum_{i=1}^N \sum_{j=1}^N c_{i,j} S_{i,j}(t)\Big| X(0) = (d, 0, \ldots, 0)\Big]. \tag{3}$$

Here $S_{i,j}(t)$ is the random number of switches from queue i to queue j in time interval $[0, t]$. Expectation $\mathbb{E}^f$ must be calculated with respect to the control policy f. The policy f^* is said to be optimal when for any admissible policy f,

$$g^* := g^{f^*} = \min_f g^f. \tag{4}$$

For our purposes in case of a general model, we combine simulation modelling and neural networks. To verify the results of solving the optimization problem (4), we formulate an appropriate Markov decision problem, for which we compute using a policy iteration algorithm, see e.g. in Howard [16], Puterman [25], Tijms [30], the optimal control policy and the corresponding minimum average cost g^*.

3. Markov decision problem formulation

Assume that the inter-arrival and service times are exponentially distributed, i.e. $\nu_i \sim \mathcal{E}(\lambda_i)$ and $\zeta_i \sim \mathcal{E}(\mu_i), i \in I$. Under markovian assumption the process (1) is a continuous-time Markov chain with a state space E. The MDP associated with this Markov process is represented as a five-tuple:

$$(E, A, \{A(x), x \in E\}, \lambda_{xy}(a), c(x, a)), \tag{5}$$

where state space E, action spaces A and $A(x)$ have been already defined in the previous section.

– λ_{xy} is a transition intensity to go from state x to state y by choosing a control action a are defined as

$$\lambda_{xy}(a) = \begin{cases} \lambda_i & y = x + \mathbf{e}_i, \\ \mu_i & y = x - \mathbf{e}_i,\ d(x) = i,\ q_i(x) > 1, \\ \mu_i & y = x - \mathbf{e}_i + (a - i)\mathbf{e}_0,\ d(x) = i,\ q_i(x) = 1,\ a \in A(x - \mathbf{e}_i), \\ 0 & \text{otherwise for } y \neq x, \end{cases} \tag{6}$$

where $\lambda_{xx} := \lambda_{xx}(a) = -\sum_{y\neq x} \lambda_{xy}(a)$.

– $c(x, a)$ is an immediate cost in state $x \in E$ by selecting an action a,

$$c(x, a) = \sum_{i=1}^N c_i q_i(x) + \mu_j c_{j,a} 1_{\{d(x)=j, q_j(x)=1\}}.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Here the first summand denotes the total holding cost of customers in all parallel queues in state x which is independent of a control action. Let $c(x) = \sum_{i=1}^{N} c_i q_i(x)$ and if $c_i = 1, i \in I$, we get the number of customers in state x. The second summand includes the fixed cost $c_{j,a}$ for switching the server from the current queue j to the next queue numbered by a.

The optimal control policy f^* and the corresponding average cost g^{f^*} are the solutions of the system of Bellman optimality equations,

$$Bv(x) = -\lambda_{xx} v(x) + g = \Big[\sum_{i=1}^{N} \lambda_i + \mu_j 1_{\{d(x)=j, q_j(x)\geq 1\}}\Big] v(x) + g,\ x \in E, \tag{7}$$

where B is a dynamic programming operator acting on value function $v : E \to \mathbb{R}$.

Proposition 1. *The dynamic programming operator B is defined as*

$$\begin{aligned} Bv(x) = c(x) &+ \sum_{i=1}^{N} \lambda_i v(x + \mathbf{e}_i) + \mu_j v(x - \mathbf{e}_j) 1_{\{d(x)=j, q_j(x)>1\}} \\ &+ \mu_j \min_{a \in A(x-\mathbf{e}_j)} \{v(x - \mathbf{e}_j + (a-j)\mathbf{e}_0) + c_{j,a}\} 1_{\{d(x)=j, q_j(x)=1\}},\ x \in E. \end{aligned} \tag{8}$$

Proof. From the Markov decision theory, e.g. [25,30], it is known that for Markov chain with continuous time the operator B is defined as $Bv(x) = \min_a \Big[c(x,a) + \sum_{y \neq x} \lambda_{xy} v(y)\Big]$. This equality for the proposed system can be obviously rewritten in form (8). In this equation, the first term $c(x)$ represents the immediate holding cost of customers in state x. The second term by λ_i describes the changes in value function due to new arrivals to the system. The third term by μ_j for $q_j(x) > 1$ stands for the value function by service completion in the queue j where there are customers waiting for service. The last term by μ_j for $q_j(x) = 1$ describes also a service completion which leads now to the state with an empty queue when a control action must be performed. Hence only the last term occurs with a min operator. □

Note that the state space of the Markov decision model is countable infinite and the immediate costs $c(x,a)$ are unbounded. The existence of the optimal stationary policy and convergence of the policy iteration algorithm can be verified for the system under study in a similar way as in Özkan and Kharoufeh [24], where first, the convergence of the value iteration algorithm for the equivalent discounted model is proved, and then, using the criteria proposed in Sennott [28], this result is extended to the policy iteration algorithm for the average cost criterion.

To solve equations (8) in the policy iteration algorithm which calculates the optimal control policy, we convert the multidimensional state space into a one-dimensional space by mapping $\Delta : E \to \mathbb{N}_0$. Thus, the state $x = (d, q_1, \ldots, q_N)$ can be rewritten in the following form:

$$s := \Delta(x) = d(x)\beta_{1,N} + \sum_{i=1}^{N} q_i(x)\beta_{i,N-1}, \tag{9}$$

where $\beta_{i,j} = \prod_{k=i}^{j}(B_k + 1)$ with $\beta_{N,N-1} = 1$. The notation $\Delta^{-1}(s)$ will be used for the inverse function. In one-dimensional case the state transitions can be expressed as

$$\begin{aligned} \Delta(x \pm \mathbf{e}_i) &= \Delta(x) \pm \beta_{i,N-1}, \\ \Delta(x + (a-j)\mathbf{e}_0) &= \Delta(x) + (a-j)\beta_{1,N}. \end{aligned}$$

To calculate the optimal policy the buffer sizes must be truncated, i.e. $B_i < \infty$. In this case the set of states E is finite with a cardinality $|E| = N\beta_{1,N}$. To start the algorithm an initial policy is required. The LQF policy is used as initial policy in algorithm 1 which consists of two main steps: Policy evaluation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

and policy improvement. In first step for the given initial control policy the system of linear equations with constant coefficients must be solved, and the value function $v(s)$ for one of the states can be assumed to be an arbitrary constant, e.g. $v(0) = 0$ with $d = 1$ and $q_i = 0$. In this case we obtain from the optimality equation (7) the equality $g = \sum_{i=1}^{N} \lambda_i v(\beta_{i,N-1})$. The remaining equations can be solved numerically. As a solution we get the $|E|$ values for the function $v(s)$ and the current value of the average cost g. In the next policy improvement step, a control action a that minimizes the test value in the right-hand side of equation (7) must be evaluated. The algorithm generates a sequence of control policies that converges to the optimum one. The convergence of the algorithm requires that the control actions in two adjacent iterations coincide in each state. To avoid policy improvement bouncing between equally good control actions in a given state, one can simply keep the previous control action unchanged if the corresponding test function is at least as large as for any other policy in determining the new policy. Also, in order to fix the convergence of the algorithm, one can use the values of average costs, the variation of which should be for example less than a given some small value.

Algorithm 1 Policy iteration algorithm

procedure PIA($N, B_i, \lambda_i, \mu_i, c_i, c_{i,j}, i, j \in I$)

▷ Initial policy

$$f^{(0)}(s) = \begin{cases} \text{Random}\{\arg\max_{j \in I}\{q_j(\Delta^{-1}(s))\}\} & \text{if } d(\Delta^{-1}(s)) = i \in I,\ q_i(\Delta^{-1}(s)) = 0 \\ 0 & \text{otherwise} \end{cases}$$

$n \leftarrow 0$

$g^{(n)} \leftarrow \sum_{i=1}^{N} \lambda_i v^{(n)}(\beta_{i,N-1})$ ▷ Policy evaluation

for $s = 1$ **to** $|E|$ **do**

$$\begin{aligned} v^{(n)}(s) \leftarrow \frac{1}{\sum_{i=1}^{N} \lambda_i + \mu_j 1_{\{q_j(\Delta^{-1}(s))>0\}}} \Big[& c(\Delta^{-1}(s)) + \mu_j c_{j,a} 1_{\{d(\Delta^{-1}(s))=j, q_j(\Delta^{-1}(s))=1\}} - g^{(n)} \\ & + \sum_{i=1}^{N} \lambda_i [v^{(n)}(s + \beta_{i,N-1}) 1_{\{q_i(\Delta^{-1}(s))<B_i\}} + v(s) 1_{\{q_i(\Delta^{-1}(s))=B_i\}}] \\ & + \mu_j v^{(n)}(s - \beta_{j,N-1}) 1_{\{d(\Delta^{-1}(s))=j, q_j(\Delta^{-1}(s))>1\}} \\ & + \mu_j v^{(n)}(s - \beta_{j,N-1} + (a-j)\beta_{1,N}) 1_{\{d(\Delta^{-1}(s))=j, q_j(\Delta^{-1}(s))=1\}} \Big], \\ a \leftarrow f^{(n)}(s - \beta_{j,N-1}) & \end{aligned}$$

end for

▷ Policy improvement

$$f^{(n+1)}(s) \leftarrow \arg\min_{a \in A(s-\beta_{j,N-1})} \{c_{j,a} + v^{(n)}(s - \beta_{j,N-1} + (a-j)\beta_{1,N})\} 1_{\{d(\Delta^{-1}(s))=j, q_j(\Delta^{-1}(s))=1\}}$$

if $f^{(n+1)}(s) \leftarrow f^{(n)}(s)$, $s \in \{0, 1, \ldots, |E|\}$ **then return** $f^{(n+1)}(s), v^{(n)}(s), g^{(n)}$

else $n \leftarrow n + 1$, **go to step 4**

end if

end procedure

Example 1. Consider the queueing system with $N = 4$ queues. The buffer sizes are equal to $B_i = 10, i \in I$. At these settings the number of states already reaches large values, $|E| = 58564$, which confirms one of significant restrictions on application of dynamic programming for this type of control problems. The switching costs are defined as $c_{i,j} = j - i + 4 \mod 4$, so that it costs more to switch to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

queues with a higher service rate. The values of system parameters λ_i, μ_i, c_i and $c_{i,j}$ are summarized in Table 1.

Table 1. System parameters

i	1	2	3	4
λ_i	0.05	0.10	0.15	0.20
μ_i	3.750	1.875	1.250	0.938
c_i	1	1	1	1
$c_{i,1}$	0	3	2	1
$c_{i,2}$	1	0	3	2
$c_{i,3}$	2	1	0	3
$c_{i,4}$	3	2	1	0

These values correspond to the system load $\rho = \sum_{i=1}^{N} \rho_i = 0.4$ that is the system is stable. This value is enough small to ensure on the one hand that the system is sufficiently loaded so that states appear where all queues are not empty, and on the other hand to minimise the probability of losing an arriving customer for given rather small buffer sizes. The solution of the large system of optimality equations is carried out numerically. The optimized average cost $g^* = 2.5632$.

Table 2. The optimal scheduling policy for selected states

(d, q_1, q_2, q_3, q_4)	0	1	2	3	4	5	6	7	8	9	10
$(1, 0, q_2, 1, 1)$	4	4	4	4	4	4	4	4	4	4	4
$(1, 0, q_2, 3, 3)$	4	4	4	4	4	3	3	3	3	3	3
$(1, 0, q_2, 5, 5)$	4	3	3	3	3	3	3	2	2	2	2
$(1, 0, q_2, 8, 8)$	3	3	3	3	2	2	2	2	2	2	2
$(1, 0, q_2, 9, 9)$	3	3	2	2	2	2	2	2	2	2	2
$(2, q_1, 0, 1, 1)$	4	4	4	4	4	4	4	4	4	4	4
$(2, q_1, 0, 3, 3)$	4	4	4	4	4	4	3	3	3	3	3
$(2, q_1, 0, 5, 5)$	3	3	3	3	3	3	3	3	3	3	3
$(2, q_1, 0, 8, 8)$	3	3	3	3	3	3	3	3	3	3	3
$(2, q_1, 0, 9, 9)$	3	3	3	3	3	3	3	3	1	1	1

Using the Algorithm 1 we calculate the optimal scheduling policy. For some of states with fixed number of customers in the third and fourth queues and varied number of customers in the first two queues the control actions are listed in Table 2. The first row of the table contains the values of the number of customers q_2 and q_1 in the second or first queue when a decision is made respectively when the first or second queue is emptied. The first column contains some selected states of the system for the fixed levels q_3 and q_4 of the third and fourth queues. As we can see, the optimal scheduling policy has a complex structure with a large number of thresholds, making it difficult to obtain any acceptable heuristic solution explicitly. The $c\mu$-rule as expected is not optimal here, $g_{c\mu} = 6.7237$ that is almost two and a half times more than the value of the average cost under the optimal policy. When the values q_1 and q_2 are small, the router directs the server to serve the queues with low service rates, in this case the switching costs are low as well. According to the optimal scheduling the initiative to route a server to the queue with a higher service rate and switching costs increases as the length of the first two queues increases.

Example 2. In this example we increase the arrival rates λ_i as given in Table 3. The other parameters are fixed at the same values as in the previous example. The load factor now is $\rho = 0.64$, and the corresponding optimized average cost is $g^* = 3.8201$ and $g_{c\mu} = 7.0420$.

Table 3. System parameters

i	1	2	3	4
λ_i	0.08	0.16	0.24	0.32

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Table 4. The optimal scheduling policy for selected states

(d, q_1, q_2, q_3, q_4)	0	1	2	3	4	5	6	7	8	9	10
$(1, 0, q_2, 1, 1)$	3	2	2	2	2	2	2	2	2	2	2
$(1, 0, q_2, 3, 3)$	3	2	2	2	2	2	2	2	2	2	2
$(1, 0, q_2, 5, 5)$	3	2	2	2	2	2	2	2	2	2	2
$(1, 0, q_2, 8, 8)$	3	2	2	2	2	2	2	2	2	2	2
$(1, 0, q_2, 9, 9)$	3	2	2	2	2	2	2	2	2	2	2
$(2, q_1, 0, 1, 1)$	3	3	1	1	1	1	1	1	1	1	1
$(2, q_1, 0, 3, 3)$	3	3	1	1	1	1	1	1	1	1	1
$(2, q_1, 0, 5, 5)$	3	1	1	1	1	1	1	1	1	1	1
$(2, q_1, 0, 8, 8)$	3	1	1	1	1	1	1	1	1	1	1
$(2, q_1, 0, 9, 9)$	3	1	1	1	1	1	1	1	1	1	1

The table of scheduling policy shows that as the system load increases the router switches the server to queue 2 or to queue 1 with a higher service rates at almost all queue lengths q_1 and q_2 respectively.

4. Event based simulation for the general model

To simulate the proposed queueing system we use an event based simulation. This technique is suitable for random process evaluation where it is sufficient to have the information about the time instants when changes in states occur. Such changes will refer as events. Note that although simulation modelling is extensively used in queueing theory, many papers lacks explicitly described algorithms that readers can use for independent research. For more information on simulation methods with applications to single- and multi-server queueing systems we can recommend Ebert et al. [7] and Franzl [11]. In this regard, it will not be certainly superfluous, if we preset and discuss here an algorithm for the system simulation which is not difficult to adapt for other similar systems.

In our case the events are the arrivals to one of N parallel queues and the departures of customers from the queue d currently being served by the server. The present time is selected as a global time reference.

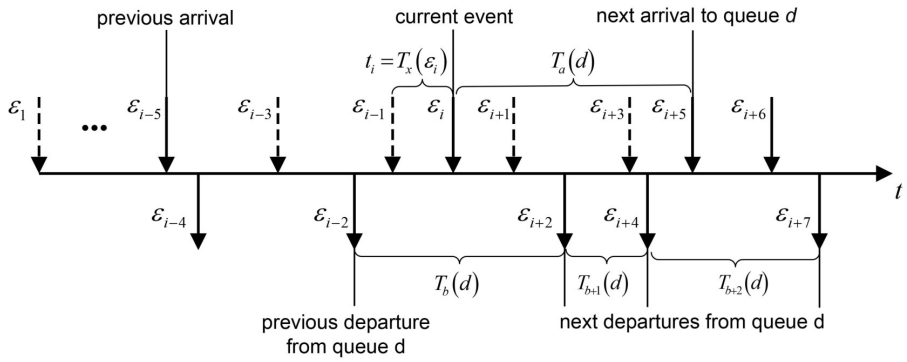

Figure 2. The time assignment for the present time based simulation

In Figure 2, on the time axis we mark the moments of arrival of new applications and the moments of their service in a fixed queue with number d by means of arrows above and below the axis respectively. The dotted arrows indicate the arrival of new applications in other queues. The successive events are denoted by ε_i and the corresponding time moments by $t(\varepsilon_i)$. In the proposed queue simulation algorithm 2 all the times are referred to the present time. Suppose that at the present moment of time there is a new arrival to the queue with the number d, which is serviced by the server, i.e. $t(\varepsilon_i) = 0$. Denote by $T_x(\varepsilon_i)$ the holding time of the system in state x up to the occurrence of the event ε_i. According to the time schema the holding time in a previous state is defined as $t_i = \min\{T_x(\varepsilon_i), T_b(d) - T_x(\varepsilon_{i-1}), \dots\} = T_x(\varepsilon_i)$, where $T_x(\varepsilon_i)$ is a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

remaining inter-arrival time to the queue d, $T_b(d)$ stands for the generated service time after the event ε_{i-2} of the previously occurred departure and the dots replace the time intervals associated with arrivals of customers in other queues. The next event is determined then by subtracting the holding time t_i from the all event time intervals. In this case the current event is a new arrival. Thus the holding time t_{i+1} in state up to the event ε_{i+1} of an arrival to some other queue which not equal to d is of the form $t_{i+1} = \min\{T_a(d), T_b(d) - \sum_{j=i-1}^{i} T_x(\varepsilon_j), \dots\} = T_x(\varepsilon_{i+1})$. The subsequent holding times are calculated as follows, $t_{i+2} = \min\{T_a(d) - T_x(\varepsilon_{i+1}), T_b(d) - \sum_{j=i-1}^{i+1} T_x(\varepsilon_j), \dots\} = T_x(\varepsilon_{i+2}) = T_b(d) - \sum_{j=i-1}^{i+1} T_x(\varepsilon_j)$, i.e. the event ε_{i+2} is then the next departure from queue d, $t_{i+3} = \min\{T_a(d) - \sum_{j=i+1}^{i+2} T_x(\varepsilon_j), T_{b+1}(d), \dots\} = T_x(\varepsilon_{i+3})$, where $T_{b+1}(d)$ is the next generated service time, $t_{i+4} = \min\{T_a(d) - \sum_{j=i+1}^{i+3} T_x(\varepsilon_j), T_{b+1}(d) - T_x(\varepsilon_{i+3}), \dots\} = T_x(\varepsilon_{i+4}) = T_{b+1}(d) - T_x(\varepsilon_{i+3})$ and $t_{i+5} = \min\{T_a(d) - \sum_{j=i+1}^{i+4} T_x(\varepsilon_j), T_{b+2}, \dots\} = T_x(\varepsilon_i + 5) = T_a(d) - \sum_{j=i+1}^{i+4} T_x(\varepsilon_j)$ is a remaining inter-arrival time for the next arrival to the queue d. Continuing the process in a similar manner, all holding times of the system in the corresponding states are evaluated. By summing up the times t_i we obtain the total simulation running time of the system $simT$. The average cost per unit of time is then obtained by division of the accumulated cost by the time $symT$.

The time instants of arrival events to the queue $q \in I$ are stored in vector variable T_a and the departure events in the queue with a number q in $T_b[q]$. The algorithm 2 contains the pseudo-code of the main elements of the event based simulation procedure.

5. Neural network architecture

In our model, we propose to equip the router with a trained neural network. This network will determine a sequence number of the queue that the server will serve next based on the information about the system state at a decision epoch when the server finishes service of a certain queue. We have chosen a simple architecture for the neural network consisting of only two layers in such a way that, on the one hand, it would have a small number of parameters for further optimization and, on the other hand, that the quality of correct classification of some fixed initial control policy would be equal at least 95%. The proposed neural network has one linear layer which represents an affine transformation and softmax normalization layer as illustrated in Figure 3.

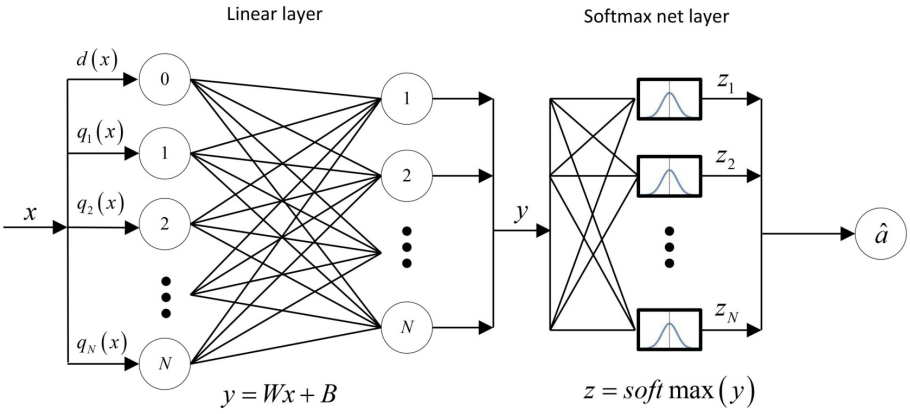

Figure 3. Neural network architecture

The input includes $N+1$ neurons according to the system state $x = (d, q_1, \dots, q_N)$, where $q_d(x) = 0$. The neuron 0 gets the information on $d(x)$, the ith neuron for $i \in I$ gets the information on the state of ith queue. When the server finishes service at the queue d, then the neural network classifies this state to one of N classes which defines a current control action $a \in A$ for the state x. The

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Algorithm 2 Queue simulation algorithm

procedure QSIM($N, B_i, A_i, B_i, c_i, c_{i,j}, i, j \in I, \theta, n_{\max}, n_{\min}$) ▷ Initialization
  $T_a = (0, \ldots, 0)$, $|T_a| = N$, $T_b = ((\infty), \ldots, (\infty))$, $|T_b| = N$, $xT = 0$, $i = 0$, $sc = 0$
  $d = \text{Random}[\{1, \ldots, N\}]$, $x = (d, 0, \ldots, 0)$, $|x| = N + 1$
  while $i < n_{\max}$ **do** ▷ State recording
    $t_i \leftarrow \min(T_a, \min(T_b[1]), \ldots, \min(T_b[N]))$
    $T_a \leftarrow T_a - t_i$
    for $q = 1$ **to** N **do**
      $T_b[q](2 : |T_b[q]|) \leftarrow T_b[q](2 : |T_b[q]|) - t_i$
    end for
    if $i > n_{\min}$ **then**
      $simT \leftarrow simT + t_i$ ▷ Simulation time
      $xT \leftarrow xT + t_i \sum_{j=1}^{N} c_j x[j+1] + sc$ ▷ Sum up the cost
    end if
    $cs \leftarrow 0$
    for $q = 1$ **to** N **do**
      if $(q = d \,\&\, T_a[q] \leq \varepsilon)$ **then return**
        $T_a[q] \leftarrow \text{RandomVariate}[A_q(t)]$ ▷ Generate interarrival time
        $x \leftarrow x + \mathbf{e}_{q+1}(N+1), i \leftarrow i + 1$
        if $|T_b[q]| \leq 1$ **then return**
          $T_b[q] \leftarrow (T_b[q], \text{RandomVariate}[B_q(t)])$ ▷ Generate service time
        end if
      end if
      if $(q = d \,\&\, T_a[q] > \varepsilon \,\&\, |T_b[q] \leq \varepsilon| > 0)$ **then return**
        index $\leftarrow T_b[q] \leq \varepsilon$ ▷ Index of the current departure
        $T_b[q] \leftarrow (T_b[q] \setminus T_b[q][\text{index}])$ ▷ Remove current departure
        $x \leftarrow x - \mathbf{e}_{q+1}(N+1)$
        if $x[q+1] \geq 1$ **then return**
          $T_b[q] \leftarrow (T_b[q], \text{RandomVariate}[B_q(t)])$
        end if
        if $x[q+1] = 0$ **then return**
          $a \leftarrow f(x, \theta), d \leftarrow a$ ▷ New server scheduling
          $x \leftarrow x + (a - q)\mathbf{e}_1(N+1)$
          $sc = c_{q,a}$
          if $x[a+1] > 0$ **then return**
            $T_b[a] \leftarrow (T_b[a], \text{RandomVariate}[B_a(t)])$ ▷ Generate service time
          end if
        end if
      end if
      if $(q \neq d \,\&\, T_a[q] \leq \varepsilon)$ **then return**
        $T_a[q] \leftarrow \text{RandomVariate}[A_q(t)]$ ▷ Generate interarrival time
        $x \leftarrow x + \mathbf{e}_{q+1}(N+1), i \leftarrow i + 1$
      end if
    end for
  end while
  $g \leftarrow xT/simT$
end procedure

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

hidden linear layer consists of N neurons $y = (y_1, \dots, y_N)'$ which are connected with an input neurons via the system of linear equations

$$\begin{aligned} y_1 &= w_{1,0}x_0 + w_{1,1}x_1 + \cdots + w_{1,N}x_N + b_1 \\ y_2 &= w_{2,0}x_0 + w_{2,1}x_1 + \cdots + w_{2,N}x_N + b_2 \\ &\dots \\ y_N &= w_{N,0}x_0 + w_{N,1}x_1 + \cdots + w_{N,N}x_N + b_N, \end{aligned}$$

or in matrix form $y = Wx + B$ with $W \in \mathbb{R}^{N\times(N+1)}$ and $B \in \mathbb{R}^N$, where

$$W = \begin{pmatrix} w_{1,0} & w_{1,1} & \dots & w_{1,N} \\ w_{2,0} & w_{2,1} & \dots & w_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N,0} & w_{N,1} & \dots & w_{N,N} \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} \text{ and } B = (b_1, b_2, \dots, b_N)' \tag{10}$$

with $w_i = (w_{i,0}, w_{i,1}, \dots, w_{i,N})$ are respectively the matrix of weights and the vector of biases of the given neural network which must be estimated by means of the training set. The softmax layer $z = \text{softmax}(y)$ is a final layer of the multiclass classification. The softmax layer generates as an output the vector of N estimated probabilities of the input sample y_i, where the ith entry is the likelihood that x belongs to class i. The vector y is normalized by the transformation

$$z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} = \frac{1}{\sum_{i=1}^N e^{y_i}} \begin{pmatrix} e^{y_1} \\ e^{y_2} \\ \vdots \\ e^{y_N} \end{pmatrix}.$$

The class number is then defined as $\hat{a} = \arg\max z_i$. Hence, the output z is a mapping of the form $z = \varphi(x, \theta)$, where $\theta \in \mathbb{R}^{N(N+2)}$ is the parameter vector of the neural network which includes all entries of the weight matrix $W \in \mathbb{R}^{N\times(N+1)}$ and the bias vector $B \in \mathbb{R}^N$, i.e.

$$\theta = (w_1, w_2, \dots, w_N, B'). \tag{11}$$

The values of the parameter vector θ of the initial control policy, which in the next section will be used as a starting solution for optimization procedure, are obtained by training the neural network on some known heuristic control policy. In our case this policy is the LQF. In the training phase the following optimization problem must be solved given the training set $\{x^{(k)}\}_{k=1}^m \to \{a^{(k)}\}_{k=1}^m$,

$$\theta^* = \arg\min_\theta \frac{1}{m} \sum_{k=1}^m l_k(\theta), \tag{12}$$

where a non-negative loss function

$$l_k(\theta) = -\sum_{i=1}^N 1_{\{a^{(k)}=i\}} \ln z_i^{(k)}$$

with $z_i^{(k)} = \mathbb{P}[a^{(k)} = i | x^{(k)}, \theta]$ takes the value 0 only if the class of the kth element of a sample is i, i.e. $\hat{a} = a^{(k)}$. The problem (12) can be solved in a usual way by the stochastic gradient descent method, where a single learning rate η to update all parameters is maintained. The corresponding iterative expression is given below,

$$\theta^{(n)} = \theta^{(n-1)} - \eta \nabla_\theta \Big(\frac{1}{m} \sum_{k=1}^m l_k(\theta^{(n-1)}) \Big),$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

where ∇_θ is a Nabla-operator defining the gradient of the function relative to the parameter vector θ. In our calculations we use the adaptive moment estimation algorithm (ADAM) to solve the problem (12). It updates iteratively the parameters of the neural network based on training data. The ADAM calculates independent adaptive learning rates for the elements of θ by evaluating the first-moment and second moment estimation of the gradient. The method is simple to implement, computationally efficient, requires little memory and is invariant to diagonal changes in gradients. The further detailed information regarding ADAM algorithm can be found in Kingma and Ba [17]. Despite of fact that the ADAM algorithm can be found in various sources, we have also chosen to cite it in this article. The main steps required for the iterative updating the parameter vector θ are summarized in the algorithm 3. The parameters of the algorithm 3 are fixed to $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$

Algorithm 3 Adaptive moment estimation algorithm

procedure ADAM(η,β_1,β_2,ε,δ)
  $M_1^{(0)} \leftarrow (0,\ldots,0)$ ▷ Initialisation of the moment 1
  $M_2^{(0)} \leftarrow (0,\ldots,0)$ ▷ Initialisation of the moment 2
  $CI \leftarrow 0$ ▷ Convergence index
  $n \leftarrow 0$
  while $CI = 0$ **do**
    $n \leftarrow n+1$
    $G^{(n)} \leftarrow \nabla_\theta\left(\frac{1}{m}\sum_{k=1}^{m} l_k(\theta^{(n-1)})\right)$ ▷ Calculate the gradient at step n
    $M_1^{(n)} \leftarrow \beta_1 M_1^{(n-1)} + (1-\beta_1)G^{(n)}$ ▷ Update the biased first moment
    $M_2^{(n)} \leftarrow \beta_2 M_2^{(n-1)} + (1-\beta_2)(G^{(n)})^2$ ▷ Update the biased second moment
    $\hat{M}_1^{(n)} \leftarrow \frac{M_1^{(n)}}{1-\beta_1^n}$ ▷ The bias corrected first moment
    $\hat{M}_2^{(n)} \leftarrow \frac{M_2^{(n)}}{1-\beta_2^n}$ ▷ The bias corrected second moment
    $\theta^{(n)} = \theta^{(n-1)} - \eta\frac{\hat{M}_1^{(n)}}{\sqrt{\hat{M}_2^{(n)}+\varepsilon}}$ ▷ Update the parameter vector
    if $|\theta^{(n)} - \theta^{(n-1)}| < \delta$ **then return** $\theta^{(n)}$
      $CI \leftarrow 1$ ▷ Check the convergence
    end if
  end while
end procedure

and $\delta = 0.001$. The classification accuracy of the proposed neural network trained on the LQF policy is over 97%. The test phases of the trained network were conducted on system states with a queue length of up to 100 customers per queue. Thus this starting network can be used to generate control actions of the initial control policy for subsequent parameters' optimization of this neural network.

6. Optimization of the neural network based allocation policy

Denote by θ the known parameter vector of the trained neural network as was defined in (11). The function $g(\theta)$ means the average cost for the queueing system where the router chooses an action obtained from the trained neural network with the parameter vector θ. We adapt further a simulated annealing method described in algorithm 4 for discrete stochastic optimization of the average cost function

$$g^* = \min_\theta g(\theta),\ \theta^* = \arg\min_\theta g(\theta) \tag{13}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

with a multidimensional parameter vector θ. This algorithm is quite straightforward. It needs some starting solution and in each iteration the algorithm evaluates for the randomly selected neighbor values of the function parameters the corresponding function value. If the neighbor occurs to be better than the current solution with respect to value of the objective function, algorithms replaces the current solution with a new one. If the neighbor value is worse, the algorithm keeps the current solution with a high probability and chooses a new value with a specified low probability. The simulated annealing

Algorithm 4 Simulated annealing algorithm

procedure SA($T(n)$,Δ,m,η,τ,ν,$\theta_{\min}$,$\theta_{\max}$) ▷ Initialisation
  $\theta^{(0)} \leftarrow (w_{1,\mathrm{LQF}}, w_{2,\mathrm{LQF}}, \ldots, w_{N,\mathrm{LQF}}, B'_{\mathrm{LQF}})$
  $n \leftarrow 0$
  $\bar{g}(\theta^{(n)}) \leftarrow \frac{1}{m}\sum_{k=1}^{m} \mathrm{QSIM}(\ldots, \theta^{(n)})$
  $g^* \leftarrow \bar{g}(\theta^{(n)}), \theta^* \leftarrow \theta^{(n)}$
  while $T(n) > \tau || n < \nu$ **do**
    $n \leftarrow n + 1$ ▷ Perturbation
    $i \leftarrow \mathrm{Random}[\{1, \ldots, N(N+2)\}]$
    $\xi \leftarrow \mathrm{Random}[\{\max\{-\eta\Delta, \theta_{\min} - \theta_i^{(n-1)}\}, \ldots, \min\{\eta\Delta, \theta_{\max} - \theta_i^{(n-1)}\}\}]$
    $\theta^{(n)} \leftarrow \theta^{(n-1)} + \xi e'_i$
    $\bar{g}(\theta^n) \leftarrow \frac{1}{m}\sum_{k=1}^{m} \mathrm{QSIM}(\ldots, \theta^{(n)})$ ▷ Acceptance
    if $\bar{g}(\theta^{(n)}) - g^* - S_{g(\theta^{(n)}),g(\theta^{(n-1)})} t_{2m-2;1-\alpha} > 0$ **then return**
      $p \leftarrow e^{-\frac{\bar{g}(\theta^{(n)}) - g^* - S_{g(\theta^{(n)}),g(\theta^{(n-1)})} t_{2m-2;1-\alpha}}{T(n)}}$
    else $p \leftarrow 1$
    end if
    $u \leftarrow \mathrm{Random}[]$
    if $p \geq u$ **then return** $g^* \leftarrow \bar{g}(\theta^{(n)}), \theta^* \leftarrow \theta^{(n)}$
    else $\theta^{(n)} \leftarrow \theta^{(n-1)}, m \leftarrow m + 1$
    end if
  end while
end procedure

requires the finite discrete space for the parameters of the optimized function. It is assumed that all weights and biases of the neural network summarized in the vector θ take values in the interval $[\theta_{\min}, \theta_{\max}]$ with a low bound $\theta_{\min}$ and an upper bound $\theta_{\max}$. Moreover, this interval is quantized in such a way that $\theta_i, i = 1, \ldots, N(N+2)$, takes only discrete values $\theta_{\min} + k\Delta, k = 0, 1, \ldots, Q$, where $Q = \frac{\theta_{\max} - \theta_{\min}}{\Delta}$ is a quantization level. Note that the domains for the elements of the parameter vector θ can be specified separately, and the values of the vector obtained by training the neural network based on the optimal policy of the Markov model will be suitable for determining the possible maximum and minimum bounds. In this case it is possible to achieve faster convergence of algorithm 4 to the optimal value.

Since the average cost function g can not be calculated analytically, for this purpose a simulation technique is used. As it is shown in algorithm 4, at each iteration at the step where the current solution can be accepted with a given probability we need to calculate the difference between the object functions. Due to the fact that this function can be calculated only numerically, it is necessary to check whether this difference is statistically significant at each iteration of the algorithm. The algorithm is modified in such a way that the t-test for two samples is used to compare the expected values of two

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

normally distributed samples with unknown but equal variances. Denote by θ_1 and θ_2 respectively the current and the modified parameter vector and by

$$\bar{g}(\theta_1) = \frac{1}{m}\sum_{k=1}^{m} g^{(k)}(\theta_1),\ \bar{g}(\theta_2) = \frac{1}{m}\sum_{k=1}^{m} g^{(k)}(\theta_2) \tag{14}$$

two corresponding first empirical moments of the objective function. According to the t-test the null hypothesis which states that for the modified vector the average cost is statistically smaller then the previous solution is rejected if

$$\bar{g}(\theta_2) - \bar{g}(\theta_1) - S_{g(\theta_1),g(\theta_2)} t_{2m-2;1-\alpha} > 0, \tag{15}$$

where $t_{m;q}$ stands for the q-quantile of the t-distribution and statistics $S_{g(\theta_1),g(\theta_2)}$ is defined as

$$S_{g(\theta_1),g(\theta_2)} = \sqrt{\frac{V^{(m)}_{g(\theta_1)} + V^{(m)}_{g(\theta_2)}}{m}}, \tag{16}$$

with empirical variances $V^{(m)}_{g(\theta_1)}$ and $V^{(m)}_{g(\theta_2)}$.

Below we briefly describe the main steps of the algorithm 4. At the initialisation step of the algorithm, the neural network is trained based on the LQF control policy. The parameter vector is then equal to the initial vector $\theta^{(0)}$ to be optimized. The simulation algorithm 2 is then used to calculate the initial sample $\{g^{(k)}(\theta^{(0)})\}_{k=1}^{m}$ with $g^{(k)}(\theta^{(0)}) = \mathrm{QSIM}(\dots)$ of the average cost function for a given initial parameter vector $\theta^{(0)}$ and the corresponding first empirical moment $\bar{g}(\theta^{(0)})$. These values are set as the current solution g^* and θ^* to the optimization problem (13). At the perturbation step, a randomly chosen element of the previous parameter vector $\theta^{(n-1)}$ must be randomly perturbed on the specified set

$$L(i) = \{\max\{\theta_i^{(n-1)} - \eta\Delta, a_{\min}\}, \dots, \min\{\theta_i^{(n-1)} + \eta\Delta, a_{\max}\}\}$$

of admissible discrete domain. For the new parameter vector $\theta^{(n)}$ next sample $\{g^{(k)}(\theta^{(n)})\}_{k=1}^{m}$ of average costs must be calculated together with the first empirical moment $\bar{g}(\theta^{(n)})$. At the acceptance step, a new policy $\theta^{(n)}$ can be accepted as a current solution with a probability p defined as

$$p = \begin{cases} 1 & \text{if } \bar{g}(\theta^{(n)}) \le g^* \\ e^{-\frac{\bar{g}(\theta^{(n)}) - g^* - S_{g(\theta^{(n)}),g(\theta^{(n-1)})} t_{2m-2;1-\alpha}}{T(n)}} & \text{if } \bar{g}(\theta^{(n)}) > g^*, \end{cases}$$

where $T(n)$ is the temperature at the nth iteration. If a new policy $\theta^{(n)}$ is accepted, then it is defined together with a corresponding average cost $\bar{g}(\theta^{(n)})$ as a current solution. Otherwise, the last change in the parameter vector $\theta^{(n-1)}$ must be reversed, i.e. $\theta^{(n)} = \theta^{(n-1)}$ and the sample size m for calculating the first moments is updated. Then the perturbation step must be repeated. For termination of the algorithm the stopping criteria $T(n) < \tau$ or $n < \nu$ is used.

We note that the classical simulated annealing method generates for some function $g(\theta)$ a sample $\theta^{(n)}$ which for the constant temperature $T(n) = T$ can be interpreted as a realization of a homogeneous Markov chain $\{\Theta_n\}_{\{}n \in \mathbb{N}_0\}$ with transition probabilities

$$p_{\theta_i,\theta_j} = \mathbb{P}[\Theta_{n+1} = \theta_j | \Theta_n = \theta_i] = \frac{1}{|L(i)|}\mathbb{P}\left[U_n \le e^{-\frac{g(\theta_j) - g(\theta_i)}{T}}\right],\ \theta_j \in L(i), \tag{17}$$

where U_n is a uniformly distributed random variable on the interval $[0, 1]$. It is easy to show that the modified transition probabilities, where the objective function is calculated numerically, converges to the transition probabilities (17) which in turn can guarantee the convergence to an optimal solution.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Proposition 2. *The acceptance probability $p(n)$ satisfies the limit relation*

$$\lim_{n\to\infty} p(n) = \lim_{n\to\infty} \mathbb{P}\Big[U_n \le e^{-\frac{\bar{g}(\theta_j)-\bar{g}(\theta_i)-S_{g(\theta_j),g(\theta_i)}t_{2m-2;1-\alpha}}{T}}\Big] = \mathbb{P}\Big[U_n \le e^{-\frac{g(\theta_j)-g(\theta_i)}{T}}\Big]. \tag{18}$$

Proof. The probability $\mathbb{P}[U_n \le X]$ can be obviously rewritten as

$$\mathbb{P}[U_n \le X] = \int_0^1 \mathbb{P}[u \le X] f_{U_n}(u)du = \mathbb{E}[X],$$

where $X = e^{-\frac{\bar{g}(\theta_j)-\bar{g}(\theta_i)-S_{g(\theta_j),g(\theta_i)}t_{2m-2;1-\alpha}}{T}}$. Then the following relation holds,

$$\lim_{n\to\infty} \mathbb{E}\Big[e^{-\frac{\bar{g}(\theta_j)-\bar{g}(\theta_i)-S_{g(\theta_j),g(\theta_i)}t_{2m-2;1-\alpha}}{T}}\Big] = \mathbb{E}\Big[e^{-\frac{g(\theta_j)-g(\theta_i)}{T}}\Big],$$

due to the strong law of large numbers and the fact that for $n \to \infty$ the sample size $m \to \infty$ and hence

$$\lim_{m\to\infty} S_{g(\theta_j),g(\theta_i)} = \lim_{m\to\infty} \sqrt{\frac{\sigma_j^2+\sigma_i^2}{m}} = 0.$$

□

7. Numerical analysis

Consider the queueing system with $N = 4$. We first analyze a Markov model, where the parallel queues are of the type $M/M/1$ with $\nu_i \sim \mathcal{E}(\lambda_i)$ and $\zeta_i \sim \mathcal{E}(\mu_i), i \in I$, the coefficient of variation $CV_{\nu_i}^2 = CV_{\zeta_i}^2 = 1$. The values of system parameters λ_i and μ_i are fixed as in examples 1 and 2 which will refer to as Case 1 and Case 2. For this model it is possible to compare the optimization results obtained by combining the simulation, trained neural network and simulated annealing algorithm with the results of the policy iteration algorithm. In Cases 1 and 2 the weights and biases of the trained neural network based on PIA take respectively the following values

$$W_{\text{PIA}} = \begin{pmatrix} 0.4 & 3.2 & 0.3 & 0.1 & 0.2 \\ -0.3 & -3.8 & 0.8 & 0.2 & 0.2 \\ 0.1 & -2.9 & -3.6 & 0.4 & 0.3 \\ 0.4 & -0.3 & -1.6 & -1.3 & 0.3 \end{pmatrix} \qquad B_{\text{PIA}} = (-1.6, 1.0, 0.9, 0.4),$$

$$W_{\text{PIA}} = \begin{pmatrix} 0.5 & 2.0 & 0.2 & 0.0 & 0.3 \\ -0.3 & -2.0 & 0.7 & 0.0 & 0.3 \\ 0.2 & -1.3 & -2.1 & 0.0 & 0.4 \\ 0.1 & 0.0 & -1.0 & 0.0 & 0.3 \end{pmatrix} \qquad B_{\text{PIA}} = (-1.1, 1.1, 0.6, 0.0).$$

Based on these values, in the simulation annealing algorithm 4 we can define the domain for each element of the vector θ. In our experiments for simplicity we set general boundaries for all elements at values $\theta_{\min} = -6$ and $\theta_{\max} = 6$. The length of the increment $\Delta = 0.1$ implies the quantization level $Q = 120$. Next we set $\eta = 6, \nu = 200$, and $T(n) = \frac{0.2}{\log(n)}$. As an initial value we take the parameter vector obtained for the LQF policy. For the initial control policy one could also choose the policy $W_{\text{PIA}}, B_{\text{PIA}}$ obtained by algorithm 1. However, we would like to check the convergence of the algorithm when choosing not the best initial solution, since in general case one usually chooses either some heuristic policy or an arbitrary one. The empiric average cost $\bar{g}(\theta^{(n)})$ for each iteration step is calculated based on sample with a size $m \ge 20$. The accumulation of sample data in QSIM algorithm 2 is carried out after 1000 customers have entered the system and is completed after 5000 customers have entered the system.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

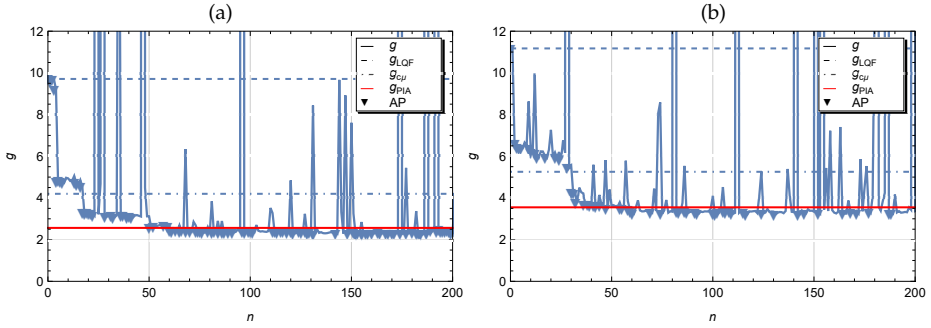

Figure 4. Iteration steps for g with $\nu_i \sim \mathcal{E}(\lambda_i)$ and $\zeta_i \sim \mathcal{E}(\mu_i)$ for Case 1 (a) and Case 2 (b)

Application of the algorithm 4 to a Markov model leads to the following optimal solutions:

Case 1: Optimal solution is reached at $n = 184$, $g^* = g(\theta^*) = 2.2436$,

$$W_{\mathrm{LQF}} = \begin{pmatrix} 0.5 & 2.0 & -1.0 & 0.7 & -0.9 \\ 0.3 & -0.7 & 1.6 & -0.7 & -0.9 \\ 0.4 & -0.6 & -1.3 & 2.0 & -0.8 \\ 0.0 & -0.6 & -1.3 & -1.8 & 1.9 \end{pmatrix} \Rightarrow W_{\mathrm{SA}} = \begin{pmatrix} 0.7 & 5.3 & -0.3 & -0.8 & 0.9 \\ 0.4 & -2.8 & 1.6 & 0.2 & -0.1 \\ 0.4 & -5.7 & -5.9 & 1.6 & 0.9 \\ 1.0 & -0.7 & -2.3 & -2.8 & 1.2 \end{pmatrix}$$

$$B_{\mathrm{LQF}} = (0.5, -0.1, -0.2, -0.2)' \qquad \Rightarrow B_{\mathrm{SA}} = (-1.8, -0.2, -0.6, -0.3)'.$$

Case 2: Optimal solution is reached at $n = 188$, $g^* = g(\theta^*) = 3.2279$,

$$W_{\mathrm{LQF}} = \begin{pmatrix} 0.5 & 2.0 & -1.0 & 0.7 & -0.9 \\ 0.3 & -0.7 & 1.6 & -0.7 & -0.9 \\ 0.4 & -0.6 & -1.3 & 2.0 & -0.8 \\ 0.0 & -0.6 & -1.3 & -1.8 & 1.9 \end{pmatrix} \Rightarrow W_{\mathrm{SA}} = \begin{pmatrix} 0.4 & 5.3 & -0.6 & 0.8 & 0.4 \\ -0.2 & -3.2 & 1.8 & 0.0 & 0.3 \\ 0.5 & -3.1 & -3.9 & 1.4 & 0.0 \\ 0.4 & -1.0 & -1.0 & -3.5 & 1.3 \end{pmatrix}$$

$$B_{\mathrm{LQF}} = (0.5, -0.1, -0.2, -0.2)' \qquad \Rightarrow B_{\mathrm{SA}} = (-2.5, 0.6, 0.0, 0.6)'.$$

The optimization process of the scheduling policy is illustrated in Figure 4. In addition to the numerical results of calculating the average cost at each iteration step of the simulated annealing algorithm, the figures show horizontal dotted and dash-dotted lines respectively at the level of the average cost $g_{\mathrm{LQF}} = 9.7093$ and $g_{c\mu} = 4.1984$ in figure labeled by (a) and $g_{\mathrm{LQF}} = 11.1740$ and $g_{c\mu} = 5.2546$ in figure labeled by (b) for the LQF and $c\mu$ heuristic policies. As expected, control policy LQF implies too high average cost. The results look much better for policy $c\mu$, but still the presence of switching costs significantly worsens the performance of this policy. The red horizontal line indicates the average cost $g_{\mathrm{PIA}} = 2.5632$ and $g_{\mathrm{PIA}} = 3.5500$ obtained by solving the Markov decision problem using the policy iteration algorithm 1. We can observe that the values are quite close to those obtained by random search. However, some small difference may be due, firstly, to the fact that the simulation is used for the calculations and the results have a certain scattering, and ,secondly, we do not exclude the influence of boundary states in the Markov model, where a buffer size truncation has been used. In the figures, we have also marked with triangles those iteration steps with accepted policy (AC) where the perturbed parameter vector has been accepted. The number of accepted points in Case 1 and 2 is equal respectively to 98 and 110. From above results in case of exponential time distributions we can make the following observations. If the parameter vector $\theta^{(0)}$ with elements W_{PIA} and B_{PIA} is used for the initial scheduling policy, then one can expect the faster convergence of the simulated annealing algorithm to the optimal solution which was confirmed numerically. If an optimal policy for a controlled Markov process is not available, e.g. when the number of queues is too large, in this case it is reasonable to use the static $c\mu$-rule as the initial policy.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

The Figure 5 displays experiments realized for the queues of the type $D/D/1$ with deterministic inter-arrival and service times which are equal to corresponding mean values $\frac{1}{\lambda_i}$ and $\frac{1}{\mu_i}$ of the Markov model. Here the coefficient $CV^2_{\nu_i} = CV^2_{\zeta_i} = 0$. The SA algorithm converges to the values $g^* = 1.6500$ and $g^* = 2.0326$ respectively for Case 1 and 2 with the following optimal policies,

Case 1:

$$W_{\mathrm{LQF}} = \begin{pmatrix} 0.5 & 2.0 & -1.0 & 0.7 & -0.9 \\ 0.3 & -0.7 & 1.6 & -0.7 & -0.9 \\ 0.4 & -0.6 & -1.3 & 2.0 & -0.8 \\ 0.0 & -0.6 & -1.3 & -1.8 & 1.9 \end{pmatrix} \Rightarrow W_{\mathrm{SA}} = \begin{pmatrix} 0.4 & 2.5 & -0.7 & -0.3 & -0.2 \\ 0.4 & -3.6 & 1.6 & 0.5 & -0.5 \\ 0.5 & -5.8 & -1.2 & 1.2 & 0.0 \\ 0.9 & -0.1 & -3.5 & -2.9 & 1.2 \end{pmatrix}$$

$$B_{\mathrm{LQF}} = (0.5, -0.1, -0.2, -0.2)' \qquad \Rightarrow B_{\mathrm{SA}} = (0.1, 0.5, 0.8, 0.6)'.$$

Case 2:

$$W_{\mathrm{LQF}} = \begin{pmatrix} 0.5 & 2.0 & -1.0 & 0.7 & -0.9 \\ 0.3 & -0.7 & 1.6 & -0.7 & -0.9 \\ 0.4 & -0.6 & -1.3 & 2.0 & -0.8 \\ 0.0 & -0.6 & -1.3 & -1.8 & 1.9 \end{pmatrix} \Rightarrow W_{\mathrm{SA}} = \begin{pmatrix} 0.1 & 4.5 & -1.2 & 0.7 & 0.9 \\ -0.7 & -2.6 & 1.8 & 0.0 & 0.5 \\ 0.0 & -0.3 & -3.8 & 0.6 & 0.6 \\ 0.5 & -1.1 & -3.4 & -1.0 & 0.7 \end{pmatrix}$$

$$B_{\mathrm{LQF}} = (0.5, -0.1, -0.2, -0.2)' \qquad \Rightarrow B_{\mathrm{SA}} = (-2.0, -0.3, 0.3, -1.0)'.$$

The average costs for heuristic policies take the values $g_{\mathrm{LQF}} = 3.7333, g_{c\mu} = 2.8000, g_{\mathrm{PIA}} = 1.6500$ and $g_{\mathrm{LQF}} = 5.0133, g_{c\mu} = 3.9866, g_{\mathrm{PIA}} = 2.7373$.

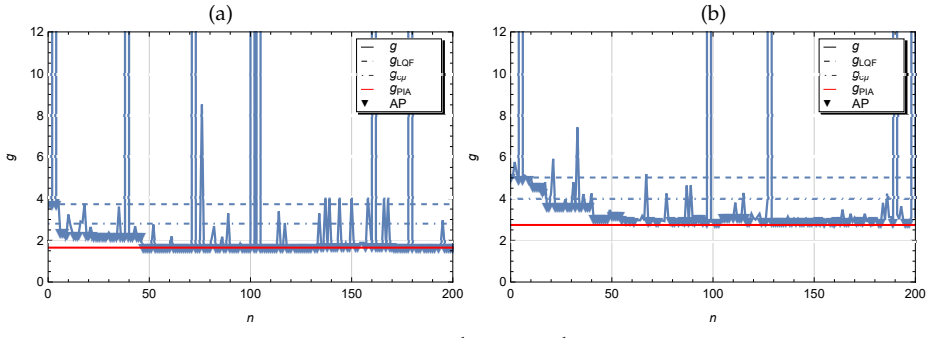

Figure 5. Iteration steps for g with $\nu_i = \frac{1}{\lambda_i}$ and $\zeta_i = \frac{1}{\mu_i}$ for Case 1 (a) and Case 2 (b)

It is observed that the optimal policy obtained by the SA algorithm is quite close to those obtained by the PIA. Nevertheless, from experiment to experiment certain deviations in the value of the average costs may appear. Therefore it is of interest for us to check whether such differences are statistically significant.

Further we analyse how sensitive is the optimal policy obtained in exponential case by the SA algorithm to the shape of arrival and service time distributions. The following distributions will be used to calculate the optimal control policy in the non-exponential case: gamma $\mathcal{G}(\alpha, \beta)$, log-normal $\mathcal{LN}(\mu, \sigma)$ and Pareto $\mathcal{PR}(\alpha, k)$ distributions , where two last options belong to a set of heavy tail distributions. The parameters of these distributions are chosen so that the first two moments coincide with the moments of the Markov counterpart. Thus, they need to be represented as functions depending on the corresponding sample moments as in a moment method for the parameter estimation. In the following experiments the first moments of the inter-arrival and service times are fixed at values of Case 2, and the squared coefficient of variation is varied as $CV^2_{\nu_i} = CV^2_{\zeta_i} = 0.5$ and $CV^2_{\nu_i} = CV^2_{\zeta_i} = 20$. Denote by $\{Z^{(k)}\}_{k=1}^m$ a sample random variable Z distributed according to the proposed distributions

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

with two first sample moments $\bar{Z}$, $\bar{Z}_2$ and squared empirical coefficient of variation $CV_Z^2 = \frac{\bar{Z}_2}{\bar{Z}} - 1$. Then for the gamma distribution $Z \sim \mathcal{G}(\alpha, \beta)$ with a PDF

$$f_Z(z) = \begin{cases} \frac{\beta(\beta z)^{\alpha-1} e^{-\beta z}}{\Gamma(\alpha)} & z \geq 0, \\ 0 & z < 0 \end{cases}$$

the parameters $\alpha > 0$ and $\beta > 0$ satisfy the relations,

$$\alpha = \frac{1}{CV_Z^2}, \; \beta = \frac{\alpha}{\bar{Z}}.$$

In case of the lognormal distribution $Z \sim \mathcal{LN}(\mu, \sigma)$ with a PDF

$$f_Z(z) = \frac{1}{\sigma z} \Phi\Big(\frac{\ln(z) - \mu}{\sigma}\Big), \; z > 0,$$

the parameters $\mu \in \mathbb{R}$ and $\sigma > 0$ are calculated by

$$\sigma = \sqrt{\ln(1 + CV_Z^2)}, \; \mu = \ln(\bar{Z}) - \frac{\sigma^2}{2}.$$

In case of a Pareto distribution $Z \sim \mathcal{PR}(k, \alpha)$ with a PDF

$$f_Z(z) = \begin{cases} \frac{\alpha k^\alpha}{x^{\alpha+1}} & x \geq k \\ 0 & x < k \end{cases}$$

the parameters $k > 0$ and $\alpha > 0$ are calculated by relations

$$\alpha = 1 + \frac{\sqrt{1 + CV_Z^2}}{CV_Z}, \; k = \frac{\alpha - 1}{\alpha} \bar{Z}.$$

The parameters of the proposed distributions are listed in Tables 5 and 6 respectively for the inter-arrival and service time distributions.

Table 5. The distribution parameters for inter-arrival times, $CV_{v_i}^2 = 0.5$ (a) and $CV_{v_i}^2 = 20$ (b)

(a)

i	1	2	3	4
$\mathcal{G}(\alpha_i, \beta_i)$	$(2.00, 0.16)$	$(2.00, 0.32)$	$(2.00, 0.48)$	$(2.00, 0.64)$
$\mathcal{LN}(m_i, \sigma_i)$	$(2.323, 0.637)$	$(1.629, 0.637)$	$(0.937, 0.637)$	$(0.637, 0.637)$
$\mathcal{PR}(k_i, \alpha_i)$	$(7.925, 2.732)$	$(3.962, 2.732)$	$(2.642, 2.732)$	$(1.981, 2.732)$

(b)

i	1	2	3	4
$\mathcal{G}(\alpha_i, \beta_i)$	$(0.05, 0.004)$	$(0.05, 0.008)$	$(0.05, 0.012)$	$(0.05, 0.016)$
$\mathcal{LN}(m_i, \sigma_i)$	$(1.003, 1.745)$	$(0.310, 1.745)$	$(-0.095, 1.745)$	$(-0.383, 1.745)$
$\mathcal{PR}(k_i, \alpha_i)$	$(6.326, 2.025)$	$(3.163, 2.025)$	$(2.109, 2.025)$	$(1.582, 2.025)$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Table 6. The distribution parameters for service times, $CV^2_{\zeta_i} = 0.5$ (a) and $CV^2_{\zeta_i} = 20$ (b)

(a)

i	1	2	3	4
$\mathcal{G}(\alpha_i, \beta_i)$	$(2.00, 7.500)$	$(2.00, 3.750)$	$(2.00, 2.500)$	$(2.00, 1.875)$
$\mathcal{LN}(m_i, \sigma_i)$	$(-1.524, 0.637)$	$(-0.831, 0.637)$	$(-0.426, 0.637)$	$(-0.138, 0.637)$
$\mathcal{PR}(k_i, \alpha_i)$	$(0.169, 2.732)$	$(0.338, 2.732)$	$(0.507, 2.732)$	$(0.676, 2.732)$

(b)

i	1	2	3	4
$\mathcal{G}(\alpha_i, \beta_i)$	$(0.05, 0.198)$	$(0.05, 0.094)$	$(0.05, 0.063)$	$(0.05, 0.047)$
$\mathcal{LN}(m_i, \sigma_i)$	$(-2.844, 1.745)$	$(-2.151, 1.745)$	$(-1.745, 1.745)$	$(-1.458, 1.745)$
$\mathcal{PR}(k_i, \alpha_i)$	$(0.135, 2.025)$	$(0.269, 2.025)$	$(0.405, 2.025)$	$(0.539, 2.025)$

The sensitivity of the optimal control policy to the shape of the distributions is tested by means of a two-sided t-test for samples with unknown but equal variances. Let g_{exp} and g_{opt} are the samples of the average cost values obtained for the optimal control policy in case of exponentially distributed times and for the system with proposed distributions for the inter-arrival and service times. These samples of size m are associated with the normally distributed random variables $Z_{\text{exp}} \sim \mathcal{N}(\mu_{g_{\text{exp}}}, \sigma_{g_{\text{exp}}})$ and $Z_{\text{opt}} \sim \mathcal{N}(\mu_{g_{\text{opt}}}, \sigma_{g_{\text{opt}}})$, where $\mu_{g_{\text{exp}}}, \mu_{g_{\text{opt}}} \in \mathbb{R}$ and $\sigma_{g_{\text{exp}}} = \sigma_{g_{\text{opt}}} > 0$. The test is defined then as

$$\mathcal{H}_0 : \mu_{g_{\text{exp}}} = \mu_{g_{\text{opt}}} \quad \mathcal{H}_1 : \mu_{g_{\text{exp}}} \neq \mu_{g_{\text{opt}}} \quad p = \mathbb{P}\left[\frac{\bar{g}_{\text{exp}} - \bar{g}_{\text{opt}}}{S_{g_{\text{opt}}, g_{\text{exp}}}} > t_{2m-2;1-\frac{\alpha}{2}}\right],$$

where statistics $S_{g_{\text{opt}}, g_{\text{exp}}}$ is calculated by (16). The results of tests in form of the p-value, the values of the average costs $\bar{g}_{\text{exp}}$ and $\bar{g}_{\text{exp}}$ together with their 95% confidence intervals are summarized in Tables 7 and 8 for the systems with different inter-arrival and service time distributions with the smaller and greater levels of dispersion around the mean, d.h. for $CV^2_{\nu_i} = CV^2_{\zeta_i} = 0.5$ in Table 7 and $CV^2_{\nu_i} = CV^2_{\zeta_i} = 20$ in Table 8.

Table 7. Comparison of optimal policies $CV^2_{\nu_i} = CV^2_{\zeta_i} = 0.5$

service \ arrival	$\mathcal{G}$	$\mathcal{LN}$	$\mathcal{PR}$
$\mathcal{G}$	3.0836 ± 0.0286 3.0491 ± 0.0576 $p = 0.2964$	3.0556 ± 0.0196 3.0203 ± 0.0301 $p = 0.0569$	3.0096 ± 0.0437 3.0083 ± 0.0726 $p = 0.9736$
$\mathcal{LN}$	3.0818 ± 0.0291 3.0445 ± 0.0233 $p = 0.0527$	3.0654 ± 0.0227 3.0282 ± 0.0364 $p = 0.0931$	3.0347 ± 0.0622 3.0351 ± 0.0877 $p = 0.9881$
$\mathcal{PR}$	3.0904 ± 0.0485 3.0168 ± 0.0701 $p = 0.0.0942$	3.1142 ± 0.0539 3.0572 ± 0.0614 $p = 0.1749$	3.3081 ± 0.4249 3.1435 ± 0.1305 $p = 0.4709$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 May 2023 doi:10.20944/preprints202305.0080.v1

Table 8. Comparison of optimal policies $CV_{\nu_i}^2 = CV_{\zeta_i}^2 = 20$

service \ arrival	$\mathcal{G}$	$\mathcal{LN}$	$\mathcal{PR}$
$\mathcal{G}$	44.5518 ± 5.3662 40.8015 ± 4.0916 $p = 0.2788$	48.3524 ± 13.0935 38.6532 ± 15.3943 $p = 0.3493$	19.1573 ± 3.7810 16.3102 ± 1.6154 $p = 0.1793$
$\mathcal{LN}$	44.0659 ± 4.6092 41.6925 ± 5.4512 $p = 0.5162$	26.7610 ± 6.0684 28.8180 ± 8.7892 $p = 0.7067$	9.6126 ± 1.9352 11.9165 ± 4.6811 $p = 0.3759$
$\mathcal{PR}$	36.3436 ± 4.0311 34.1937 ± 2.4608 $p = 0.3749$	32.9247 ± 11.1232 24.4347 ± 4.1215 $p = 0.1656$	5.6667 ± 0.7101 6.4067 ± 1.5618 $p = 0.4008$

From numerical examples it is observed that the shape of distributions has a high level of influence over the value of the average cost functions $\bar{g}_{\text{exp}}$ and $\bar{g}_{\text{opt}}$. However, an examination of the entries in last two tables reveals that in all experiments the p-value exceeds a significance level $\alpha = 0.05$. Furthermore, it is worth noting that in most cases this exceeding is significant. In this regard, the statistical test fails to reject null hypothesis at a given signigicance level, in other words, the average cost values are statistically equal and the corresponding optimal control policies are equivalent. Thus, for practical purposes in general queueing systems one can either apply the proposed optimization method, or use the control policy optimized for the exponential model as a suboptimal scheduling policy.

8. Conclusion

In this paper we combined the queue simulation technique, neural network and simulated annealing optimization to calculate the optimal scheduling policy and optimized average cost function in a general single-server queueing system with multiple parallel queues. The proposed combination of tools is sufficiently versatile to solve discrete optimization problems occurring by resource allocation in complex queueing systems and networks. Numerical results demonstrate the effectiveness of the proposed approach. The optimal solution seems to be statistically insensitive to form of the inter-arrival and service time distributions if the first two moments are the same. Therefore, the optimal policy in exponential case can be treated as a suboptimal policy and the corresponding trained neural network can be used by router in queueing systems with arbitrary distributions. The possibility to compose the reinforcement learning algorithms and neural networks to solve optimization problems in general controlled queueing models could be considered as a further line of reseach.

Acknowledgements

Supported by Johannes Kepler Open Access Publishing Fund. The reported study was funded by RSF, project number 22-49-02023 (recipient V. Vishnevsky). This paper has been supported by the RUDN University Strategic Academic Leadership Program (recipient D. Efrosinin).

References

1. Aarts, E.; Korst, J. *Simulated annealing and Boltzmann machines*. John Wiley & Sons **1989**.
2. Ahmed, M.A. A modification of the simulated annealing algorithm for discrete stochastic optimization. *Engineering Optimization* **2007** *39(6)*, 701–714.
3. Avram, F.; Gómez-Corral, A. On the optimal control of a two-queue polling model. *Operations Research Letters* **2006** *34(3)*, 339–348.
4. Buyukkoc, C.; Varaiya, P.; Walrand, I. The $c\mu$ rule revisited. *Advances in Applied Probability* **1985** *17*, 237–238.
5. Cox, D.R.; Smith, W.L. *Queues*. Chapman & Hall, London **1991**.
6. Duenyas, I.; Van Oyen, M.P. Stochastic scheduling of parallel queues with set-up costs. *Queueing Systems* **1995** *19*, 421–444.

7. Ebert, A.; Wu, P.; Mengersen, K.; Ruggeri, F. Computationally efficient simulation of queues: The *R* package queuecomputer. *Journal of Statistical Software* **2020**, *95(5)*.
8. Efrosinin, D. *Controlled queueing systems with heterogeneous servers. Dynamic optimization and monotonicity properties*. VDM Verlag, Saarbr"ucken **2008**.
9. Efrosinin, D.; Stepanova, N. Estimation of the optimal threshold policy in a queue with heterogeneous servers using a heuristic solution and artificial neural networks. *Mathematics* **2021**, 9, 1267.
10. Efrosinin, D.; Rykov, V.; Stepanova, N.Evaluation and prediction of an optimal control in a processor sharing queueing system with heterogeneous servers. *In: Vishnevsky, V.M., Samouylov, K.E., Kozyrev, D.V. (eds) Distributed Computer and Communication Networks. DCCN 2020. Lecture Notes in Computer Science* **2020**, *12563*, 450–462.
11. Franzl, G. Queueing models for multi-service networks. *PhD thesis, Technique University of Vienna* **2015**.
12. Gallo, C.; Capozzi, V. A simulated annealing algorithm for scheduling problem. *Open Journal of Applied Mathematics and Physics* **2019** *7(11)*.
13. Gorbunova, A.V.; Vishnevsky, V. Evaluation of the Performance Parameters of a Closed Queuing Network Using Artificial Neural Networks. *In: Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V. (eds) Distributed Computer and Communication Networks: Control, Computation, Communications. DCCN 2021. Lecture Notes in Computer Science* **2021**, *13144*, 265—278.
14. Gosavi, A. *Simulation-based optimization*. Springer, New-York **2015**.
15. Hofri, M.; Ross, K.W. On the optimal control of two queues with server setup times and its analysis. *SIAM Journal on Computing* **1987**, *16*, 399–420.
16. Howard, R.A. *Dynamic programming and Markov processes* **1960**.
17. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980* **2017**.
18. Kohonen, T. The self-organizing map. *Proceedings of the IEEE* **1990** *78(9)*, 1464–1480.
19. Koole, G. Assigning a single server to inhomogeneous queues with switching costs. *CWI Report BS-R9405* **1994**.
20. Kyritsis, A.I.; Deriaz, M. A machine mearning approach to waiting time prediction in queueing scenarios. *2019 Second International Conference on Artificial Intelligence for Industries (AI4I), Laguna Hills, CA, USA* **2019** 17–21.
21. Liu, Z.; Nain, P., Towsley, D. On optimal polling policies. *Queueing Systems and Their Applications* **1992** *11*, 59–83.
22. Matsumoto, Y. On optimization of polling policy represented by neural network. *Computer Communication Review* **1994** *4*, 181–190.
23. Nii, S.; Okuda, T; Wakita, T. A performance evaluation of queueing systems by machine learning. *IEEE International Conference on Consumer Electronics (ICCE-Taiwan)* **2020**.
24. Özkan, E.; Kharoufeh, J. (2014). Optimal control of a two-server queueing system with failures. *Probability in the Engineering and Informational Sciences* **2014** *28(4)*, 489–527.
25. Puterman, M.L. *Markov decision process*. Wiley series in Probability and Mathematcal Statistics **1994**.
26. Sherzer, E.; Senderovich, A.; Baron, O.; Krass, D. Can machines solve general queueing systems? *arXiv preprint arXiv:2202.01729* **2022**.
27. Sivakami, S.M.; Senthil, K.K.; Yamini, S.; Palaniammal, S. Artificial neural network simulation for Markovian queueing models. *Indian Journal of Computer Science and Engineering* **2020** *11(2)*, 127–134.
28. Sennott, L.I. Average cost optimal stationary policies in infinite state Markov decision processes with unbounded costs. *Operations Research* **1989** *37*, 626-633.
29. Stintzing, J.; Norrman, F. Prediction of Queuing Behaviour through the Use of Artificial Neural Networks. Available online: http://www.diva-portal.se/smash/get/diva2:1111289/FULLTEXT01.pdf **2017**.
30. Tijms, H.C. *Stochastic models. An algorithmic approach*. John Wiley & Sons **1994**.
31. Vishnevsky, V.; Klimenok, V.; Sokolov, A.; Larionov, A. Performance evaluation of the priority multi-server system $MMAP/PH/M/N$ using machine learning methods. *Mathematics* **2021**, *9*, 3236.
32. Vishnevsky, V.; Gorbunova, A.V. Application of machine learning methods to solving problems of queuing theory. *In: Dudin, A., Nazarov, A., Moiseev, A. (eds) Information Technologies and Mathematical Modelling. Queueing Theory and Applications. ITMM 2021. Communications in Computer and Information Science* **2022**, *1605*, 304–316.

33. Vishnevsky, V.; Semenova O. *Polling systems theory and applications for broadband wireless networks*. LAP LAMBERT Academic Publishing GmbH **2012**.
34. Vishnevsky, V.; Semenova O. Polling systems and their application to telecommunication networks. *Mathematics* **2021**, *9*, 117.
35. Vishnevsky, V.; Semenova, O.; Bui, D.T. Using a machine learning approach for analysis of polling systems with correlated arrivals. In: Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V. (eds) *Distributed Computer and Communication Networks: Control, Computation, Communications. DCCN 2021. Lecture Notes in Computer Science* **2021** *13144*, 336–-345.