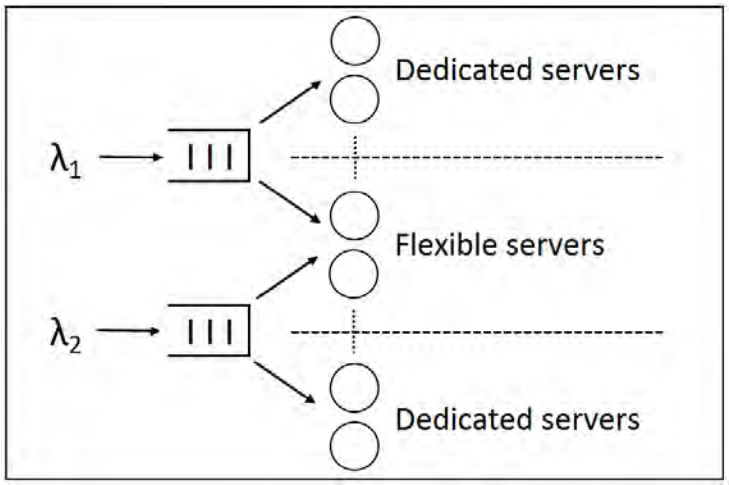
Approximating queueing functions with simulation and data analysis



Ivo van der Stap

Research Paper Business Analytics

Supervisor: René Bekker

Vrije Universiteit
Faculty of Exact Sciences
De Boelelaan 1081
1081 HV Amsterdam

29-05-2016

Preface

This paper was written for my study MSc. Business Analytics. In preparation for writing a Master Thesis, students have to write a short paper about a subject of their own choice. This paper was written in the period April-May 2016. It turned out to be a difficult problem I created for myself and not all results are what I had hoped for. On the positive side, some results are actually the approximations I was looking for and I did learn a lot. It wouldn't have been any fun anyway if it would have been an easy problem.

I'd like to thank my supervisor René Bekker for his help and numerous ideas to progress the paper.

Enjoy reading!

Abstract

In this paper we explore the possibilities of data mining techniques on data obtained with a queueing simulation. The simulation model has two queues, where each queue has dedicated servers, and both queues share flexible servers as well.

An approximation was found for the probability of delay using a product of two Erlang C systems and making a correction on the error with a decision tree. The final approximation estimates the probability of delay with an R^2 of 0.982

We also approximated the mean waiting time in case of delay by modeling it as single system that suffers from a Poisson overflow by another system. However in various cases this approximation differed significantly from the actual waiting time. A possible explanation is that the overflow in this system often does not approximate Poisson arrivals.

Contents

1. Introduction

Problem formulation

In queueing theory, we are usually interested in exact expressions of key performance indicators (KPI's). Two of the most used KPI's are the expected waiting time in the queue and probability of waiting longer than some set time t in the queue. For the simplest queues or queueing networks, like M/G/1 queues and Jackson networks, exact expressions are known for such KPI's. When networks or queues are more complex approximation formulas can be constructed using mathematical techniques. A well-known example is Kingsman's formula [1] which approximates the mean waiting time in the GI/G/1 queue by using the coefficient of variation for the interarrival and service times. Many other papers are written about queueing approximations; Shanthikumar and Buzacott compare different approximations for the single server queue in their paper[2] and Ward Whitt has written lots of papers about approximations, for example about the GI/G/M queue [3] or about queues in heavy traffic[4].

For many systems it is difficult to derive these exact or approximate expressions, this is due to the fact that also the asymptotic analysis is complicated or intractable. When these problems arise in a professional setting, the usual approach is to use simulation to estimate the KPI's, tweaking the parameters (number of servers for example) for every run and comparing results. Afterwards these simulation results are only used to implement the best system and the results are not used to find approximate formulas of KPI's for the simulated systems.

Research question and structure of paper

In this research paper the following question is answered:

"Is it possible to create KPI approximations for a queueing network by fitting functions on simulation data with techniques from data mining?"

The KPI's focused on in the paper are the probability of waiting, and the expected waiting time given waiting. For this a queueing network has been simulated. The network that has been simulated is a two queue network, where each queue has dedicated servers. There are flexible servers as well, that can process customers from both queues in a first-come, first-served discipline. A more in depth look and notation of the network is in chapter 2. The building, running and testing of the simulation as well as the settings used for the simulation will be explained in the chapter 3. After that the work done on estimating the KPI's using a combination of queueing theory and data analysis is in chapter 4. Finally chapters 5 and 6 have results, conclusions and recommendations.

2. Queueing network

To answer the research question, we consider the expected waiting time and the probability of waiting for the system depicted in figure 1

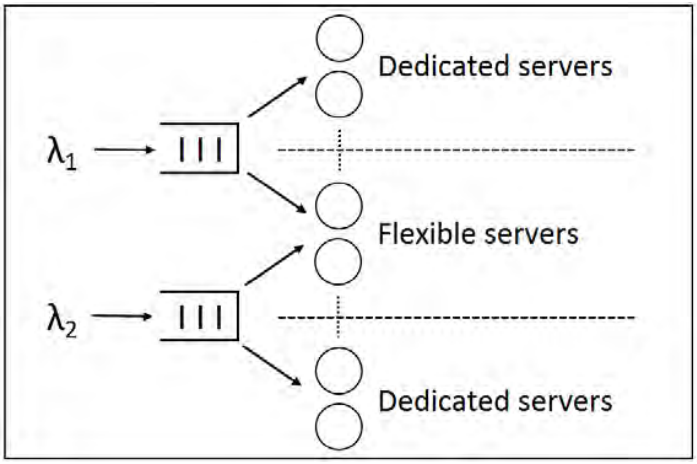


Figure 1: The simulated network

There are two separate queues, in each queue different customers arrive, c_1 and c_2. The customers arrive according to Poisson processes with rates λ_1 and λ_2, respectively. The service times of customers of types c_1 and c_2 are exponentially distributed with rates μ_1 and μ_2, respectively. Then there is a number of dedicated servers, s_1 and s_2, that can process customers of types c_1 and c_2. Finally there is a number of flexible servers of type s_f that can process both types of customers. The flexible servers will process customers according to the First-In-First-Out principle. The rate at which the flexible servers process customers is the same as the dedicated servers. So when a flexible server processes a customer of type c_1, the server rate becomes μ_1 and when it processes a customer of type c_2, the rate becomes μ_2.

An important aspect of queueing is the stability of the system. It has to be ensured that all work can be processed by the servers. In the long run queues grow to infinity if this is not the case. In this specific network there are 3 stability conditions that have to be fulfilled. These are:

1. The amount of work for customers of type 1 can be done by the combination of the servers of type 1 plus servers of type f. Acting as if the flexible servers are dedicated to only type 1.
   $$\rho_1 = \lambda_1/((s_1 + s_f)\,\mu_1) < 1$$
2. Similar for type 2:
   $$\rho_2 = \lambda_2/((s_2 + s_f)\,\mu_2) < 1$$
3. Under above settings, it's possible that the required amount of work for the flexible servers exceeds 100%. To prevent this from happening a third stability condition is required. This is the total utilization of the system ρ_t
   $$\rho_t = (\lambda_1/\,\mu_1 + \lambda_2/\,\mu_2)\,/(s_1 + s_2 + s_f) < 1$$

A practical example of the simulated network is a call center where there are two types of customers having questions about either subject 1 or subject 2. There are three types of call centers employees. The first two types of employees can answers questions about subject 1 and subject 2, the third type are highly skilled employees who can answer questions about both subjects. In this network, questions

about one subject can take structurally longer to get service compared to questions about the other subject.

Some special cases can occur that have known solutions. The first case is when $s_f = 0$, in which case we have two separate M/M/C systems. The M/M/C system is also known as Erlang C and depicted in Figure 2. Another special case is when $\lambda_1 = 0$ or $\lambda_2 = 0$, where the flexible servers act as dedicated servers. This corresponds to an M/M/C system as well.

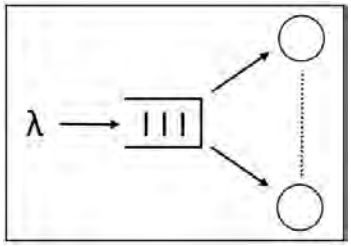


Figure 2: The Erlang C model

3. Simulation

Building the simulation.

The simulation was built in the statistical programming language R. Some of the advantages of using R are quick and easy functions for drawing random numbers, the ability to do both the analysis and the simulation in the same language and the power and simplicity of vector and matrix manipulation. A disadvantage of using R is that it is considered to be a slow language, especially when using for loops and lookups. Even with this disadvantage, and after several language specific optimizations, about half a billion arrivals could be simulated within a day on an Intel®Core™ i7-4700MQ CPU @ 2.40GHz with 16GB RAM, which is enough for this paper.

The general structure of the simulation is a discrete event simulation. The simulation uses a table where every customer has its own row. The columns in the table are:

1. Arrival times. This is created immediately at the start of the simulation.
2. Arrival types. Contains 1 or 2 to indicate which type of customer arrives. Created at the start of the simulation.
3. Service times. Created at the start of the simulation.
4. Server-in times. This has all the times when the customer enters the server, so after waiting in queue.
5. Server-out times. This contains the times when the customer leaves the system
6. Server-type. Contains 1, 2 or f to indicate the type of server serving the customer.

The main program loops through the arrival times and server-out columns updating Server-in, Server-out, and Server-type until the last customer has left the system. Some extra variables are updated as well like number of customers in each queue and first customer leaving.

In the form of a flowchart the simulation looks like figure 3. The simulation loops through this flowchart top to bottom until all customers have left the system.

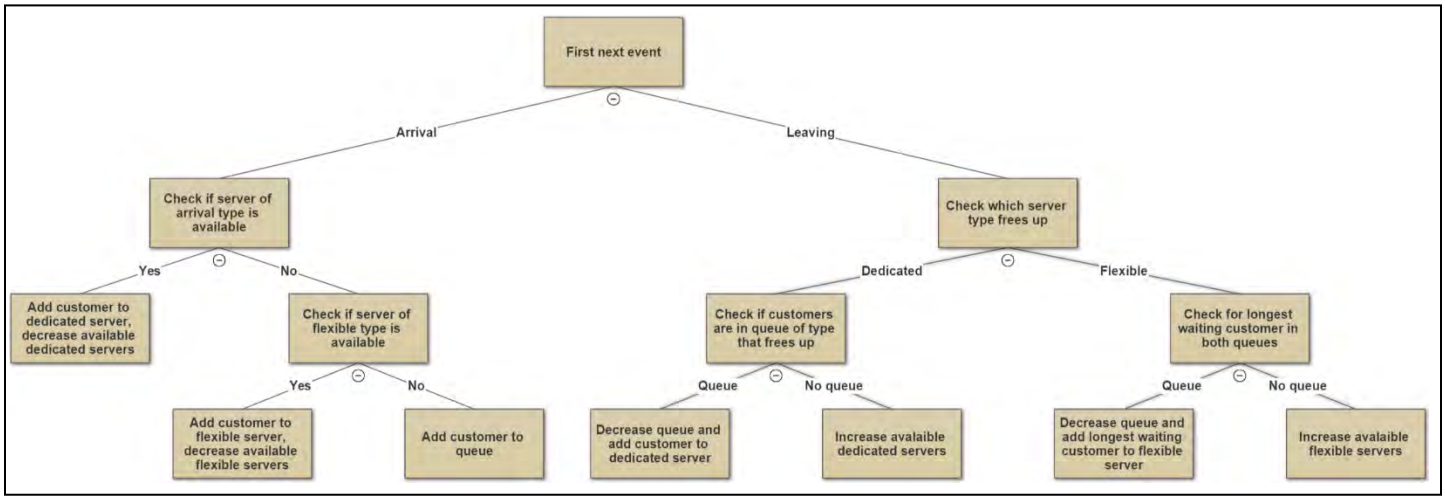


Figure 3: Main flow of the program

Parameter settings for the simulation

There are 7 parameters to be set for every simulation. In this paragraph the parameter settings are explained, starting with the number of servers. For this paper we made the decision to always have at least one of every server type. The maximum number of servers of each type was chosen at 4. This maximum is chosen reasonably small such that each possible configuration can be simulated a few times.

The service rate is chosen such that the servers do not work at different time scales. The maximum difference between both service rates is a factor 10. With this maximum the servers do not work at different orders of magnitude.

After the number of servers and the server rate have been set the arrival rates can be set such that they meet the stability requirements, given in chapter 2. However if one of the server utilization is really low, the overlap between the two systems is very small, effectively making it one Erlang C model, and one model with no waiting at all. On the other side if the server utilization is very close to 1 the average queues tend to grow to very large numbers. Simulation of queueing systems with utilizations close to 1 is a challenging topic. To avoid such issues all arrival rates have been chosen such that ρ_1, ρ_2, and ρ_t are all between 0.30 and 0.98. In table 1, a summary of the parameter settings can be found.

Variable	Setting	Variable type
s_1	Discrete random number from 1:4	Number of servers of type 1
s_2	Discrete random number from 1:4	Number of servers of type 2
s_f	Discrete random number from 1:4	Number of servers of flexible type
μ_1	Continuous random number drawn from the uniform distribution between 0.1 and 1	Service rate of customers of type 1
μ_2	Continuous random number drawn from the uniform distribution between 0.1 and 1	Service rate of customers of type 2
λ_1	Continuous random number such that ρ_1 is between 0.30 and 0.98	Arrival rate of customers of type 1
λ_2	Continuous random number such that both ρ_2 and ρ_t are between 0.30 and 0.98	Arrival rate of customers of type 2

Table 1: Simulation settings

Running the simulation

To give good estimations of the KPI's and have room for data analysis a few hundred simulations were run drawing new parameters for every simulation from table 1. For this paper each of these runs consists of several sub runs. This is for two reasons:

1. It is easier to give confidence intervals for the KPI's with several sub runs because customers within the same run are interdependent; waiting customers usually create more waiting customers. All different sub runs are independent of each other.
2. The simulation uses lookups in the in-memory-table to find the longest waiting customers when a server becomes available. Using one simulation and one huge table makes these lookups really slow compared to a lot of small simulations.

However, a disadvantage of using several sub runs is that the KPI's should be about the long term averages while every sub run starts with empty queues and empty servers. Using lots of runs these repeating empty queues influence the averages significantly. This is why a startup period is necessary for every run. Using a startup period by simulating a big number of customers and only starting the "true" simulation after this startup will have some runs starting with empty queues, and others runs with long queues.

To find out how much customers is a big number of customers several settings for this start up number were tested for theoretically known queues. For the M/M/1 queue with a utilization factor ρ = 0.9 and with a 200 customer startup, and 1,000 "real" customers in every sub run the simulation did not converge to its theoretically expected waiting time, even after ten thousands of these runs. Apparently after 200 arriving customers the empty starting state of the system still influenced the expected waiting time significantly.

After this a 1,000 customer startup was used for every independent sub run of the simulation, which did converge for a M/M/1 queue with ρ = 0.9. However when testing the simulation with the parameters from table 1, it did not converge to the true average on these simulations. This was found by comparing the mean waiting time for the first customers after the startup period with the mean waiting time of the last customers after the startup period. In some simulation cases there was a significant difference. In one simulation case with randomly drawn parameters it led to graph in Figure 4, which shows the mean waiting time of the nth-customer after startup on the x-axis:

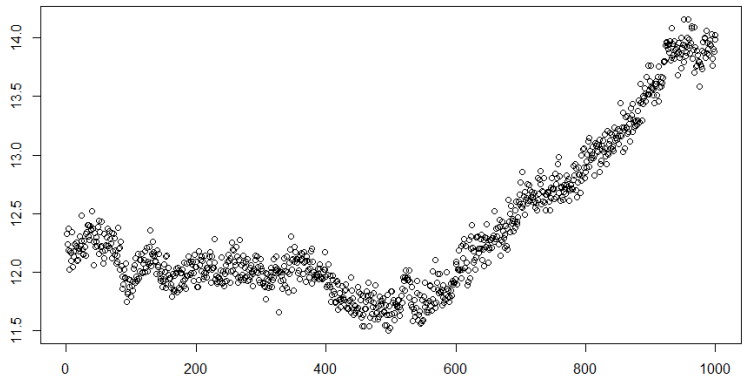


Figure 4: Mean waiting time is still growing for customers arriving later

In Figure 4 the first customers after the startup period still have shorter queues on average than the customers that come later in the system. This means that the startup number of customers should be larger than the 1,000 to prevent some customers getting influenced by the empty starting state. To be safe the startup number of customers was set to 10,000 and this growing relationship was not found anymore.

Finally the simulation was run, doing 100 sub runs per simulation, each containing 5000 customers, yielding a total of 500,000 customers. Running this for a day resulted in a total of 315 simulations with different settings. Because the roles of the queue 1 and 2 can be exchanged every simulation could be used twice to approximate the KPI's. In other words, it is possible to estimate the KPI's for both customers of type and c_1 and c_2 respectively.

4. Approximations

In this chapter several approximations are made for the KPI's. The Erlang C model is used several times in this chapter; a short introduction to the Erlang C formulas and notation will thus be given first. After this introduction a section is dedicated to a lower bound for the expected waiting time in the two queue system using the Erlang C formulas. After that paragraph the approximations will be made.

Erlang C

The model with Poisson arrivals, exponentially distributed service times and more than one server is named the Erlang C model. The formulas for the Erlang C model can be found in a lot of textbooks; in this paper we will use the formulas and notation used in the educational textbook from Koole[5]. The expected waiting time is given by a formula where the denominator represents the expected waiting time in case of delay and the numerator the probability of waiting. In the denominator s is the number of servers, μ is the service rate and λ is the arrival rate. In the numerator of the formula a is the offered load λ/μ and C(s,a) is the probability of delay.

$$\mathbb{E}W_Q = \frac{C(s,a)}{s\mu - \lambda}$$

C(s,a) is given by the following formula, in this formula π(j) is the stationary distribution of j customers in the system:

$$C(s,a) = \sum_{j=s}^{\infty} \pi(j) = \frac{a^s}{(s-1)!\,(s-a)} \left[\sum_{j=0}^{s-1} \frac{a^j}{j!} + \frac{a^s}{(s-1)!\,(s-a)} \right]^{-1}$$

With the stationary distributions given by:

$$\pi(j) = \begin{cases} \frac{a^j}{j!}\pi(0), & if\ j < s \\ \frac{a^j}{s!\,s^{j-s}}, & otherwise \end{cases}$$

And the probability of being empty:

$$\pi(0)^{-1} = \sum_{j=0}^{s-1} \frac{a^j}{j!} + \frac{a^s}{(s-1)!\,(s-a)}$$

Lower bound on the expected waiting time using Erlang C

As explained in chapter 3, a special case occurs when $\lambda_2 = 0$. In this case the modeling of customers of type c_1 corresponds to an Erlang C model. However, even when $\lambda_2 > 0$ the model for c_1 can still be approximated as Erlang C model. In this case all flexible servers are dedicated to customers of type c_1, with customers of type c_2 coming into the system and blocking some of the capacity

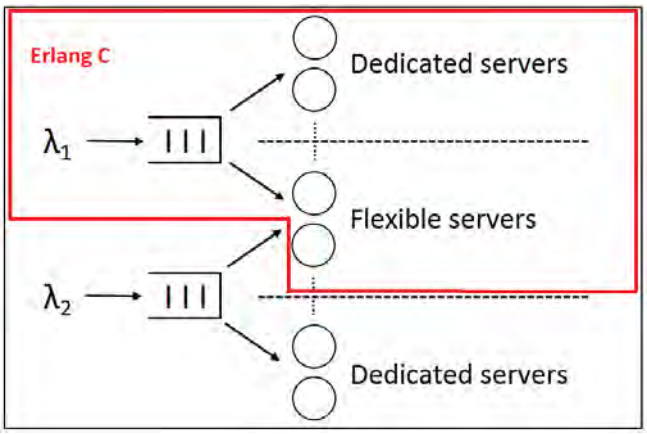


Figure 5: When ignoring customers of type 2 it becomes an Erlang C system

This occasional blocking will make the expected waiting time of customers of type c_1 bigger. Decreasing the average availability of servers will never decrease waiting times, so the Erlang C model can be used as a lower bound of the function of the expected waiting time in the network.

Besides functioning as a lower bound for the expected waiting time of the simulated network, the structure of the formula for the expected waiting time can also function as a framework for the formula to look for. It hits the core of queueing theory because it can be broken up in the two essential parts: probability of delay and mean waiting time in case of delay. We will try to approximate these two useful KPI's separately from here on.

Estimating probability of delay

First a formula for the probability of delay will be constructed. Because the network resembles a combination of two Erlang C systems it makes sense to use those formulas as a basis for the new approximation.

The probability estimation is achieved by acting as if the original ***two-queue-system*** consists of two independent ***Erlang C sub-systems***. For each of these two independent subsystems, the stationary probabilities are calculated. By taking the product of the stationary distributions of the two subsystems an estimation is made for the probability of delay.

In figure 6, a product of two Gaussian functions is shown. When for example "delay" would be experienced when the sum of the Gaussian functions is positive, the probability of this could be calculated by taking the volume under the region where this product is positive. In a similar way, an approximation of the stationary state for the two-queue-system could be calculated by taking the product of the two subsystems.

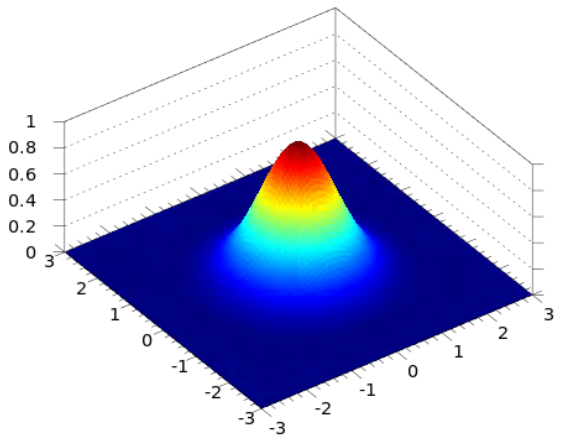


Figure 6: Visualization [6] of the product of two Gaussian functions.

Each subsystem n has $s_n + s_f$ servers available, arrival rate λ_n and service rate μ_n. Some notation for the two sub-systems is required from here on: $\pi(f_n = i)$ denotes the stationary distribution where i flexible servers are utilized by customer type n. In this definition, if i = 0, exactly 0 flexible servers are occupied and all regular servers are occupied.

There are three cases that can be calculated using the Erlang C utilization formulas from above:

1. Occupation of $f_1 < 0$ flexible servers, in other words there are is at least one dedicated servers available:

$$\pi(f_1 < 0) = \sum_{k=0}^{s_1 - 1} \pi(k)$$

2. Occupation from exactly all dedicated servers occupied, and k flexible servers, where k is between 0 and s_f-1. In other words $f_1 = 0$ to $f_1 = s_f - 1$ flexible servers occupied:

$$\pi(f_1 = k) = \pi(s_1 + k)$$

3. All flexible servers $f_1 = s_f$ being occupied:

$$\pi(f_1 = s_f) = C(s, a) = \sum_{k=s_1+s_f}^{\infty} \pi(k)$$

If both subsystems would act like independent Erlang C systems, each having s_f flexible servers, the long term average of the events $f_1 = i$ and $f_2 = j$ would be given by the product of these

$$\pi(f_1 = i, f_2 = j) = \pi(f_1 = i) * \pi(f_2 = j)$$

In the example in figure 7, with 4 “flexible” servers per subsystem, the product $\pi(f_1 = 3) * \pi(f_2 = 2)$ is the long term average where 5 “flexible” servers are in use, 3 by type c_1 and 2 by type c_2. Normally this would not cause any of the two subsystems into a delay-state. However when the server capacity is shared using at least the number of available shared flexible servers does cause the two-queue-system

into a delay-state. That's why for an estimation of the new delay odds these situation will be summed on top of the old delay probability C(s,a).

ODDS	$\pi(f_1=4)$
$\pi(f_2<0)$	$\pi(f1=4)*\pi(f2<0)$

ODDS	$\pi(f_1=0)$	$\pi(f_1=1)$	$\pi(f_1=2)$	$\pi(f_1=3)$	$\pi(f_1=4)$
$\pi(f_2=0)$	$\pi(f1=0)*\pi(f2=0)$	$\pi(f1=1)*\pi(f2=0)$	$\pi(f1=2)*\pi(f2=0)$	$\pi(f1=3)*\pi(f2=0)$	$\pi(f1=4)*\pi(f2=0)$
$\pi(f_2=1)$	$\pi(f1=0)*\pi(f2=1)$	$\pi(f1=1)*\pi(f2=1)$	$\pi(f1=2)*\pi(f2=1)$	$\pi(f1=3)*\pi(f2=1)$	$\pi(f1=4)*\pi(f2=1)$
$\pi(f_2=2)$	$\pi(f1=0)*\pi(f2=2)$	$\pi(f1=1)*\pi(f2=2)$	$\pi(f1=2)*\pi(f2=2)$	$\pi(f1=3)*\pi(f2=2)$	$\pi(f1=4)*\pi(f2=2)$
$\pi(f_2=3)$	$\pi(f1=0)*\pi(f2=3)$	$\pi(f1=1)*\pi(f2=3)$	$\pi(f1=2)*\pi(f2=3)$	$\pi(f1=3)*\pi(f2=3)$	$\pi(f1=4)*\pi(f2=3)$
$\pi(f_2=4)$	$\pi(f1=0)*\pi(f2=4)$	$\pi(f1=1)*\pi(f2=4)$	$\pi(f1=2)*\pi(f2=4)$	$\pi(f1=3)*\pi(f2=4)$	$\pi(f1=4)*\pi(f2=4)$

NO DELAY
DELAY IN ONE-QUEUE-SYSTEM
DELAY IN TWO-QUEUE-SYSTEM

Figure 7: Old and new delay probabilities

Summing only over the old delay-state and the new delay-states, the estimated probability of delay for customers of type 1 will thus be given by the following formula, which will be denoted as **$C^*(s,a)$** :

$$C^*(s,a) = \mathbb{P}(W_{Q1} > 0) = \pi(f_1 = s_f) * \pi(f_2 < 0) + \sum_{i=0}^{s_f} \sum_{j=s_f-i}^{s_f} \pi(f_1 = i) * \pi(f_2 = j)$$

This formula is used on the dataset from the simulation, resulting in the following plot:

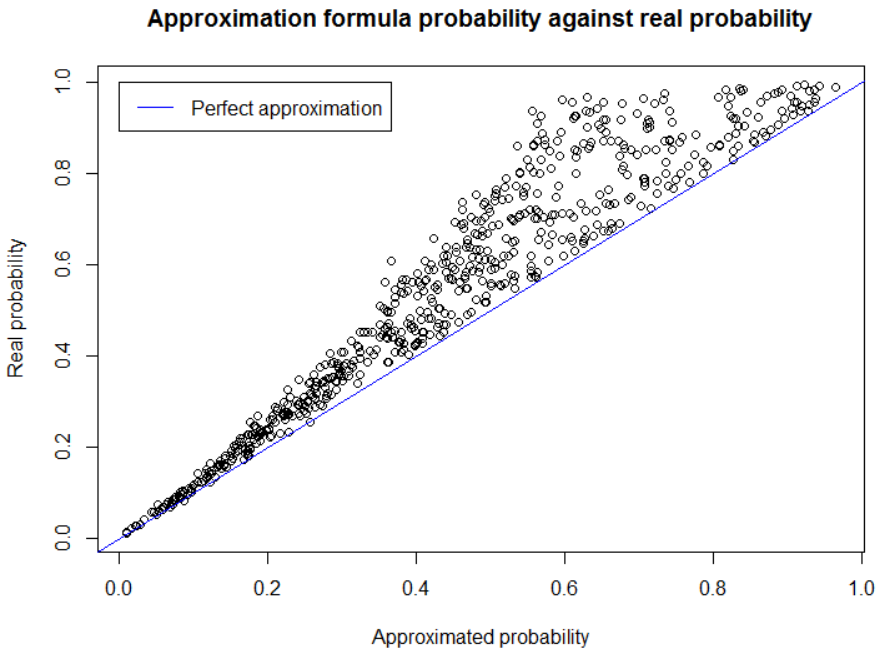


Figure 8: The approximated probability of delay is shown against the real probability of delay

Looking at figure 8, the approximation looks like a lower bound on the real probability. It seems as if the approximation formula always underestimates the real delay probability. Especially when the approximated probability becomes larger, in the 0.4-0.8 range, the difference between the real probability and the approximated probability can be significant.

The absolute difference between the real probability and the approximated probability of C*(s,a) is calculated; from here on we'll call this the error. If, for example, the approximation has an output of 0.5 for the probability of delay and the real probability of delay is 0.6, the error is 0.1

In figure 9 we plot the errors of the approximation against ρ_t. The first observation is that the errors become bigger on average as ρ_t gets closer to 1. A possible explanation of this is that the approximation assumes that the subsystems do not influence each other while in reality, the subsystems influence each other, and this influence becomes larger as ρ_t grows.

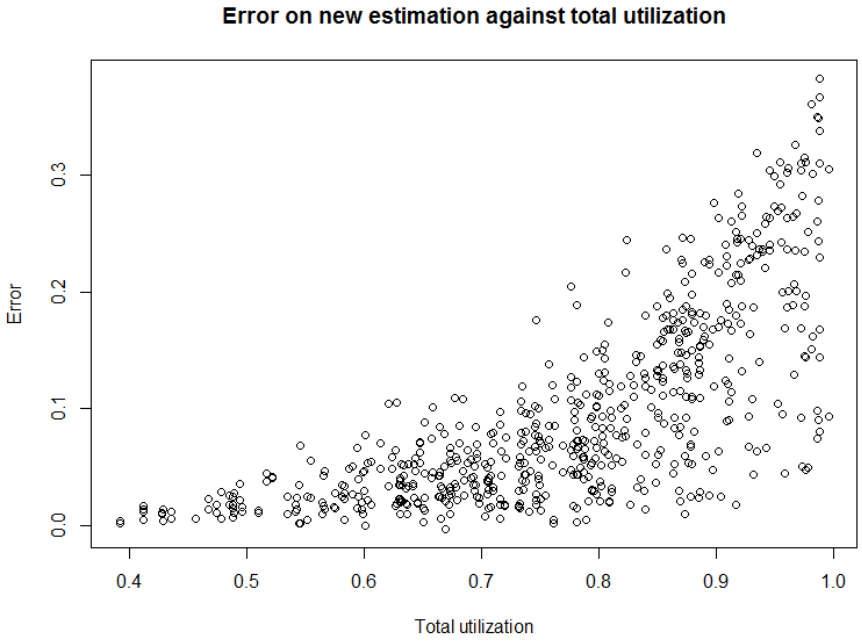


Figure 9: The errors on the new approximation visualized

It also strongly depends on the value of ρ_1 as shown in the following plots:

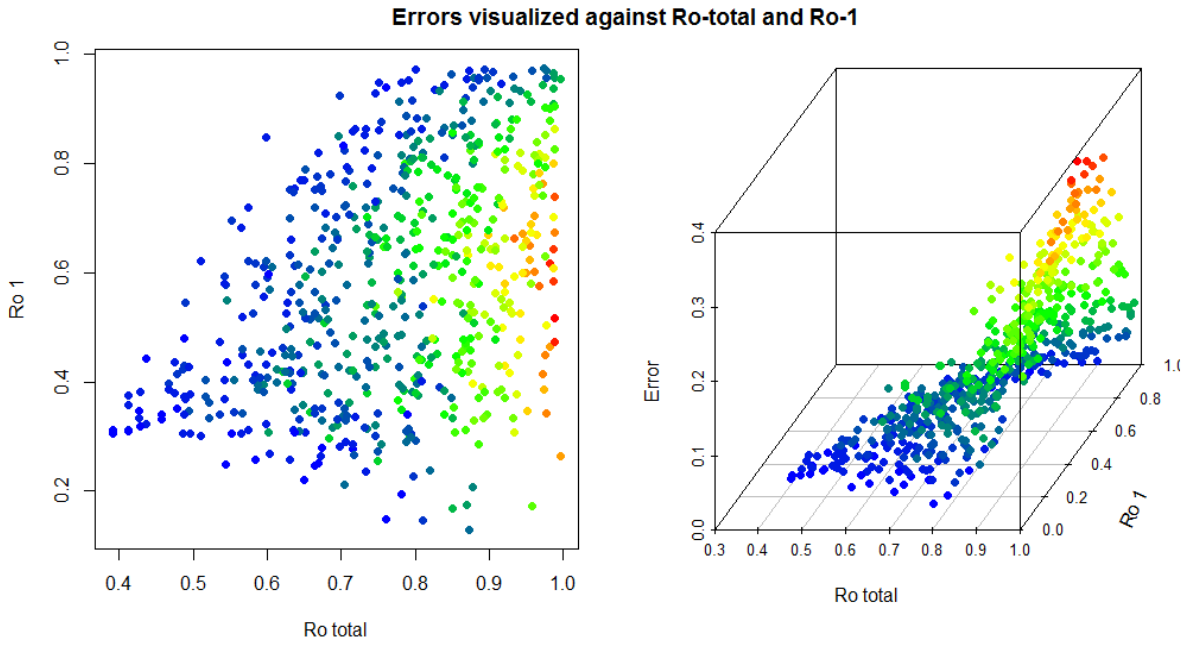


Figure 10: The colors represent the size of the error, with blue being small errors and red big errors

In figure 10, the errors are biggest when ρ_1 is between 0.4 and 0.7 and ρ_t is very close to 1. What would be ideal here is function fitted to this "hill-shape". As estimation a decision tree can predict the error.

The “rpart” package from R is used to build the decision tree. By default, rpart uses entropy to make the splits the dataset in different branches. Because the plots from above show that the load is a crucial factor in explaining the error on our first prediction, we use the load factors as predictors.

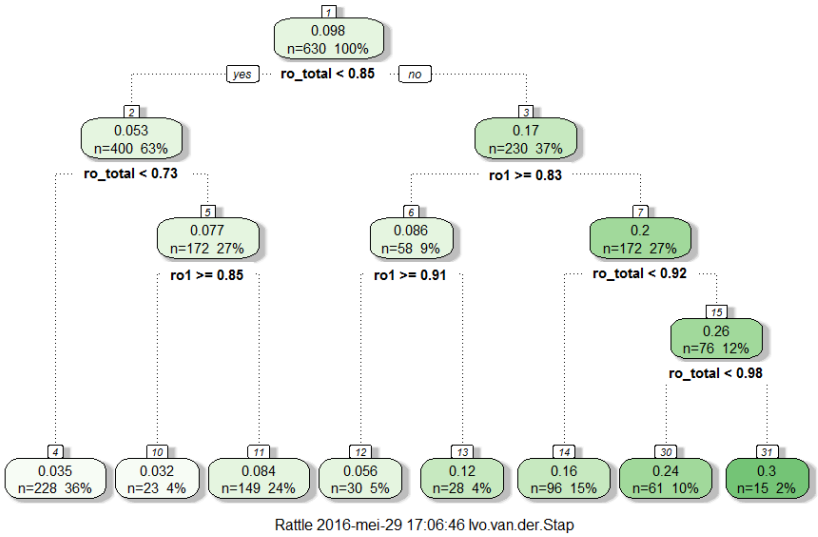


Figure 11: Decision tree boundaries to estimate the error on the first prediction

In Figure 11, the decision tree is plotted. The numbers in the boxes show the predicted error on top and the number of occurrences in the dataset on the bottom. The bottom left box thus shows, that in 228 cases (36% of total dataset), ρ_t was smaller than 0.73, and the average error in this case was 0.035. This means that the correction on C*(s,a) is 0.035 for these cases.

This makes the final estimation a combination of the adjusted C*(s,a) and the “error fixing” of the decision tree.

$$\mathbb{P}(W_{Q1} > 0) = C^*(\mathrm{s},\mathrm{a}) + \mathrm{DecisionTree}$$

In the chapter *results* a short analysis is provided how well this approximation works.

Estimating mean waiting time in case of delay

It is difficult to estimate the mean waiting time in case of delay. We tried to model the system as an M/G/c model. In this model we interpret it as an isolated system, just as explained in the chapter *Lower limit on the expected waiting time using Erlang C*. From here on we will look at the system from the perspective of one queue only, type c_1. We assume that the customers outside of this system influence the isolated system by “overflow”. We assume this overflow to be a Poisson arrival process, and have a minimum arrival rate 0, and a maximum rate λ_2. We’ll denote overflow fraction of type 2 customers as *p*. As goal, we’ll assume the independent Poisson overflow assumption as valid and try to find a way to estimate p in terms of the 7 variables in the model.

Visualized, we try to approximate p in the following system:

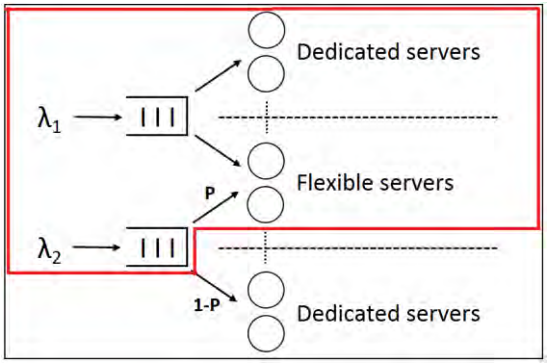


Figure 12: Mean waiting time approximation with overflow $p\lambda_2$

A general formula is known to approximate the M/G/c queue, already given in 1959 [7]. According to Whitt [3], this formula is usually an excellent approximation of most M/G/c systems. The formula is given by:

$$\mathbb{E}W_Q = \frac{C(s,a)}{s\mu - \lambda} \cdot \frac{1+c^2}{2}$$

In this formula c^2 is the coefficient of variation of the service times. Several parameters in this equation are unknown; however they can all be expressed in terms of p and other known expressions, leaving only p unknown.

From here on we assume this one-queue system with overflow (M/G/c) as a good approximation for the two-queue-system. Next we will identify the parameters of this system.

The arrival rate is given by

$$\lambda = \lambda_1 + p\lambda_2$$

The service rate is given by

$$\mu = 1/\mathbb{E}S$$

With

$$\mathbb{E}S = \frac{\lambda_1}{\lambda_1 + p\lambda_2} \cdot \frac{1}{\mu_1} + \frac{p\lambda_2}{\lambda_1 + p\lambda_2} \cdot \frac{1}{\mu_2}$$

The coefficient of variation of the service times is given by:

$$c^2 = \frac{\mathbb{V}ar\ S}{(\mathbb{E}S)} = \frac{\mathbb{E}S^2 - (\mathbb{E}S)^2}{(\mathbb{E}S)^2} = \frac{\mathbb{E}S^2}{(\mathbb{E}S)^2} - 1$$

With

$$\mathbb{E}S^2 = \frac{\lambda_1}{\lambda_1 + p\lambda_2} \cdot \frac{2}{{\mu_1}^2} + \frac{p\lambda_2}{\lambda_1 + p\lambda_2} \cdot \frac{2}{{\mu_2}^2}$$

A decent approximation for the probability of delay C(s,a) has already been given above, however, for data analysis purpose the probability of waiting of the simulation will be used.

As we assume the overflow model as a good way to approximate the two-queue-system, the goal is to estimate p in terms of the parameters from the two-queue-system. The probability p could be calculated by plugging all known formulas into the expression for the expected waiting time in the queue. After that we assume to know the true waiting times from the simulation, and use these as well; after that p could be isolated. However p can be found as well by following the curve that p produces as it grows bigger. By using the fact that the mean waiting time will only grow as p grows larger it is a matter of trying. It starts at p = 0 and can become p = 1 at most. So starting at p=0 and very slowly growing towards 1 until the right side of the expression gets bigger than the expected waiting time at the left side of the expression. We show this for one case in Figure 13.

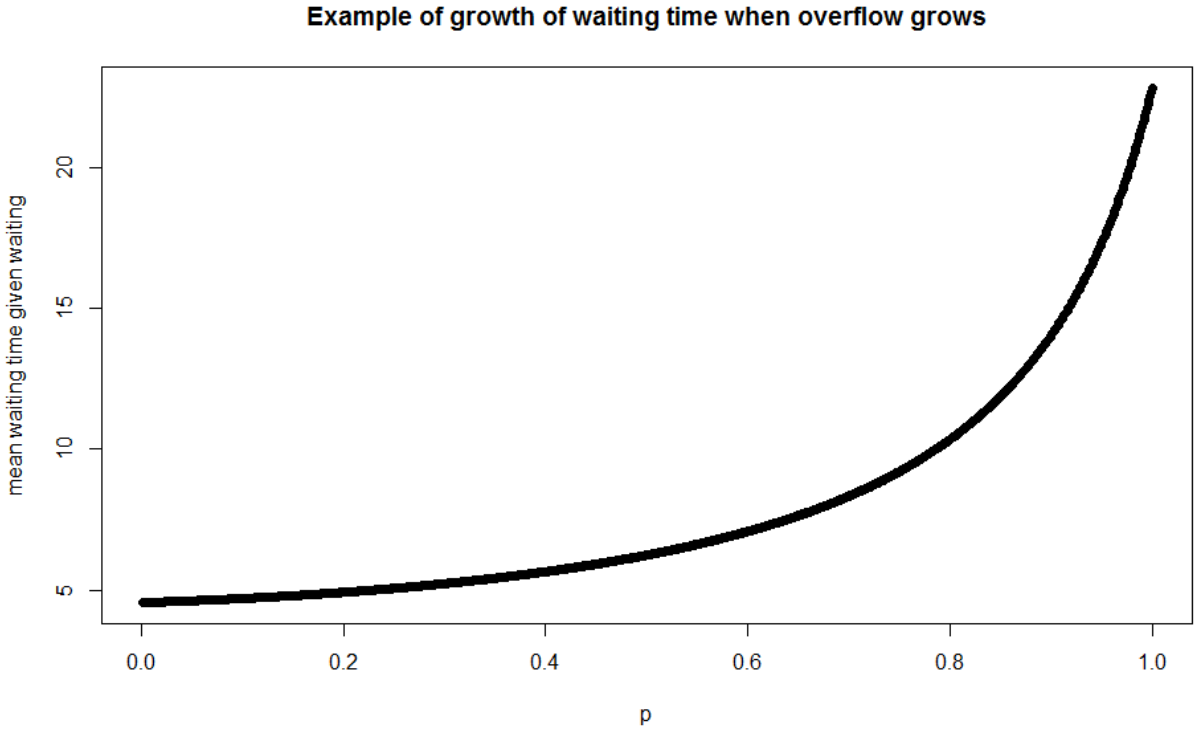


Figure 13: In this picture it can be seen that finding p is just a matter of following the curve

When using this for the 630 results from the simulation in 63 (exactly 10%) cases the mean waiting time in case of delay is actually only achievable with a negative overflow. When just focusing on the cases that at least have some overflow it is also hard to approximate p. We know the value of p from the simulation, however no clear relation is found between p and the simulation parameters. In figure 14 we plot the overflow against several parameters and against the difference between C(s,a) and C*(s,a) from last paragraph.

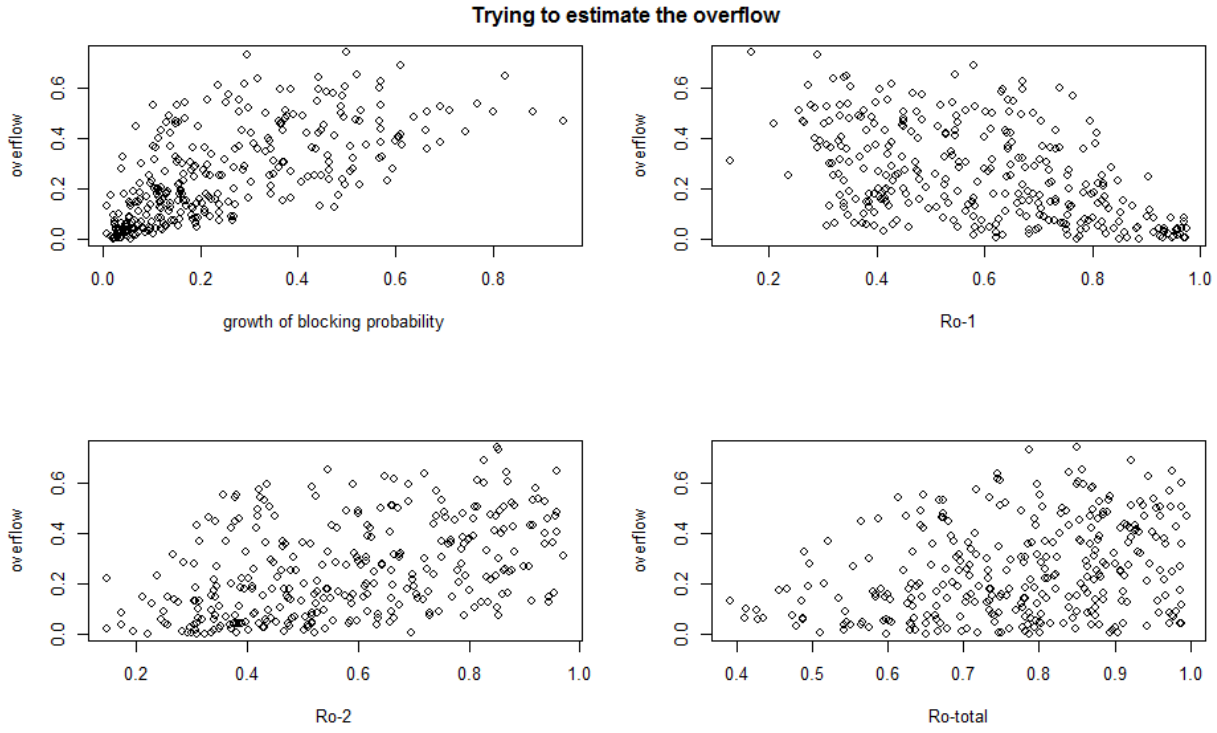


Figure 14: Testing the overflow p on the y-axis against several variables

A big drawback of this model is that the mean waiting time in case of delay can only grow because of the overflow. However, as shown before, in a significant part of the cases this waiting time was actually smaller than in the system without any overflow. This shows that the assumption of a Poisson overflow process might not be a realistic assumption. The overflow could still be Poisson distributed a fraction of the time, and thus be state dependent. Especially when the service rate μ_2 is very large compared to μ_1 customers of type 1 delay more often for a shorter time.

5. Results

The results from “Estimating probability of delay” leads to the following plot, when applied to the data:

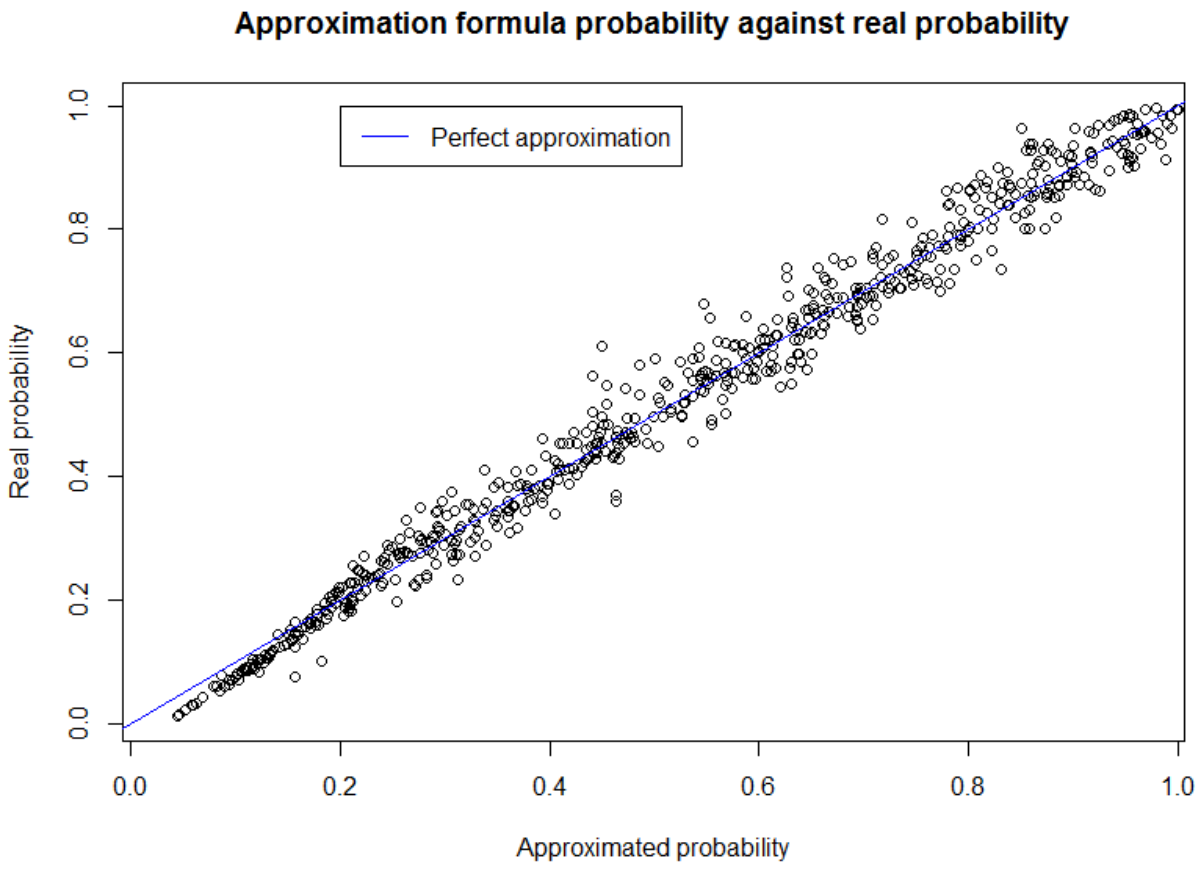


Figure 15: Final approximation visualized

In general, the approximation for the probability of delay works quite well. The r^2 (coefficient of determination) for this model is 0.982. However there are still some points that are off by quite a big margin, the biggest being 0.16, this seems to happen most of the time when the approximated probability is between 0.4 and 0.6. It seems that part of this happens because of the decision tree. In general decision trees work very well on linear decision boundaries, and not so well on gradually increasing functions. However, when looking more closely at figure 10, it does look more like a function that increases gradually, especially in the ρ_t direction. For a follow up research, it would probably be best to estimate the surface by a better function than a decision tree.

Unfortunately no really good prediction could be given for the mean waiting in case of a delay. What was tried was to model and then estimate p in terms of parameters. However as shown in Figure 14, it is unclear what parameters really influence the overflow p. This overflow model probably does not work because the arrival rate by overflow is considered to be a Poisson process, however they (probably) arrive with significantly more variance. This approximation especially does not work when $\mu_2 > \mu_1$. The 10% of the cases where the overflow was “negative” all had $\mu_2 > \mu_1$

6. Conclusion and recommendations

The research question was the following: “Is it possible to create KPI approximations for a queueing network by fitting functions on simulation data with techniques from data mining?” This paper shows that it is at least possible in some cases. The probability of waiting can well be approximated by a decomposition of two Erlang C models in addition to a refinement using a decision tree. The mean waiting turns out to be hard to approximate.

There are also several drawbacks of a data analysis approach:

1. The amount of time spent fitting functions and analysis was at least a factor 20 times bigger as the time it took to build and run the simulation. It is also difficult to do it without any knowledge of queueing theory. The form of these functions is usually a sum over lots of different states the system can be in and using only data analysis techniques these sums are hard to find. Maybe genetic algorithms would work better here, however it is recommended to start such an approach with less complex systems.
2. The accuracy of a simulation will, given enough simulation time, beat the accuracy of an approximation function on such a simulation. Even when the standard error on the estimation is known to be small.
3. It is unclear how well the approximation formulas generalize for cases outside the simulation boundaries as this has not been tested. Upper and lower boundaries on the KPI’s can be given if they fit within the simulation boundaries. However it’s hard to give these boundaries when the systems fall out of the simulation boundaries. For example it is unsure how well the function found in this paper works if the number of flexible servers is a bigger than 4, or what happens when the utilization is bigger or smaller than the utilization in the simulation.

This is why we would only recommend the approximation approach from this paper in certain specific cases. An example of such a case is when the networks or queues approximated are recurring very often and the influence of some parameters needs to be estimated. It is also useful to know approximation formulas for optimization purposes.

Maybe that the ‘overflow model’ could actually be used under certain circumstances, as in 90% of the cases there actually was a positive overflow p. However it would be important to find out which parameter settings this would actually work for and for which settings it would not.

Would we use a data analysis approach again, we would probably not use random settings for the simulation. It’s much easier to estimate the influence of these settings by fixing all parameters except for one parameter.

7. Literature

[1] Kingman, J. F. C. (1961). *"The single server queue in heavy traffic".* Mathematical Proceedings of the Cambridge Philosophical Society 57 (4): 902-904

[2] Shanthikumar, J. G., and J. A. Buzacott. (1980) *"On the approximations to the single server queue."* International Journal of Production Research 18.6: 761-773.

[3] Whitt, W. (1993) *"Approximations for the GI/G/m queue."* Production and Operations Management 2.2: 114-161.

[4] Halfin, S, and Whitt, W. (1981) *"Heavy-traffic limits for queues with many exponential servers."* Operations research 29.3: 567-588.

[5] Koole, G. (2010) *“Optimization of Business Processes”*

[6] Image from *https://en.wikipedia.org/wiki/Gaussian_function*

[7] Lee, A. M. and Longton, P. A. (1959). *"Queueing Processes Associated with Airline Passenger Check-in".* Journal of the Operational Research Society 10: 56-71