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

SYNTHETIC INTELLIGENCE: LEARNING THE ERLANG BLOCKING FORMULA USING SIMULATION

Jensen Andre
Jamol Pender

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

ABSTRACT

The Erlang blocking formula is a seminal result in queueing theory, representing a fundamental contribution to applied probability. In this study, we utilize machine learning techniques and stochastic simulation to learn the Erlang blocking formula directly from simulated blocking data. By focusing on the blocking probability as a function of model parameters, we can leverage learning methods to bypass distributional assumptions about queueing models and instead learn from simulated data. We use this "synthetic" approach to construct new "learned" formulas for the blocking probability of the G/G/C/C queue, which currently lacks any closed-form steady state formulas. Finally, we illustrate that our machine learning approach is more accurate than approximations derive from heavy traffic limit theory.

1 INTRODUCTION

The M/M/C/C model, also known as the stationary Erlang loss model, is a classic queueing system that has a Poisson arrival process with rate λ, independent and identically distributed service times from an exponential distribution with rate μ, and C parallel servers with no extra waiting spaces. This model was first introduced in a pioneering paper by Agner Erlang (Erlang 1917), which is considered the foundational work in queueing theory. The distinguishing feature of the Erlang loss queue is that if a customer arrives while all the servers are busy, then the customer is immediately lost and never receives service. The model has wide-ranging applications in various domains, such as telecommunication networks, transportation and parking systems, healthcare systems, and even education systems, as illustrated by Miles (2014), Hampshire and Shoup (2018), Hampshire et al. (2020), and De Bruin et al. (2010). Originally, the Erlang loss queue was developed to model the arrival of telephone calls to a telephone exchange, with the aim of determining the optimal number of trunklines required to provide a high quality of service while minimizing waste.

Since Erlang's groundbreaking work, the Erlang loss model has been generalized in various directions. For example, Erlang himself studied a variation of the model in which customers who find all servers busy join a queue rather than being lost, leading to the development of the Erlang-C formula. Another extension is the loss network model (Kelly 1991), which generalizes the M/M/C/C model to multiple dimensions and captures the interaction between calls and trunklines. In addition, the Ph/M/C/C+K model has been fully analyzed by Neuts, Marcel F (1981) using classic matrix geometric methods, while more recent work by Mandjes et al. (2017) has analyzed a Markov modulated version of the Erlang loss queue.

In his seminal paper, Erlang demonstrated that it was possible to derive a formula for the blocking probability in a M/M/C/C queue, which is widely known as the Erlang blocking formula. Interestingly, Erlang also discovered that when using deterministic service times, the derived formula was insensitive to the service time distribution, implying that the service distribution did not affect the applicability of the

Erlang blocking formula. However, this observation by Erlang was based on empirical evidence, and he did not provide a formal proof of the insensitivity in the paper. It was only through the subsequent work of Sevast'yanov (1957) that a formal proof was derived for the insensitivity of the blocking formula to the service time distribution.

Furthermore, it is worth noting that the insensitivity property is not unique to the Erlang blocking formula. For example, it has been shown that loss networks, which are a multi-dimensional generalization of the Erlang loss model, also exhibit the insensitivity property with respect to the distribution of the holding times of each of the user classes (Kelly 1991). This property has important implications for the practical applicability of queueing models, as it allows the use of simpler models with deterministic or exponential service times to accurately approximate the behavior of more complex systems with general service time distributions.

Despite the insensitivity result for the service time distribution, the arrival distribution remains a crucial factor in determining the blocking probability. It is well-known that the Erlang blocking formula fails to hold for non-Poisson arrival distributions, making it necessary to develop approximations for the blocking probability. Several researchers have attempted to find such approximations, including those proposed by Whitt (Whitt 1984), Borovkov (Borovkov 1967), Jagerman (Jagerman 1974), and Beaubrun (Beaubrun et al. 2007). However, these approximations are often limited to specific regimes of parameter spaces, exploiting the central limit theorem or heavy traffic limit theory. Hence, the question of finding a good approximation for the blocking probability in any regime remains. In our work, we address this challenge by leveraging stochastic simulation and data-driven methods to uncover new blocking probability formulas for the G/G/C/C queue.

In this paper, our goal is to find an approximate probability blocking formula for the G/G/C/C queue using data driven methods and we also evaluate these data driven approximations against traditional applied probability formulas. To this end, we list the contributions of our work below.

1.1 Contributions of Our Work

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

- What machine learning methods work well at learning the Erlang blocking formula for the G/G/C/C queue?
- Is learning the Erlang blocking formula easier in the heavy traffic setting?
- How difficult is learning the Erlang blocking formula for increasing server utilization?

1.2 Organization of the Paper

The remainder of the paper is organized as follows. Section **??** introduces the Lindley recursion and gives a brief history of the formula. In Section **??**, 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 learning the Lindley recursion. Finally, a conclusion is given in Section **??**.

2 ERLANG BLOCKING FORMULA

The Erlang-B formula, developed by Agner Erlang in 1917 Erlang (1917), was a significant milestone in queueing theory and telecommunications. Erlang is widely credited with being the first to study teletraffic characteristics, and in 1909, he published a pioneering paper that provided a data-driven analysis of teletraffic characteristics Erlang (1909). His work aimed to understand the probability of all circuits being busy, which was crucial for designing efficient telecommunication networks.

The Erlang loss model, also known as M/M/C/C, is a stationary model that assumes a Poisson arrival process, independent and identically distributed service times from an exponential distribution, and c parallel

servers with no extra waiting spaces. However, most communication networks experience non-stationary conditions, which require more sophisticated models to analyze. Moreover, many of the literature on stationary processes do not easily carry over to non-stationary models, requiring additional insight and analysis.

The concepts and mathematics introduced by Agner Krarup Erlang have broad applicability beyond telephony. They apply wherever users arrive more or less at random to receive exclusive service from any one of a group of service-providing elements without prior reservation, for example, where the service-providing elements are ticket-sales windows, toilets on an airplane, or motel rooms. However, Erlang's models do not apply where the service-providing elements are shared between several concurrent users or different amounts of service are consumed by different users, as in circuits carrying data traffic.

The central goal of Erlang's traffic theory is to determine the exact number of servers required to satisfy the arrival demand without over-provisioning. To achieve this, a quality-of-service (QoS) target is set, which becomes the target probability of call blocking (Pb) when using the Erlang B formula. This formula provides a powerful tool for designing efficient telecommunication networks that can handle high traffic demands while maintaining a satisfactory QoS level.

The Erlang blocking formula is given by the following expression

$$\lim_{t\to\infty}\mathbb{P}(Q(t)=C) \quad \equiv \quad \beta(C,\rho)=\frac{\frac{\rho^C}{C!}}{\sum_{j=0}^{C}\frac{\rho^j}{j!}}$$

where $\rho=\frac{\lambda}{\mu}$. It can be shown that the Erlang blocking formula also satisfies the following recursion formula

$$\beta(C,\rho)=\frac{\rho\cdot\beta(C-1,\rho)}{\rho\cdot\beta(C-1,\rho)+C}$$

where the recursion is initialized with $\beta(0,\rho)=1$.

The recursion for the Erlang blocking formula is useful when the number of servers is large since computing the factorial with respect to the number of servers might be computationally hard. Thus, the recursion is more stable than computing the formula directly for large values of C. With Erlang blocking formula and its recursion in hand, we are interested in the question of whether we can learn the Erlang blocking formula directly from data using the M/G/C/C blocking probability formula. Thus, in this context we would be given the following data

- $X_i^{(1)}$ = mean of the inter-arrival distribution in the i^{th} example
- $X_i^{(2)}$ = mean of the service distribution in the i^{th} example
- $X_i^{(3)}$ = number of servers used in the i^{th} example
- R_i = the probability of being blocked in the i^{th} example (Erlang blocking formula).

With this data, we can now leverage machine learning techniques to find a function $f(\cdot)$ so that

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

and

$$R_i=\beta\left(X_i^{(3)},\frac{X_i^{(2)}}{X_i^{(1)}}\right)$$

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

is minimized. In a linear regression framework the goal is to find the coefficients $\beta_0, \beta_1, \beta_2$ in order to minimize the following loss function

$$\min_{\alpha_0,\alpha_1,\alpha_2,\alpha_3} \sum_{i=1}^{n} \left(R_i - \alpha_0 - \alpha_1 X_i^{(1)} - \alpha_2 X_i^{(2)} - \alpha_3 X_i^{(3)} \right)^2.$$

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.

Table 1: Linear Regression for Blocking Formula for M/M/C/C 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
$R_n(x)$ AMSE	1.94	6.37	2.84e+1	2.16e+2	2.14e-2	7.18e-2	3.41e-1	2.26

3 LEARNING A NEW BLOCKING FORMULA FOR THE G/G/C/C QUEUE

In the previous section, we demonstrated our ability to learn the probability of blocking in the M/G/C/C queue. While we could directly compare our learned formula to the exact formula due to the closed form formula for the Erlang blocking formula, the same cannot be done for the G/G/C/C blocking probability. The G/G/C/C blocking probability is sensitive to the arrival distribution, and there are no exact closed form formulas for it. As such, we cannot replicate the approach we used for the M/G/C/C queue. Instead, we turn to approximations based on heavy traffic limit theory, such as those presented in Whitt (1984), Borovkov (1967), Jagerman (1974), and Beaubrun et al. (2007). To learn a new mean queue length formula for the G/G/C/C queue, we propose using stochastic simulation to estimate the blocking probability, which allows us to obtain the value R_i for each set of parameters.

- Develop a discrete-event simulation model of the G/G/C/C queue.
- Use the simulation to generate data on the blocking probability for each set of model parameters (departure samples only).
- Split the generated data into training and test sets.
- Train multiple machine learning models using the training data to develop predictive models.
- Apply the trained models to the test data to make predictions and assess their performance.
- Evaluate the performance of the machine learning models on the test data using a metric such as average mean squared error.

$$D_n = A_n + S_n \cdot \{Q_n < K\} \tag{1}$$

3.1 How the Data is Collected and Analyzed

This paper combines simulation and machine learning to approximate a function. We consider the simulation step as crucial because it generates the training data that the machine learning methods use. Thus, the simulation step is analogous to a data collection step, except that we have more control over the data

since we can simulate it. In this section, we describe in more detail how we use simulation to generate the training data and how we produce the numerical results presented in this paper. Before delving into the different machine learning methods used for predicting response times, we first outline the simulation process for generating the necessary data.

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.

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

where

$$\hat{R}_i = f\left(X_i^{(1)}, X_i^{(2)}, X_i^{(3)}, X_i^{(4)}\right)$$

for some function $f(\cdot)$ and

$$R_i = \text{Simulated Estimate of the Blocking Probability}$$

and

- $X_i^{(1)}$ = mean of the inter-arrival distribution in the i^{th} example
- $X_i^{(2)}$ = mean of the service distribution in the i^{th} example
- $X_i^{(3)}$ = variance of the inter-arrival distribution in the i^{th} example
- $X_i^{(4)}$ = variance of the service distribution in the i^{th} example
- R_i = the simulated blocking probability in the i^{th} example.

Thus, we aim to predict the blocking probability, R_i, of the i^{th} example by using information such as the mean of the inter-arrival distribution $X_i^{(1)}$, the mean of the service distribution $X_i^{(2)}$, the variance of the inter-arrival distribution $X_i^{(3)}$, and the variance of the service distribution $X_i^{(4)}$.

3.2 Blocking Probability Approximations

Regarding the probability of blocking there are many approximations. One of the most used approximations is based on the conditioning heuristic of Whitt (1984) or the heavy traffic limits of Borovkov (1967). It approximates the blocking probability of a G/G/C/C queue by the following expression

$$\text{Heavy Traffic Blocking Probability} \quad = \quad \sqrt{\rho/z}\frac{\phi\left(\frac{C-\rho}{\sqrt{\rho z}}\right)}{\Phi\left(\frac{C-\rho}{\sqrt{\rho z}}\right)} \tag{2}$$

where $\rho = \lambda/\mu$ and z is the asymptotic peakedness of the arrival process with respect to the service time distribution as defined in Whitt (1984). Moreover, $\phi(\cdot)$ and $\Phi(\cdot)$ are the standard Gaussian pdf and cdf respectively.

The second approximation we explore for the blocking probability is credited to Walter Hayward of Bell Laboratories, and hence is called the Hayward approximation. A paper by Fredericks (1980) gives some heuristic insights into why the approximation works well for G/G/C/C loss queues. At its core, the Hayward approximation uses the well-known Erlang loss formula $\beta(C,\rho)$ and scales both parameters by the peakedness of the queue length process of the infinite server queue z. Thus, the Hayward approximation becomes

$$\text{Hayward Blocking Probability} \quad = \quad \beta(C/z, \rho/z). \tag{3}$$

It is important to note that while the infinite server heavy traffic approximation provides an estimate of the blocking probability of the exact system, the Hayward approximation uses the exact blocking probability of an approximate system. Although the two approximations differ in approach, they converge to the same limit in the heavy traffic setting. This was demonstrated by Li and Whitt (2014). Since the queues we are dealing with may not be in heavy traffic, we will use both approximations as benchmarks for our machine learning methods.

Another important thing to recognize is that the Hayward approximation is written in terms of scaled Erlang loss recurision formula. However, it is not clear how to write a recursion that is written in terms of integers for its first variable to one that takes in real numbers as its first argument. Thus, we develop Proposition 1 to help understand how to transform a formula for integers into one that is valid for real numbers.

Proposition 1 Let $\beta(C,\rho)$ be the recursion for the Erlang blocking formula. Then, the recursion can be written in terms of incomplete gamma functions i.e.

$$\beta(C,\rho) = \frac{\frac{\Gamma(C+1,\rho)}{\Gamma(C+1)} - \frac{\Gamma(C,\rho)}{\Gamma(C)}}{\frac{\Gamma(C+1,\rho)}{\Gamma(C+1)}}. \tag{4}$$

Proof. The first observation is that the Poisson distribution cdf can be written in terms of incomplete gamma functions. In particular, we have the following expression

$$\sum_{j=0}^{C} \frac{e^{-\rho}\rho^j}{j!} = \frac{\Gamma(C+1,\rho)}{\Gamma(C+1)}.$$

Thus, we have

$$\begin{aligned}
\beta(C,\rho) &= \frac{\frac{\rho^C}{C!}}{\sum_{j=0}^{C}\frac{\rho^j}{j!}} \\
&= \frac{\frac{\rho^C e^{-\rho}}{C!}}{\sum_{j=0}^{C}\frac{e^{-\rho}\rho^j}{j!}} \\
&= \frac{\frac{\Gamma(C+1,\rho)}{\Gamma(C+1)} - \frac{\Gamma(C,\rho)}{\Gamma(C)}}{\frac{\Gamma(C+1,\rho)}{\Gamma(C+1)}}.
\end{aligned}$$

□

Using the equivalent expression given in Equation 4, we can write the Hayward approximation in terms of incomplete gamma functions Therefore the Hayward approximation can be written as

$$\text{Hayward Blocking Probability} = \beta(C/z,\rho/z) = \frac{\frac{\Gamma((C+1)/z,\rho/z)}{\Gamma((C+1)/z)} - \frac{\Gamma(C/z,\rho/z)}{\Gamma(C/z)}}{\frac{\Gamma((C+1)/z,\rho/z)}{\Gamma((C+1)/z)}}. \tag{5}$$

We will leverage this new incomplete gamma function representation for the Hayward formula when we compare our data driven approximations versus the heavy traffic approximations.

4 ML METHODS FOR LEARNING THE ERLANG BLOCKING FORMULA

4.1 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 $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ in order to minimize the following loss function

$$\min_{\alpha_0,\alpha_1,\alpha_2,\alpha_3,\alpha_4} \sum_{i=1}^{n} \left(R_i - \alpha_0 - \alpha_1 X_i^{(1)} - \alpha_2 X_i^{(2)} - \alpha_3 X_i^{(3)} - \alpha_4 X_i^{(4)} \right)^2$$

in terms of the independent variables $\left(X_i^{(1)}, X_i^{(2)}, X_i^{(3)}, X_i^{(4)} \right)$. 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.

Table 2: 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

4.2 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 3 - **??**, 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.

4.3 Gaussian Processes

In this section, we describe the performance of Gaussian processes on predicting the response times of the processor sharing queue. In Tables 4 - **??**, 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,

Table 3: 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

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

Table 4: 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

4.4 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 5 - **??**, we observe that the performance of K-nearest neighbors 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 5: 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

4.5 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 6 - **??**, 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 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 6: 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

4.6 Summary of Numerical Results

So far, we have shown that machine learning can do well at predicting

ACKNOWLEDGMENTS

WE love so and so...

REFERENCES

Beaubrun, R., S. Pierre, and J. Conan. 2007. "Analysis of traffic distribution and blocking probability in future wireless networks". *International Journal of Wireless Information Networks* 14(1):47–53.

Borovkov, A. 1967. "On limit laws for service processes in multi-channel systems". *Siberian Mathematical Journal* 8(5):746–763.

De Bruin, A. M., R. Bekker, L. Van Zanten, and G. Koole. 2010. "Dimensioning hospital wards using the Erlang loss model". *Annals of Operations Research* 178(1):23–43.

Erlang, A. K. 1909. "The theory of probabilities and telephone conversations". *Nyt. Tidsskr. Mat. Ser. B* 20:33–39.

Erlang, A. K. 1917. "Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges". *Post Office Electrical Engineer's Journal* 10:189–197.

Fredericks, A. 1980. "Congestion in blocking systems—a simple approximation technique". *Bell System Technical Journal* 59(6):805–827.

Hampshire, R. C., S. Bao, W. S. Lasecki, A. Daw, and J. Pender. 2020. "Beyond safety drivers: Applying air traffic control principles to support the deployment of driverless vehicles". *PLoS one* 15(5):e0232837.

Hampshire, R. C., and D. Shoup. 2018. "What share of traffic is cruising for parking?". *Journal of Transport Economics and Policy (JTEP)* 52(3):184–201.

Jagerman, D. L. 1974. "Some properties of the Erlang loss function". *Bell System Technical Journal* 53(3):525–551.

Kelly, F. P. 1991. "Loss networks". *The annals of applied probability*:319–378.

Li, A. A., and W. Whitt. 2014. "Approximate blocking probabilities in loss models with independence and distribution assumptions relaxed". *Performance Evaluation* 80:82–101.

Mandjes, M., P. G. Taylor, and K. De Turck. 2017. "The Markov-modulated Erlang loss system". *Performance Evaluation* 116:53–69.

Miles, A. 2014. "No Classroom Left Uncovered: An Application of Queueing Theory to Public Schools' Substitute Teacher Systems".

Neuts, Marcel F 1981. "Matrix-geometric solutions in stochastic models, volume 2 of Johns Hopkins Series in the Mathematical Sciences".

Sevast'yanov, B. A. 1957. "An ergodic theorem for Markov processes and its application to telephone systems with refusals". *Theory of Probability & Its Applications* 2(1):104–112.

Whitt, W. 1984. "Heavy-Traffic Approximations for Service Systems With Blocking". *AT&T Bell Laboratories Technical Journal* 63(5):689–708.

AUTHOR BIOGRAPHIES

JENSEN ANDRE is a undergraduate student in Computer Science at Cornell University. His research interests are in cloud computing and working at Google, machine learning and healthcare. His e-mail address is sl3243@cornell.edu.

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/.