Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

UNITING SIMULATION AND MACHINE LEARNING FOR RESPONSE TIME PREDICTION IN PROCESSOR SHARING QUEUES

Jamol Pender
Elena Zhang

Operations Research and Information Engineering
Cornell University
Ithaca, NY 14853, USA

ABSTRACT

Processor sharing queues are used in a variety of settings in telecommunications and internet applications and are known for being fair. In this paper, we study the possibility of accurately predicting the response times in real time of the G/G/1 processor sharing queue. To this end, we combine stochastic simulation with supervised learning machine learning methods. We show that many of the current machine learning algorithms can perform well at predicting response times in real-time.

1 INTRODUCTION

In this paper, we analyze the single-server queue operating under the processor sharing (PS) service discipline. In this G/G/1/PS queue, jobs arrive to the queue according to a general arrival process. Moreover, each such job has associated with it a processing time, which is a random variable that represents the amount of time that the server must spend working on this job to complete its service if it were the only job in the queue. In the processor sharing queue, jobs never wait and are immediately served when they enter the queue. However, in a processor sharing queue, the processor is split among all of the jobs in a even manner i.e. if there are n jobs, then each job receives $1/n$ of the processor's effort. As a result, in the processor sharing queue, it is possible that future arrivals can affect the response time of jobs that have already arrive since each job experiences additional slowdown when new jobs arrive. Unlike the first in first out (FIFO) G/G/1 queue, even if one were given all information about all previous arrivals, one cannot perfectly predict the response time of a job. Thus, one of the most important quantities of interest is to predict the response time of a job. This is the main focus of this paper.

The processor sharing queue is well studied since its inception in Kleinrock (1967), Kleinrock (1964), Kleinrock (1970), Kleinrock et al. (1971) and Kleinrock and Muntz (1972) as the limit of round robin queueing systems. It is well known that the unconditional steady-state response time distribution is approximately exponential. This implies that prediction is almost hopeless in this setting. It is also well known in Whitt (1999b) that the unconditional steady-state response-time in the M/G/1/PS queue is asymptotically exponential in the heavy traffic regime. However, when limited information is available, it is an important question to know whether response times can be reliably predicted from basic queueing information. Despite its importance there is only one paper by Ward and Whitt (2000) that attempts to predict response times in the single server processor sharing queue and quite a few papers in the FCFS setting i.e. Whitt (1999b), Whitt (1999a), Ibrahim and Whitt (2009), Ibrahim and Whitt (2011) and Shah et al. (2019). Unlike the G/G/1/FIFO queue, PS response times are not necessarily known at the time of arrival. This is simply because the response time depends on the number of jobs that arrive after a job. This feature of PS queues is precisely what makes it interesting to study from a prediction point of view and is one of the main goals of this work. Here in this work, we provide a new framework for response time

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

prediction in single server queues with the PS queueing discipline using simulation as a data generator and machine learning as a predictive tool. By combining these two areas, we are able to develop new ways of predicting response times in PS queues and give incoming jobs accurate estimates of departure.

One important feature of our work is that our data driven approach does not assume anything about the dependence of the arrival and service distributions. Much of the theoretical work assumes Poisson arrivals, however, recent literature shows that arrivals may not be Poisson, see for example Sriram and Whitt (1986), Daw and Pender (2018c), Daw and Pender (2018b), Koops et al. (2018), Daw and Pender (2018a), Saha et al. (2019), Daw et al. (2020).

This paper is valuable in many ways. First we introduce the idea of simulation as a data generation mechanism for machine learning. This is similar to reinforcement learning, but is still quite different since our model is not restricted to be in the class of Markov decision processes. Second, we show that machine learning can be quite helpful at predicting response times. This is relevant in that we can use these predictions in data centers as a way of partitioning traffic from one server to another.

1.1 Contributions of Our Work

By using stochastic simulation for the G/G/1/PS queue and leveraging machine learning methods, we provide new insights to the following questions:

- What machine learning methods work well at predicting response times for the G/G/1/PS queue?
- How valuable is additional information when predicting response times of the G/G/1/PS queue?

1.2 Organization of the Paper

The remainder of the paper is organized as follows. Section 2 introduces the processor sharing queue and provides a brief history of its construction. In Section 3, we present our simulation and machine learning results for various methods. We also explain how various parameters of the queueing model affect the performance of predicting the response times of processor sharing queues. Finally, a conclusion is given in Section 4.

2 THE PROCESSOR SHARING QUEUE

To the authors knowledge, there are not many papers that discuss response time prediction for processor sharing queues in real time. Most research explores the use of fluid and diffusion limits or Laplace transforms to construct an approximate expected response time, see for example Hampshire et al. (2006), Whitt (1999b), Zhang et al. (2011), Ramanan et al. (2003). Moreover, the approximation might only depend on the number in the system and the job size and not other forms of information. One reason is that it is very difficult if not impossible to construct analytical estimates for the conditional mean response time given different forms of information other than the queue length or job size. To this end, we leverage machine learning to understand how well current machine learning methods can use this additional information in the context of response time prediction. Hence, we develop the following approach for generating response time predictions for the G/G/1/PS queue:

1. Construct a discrete-event simulation of the G/G/1/PS queue.
2. The discrete-event simulation is then used to record for each job, the job size, the queue length upon arrival, the response time, and the amount of work in the system upon arrival.
3. Partition the data generated from the discrete-event simulation into training and test sets.
4. Train various machine learning methods on the training sets and develop predictive models.
5. Use the predictive models generated from the training sets and make predictions on the test sets.
6. Evaluate via some metric like average mean square error which machine learning methods perform well on the test data.

2.1 How the Data is Collected and Analyzed

This paper leverages simulation and machine learning together for prediction purposes. In this regard, we view the simulation part as essential because it is by which we generate the training data used by the machine learning methods. Thus, the simulation step should be viewed as a data collection step. Unlike most data collection situations, via simulation, we can collect whatever data we want as long as it can be simulated. Before we go into more detail describing the various machine learning methods that we use to predict the response times, we will describe how we generate the training data via simulation. Moreover, we outline how we generate the numerical results that follow.

The experiments that follow were all trained on $10,000$ simulated waiting time observations and are implemented or tested on 10 independent sets for all methods and averaged. All numbers that are reported are based on the sample mean squared error for the 10,000 samples i.e.

$$\textit{Average MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(R_i - \hat{R}_i\right)^2.$$

where

$$\hat{R}_i = f\left(X_i^{(1)}, X_i^{(2)}, X_i^{(3)}\right)$$

for some function $f(\cdot)$ and

- $X_i^{(1)}$ = size of the i^{th} job
- $X_i^{(2)}$ = amount of work in system at time of arrival of the i^{th} job
- $X_i^{(3)}$ = queue length at time of arrival of the i^{th} job
- R_i = the response time of the i^{th} job.

Thus, we aim to predict the response time, R_i, of the i^{th} job by using information such as the job size $X_i^{(1)}$, the amount of work in the system at the time of arrival of the i^{th} job $X_i^{(2)}$, and the queue length at the time of arrival of the i^{th} job $X_i^{(3)}$. We will consider both Markovian and non-Markovian queueing models and we will use the parameter $\rho = \frac{\lambda}{\mu}$ throughout the rest of the paper. Finally all numbers reported in tables are AMSE for the test sets and not the training sets since for all of our examples, the test sets performed worst than the training sets.

2.2 Arrival and Service Distributions

In this paper, we simulated the processor sharing queues using exponential and non-exponential distributions. For the exponential case, we assumed that the inter-arrival times occurred as exponential random variables with rate parameter λ and the job sizes were exponential with rate parameter μ. For the non-exponential case, we considered log-normal random variables for the inter-arrival times and job sizes given by $\text{LN}(\mu_a, \sigma_a)$ and $\text{LN}(\mu_s, \sigma_s)$ for the inter-arrivals and the job sizes respectively. For consistency of results, we ensured that the mean and variance of the log-normal random variables were the same as the exponential settings. This makes it easier for comparison purposes. Finally, all calculations were done using Python's sci-kit learn packages.

3 MACHINE LEARNING METHODS FOR THE PROCESSOR SHARING QUEUE

3.1 Linear Regression

In this section, we aim to understand how well linear regression will perform in predicting response times for the processor sharing queue. In the linear regression framework our goal is to find the coefficients

$\beta_0, \beta_1, \beta_2, \beta_3$ in order to minimize the following loss function

$$\min_{\beta_0,\beta_1,\beta_2,\beta_3} \sum_{i=1}^{n} \left(R_i - \beta_0 - \beta_1 X_i^{(1)} - \beta_2 X_i^{(2)} - \beta_3 X_i^{(3)} \right)^2$$

in terms of the independent variables $\left(X_i^{(1)}, X_i^{(2)}, X_i^{(3)} \right)$. Linear regression is an attractive prediction method because it is quite simple and gives a natural interpretation of just weighting the available information upon arrival to construct a prediction. In fact, if one just receives the queue length as the information i.e $\beta_0 = \beta_1 = \beta_2 = 0$, then the linear regression can be interpreted as a statistical version of Little's law as it relates the queue length to the response time in a linear relationship.

In Tables 1 - 4, we observe that the performance of linear regression improves as we scale the arrival rate and service rate (large parameter setting). Moreover, we also observe that the performance degrades as we let λ approach μ i.e. we are in heavy traffic. Surprisingly, we also observe that the coefficients for the linear regression are quite similar for specific offered load values. One should also note that the performance is quite similar in regardless if we use a Markovian model where the inter-arrival and service times are exponentially distributed or a non-Markovian model. In all of the simulation experiments, we observe that the intercept i.e. β_0 is always negative. This suggests that the other coefficients are overestimating the response time. However, as we scale up the arrival rate and service rate parameters, the magnitude of the shift decreases significantly and approaches zero. We also observe that the coefficient β_1 is proportional to the offered load parameter λ/μ since we know that the response time is of the order $x/(1-\rho)$, see for example Ward and Whitt (2000). The coefficient β_2, which measures the impact of the work in the system upon arrival, is also positive. It gets smaller as the arrival rate and service rate get large and it also gets slightly larger as λ approaches μ. Finally, we observe the parameters β_3 is pretty consistent in all parameter regimes.

We can also connect the linear regression to the conditional mean response time with job size x and n customers at arrival given in Asare and Foster (1983) by the following formula

$$R_n(x) = \frac{x}{1-\rho} + \frac{n}{\mu-\lambda} \cdot \left(1 - e^{-(\mu-\lambda)x}\right) - \frac{\lambda}{(\mu-\lambda)^2} \cdot \left(1 - e^{-(\mu-\lambda)x}\right).$$

Thus, the signs and magnitudes of the coefficients are well approximated by this formula in the M/M/1/PS model. We find that the AMSE for the conditional mean response time is decent, however, does not do better than the linear regression in most settings. We observe that it also does not do better than most of the machine learning methods. This is possible as additional information is added to the prediction for the other methods. We also observe that conditional mean response time is about as accurate for the M/M/1 queue and the G/G/1 queue as well.

Table 1: Linear Regression Response Time Prediction M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Linear (AMSE)	1.56	8.37	8.21e+1	2.92e+3	1.85e-2	1.06e-1	1.02	2.20e+1
β_0	-1.01	-2.96	-8.03	-4.54e+1	-4.54e+1	-2.79e-1	-9.29e-1	-5.19
β_1	2.01	4.18	9.52	4.78e+1	1.97	3.98	$1.08e_1$	5.48e+1
β_2	3.36e-1	4.08e-1	4.55e-1	4.97e-1	3.37e-2	4.13e-2	4.40e-2	5.06e-2
β_3	3.30e-1	3.92e-1	4.28e-1	4.71e-1	3.40e-1	3.81e-1	4.62e-1	4.47e-1
$R_n(x)$ AMSE	1.94	6.37	2.84e+1	2.16e+2	2.14e-2	7.18e-2	3.41e-1	2.26

Table 2: Linear Regression Response Time Prediction M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
Linear (AMSE)	1.40e-4	9.92e-4	1.24e-2	4.73e-1	2.00e-6	9.00e-6	9.40e-5	1.20e-3
β_0	-9.84e-3	-2.81e-2	-9.52e-2	-6.41e-1	-1.01e-3	-2.72e-3	-8.90e-3	-4.15e-2
β_1	1.98	4.08	1.10e+1	6.70e+1	2.01	3.95	1.05e+1	4.48e+1
β_2	3.43e-3	3.89e-3	4.31e-3	4.65e-3	3.38e-4	4.09e-4	4.42e-4	4.66e-4
β_3	3.07e-1	3.95e-1	4.77e-1	5.08e-1	3.20e-1	3.72e-1	4.43e-1	4.75e-1
$R_n(x)$ AMSE	3.53e-1	4.02e-1	4.50e-1	4.83e-1	3.36e-1	3.79e-1	4.19e-1	4.68e-1

Table 3: Linear Regression Response Time Prediction G/G/1 Queue.

$1/\mu_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\sigma_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\mu_s$	1	1	1	1	10	10	10	10
$1/\sigma_s$	1	1	1	1	10	10	10	10
Linear (AMSE)	1.00	6.84	7.21e+1	1.82e+3	9.78e-3	1.05e-1	5.44e-1	4.61e+1
β_0	-1.13	-2.86	-7.29	-5.93e+1	-1.09e-1	-2.86e-1	-6.90e-1	-6.44
β_1	1.96	3.86	8.56	6.20e+1	1.92	3.88	8.32	6.60e+1
β_2	4.41e-1	5.50e-1	5.86e-1	6.39e-1	4.29e-2	5.22e-2	5.49e-2	6.17e-2
β_3	2.22e-1	2.40e-1	2.91e-1	3.24e-1	2.35e-1	2.72e-1	3.17e-1	3.44e-1
$R_n(x)$ AMSE	1.58	5.78	22.88e+1	3.68e+2	1.55e-2	7.68e-2	2.76e-1	3.42

Table 4: Linear Regression Response Time Prediction G/G/1 Queue.

$1/\mu_a$	50	75	90	99	500	750	900	990
$1/\sigma_a$	50	75	90	99	500	750	900	990
$1/\mu_s$	100	100	100	100	1000	1000	1000	1000
$1/\sigma_s$	100	100	100	100	1000	1000	1000	1000
Linear (AMSE)	1.15e-4	9.37e-4	1.45e-2	3.83e-1	8.46e-7	1.80e-5	5.80e-5	1.61e-2
β_0	-1.11e-2	-2.78e-2	-8.19e-2	-5.58e-1	-1.11e-3	7.00e-6	-7.96e-3	-4.97e-2
β_1	1.90	3.82	9.52	5.91e+1	1.94	3.76	9.34	5.13e+1
β_2	4.61e-3	5.10e-3	5.95e-3	6.41e-3	4.29e-4	5.61e-4	5.82e-4	5.70e-4
β_3	2.13e-1	2.81e-1	2.88e-1	3.13e-1	2.41e-1	2.71e-1	3.29e-1	4.11e-1
$R_n(x)$ AMSE	1.69e-4	7.11e-4	4.85e-3	3.20e-2	1.51e-4	6.00e-6	3.20e-5	1.04e-3

3.2 Gaussian Processes

In this section, we describe the performance of Gaussian processes on predicting the response times of the processor sharing queue. In Tables 5 - 8, we observe that the performance of Gaussian processes improves as we scale the arrival rate and service rate (large parameter setting). Moreover, we also observe that the performance degrades as we let λ approach μ i.e. we are in heavy traffic. The performance under a Markovian setting or a non-Markovian setting is roughly the same. We note that the Gaussian processes do not do as well as linear regression when the arrival rate and service rates are small, however, Gaussian processes work well when the rates are large. We compare three different kernels (squared exponential,

Matern 5/2, rational quadratic) and observe that the rational quadratic and squared exponential seemed to work uniformly better than the Matern kernel.

Table 5: Gaussian Processes Prediction M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
Squared Exponential	3.98e+4	9.45e+6	5.50e+7	1.45e+8	7.84e-2	6.38e-1	8.49e+1	3.01e+2
Matern 52	4.33e+4	6.17e+3	1.12e+5	2.67e+7	1.50e+1	5.36e+1	4.10e+2	6.76e+3
Rational Quadratic	4.05e+1	1.31e+2	1.66e+2	2.16e+3	2.77e-1	1.34	7.34	4.09e+1

Table 6: Gaussian Processes Prediction M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	9900
Squared Exponential	1.50e-4	9.53e-4	8.98e-3	5.37e-1	1.84e-6	6.00e-6	4.40e-5	8.68e-4
Matern 52	7.96e-4	5.42e-3	2.37e-2	8.51e-1	1.19e-4	4.08-4	5.10e-5	3.93e-3
Rational Quadratic	8.56e-4	3.99e-2	9.48e-2	1.48	7.32e-6	5.20e-5	3.72e-4	3.82e-3

Table 7: Gaussian Processes Prediction G/G/1 Queue.

$1/\mu_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\sigma_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\mu_s$	1	1	1	1	10	10	10	10
$1/\sigma_s$	1	1	1	1	10	10	10	10
Squared Exponential	1.55e+6	9.87e+5	1.78e+7	2.53e+8	1.93e-2	5.29	2.31e+2	3.62e+4
Matern 52	9.29e+2	4.76e+3	3.04e+5	4.93e+5	1.01e+1	3.54e+1	7.53e+2	2.09e+2
Rational Quadratic	2.16e+1	9.62e+1	3.07e+2	7.05e+3	2.02e-1	3.72	3.37	1.82e+1

Table 8: Gaussian Processes Prediction G/G/1 Queue.

$1/\mu_a$	50	75	90	99	500	750	900	990
$1/\sigma_a$	50	75	90	99	500	750	900	990
$1/\mu_s$	100	100	100	100	1000	1000	1000	1000
$1/\sigma_s$	100	100	100	100	1000	1000	1000	1000
Squared Exponential	1.14e-4	1.22e-3	9.26e-3	1.32	1.29e-6	2.20e-5	5.14e-4	1.84e-3
Matern 52	6.33e-2	3.70e-2	2.88	4.62	8.94e-6	2.10e-5	2.14e-3	1.01e-1
Rational Quadratic	7.46e-4	1.11e-2	7.55e-2	1.75	2.17e-6	8.60e-5	2.71e-3	5.53e-3

3.3 Deep Neural Networks

In this section, we describe the performance of deep neural networks on predicting the response times of the processor sharing queue. In Tables 9 - 12, we observe that the performance of deep neural networks improves as we scale the arrival rate and service rate (large parameter setting). We also observe that the performance degrades as we let λ approach μ, which implies that we are in the heavy traffic regime. The performance under a Markovian setting or a non-Markovian setting is roughly the same, however, there

are some settings where DNN does better for the G/G/1 and vice versa. In contrast to Gaussian processes, deep neural networks appear to work as well as linear regression when the arrival rate and service rates are small, however, deep neural networks outperform the linear regression when the rates are large and the performance is comparable to Gaussian processes in the large parameter regime. We compare three different activation functions (ReLU, Sigmoid, Tanh) and observe that ReLu slightly outperforms Sigmoid and Tanh uniformly. Moreover, we compare different numbers of layers and observe that with both 100 and 400 layers, the performance is very similar.

Table 9: Deep Neural Network Prediction M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	0.5	0.75	0.9	.99	5	7.5	9	9.9
ReLU (100 layers)	1.14	1.09	3.19	207.02	1.57e-2	4.48e-2	.55	4.57
ReLU (400 layers)	1.14	1.12	3.13	209.45	1.55e-2	4.50e-2	.56	4.39
Tanh (100 layers)	1.14	1.17	3.24	338.77	1.93e-2	4.81e-2	.61	4.93
Tanh (400 layers)	1.15	1.18	3.32	291.34	2.12e-2	4.84e-2	.66	5.37
Sigmoid (100 layers)	1.14	1.20	3.46	351.25	2.05e-2	6.68e-2	1.12	6.96
Sigmoid (400 layers)	1.31	1.20	3.43	301.01	2.10e-2	6.89e-2	1.38	1.07

Table 10: Deep Neural Network Prediction M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
ReLU (100 layers)	1.91e-4	1.29e-3	9.10e-3	1.39e-1	6.45e-6	2.60e-5	1.95e-4	4.84e-3
ReLU (400 layers)	1.46e-2	9.98e-4	7.34e-3	1.23e-1	5.96e-6	2.00e-5	1.91e-4	4.84e-3
Tanh (100 layers)	1.86e-4	1.51e-3	9.87e-3	1.22e-1	1.13e-5	2.90e-5	2.22e-4	4.96e-3
Tanh (400 layers)	1.62e-4	1.33e-3	7.92e-3	1.47e-1	6.36e-6	1.90e-5	1.86e-4	4.94e-3
Sigmoid (100 layers)	3.77e-4	2.18e-3	1.83e-2	3.17e-1	6.80e-6	2.70e-5	2.06e-4	4.83e-3
Sigmoid (400 layers)	2.84e-4	1.78e-3	1.52e-2	2.53e-1	6.32e-6	2.60e-5	1.95e-4	4.93e-3

Table 11: Deep Neural Network Prediction G/G/1 Queue.

$1/\mu_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\sigma_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\mu_s$	1	1	1	1	10	10	10	10
$1/\sigma_s$	1	1	1	1	10	10	10	10
ReLU (100 layers)	7.58e-1	4.02	9.12	4.32e+2	8.20e-3	5.14e-2	2.37e-1	7.86e-1
ReLU (400 layers)	7.56e-1	4.05	8.93	4.57e+2	8.24e-3	5.30e-2	2.40e-1	7.90e-1
Tanh (100 layers)	7.91e-1	4.24	134.97	7.27e+2	9.80e-3	5.39e-2	2.50e-1	8.38e-1
Tanh (400 layers)	8.10e-1	4.23	1.20.52	5.97e+2	9.57e-3	5.34e-2	2.53e-1	8.31e-1
Sigmoid (100 layers)	8.40e-1	4.61	115.16	1.04e+3	9.24e-3	8.49e-2	3.70e-1	9.05e-1
Sigmoid (400 layers)	9.19e-1	4.75	109.07	8.32e+2	9.37e-3	8.45e-2	5.26e-1	9.30e-1

3.4 K-Nearest Neighbors

In this section, we describe the performance of K-nearest neighbors on predicting the response times of the processor sharing queue. In Tables 13 - 16, we observe that the performance of K-nearest neighbors

Table 12: Deep Neural Network Prediction G/G/1 Queue.

$1/\mu_a$	50	75	90	99	500	750	900	990
$1/\sigma_a$	50	75	90	99	500	750	900	990
$1/\mu_s$	100	100	100	100	1000	1000	1000	1000
$1/\sigma_s$	100	100	100	100	1000	1000	1000	1000
ReLU (100 layers)	1.52e-4	1.68e-3	5.41e-3	3.22e-1	4.53e-5	4.10e-5	5.45e-4	6.26e-3
ReLU (400 layers)	1.25e-4	1.09e-3	4.61e-3	3.12e-1	3.69e-6	3.20e-5	5.11e-4	6.27e-3
Tanh (100 layers)	1.57e-4	1.92e-3	8.23e-3	3.37e-1	1.07e-5	5.50e-5	6.10e-4	6.41e-3
Tanh (400 layers)	1.44e-4	1.53e-3	6.74e-3	3.45e-1	3.90e-6	9.90e-5	6.13e-4	6.56e-3
Sigmoid (100 layers)	3.32e-4	2.62e-3	1.42e-2	6.36e-1	4.72e-6	3.70e-5	4.85e-4	6.34e-3
Sigmoid (400 layers)	2.70e-4	2.17e-3	7.41e-3	7.38e-1	4.71e-6	4.00e-5	4.77e-4	6.75e-3

improves as we scale the arrival rate and service rate (large parameter setting). We also observe that the performance degrades as we let λ approach μ, which implies that we are in the heavy traffic regime. Unlike the other methods, we observe that K-NN works slightly better under the non-Markovian setting when compared to the M/M/1 model. In contrast to the other methods, K-NN also appears to work well in the small parameter regime setting. This is mostly because, K-NN has a conditional expectation interpretation and we are finding the closest representatives to our test points. K-NN also outperforms the linear regression when the rates are large and the performance is comparable to Gaussian processes and deep neural networks in the large parameter regime. We compare three different values of K (5,10,25) and observe that the performance is similar, but $K = 5$ slightly outperforms the other values.

Table 13: K-Nearest Neighbor Prediction M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
k = 5	4.96e-3	3.12e-2	6.66e-2	2.94	2.38e-4	8.18e-4	1.80e-2	4.72e-1
k = 10	9.48e-3	1.06e-1	7.67e-2	5.06	5.24e-4	2.07e-3	3.54e-2	8.30e-1
k = 25	2.63e-2	4.20e-1	3.47e-1	2.03e+1	1.50e-3	5.46e-3	8.92e-2	1.88

Table 14: K-Nearest Neighbor Prediction M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
k = 5	3.00e-6	4.70e-5	1.16e-3	2.44e-2	1.25e-7	4.96e-7	8.00e-6	7.64e-4
k = 10	8.00e-6	1.34e-4	2.49e-3	5.25e-2	2.70e-7	1.36e-6	2.00e-5	1.87e-3
k = 25	2.3e-5	3.60e-4	5.57e-3	1.26e-1	6.11e-7	3.87e-6	4.20e-5	4.45e-3

3.5 Gradient Boosted Trees Regression

In this section, we describe the performance of boosted tree regression on predicting the response times of the processor sharing queue. In Tables 17 - 20, we observe that the performance of boosted tree regression improves as we scale the arrival rate and service rate (large parameter setting). We also observe that the performance degrades as we let λ approach μ, which implies that we are in the heavy traffic regime. Similar to K-NN, we find that boosted tree regression works slightly better under the non-Markovian setting when compared to the M/M/1 model. Boosted tree regression has very similar performance to linear regression when the rates are large and the performance is worse than Gaussian processes and deep neural networks in

Table 15: K-Nearest Neighbor Response Time Prediction G/G/1 Queue.

$1/\mu_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\sigma_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\mu_s$	1	1	1	1	10	10	10	10
$1/\sigma_s$	1	1	1	1	10	10	10	10
k = 5	8.03e-2	1.68e-2	5.72e-2	3.43e+2	1.84e-4	1.59e-3	5.07e-2	1.73e-1
k = 10	1.35e-1	3.76e-2	1.32e-1	7.59e+2	4.71e-4	3.71e-3	9.04e-2	3.76e-1
k = 25	2.35e-1	1.39e-1	7.07e-1	1.47e+3	1.37e-3	1.05e-2	1.87e-1	1.10

Table 16: K-Nearest Neighbor Response Time Prediction G/G/1 Queue.

$1/\mu_a$	50	75	90	99	500	750	900	990
$1/\sigma_a$	50	75	90	99	500	750	900	990
$1/\mu_s$	100	100	100	100	1000	1000	1000	1000
$1/\sigma_s$	100	100	100	100	1000	1000	1000	1000
k = 5	1.70e-5	6.70e-5	2.08e-3	1.73e-2	1.03e-7	1.00e-6	1.90e-5	9.88e-5
k = 10	3.50e-5	1.66e-4	4.29e-3	3.96e-2	2.20e-7	2.00e-6	3.40e-5	1.86e-3
k = 25	6.00e-5	3.74e-4	7.83e-3	1.01e-1	5.27e-7	5.00e-6	6.3e-5	3.11e-3

the large parameter regime. We compare two different values of the number of estimators (100,1000) and observe that the performance is similar, but 100 estimators seems to slightly outperform 1000 estimators.

Table 17: Boosted Trees Regression Prediction M/M/1 Queue.

μ	1	1	1	1	10	10	10	10
λ	.5	.75	.9	.99	5	7.5	9	9.9
# of Estimators = 100	1.39	9.46	2.60e+1	5.89e+2	1.31e-2	5.97e-2	4.26e-1	6.53
# of Estimators = 1000	1.80	1.12e+1	3.15e+1	6.21e+2	1.57e-2	7.73e-2	4.97e-1	6.88

Table 18: Boosted Trees Prediction M/M/1 Queue.

μ	100	100	100	100	1000	1000	1000	1000
λ	50	75	90	99	500	750	900	990
# of Estimators = 100	1.35e-4	6.88e-4	3.66e-3	5.80e-2	1.69e-6	6.50e-6	2.50e-5	4.07e-4
# of Estimators = 1000	1.67e-4	8.54e-4	4.12e-3	5.86e-2	2.00e-6	8.22e-6	2.90e-5	4.42e-4

Table 19: Gradient Boosted Trees Response Time Prediction G/G/1 Queue.

$1/\mu_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\sigma_a$	0.5	0.75	0.9	.99	5	7.5	9	9.9
$1/\mu_s$	1	1	1	1	10	10	10	10
$1/\sigma_s$	1	1	1	1	10	10	10	10
# of Estimators = 100	1.23	5.26	2.61e+1	2.09e+3	7.00e-3	5.27e-2	4.92e-1	4.00
# of Estimators = 1000	1.45	6.43	3.20e+1	2.14e+3	9.38e-3	6.61e-2	5.63e-1	4.54

Table 20: Gradient Boosted Trees Response Time Prediction G/G/1 Queue.

$1/\mu_a$	50	75	90	99	500	750	900	990
$1/\sigma_a$	50	75	90	99	500	750	900	990
$1/\mu_s$	100	100	100	100	1000	1000	1000	1000
$1/\sigma_s$	100	100	100	100	1000	1000	1000	1000
# of Estimators = 100	1.33e-4	5.90e-4	5.87e-3	5.05e-2	1.13e-6	7.00e-6	3.60e-5	8.77e-4
# of Estimators = 1000	1.61e-4	7.39e-4	6.03e-3	5.21e-2	1.43e-6	8.00e-6	4.20e-5	9.04e-4

3.6 Summary of Numerical Results

So far, we have shown that machine learning can do well at predicting the response times of the processor sharing queue. In summary, we observe that K-Nearest Neighbors works very well at predicting the response times in both the small parameter and large parameter regimes. We also see that the linear regression prediction is picking up to some extent, the conditional mean response time given the job size and queue length information, which is quite remarkable. We notice some variation from the exact formula, but that could be caused by the randomness of the data. Thus, in all it seems that one can use most machine learning methods with confidence, however, we recommend K-NN as it is fast and is quite accurate in many parameter settings.

One important observation we made was that sometimes the AMSE can be misleading since the large values can be caused by one or two large mistakes in prediction. In an effort to understand this, we plot the errors of the response times and the predictions via several methods in a histogram for the M/M/1 and G/G/1 settings in Figures 1 and 2 respectively. We find in our observations that the large mistake rule does seem to skew the AMSE values for most of the methods except linear regression, K-NN, and the conditional expectation estimator. Perhaps other performance measures can be used to capture this effect as it helps us understand the true performance of the method on most of the points.

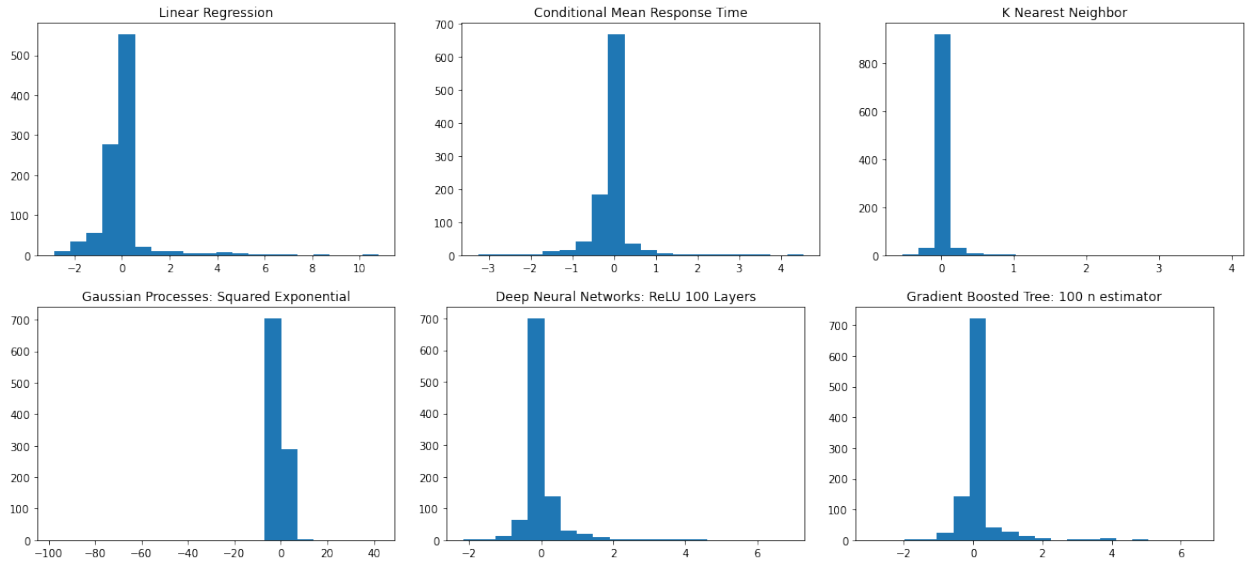

Figure 1: M/M/1 Queue

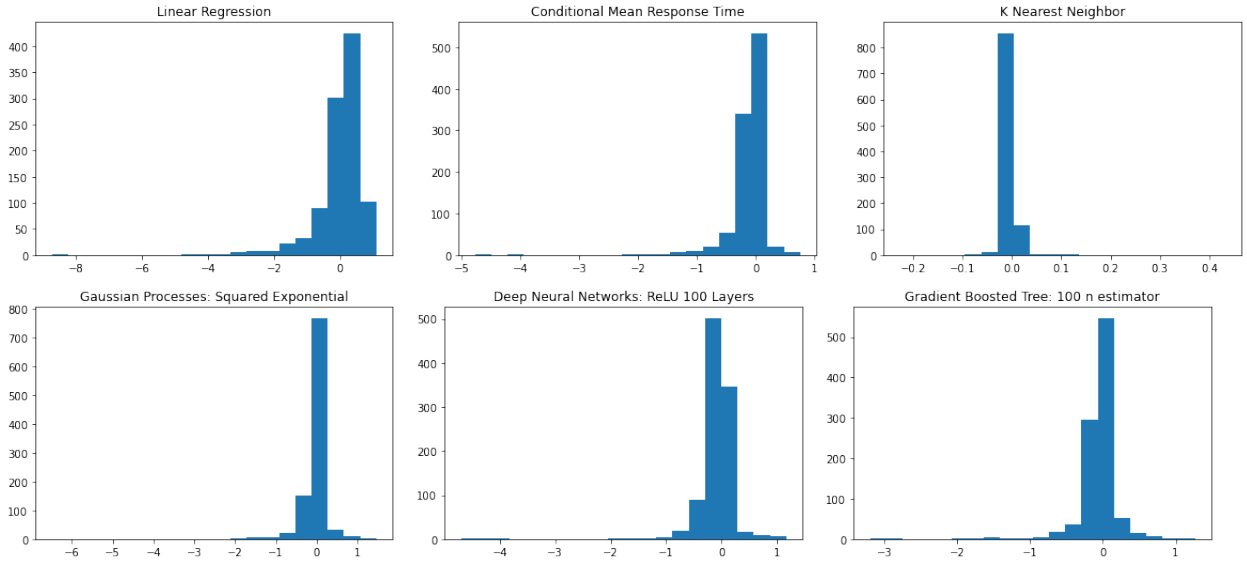

Figure 2: G/G/1 Queue

4 CONCLUSION & FUTURE DIRECTIONS

In this work, we propose combining simulation and machine learning for response time prediction for processor sharing queues. We find that K-Nearest Neighbors is a robust method that performs well for small and large parameter settings and also performs well in light and heavy traffic. However, all of the machine learning methods work reasonably well on predicting the response times.

One major future direction would be to extend this analysis to networks of processor sharing queues since it is clear in many applications that one queue is not enough, see for example Wang et al. (2021) and Pender and Phung-Duc (2016). In the network setting, it would be great to predict response times of not only an individual queue, but the entire network, if one knows the path.

Another area for future work is the area of processor sharing queues with time varying rates. There is one paper by Hampshire et al. (2006) that analyzes the processor sharing queue in the non-stationary setting. It would be great to compare the results from the Hampshire et al. (2006) to various machine learning methods that can incorporate time varying behavior.

REFERENCES

Asare, B., and F. Foster. 1983. "Conditional Response Times in the M/G/1 Processor-Sharing System". *Journal of Applied Probability*:910–915.

Daw, A., A. Castellanos, G. B. Yom-Tov, J. Pender, and L. Gruendlinger. 2020. "The Co-Production of Service: Modeling Service Times in Contact Centers Using Hawkes Processes". *arXiv preprint arXiv:2004.07861*.

Daw, A., and J. Pender. 2018a. "An Ephemerally Self-Exciting Point Process". *arXiv preprint arXiv:1811.04282*.

Daw, A., and J. Pender. 2018b. "Exact Simulation of the Queue-Hawkes Process". In *Proceedings of the 2018 Winter Simulation Conference*, edited by N. M. A. S. S. J. M. Rabe, A. A. Juan and B. Johansson, 4234–4235. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Daw, A., and J. Pender. 2018c. "Queues Driven by Hawkes Processes". *Stochastic Systems* 8(3):192–229.

Hampshire, R. C., M. Harchol-Balter, and W. A. Massey. 2006. "Fluid and Diffusion Limits for Transient Sojourn Times of Processor Sharing Queues with Time Varying Rates". *Queueing Systems* 53(1):19–30.

Ibrahim, R., and W. Whitt. 2009. "Real-time Delay Estimation Based on Delay History". *Manufacturing & Service Operations Management* 11(3):397–415.

Ibrahim, R., and W. Whitt. 2011. "Wait-time Predictors for Customer Service Systems with Time-Varying Demand and Capacity". *Operations research* 59(5):1106–1118.

Kleinrock, L. 1964. "Analysis of a Time-shared Processor". *Naval research logistics quarterly* 11(1):59–73.

Kleinrock, L. 1967. "Time-shared Systems: A Theoretical Treatment". *Journal of the ACM (JACM)* 14(2):242–261.

Kleinrock, L. 1970. "A Continuum of Time-sharing Scheduling Algorithms". In *Proceedings of the May 5-7, 1970, spring joint computer conference*, 453–458.

Kleinrock, L., and R. R. Muntz. 1972. "Processor Sharing Queueing Models of Mixed Scheduling Disciplines for Time Shared System". *Journal of the ACM (JACM)* 19(3):464–482.

Kleinrock, L., R. R. Muntz, and E. Rodemich. 1971. "The Processor-sharing Queueing Model for Time-shared Systems with Bulk Arrivals". *Networks* 1(1):1–13.

Koops, D. T., M. Saxena, O. J. Boxma, and M. Mandjes. 2018. "Infinite-Server Queues with Hawkes Input". *Journal of Applied Probability* 55(3):920–943.

Pender, J., and T. Phung-Duc. 2016. "A Law of Large Numbers for M/M/c/Delayoff-Setup Queues with Nonstationary Arrivals". In *International conference on analytical and stochastic modeling techniques and applications*, 253–268. Springer.

Ramanan, K., M. I. Reiman et al. 2003. "Fluid and heavy traffic diffusion limits for a generalized processor sharing model". *The Annals of Applied Probability* 13(1):100–139.

Saha, A., N. Ganguly, S. Chakraborty, and A. De. 2019. "Learning Network Traffic Dynamics Using Temporal Point Process". In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 1927–1935. IEEE.

Shah, A., A. Wikum, and J. Pender. 2019. "Using Simulation to Study the Last to Enter Service Delay Announcement in Multiserver Queues with Abandonment". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 2595–2605. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sriram, K., and W. Whitt. 1986. "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data". *IEEE journal on selected areas in communications* 4(6):833–846.

Wang, W., Q. Xie, and M. Harchol-Balter. 2021. "Zero Queueing for Multi-Server Jobs". *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5(1):1–25.

Ward, A., and W. Whitt. 2000. "Predicting Response Times in Processor-Sharing Queues". *Analysis of Communication Networks: Call Centres, Traffic, and Performance* 28:1.

Whitt, W. 1999a. "Improving Service by Informing Customers About Anticipated Delays". *Management science* 45(2):192–207.

Whitt, W. 1999b. "Predicting Queueing Delays". *Management Science* 45(6):870–888.

Zhang, J., J. Dai, B. Zwart et al. 2011. "Diffusion Limits of Limited Processor Sharing Queues". *Annals of Applied Probability* 21(2):745–799.

AUTHOR BIOGRAPHIES

JAMOL PENDER is an assistant professor in Operations Research and Information Engineering (ORIE) at Cornell University. He earned his PhD in the Department of Operations Research and Financial Engineering (ORFE) at Princeton University. His research interests include queueing theory, stochastic simulation, dynamical systems and applied probability. His e-mail address is jjp274@cornell.edu. His website is https://blogs.cornell.edu/jamolpender/.

ELENA ZHANG is a second-year undergraduate student in Operations Research and Information Engineering at Cornell University. Her research interests are in operations research, machine learning and applied probability and statistics. Her e-mail address is eyz5@cornell.edu.