Capacity planning in a call center

Meyfroyt, T.M.M.

Award date:
2011

Bachelorproject:
Capacity planning in a Call Center

Thomas Meyfroyt, 0649268

Bachelor Technische Wiskunde
TU Eindhoven

December 22, 2011

Contents

1 Introduction **3**

2 Problem Description **3**

3 The queueing model **5**
3.1 Assumptions about the model 5
3.2 M/M/c 5
3.3 M/M/c with abandonments 8
3.4 M/M/c with abandonments and call blending 11

4 Results of the queueing model **15**
4.1 Sensitivity analysis 17

5 Schedule generation **22**

6 Scheduling results **24**

7 Conclusion **27**

A [Table of incoming call rates during the week] **29**

B [Table of number of needed agents and threshold] **30**

C [Numerical results of determining c and c^*] **31**

D [Mathematica script] **34**

1 Introduction

In this document the bachelorproject "Capacity planning in a call center" will be discussed. The goal of the project is to make a workforce schedule for a call center in such a way that certain performance targets will be met. These performance targets deal with the number of abandoned calls and the waiting time of customers. An important aspect of the call center in question is that besides having to serve incoming calls, also outbound calls have to be made by the agents. Two different policies for dealing with these outbound calls are compared. The first policy assumes that whenever an outbound call has to be made, that this is done immediately. The second policy is the so called call blending policy, where an outbound call is made by an agent only when a predetermined number of agents is idle. Both policies are modeled by an extension of the widely used $M/M/c$ queueing model. With the information about the number of needed agents that is gotten from the queueing models a workforce schedule is made. This is done by solving a mixed integer program. We find that the call blending policy is an efficient way to make efficient schedules for call centers that have to deal with outgoing calls.

In the next chapter we will first formally describe the problem. Then a queueing model based on the widely used $M/M/c$ queueing model will be developed that is applicable to our call center. This will be followed by some results of the queueing model and a sensitivity analysis of the used parameters. After that a mixed integer program will be discussed that we will use to make the actual workforce schedule. Finally some results of this MIP will be presented and a conclusion is given.

2 Problem Description

As mentioned in the introduction, the goal of the project is to make a workforce schedule for a call center in such a way that certain performance targets will be met. The call center we will be looking at serves as a helpdesk. People call the center with a question and an agent tries to answer this question. If it is a simple question this can be done immediately. For more difficult questions the agent might have to make some calls first or look some things up before he can answer the question. There is no difference in the skills of the agents, they are all capable of answering all questions. Because only a fixed number of agents are working at a given time, they can only serve a fixed number of calls at a time. This means certain callers have to wait in a queue until an agent becomes free, before their call is answered. This queue has infinite capacity, but customers have finite patience and will abandon the system if they are put into the queue or are waiting in the queue for too long.
Of course the call center wants to deliver good service to their customers, which means minimizing their time in the system. This includes queueing time and service time. This can easily be done by always having a lot of agents scheduled, but this is very costly for the call center. That is why a balance needs to be found between the performance of the call center and the number of agents scheduled to work at a certain time. The performance of the call center is measured by so called performance measures. The following performance measures apply to the call center we will be looking at:

- The percentage of abandoned calls should be smaller than 1.5%.

- The number of calls served within 25 seconds should be bigger than 95%.

- The average speed of answer should be smaller than 10 seconds.

We split the problem of making the workforce schedule into three parts:

1. The first part is to make a mathematical model of the call center which has all the relevant aspects in it. It is clear that a queueing model can and should be used here.

2. Secondly we then need to use this model to calculate how many agents we need at each moment of our planning horizon.

3. Lastly we need to use these calculations to make the actual workforce schedule so that all the performance targets will be met. For this part we need to keep in mind how many agents the call center has available at a given time and also that agents work in shifts. This is the scheduling aspect of the project.

Each of these aspects will be discussed in their own chapter. The goal is to eventually create a workforce schedule for a typical workday of our call center. We will start by making a queueing model that is applicable to our call center.

3 The queueing model

3.1 Assumptions about the model

A lot has already been written about call center modeling and mathematics. A model that is widely used to model how a call center functions is the so called $M/M/c$ queueing model. In [3] for example, Koole discusses this model and how call center managers can and should use it. We will use the $M/M/c$ queueing model as a basis for our queueing model. The $M/M/c$ queueing model assumes that the following statements apply to our call center:

1. The arrival process (incoming calls) is *Markovian*. This means that incoming calls follow a Poisson process and we also only have one type of calls.
2. The service time distribution is *Markovian*. This means that we have exponential service times.
3. There are c parallel working servers (agents).
4. The queue has infinite capacity.
5. Customers can only leave the system when they have been serviced. In other words, there are no abandonments.

The assumptions of Markovian distributed interarrival and service times are in general relatively good assumptions for a call center. Of course this model does not completely fit our call center, but it serves as a good start. We need to adjust this model so it also includes the following two aspects of our call center:

- Customers that are put into the queue upon arrival have a probability of abandoning the system immediately or after they have spent some time in the queue.
- The call center also has to process outgoing calls, like answers to earlier received questions from a customer.

Under certain assumptions about these aspects, the $M/M/c$ model can be nicely extended to include them as shown in [2]. But let us first take a closer look at the $M/M/c$ model.

3.2 M/M/c

We shall now focus on mathematically describing the $M/M/c$ model and fully understanding it, so we can later extend the model in a way such that it fits our call center. In this chapter we will use the same structure as is used for the chapters of [5] to explain and understand our models. Like said earlier we assume that the arrival process is Markovian and let us denote the arrival rate by λ. For our call center this λ shall denote the rate at which it receives incoming calls. Furthermore we also assume that the service time distribution is Markovian, we shall denote the service rate by μ. In our model this μ will be the rate at which agents finish incoming calls. We shall denote the number of servers (or agents) by c. Note that to avoid that our queue will grow to infinity, we require that the amount of work per agent per time unit is smaller than the amount of work an agent can process per time unit. So we require $\frac{\lambda}{c} < \mu$, which gives the stability condition for an $M/M/c$ queue: $\frac{\lambda}{c\mu} < 1$.
A way to now describe our system at some time t is to count the number of customers in

the system. We will denote this state of our system by the state variable $X(t)$. So $X(t)$ is the total number of inbound calls in the system at time t. Under the made assumptions $\{X(t); t \geq 0\}$ is a so called continuous-time Markov chain (CTMC), with state space $S = \{0, 1, ...\}$. Furthermore it is also a birth-death process because our state variable can only change by 1 with each transition. This allows us to look at state-dependent birth and death rates, from which we can calculate important quantities of our system. If $X(t) = k$, then the birth rates λ_k and death rates μ_k are state-dependent as follows:

$$\lambda_k = \lambda, \quad k=0,\ 1,\ 2,\ ...$$

$$\mu_k = \begin{cases} k\mu, & k=0,\ 1,\ ...,\ c-1 \\ \\ c\mu, & k=c,\ c+1,\ ... \end{cases}$$

The flow diagram of Figure 1 shows this.

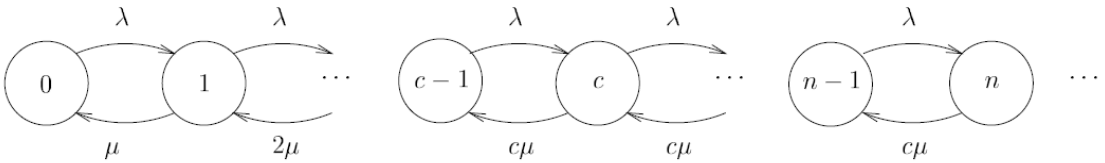

Figure 1: Flow diagram of an M/M/c model.

Now let us define π_n as the so called equilibrium probability that there are n customers in the system. This is the fraction of the time the system will be in state n on the long run i.e. $\pi_n = \lim_{t\to\infty} \mathbb{P}(X(t) = n)$. We can calculate these probabilities by equating the flow between two neighboring states, because for the system to be in equilibrium the net flow between two states should be zero. So we have:

$$\lambda\pi_{n-1} = \min(n, c)\mu\pi_n.$$

If we then define $\rho = \frac{\lambda}{c\mu}$, iterating gives us:

$$\begin{aligned} \pi_n &= \tfrac{(c\rho)^n}{n!}\pi_0, & n=0,1,...,c \\ \pi_{c+n} = \pi_c\rho^n &= \rho^n \tfrac{(c\rho)^c}{c!}\pi_0 & n=0,1,... \end{aligned}$$

Finally we can use the fact that the sum of all our equilibrium probabilities should be 1. This gives us:

$$\pi_0 = (\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!} \cdot \frac{1}{1-\rho})^{-1}.$$

These equilibrium probabilities are important because they allow us to calculate useful quantities like the probability an arriving customer is put into the queue, the mean queue length, the average waiting time and even the distribution of the waiting time. Let us first look at the probability an arriving customer is put into the queue and call it Π_w. Then from the PASTA property, which states that the probability of the system state as seen by an outside

random observer is the same as the probability of the state seen by an arriving customer, it is clear that:

$$\Pi_w = \sum_{n=c}^{\infty} \pi_n = \sum_{k=0}^{\infty} \rho^k \frac{(c\rho)^c}{c!}\pi_0 = \frac{(c\rho)^c}{c!}\pi_0 \sum_{k=0}^{\infty} \rho^k = \frac{\pi_c}{1-\rho}$$

Calculating the mean number of waiting customers is not difficult either:

$$\mathbb{E}(L) = \sum_{n=0}^{\infty} n\pi_{c+n} = \pi_c \sum_{n=0}^{\infty} n\rho^n = \pi_c \frac{\rho}{(1-\rho)^2} = \Pi_w \cdot \frac{\rho}{1-\rho}$$

With this quantity we can easily calculate the mean waiting time with Little's Law:

$$\mathbb{E}(W) = \Pi_w \cdot \frac{1}{1-\rho} \cdot \frac{1}{c\mu}$$

Calculating the waiting time distribution is a bit more difficult. Let us denote the waiting time in the queue experienced by the l'th customer by $W(l)$. Also let us denote the state the l'th customer finds the system in by $X'(l)$. We shall calculate the following limiting distribution:

$$\mathbb{P}(W > t) = \lim_{l\to\infty} \mathbb{P}(W(l) > t).$$

If we condition on the system state $X'(l)$ and use the PASTA property, we get[2]:

$$\mathbb{P}(W > t) = \lim_{l\to\infty} \sum_{n=0}^{\infty} \mathbb{P}(W(l) > t | X'(l) = c+n)\mathbb{P}(X(l) = c+n) = \sum_{n=0}^{\infty} \mathbb{P}(W > t | X' = c+n)\pi_{c+n}.$$

Note that the conditional probabilities in the expression above do not depend on l because the process $X'(\cdot)$ is Markovian and stationary. Accordingly we simplified the notation by dropping "l" with the understanding that conditioning is with respect to the stationary, random system state. Furthermore because the service times are independent, the stochastic quantities $(W|X' = c+n)$ follow an Erlang$(n+1, c\mu)$ distribution, because if a customer finds n people waiting in the queue upon arrival, he will have to wait until $n+1$ customers have received complete service before he can receive service. So if we use the CDF of an Erlang distribution we get:

$$\mathbb{P}(W > t) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} \frac{(c\mu t)^k}{k!} e^{-c\mu t}\pi_{c+n} = \sum_{k=0}^{\infty}\sum_{n=k}^{\infty} \frac{(c\mu t)^k}{k!} e^{-c\mu t}\rho^n\pi_c = \frac{\pi_c}{1-\rho}\sum_{k=0}^{\infty} \frac{(c\mu\rho t)^k}{k!} e^{-c\mu t} = \Pi_w e^{-c\mu(1-\rho)t}.$$

This yields for the following conditional probability:

$$\mathbb{P}(W > t | W > 0) = \frac{\mathbb{P}(W > t)}{\mathbb{P}(W > 0)} = \frac{\mathbb{P}(W > t)}{\Pi_w} = e^{-c\mu(1-\rho)t}.$$

Hence, the conditional waiting time $W|W > 0$ is exponentially distributed with parameter $c\mu(1-\rho)$.

These calculations have helped us to understand the $M/M/c$ model. We shall now try to extend the model in such a way that it includes customers abandoning the system.

3.3 M/M/c with abandonments

In the model we currently have all customers have infinite patience. This means that every customer that is put into the queue, will stay in the queue until he/she is served. Of course this is not a realistic assumption. Some customers do not want to wait at all and abandon the system immediately when they are put into a queue. Others will stay in the queue for some time until they are fed up with waiting and then they will leave the queue. We will now try to include this kind of behavior in our model. Let us assume that customers that arrive in the system when all agents are busy leave the system immediately with probability $(1-\gamma)$. Secondly because of the Markovian nature of the model it might be practical to assume that the time people are willing to wait in the queue also follows an exponential distribution. When compared to reality this also often seems to be a relatively good assumption [1]. So we will assume that these patience times are exponentially distributed with mean η^{-1}. Under these assumptions $\{X(t); t \geq 0\}$ is still a continuous-time Markov chain. Also it is still a birth-death process, so we can again look at the state-dependent transition rates:

$$\lambda_k = \begin{cases} \lambda, & k=0,\, 1,\, ...,\, c-1 \\ \\ \gamma\lambda, & k=c,\, c+1,\, ... \end{cases}$$

$$\mu_k = \begin{cases} k\mu, & k=0,\, 1,\, ...,\, c-1 \\ \\ c\mu + (k-c)\eta, & k=c,\, c+1,\, ... \end{cases}$$

The flow diagram of Figure 2 shows this.

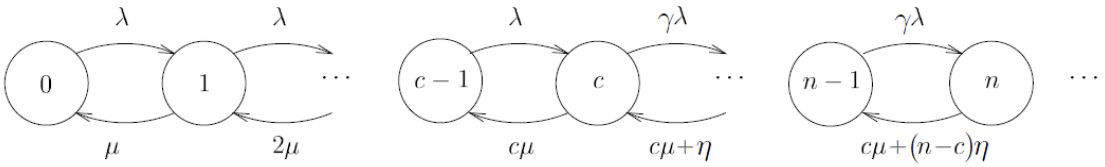

Figure 2: Flow diagram of an M/M/c model with abandonments.

Again we will determine equilibrium probabilities by equating the flows between two neighbors:

$$\lambda\pi_{n-1} = n\mu\pi_n, \quad n=0,\, 1,\, ...,\, c$$

$$\gamma\lambda\pi_{c+n-1} = (c\mu + n\eta)\pi_{c+n}, \quad n=1,\, 2,\, ...$$

Again using $\rho = \frac{\lambda}{c\mu}$, iterating gives us:

$$\pi_n = \begin{cases} [\sum_{k=0}^{\infty} \theta_k]^{-1}, & n=0 \\ \\ \theta_n\pi_0, & n=1,\, 2,\, ... \end{cases}$$

where the θ_n are as follows:

$$\theta_n = \begin{cases} \frac{(c\rho)^n}{n!}, & n=0,1,...,c \\ \\ \frac{(c\rho)^c}{c!} \cdot \prod_{k=1}^{n-c}(\frac{\gamma\lambda}{c\mu+k\eta}), & n=c+1,\ c+2,\ ... \end{cases}$$

Notice that by this small extension of the $M/M/c$ model, our expressions for the equilibrium probabilities have become more complex. This also means that expressions for quantities like the mean queue length will be more complex. But calculating these should not be a problem for a computer though so let us proceed. We will again start with the probability that an arriving customer has to wait:

$$\Pi_w = \sum_{n=c}^{\infty} \pi_n = \sum_{n=c}^{\infty} \theta_n \pi_0 = \pi_c(1 + \sum_{n=1}^{\infty} \prod_{k=1}^{n} [\frac{\gamma\lambda}{c\mu + k\eta}])$$

The mean queue length is given by the following expression:

$$\mathbb{E}(L) = \sum_{n=0}^{\infty} n\pi_{c+n} = \pi_c(\sum_{n=1}^{\infty} n \prod_{k=1}^{n} [\frac{\gamma\lambda}{c\mu + k\eta}])$$

We will now try to find the waiting time distribution in the same way as we did in the last section. However because now customers may leave the system by either abandonment or service we will make a distinction in waiting times. Let us by W^a denote the waiting time in the queue experienced by a customer which may leave the system by either abandonment or service. And let us by W^s denote the waiting time in the queue experienced by a customer which will surely receive service. Like before by $X'(l)$ we denote the state the l'th customer finds the system in when entering the system. Note that because $e^{-\eta t}$ is the probability that the patience of a customer is at least t, we have the following relationship between W^a and W^s:

$$\mathbb{P}(W^a > t) = e^{-\eta t}\mathbb{P}(W^s > t).$$

Now let us denote the waiting time in the queue experienced by the l'th customer by $W^a(l)$ (which may be terminated by either abandonement or service). We calculate the following limiting distribution:

$$\mathbb{P}(W^a > t) = \lim_{l\to\infty} \mathbb{P}\{W^a(l) > t\}.$$

If we condition on the system state $X'(l)$ like before and use the PASTA property, we get:

$$\mathbb{P}(W^a > t) = \lim_{l\to\infty} \sum_{n=0}^{\infty} \mathbb{P}\{W^a(l) > t | X'(l) = c+n\}\mathbb{P}\{X'(l) = c+n\} = \sum_{n=0}^{\infty} \mathbb{P}\{W^a > t | X' = c+n\}\pi_{c+n}.$$

This time the conditional probabilities are more difficult, because we have to take into account that people abandon the queue now. The following expression for these probabilties is given by Riordan (1962) [4]:

$$\mathbb{P}(W^a > t | X' = c+n) = e^{-\eta t(1+\psi)} \sum_{k=0}^{n} \frac{(\psi)_k(1-e^{-\eta t})^k}{k!}, \quad n \geq 0$$

where $\psi = c\mu/\eta, (\psi)_0 = 1$ *and* $(\psi)_k = (\psi)(\psi+1)\cdots(\psi+k-1)$ *for* $k \geq 1$.

This is how he derived it. Let us denote the conditional probabilities we are interested in by $w_n^a(t)$, that is the probability that the waiting time of a customer which on arrival finds n other customers waiting is at least t. Note again that this waiting time may be terminated either by service or by abandonment. Write $w_n^s(t)$ for the similar probability of a customer which is sure not to abandon the system. Then again the following relation holds:

$$w_n^a(t) = e^{-\eta t} w_n^s(t).$$

Now let us look at these $w_n^s(t)$ more closely by considering eventualities in an infinitesimal small interval δ. One out of three things can happen:

1. Nobody leaves the system. This can happen with probability $e^{-(n\eta + c\mu)\delta} = 1 - (n\eta + c\mu)\delta + O(\delta^2)$.
2. A customer has received service and leaves the system. This can happen with probability $1 - e^{-c\mu\delta} = c\mu\delta + O(\delta^2)$.
3. A customer is tired of waiting and abandons the system. This can happen with probability $1 - e^{-n\eta\delta} = n\eta\delta + O(\delta^2)$.

Note that we do not take into account the events that more than one customer leaves, because these are very unlikely ($=O(\delta^2)$) during our small time interval. We now have the following relation:

$$w_n^s(t) = (1 - (n\eta + c\mu)\delta) w_n^s(t - \delta) + (n\eta + c\mu) w_{n-1}^s(t - \delta) + O(\delta^2).$$

Moving $w_n^s(t - \delta)$ to the lefthand side, dividing by δ and letting δ go to zero gives us the following differential recurrence relation:

$$(w_n^s)'(t) = (n\eta + c\mu)[w_{n-1}^s(t) - w_n^s(t)].$$

We can solve this recurrence iteratively. We start by looking at $w_0^s(t)$:

$$(w_0^s)'(t) = (c\mu)[-w_0^s(t)].$$

Which gives us $w_0^s(t) = e^{-c\mu t}$, since $w_n^s(0) = 1$ for all n. With this we have a differential equation for $w_1^s(t)$:

$$(w_1^s)'(t) = (c\mu + \eta)[w_0^s(t) - w_1^s(t)].$$

This is an inhomogeneous first-order linear equation, which we can easily solve. We start by finding the general solution $w_H(t)$ to the homogeneous differential equation:

$$(w_H)'(t) = -(c\mu + \eta)[w_H(t)].$$

This gives us $w_H(t) = Ae^{-(c\mu+\eta)t}$, where A is a constant. We now need to find a solution $w_P(t)$ to the inhomogeneous differential equation:

$$(w_P)'(t) + (c\mu + \eta) w_P(t) = (c\mu + \eta) e^{-c\mu t}.$$

One might try a solution of the form $w_P(t) = Be^{-c\mu t}$ to find that choosing $B = \frac{c\mu+\eta}{\eta}$ gives a solution. Combining these solutions gives us the complete solution:

$$w_1^s(t) = w_H(t) + w_P(t) = Ae^{-(c\mu+\eta)t} + \frac{c\mu + \eta}{\eta} e^{-c\mu t}.$$

Finally using the fact that $w_1^s(0) = 1$ gives us $A = -\frac{c\mu}{\eta}$. And by setting $\psi = \frac{c\mu}{\eta}$ we can write the solutions we are interested in as follows, which are easily verified by the expression given by Riordan:

$$w_1^s(t) = -\psi e^{-\eta t(1+\psi)} + (1+\psi)e^{-\eta\psi t}.$$
$$w_1^a(t) = e^{-\eta t} w_1^s(t) = -\psi e^{-\eta t(2+\psi)} + (1+\psi)e^{-\eta t(1+\psi)}.$$

Using $w_1^s(t)$ we can now find a differential equation for $w_2^s(t)$, which again will be an inhomogeneous first-order linear equation which can be easily solved. Repeating this process of recursively solving differential equations leads to the expression given by Riordan. Putting all this together gives us:

$$\mathbb{P}(W^a > t) = e^{-\eta t(1+\psi)} \sum_{n=0}^{\infty} \pi_{c+n} \sum_{k=0}^{n} \frac{(\psi)_k (1-e^{-\eta t})^k}{k!}.$$

$$\mathbb{P}(W^s > t) = e^{-\eta\psi t} \sum_{n=0}^{\infty} \pi_{c+n} \sum_{k=0}^{n} \frac{(\psi)_k (1-e^{-\eta t})^k}{k!}.$$

where $\psi = c\mu/\eta, (\psi)_0 = 1$ *and* $(\psi)_k = (\psi)(\psi+1)\cdots(\psi+k-1)$ *for* $k \geq 1$.

We have now successfully extended our model to include abandonments. As mentioned earlier the call center also has to process outgoing calls, like answers to earlier received questions from a customer. So our task now is to extend our model even further so that it also includes this aspect of our call center.

3.4 M/M/c with abandonments and call blending

Before we can include outgoing calls in our model we need to decide how our call center will handle these calls. A way to deal with the outgoing calls could be by assigning a fixed number of agents to only the outgoing calls and the rest to the incoming calls. It can be shown though ([3]) that a better way to deal with incoming and outgoing calls is by so called *call blending*. Instead of assigning a fixed number of agents to ingoing or outgoing calls, they are assigned dynamically. A free agent should obviously be assigned to a waiting incoming call if any are present. A way to maximize productivity is by assigning free agents to outgoing calls if there are no waiting incoming calls. However, then every incoming call has to wait for an agent to finish their call. The solution is to keep a number of agents free for incoming calls when none are waiting. This is also known as a *threshold-type* policy. Let us try to include this policy into our model. We will denote our threshold by c^*. This means that an agent will make an outgoing call if there are at most c^* agents busy. We will assume that an agent will immediately make this outgoing call as soon as the system state reaches c^*. This implies that $X(t) > c^*$ for every t. Note that we are now assuming that there are always outgoing calls that can be made. Furthermore let us assume that outbound calls are served with rate μ_o (and let us from now on call the service rate for inbound calls μ_i). Note that by including these outbound calls in our model the state variable $X(t)$, which is the total number of inbound and outbound calls in the system at time t, does not completely describe our system anymore, because inbound and outbound calls have different service rates. A way to solve this would be to make a two-dimensional state-space, but this will make the calculation of important

quantities extremely difficult. An other intuitive solution would be to use a mean service time μ^{-1} for all calls. This would be a weighted average of μ_i^{-1} and μ_o^{-1}:

$$\frac{1}{\mu} = \frac{p}{\mu_i} + \frac{(1-p)}{\mu_o}$$

Here p is the long-run proportion of calls that are inbound and receive service. If we use this effective service rate μ the state variable $X(t)$ again fully describes our system and we have a one-dimensional state space. As we shall see however the fraction p depends on μ in this system and is unknown. To be able to calculate this fraction p and the effective service rate μ we have to define the following three quantities. Define R^a to be the rate at which inbound calls are lost due to abandonment. We then have:

$$R^a = \lambda(1-\gamma)\sum_{n=c}^{\infty}\pi_n + \eta\sum_{n=c}^{\infty}(n-c)\pi_n.$$

Secondly define R^s to be the steady-state rate of calls (inbound and outbound) served. Then it is easy to understand that:

$$R^s = \sum_{n=0}^{\infty}\min(n,c)\mu\pi_n.$$

Lastly define R^o to the steady-state rate of outbound calls served. Note that the rate of inbound calls accepted into the system that are served is $\lambda - R^a$. So we get:

$$R^o = R^s - (\lambda - R^a).$$

So the long-run proportion of calls that are inbound and served is given by:

$$p(\mu) = 1 - \frac{R^o}{R^s}.$$

This depends on μ because the steady state probabilities π_n depend on μ and they apppear in the expressions for R^a and R^s. The exact expressions for the probabilities π_n for this system will follow shortly.
We can now conclude that finding our effective service rate μ is equivalent to finding a root of the following function:

$$h(\mu) = \frac{p(\mu)}{\mu_i} + \frac{(1-p(\mu))}{\mu_o} - \frac{1}{\mu}.$$

From the continuity of R^o and R^s it follows that the function $h(\cdot)$ is also continuous. And because $0 < p(\mu_i) < 1$ and also $0 < p(\mu_o) < 1$ we have that $h(\min(\mu_i,\mu_o)) < 0 < h(\max(\mu_i,\mu_o))$. So we know that $h(\mu)$ has at least one root in $(\min(\mu_i,\mu_o), \max(\mu_i,\mu_o))$. Finding a root can be done with root-bracketing methods as is discussed in [2]. For this project the software package Mathematica was used for finding a root. Figure 3 shows a typical plot of the function $h(\mu)$ in the interval $(\min(\mu_i,\mu_o), \max(\mu_i,\mu_o))$

Like mentioned earlier, in order for us to be able to find this root we first need to define what the π_n exactly are for this system, since they apppear in the expressions for R^a and R^s. If

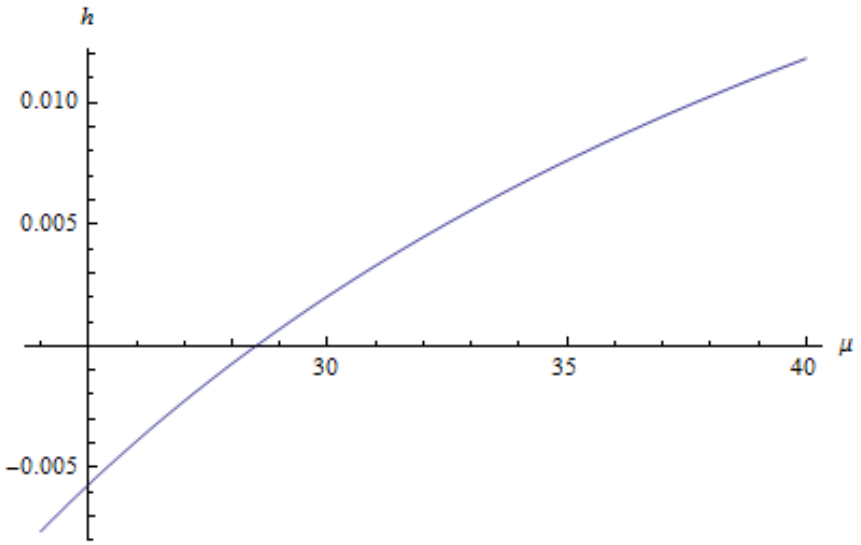

Figure 3: A typical plot of h in $(\min(\mu_i, \mu_o), \max(\mu_i, \mu_o))$.

$X(t) = k$, then the birth rates λ_k and death rates μ_k are state-dependent as follows (again note that because of our treshhold-type policy we have $X(t) > c^*$ for all t):

$$\lambda_k = \begin{cases} \lambda, & k{=}c^* + 1,...,c - 1 \\ \gamma\lambda, & \text{otherwise} \end{cases}$$

$$\mu_k = \begin{cases} k\mu, & k{=}c^* + 2,\ c^* + 3,...,c - 1 \\ c\mu + (k - c)\eta, & k{=}c,\ c + 1,\ ... \end{cases}$$

The flow diagram of Figure 4 shows this.

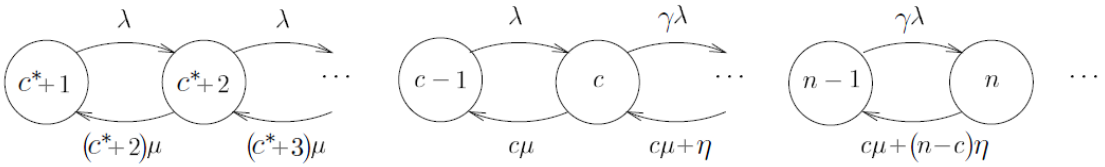

Figure 4: Flow diagram of the M/M/c model with abandonments and call blending.

Again we will determine equilibrium probabilities by equating the flows between two neighbors:

$$\lambda\pi_{n-1} = n\mu\pi_n, \quad n{=}c^* + 2,\ ...,\ c$$

$$\gamma\lambda\pi_{c+n-1} = (c\mu + n\eta)\pi_{c+n}, \quad n{=}1,\ 2,\ ...$$

Iterating gives us:

$$\pi_n = \begin{cases} [\sum_{k=c^*+1}^{\infty} \theta_k]^{-1}, & n = c^*+1 \\ \\ \theta_n \pi_{c^*+1}, & n = c^*+2, c^*+3, ... \end{cases}$$

where the θ_n are as follows:

$$\theta_n = \begin{cases} 1, & n{=}c^*+1. \\ \\ \frac{(c^*+1)!}{n!}(\frac{\lambda}{\mu})^{n-(c^*+1)}, & n{=}c^*+2,...,c-1 \\ \\ \frac{(c^*+1)!}{c!}(\frac{\lambda}{\mu})^{c-(c^*+1)} \cdot \prod_{k=1}^{n-c}(\frac{\gamma\lambda}{c\mu+k\eta}), & \text{otherwise} \end{cases}$$

These equilibrium probabilities again allow us to calculate some important quantities like before. Note that they are very similar in appearence compared to the previous section. We will again start with the probability that an arriving customer is put into the queue:

$$\Pi_w = \sum_{n=c}^{\infty} \pi_n = \sum_{n=c}^{\infty} \theta_n \pi_{c^*+1} = \pi_c (1 + \sum_{n=1}^{\infty} \prod_{k=1}^{n} [\frac{\gamma\lambda}{c\mu + k\eta}])$$

The mean queue length is now given by the following expression:

$$\mathbb{E}(L) = \sum_{n=0}^{\infty} n\pi_{c+n} = \pi_c (\sum_{n=1}^{\infty} \prod_{k=1}^{n} [\frac{n\gamma\lambda}{c\mu + k\eta}])$$

The deriviation of the waiting time distribution can be done in exactly the same way as in the previous section. So we again we have:

$$\mathbb{P}(W^a > t) = e^{-\eta t(1+\psi)} \sum_{n=0}^{\infty} \pi_{c+n} \sum_{k=0}^{n} \frac{(\psi)_k (1 - e^{-\eta t})^k}{k!}.$$

$$\mathbb{P}(W^s > t) = e^{-\eta \psi t} \sum_{n=0}^{\infty} \pi_{c+n} \sum_{k=0}^{n} \frac{(\psi)_k (1 - e^{-\eta t})^k}{k!}.$$

where $\psi = c\mu/\eta, (\psi)_0 = 1$ *and* $(\psi)_k = (\psi)(\psi+1)\cdots(\psi+k-1)$ *for* $k \geq 1$.
This completes our model.

An alternative policy: Outgoing call directly after an incoming call

In the last section we extended our model so that it includes call blending. One might wonder how well call blending actually works in comparison to a different policy for dealing with outgoing calls. For this reason we introduce a different policy. Whenever an agent finishes an incoming call which needs an outgoing call, he will immediatelly make this call. We will model this policy by using the model from the section *M/M/c with abandonments* with a service rate $\mu = (\mu_i^{-1} + a \cdot \mu_o^{-1})^{-1}$ where a is the average amount of outgoing calls an incoming call generates. Note that by doing so we are appromixating the sum of exponential distributed service times of an incoming call and a stochastic number of outgoing calls by a single exponential distributed service time with mean $\mu_i^{-1} + a \cdot \mu_o^{-1}$, where a is the expected number of outgoing calls per incoming call.

4 Results of the queueing model

In the previous chapter we created a mathematical model of the call center we are interested in. We should now be able to calculate, given an inbound call rate λ, how many agents we need to schedule and what threshold we should take, so we can reach our performance targets as cheap as possible. Remember that we are interested in the following performance measures:

- The percentage of abandoned calls.

- The percentage of calls served within 25 seconds.

- The average speed of answer.

We already know from the last chapter that the rate at which inbound calls are lost due to abandonment, R^a is as follows:

$$R^a = \lambda(1-\gamma)\sum_{n=c}^{\infty}\pi_n + \eta\sum_{n=c}^{\infty}(n-c)\pi_n.$$

And so the percentage of abandoned calls is given by: $\frac{R^a}{\lambda}\cdot 100\%$.
For the other two performance measures we can use the waiting time distribution $\mathbb{P}(W^s > t)$ we derived earlier. We use the distribution of W^s and not the distribution of W^a, because about the customers that abandon the queue we can not really say much with respect to the last two performance measures. The percentage of calls served within 25 seconds is given by the following formula:

$$\text{The percentage of calls served within 25 seconds} = (1 - \mathbb{P}(W^s > 25 \text{ seconds}))\cdot 100\%$$

The average speed of answer is simply the expectation of W^s:

$$\text{The average speed of answer} = \mathbb{E}(W^s) = \int_0^{\infty}\mathbb{P}(W^s > t)\mathrm{d}t.$$

There is another performance measure we should look at though. In the last chapter we introduced an effective service rate μ. Here μ^{-1} was a weighted average of μ_i and μ_o as follows:

$$\frac{1}{\mu} = \frac{p}{\mu_i} + \frac{(1-p)}{\mu_o}$$

Where p was the long-run proportion of calls that are inbound and receive service. This μ could be calculated for a given c and c^*, which would also give us this proportion p. In the case of our call center, each incoming call on average generates $\frac{5}{4}$ outgoing calls. So in order to be sure that the call center is able to make all these calls we require $p < \frac{4}{9}$.

With these formulas we should now be able to determine the minimum number of agents required at a given time during a typical workday of our call center. We will determine this number c for each individual interval of 30 minutes during the day and denote it by s_t $(t = 1, ..., 48)$. Also the threshold c^* that is to be used should be calculated. Before we can do this however we will first need data about the inbound call rate. This rate is given for each interval of 30 minutes during the day and can be found in Appendix A. Furthermore we need estimators for μ_i, μ_o, η and γ. We shall assume that these parameters do not change

during the course of a day. For the call center that we are looking at incoming calls have an average duration of 2.5 minutes. So we have $\mu_i = 24$ calls per hour. Outgoing calls have an average duration of 1.5 minutes, $\mu_o = 40$ calls per hour. We do not know the values of γ and η though. We will assume that 90 percent of the customers are prepared to wait if necessary, which seems reasonable, so $\gamma = 0.9$. Furthermore we will assume that the average patience of a customer is 3 minutes, so $\eta = 20$ abandonments per hour.
A good way to now determine s_t is to calculate the c and c^* for different values of λ and put them in a table. One can then just look in this table to determine s_t by looking at the value for c that is needed for the inbound call rate during the corresponding time interval. To calculate the c and c^* a value of λ one can naïvely start with $c = 2$ and $c^* = 0$ and keep increasing c and checking all c^* for that c until the performance targets are met. This could probably be done in a smarter way, but it only has to be done once, so it is not of great importance. In Appendix B such a table corresponding to our call center can be found for all integer values of λ between 1 and 100. In Appendix C some numerical results of this process of finding c and c^* for two specific values of λ can be found. In the first case we have $\lambda = 40$ and in the second case we have $\lambda = 80$. For each combination of c and c^* the performance measures are put into a table and have been made bold if the corresponding performance target is met. These results are generated with the Mathematica script that can be found in Appendix D. There are a few things we can learn from these results:
First of all notice that in the case of $\lambda = 80$ the percentage of abandoned calls for a fixed value of c is not always monotonically increasing with c^*. Note that this is only the case for small values of c^*. This also holds for the average speed of answer and the percentage of calls served within 25 seconds. We expect this to be monotonically increasing, because if you increase your threshold you will on average have less people available for incoming calls, which means the average waiting time increases. However, this unexpected behavior might be explained by the use of our effective service rate μ. If we lower our threshold we will do less incoming calls and therefore the effective service rate μ becomes smaller. And apparently for very small threshold values this decrease in μ is more significant than the fact that on average you have more agents available for incoming calls, which causes the expected waiting time to go up instead of down. We can test this theory by doing the same calculations for the case with $\lambda = 80$, but with $\mu_i = \mu_o = 32$. Because in this case we know that the effective service rate $\mu = 32$ for all possible values of c and c^*. The numerical results for this case can also be found in Appendix C. Here we find that the percentage of abandoned calls for a fixed value of c is indeed monotonically increasing with c^*. This is also true for the average speed of answer and the percentage of calls served within 25 seconds.
Secondly we can note that in both cases the performance target of having an average speed of answer smaller than 10 seconds is of low importance compared to the other targets. It is easily met before the other targets are met. The performance targets for the percentage of abandoned calls and the percentage of calls served within 25 seconds appear to be of equal importance.
Furthermore from the tables for the effective service rate we see that the requirement of making enough outgoing calls is the most difficult to meet, especially in the case of $\lambda = 80$. There the first time μ is large enough gives us the values for c and c^* we are looking for, which is at $c = 8$ and $c^* = 5$. The other performance targets are met for the first time at $c = 7$ or $c = 6$.

We can now easily determine the values for s_t with the table we generated. But before we do that one might wonder how important the parameters λ, μ_i, μ_o, γ and η are with respect to

the amount of needed agents. So we will first investigate this.

4.1 Sensitivity analysis

The inbound call rate λ

To investigate the importance of the inbound call rate λ we plot the values of c and c^* are against λ. This plot can be found in Figure 5.

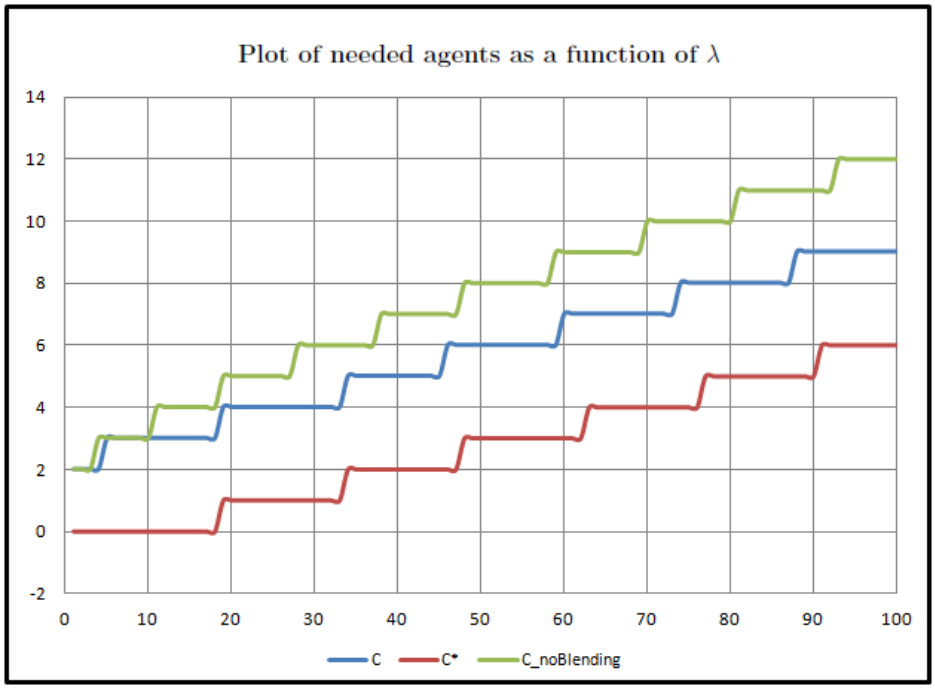

Figure 5: A plot of the needed agents against λ.

Notice that also the amount of agents corresponding with our alternative method without call blending is plot. There are a few things we can learn from this plot. Notice that for low values of λ both methods need the same amount of agents, this is what we expect, because during these non-busy periods there should be enough time to do outgoing calls directly after an incoming call without building up a queue. Furthermore for increasing values of λ the call blending policy seems to be increasingly better than the alternative method. For $\lambda = 100$ the call blending policy needs 3 agents less than the alternative method. Also if we look at the threshold c^* this always seems to be 3 less than the amount of needed agents, with a few exceptions. This could be used as a nice rule of thumb. Also the amount of agents for the call blending method increases by 1 for roughly an increase of 14 calls per hour in the rate λ.

Inbound call service rate μ_i

We will now look at the effects of changing the service rate μ_i. We will do this for two cases. For the first case we set λ to 50, which corresponds to an averagely busy period. For the

second case we set λ to 80, which corresponds to a busy period with many callers. We will vary the average incoming call duration μ_i^{-1} and calculate the amount of needed agents and the threshold. The results can be found in Figures 6 and 7.

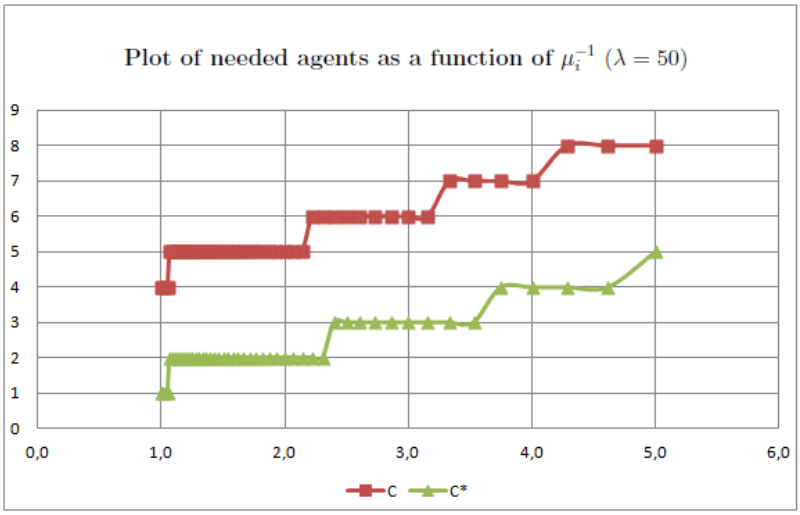

Figure 6: A plot of the needed agents against $\mu_i^{-1}(\lambda = 50)$.

From the plots we can see that the effect of increasing the average service time is bigger

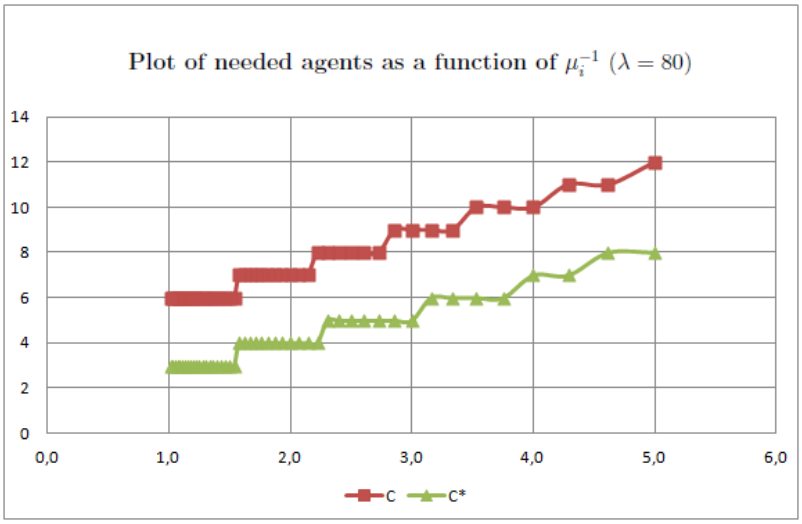

Figure 7: A plot of the needed agents against $\mu_i^{-1}(\lambda = 80)$.

during a busy period. The amount of needed agents increases faster than during an averagely busy period. During a busy period we see that an average service time of 5 minutes compared with 1 minute doubles the amount of needed agents, it increases from 6 to 12. During an average busy period it also doubles, but the difference is only 4 agents.

Outbound call service rate μ_o

We can make the same plots for the service rate of outgoing calls μ_o. These can be found in Figures 8 and 9.

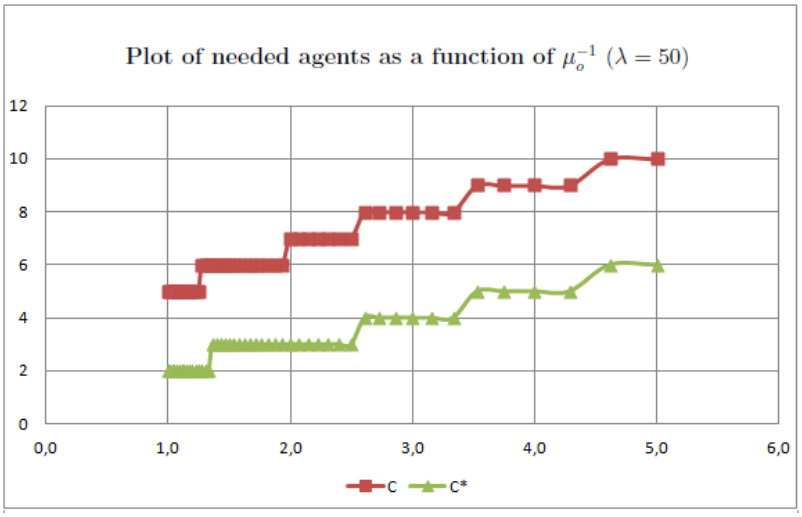

Figure 8: A plot of the needed agents against $\mu_o^{-1}(\lambda = 50)$.

Again we see that the effect of increasing the average service time are bigger during a busy

Figure 9: A plot of the needed agents against $\mu_o^{-1}(\lambda = 80)$.

period. But we also see that if we compare these plots with the previous plots that μ_o is of greater importance than μ_i. This is also logical, because 5 out of 9 calls are outgoing. If we now compare an average service time of 5 minutes with 1 minute during a busy period we see that this makes a difference of 7 agents. The amount of needed agents again doubles. For an averagely busy period this makes a difference of 5 agents.

The abandonment rate η and γ

To investigate the importance of the rate η we let it vary from 10 to 300 abandonments per hour, which is equivalent with letting the average patience of customers range from 6 minutes to 12 secondes. Again we do this for an averagely busy period and a busy period. The results can be found in Table 1.

η	c ($\lambda = 50$)	c^* ($\lambda = 50$)	c ($\lambda = 80$)	c^* ($\lambda = 80$)
10	6	3	8	5
20	6	3	8	5
30	6	3	9	5
40	7	3	9	5
50	7	3	9	5
300	7	3	9	5

Table 1: Number of agents needed for different values of η

From the table we can see that in the averagely busy case when η is 40 or larger we need 7 agents instead of 6. This means that only when the average patience of customers drops beneath 1.5 minutes we need to schedule an extra agent, but it may drop to even 12 secondes without further influencing the number of needed agents. We find a similar result for the busy case, where the turning point appears to be an average patience of 3 minutes. This tells us that the abandonment rate η is of relatively low importance. This can be easily explained with our performance targets; The number of calls served within 25 seconds should be bigger than 95%. This means that almost everyone should get service within 25 seconds. So people rarely get the chance to abandon the queue, except when they are really impatient.

To investigate the importance of γ, the probability that someone is willing to wait in the queue, we let it vary from 1 to 0.05. Again we do this for an averagely busy period and a busy period. The results can be found in Table 2. For the averagely busy case we find similar results as for η. Only when γ drops beneath 0.9 do we need to schedule an extra agent, but it may drop to 0.05 without further influencing the number of needed agents. For the busy case γ is of slightly bigger importance. In that case there are two turning points, one at 0.9 and one at 0.35. Again these results can be explained with our performance targets. Since we want an average waiting time of 10 seconds, we need to schedule a lot of agents and make sure that a queue rarely occurs. Therefore the probability to be put into a queue becomes very small and customers rarely get the chance to abandon the system. But when λ becomes larger this probability becomes bigger, in which case γ becomes of greater importance.

γ	c ($\lambda = 50$)	c^* ($\lambda = 50$)	c ($\lambda = 80$)	c^* ($\lambda = 80$)
1.00	6	3	8	5
0.95	6	3	8	5
0.90	6	3	9	5
0.85	7	3	9	5
0.80	7	3	9	5
0.75	7	3	9	5
0.70	7	3	9	5
0.65	7	3	9	5
0.60	7	3	9	5
0.55	7	3	9	5
0.50	7	3	9	5
0.45	7	3	9	5
0.40	7	3	9	5
0.35	7	3	10	5
0.30	7	3	10	5
0.25	7	3	10	5
0.20	7	3	10	5
0.15	7	3	10	5
0.10	7	3	10	5
0.05	7	3	10	5

Table 2: Number of agents needed for different values of γ

5 Schedule generation

With all the calculations done, we know for each interval of 30 minutes in the period we are interested in how many agents should be working at the call center. This does not yet give us a workforce schedule though, because agents need to work in shifts. Also we need to keep in mind that the costs of shifts might differ, and we want to keep our costs as low as possible. In [1] Koole shows that this problem can easily be translated to a MIP problem as follows:

$$\min \sum_{k=1}^{K} c_k x_k.$$

under the following constraints:

$$\begin{cases} \sum_{k=1}^{K} a_{tk} x_k \geq s_t, & t=1,...,T \\ l_n \leq \sum_{m \epsilon S_n} x_m \leq u_n, & S_n \subset 1,...,K,\ n = 1,...,N \\ x_k \geq 0 \quad \text{and integer}, & k = 1,...,K \end{cases}$$

Let us first look at the first constraint. Here x_k is the decision variable that denotes the number of shifts of type k to be used. The cost of shift k is denoted by c_k. The parameter a_{tk} equals 1 if shift k falls in interval t and 0 otherwise. As said earlier, s_t denotes the amount of agents needed in interval t. So this constraint models that we need at least s_t agents working during each interval t.
In the second constraint S_n denotes a subset of shifts and l_n is the minimum combined number of agents that need to work these shifts (think of contracts). Likewise u_l denotes the maximum combined number of agents that can work these shifts. So this constraint models that all contracts are followed and that no more than the available number of agents are scheduled.
The last constraint models that we can only plan a positive integer number of shifts.

This MIP can also be extended in a way such that it also determines which agent works which shift. This will give it a longer running time but is also necessary if there are extra constraints like an agent may not work a morning shift after a night shift, which are not unrealistic. This extension can be done as follows:

$$\min \sum_{k=1}^{K} \sum_{e=1}^{E} c_k x_{ke}.$$

under the following constraints:

$$\begin{cases} \sum_{k=1}^{K} \sum_{e=1}^{E} a_{tk} x_{ke} \geq s_t, & t=1,...,T \\ l_n \leq \sum_{k \epsilon S_n} \sum_{e=1}^{E} x_{ke} \leq u_n, & S_n \subset 1,...,K;\ n = 1,...,N \\ o_{ne} \leq \sum_{k \epsilon S_n} x_{ke} \leq b_{ne}, & S_n \subset 1,...,K;\ n = 1,...,N;\ e = 1,...,E \\ x_{ke} \epsilon \{0,1\}, & k = 1,...,K;\ e = 1,...,E, \end{cases}$$

Now the decision variable x_{ke} also depends on the employee and is binary and denotes if an employee works a certain shift. The rest of the model has not really changed. The only addition is the third constraint. Here o_{ne} is the minimum number of shifts from S_n that should be assigned to an employee. Likewise b_{ne} denotes the maximum number of shifts from S_n that can be assigned to an employee. So this constraint enables us to model things like that a morning shift should not be assigned after a night shift. Or that a certain worker can not work on Fridays.

We can extend this even further to also let the program choose a maximum number of different shifts it can assign. This is also a realistic constraint because if employees can be assigned 30 different shifts, confusion about at what time their shift starts might be the result. This extension results in the following MIP:

$$\min \sum_{k=1}^{K} \sum_{e=1}^{E} c_k x_{ke}.$$

under the following constraints:

$$\begin{cases} \sum_{k=1}^{K} y_k \leq A & \\ \sum_{e=1}^{E} x_{ke} \leq E * y_k, & k\text{=}1,...,K \\ \sum_{k=1}^{K} \sum_{e=1}^{E} a_{tk} x_{ke} \geq s_t, & t\text{=}1,...,T \\ l_n \leq \sum_{k \epsilon S_n} \sum_{e=1}^{E} x_{ke} \leq u_n, & S_n \subset 1,...,K;\ n = 1,...,N \\ o_{ne} \leq \sum_{k \epsilon S_n} x_{ke} \leq b_{ne}, & S_n \subset 1,...,K;\ n = 1,...,N;\ e = 1,...,E \\ y_k \epsilon \{0,1\}, & k = 1,...,K \\ x_{ke} \epsilon \{0,1\}, & k = 1,...,K;\ e = 1,...,E \end{cases}$$

Here y_k now is the decision variable which is 1 if shift k can be assigned and 0 otherwise. The first constraint then models that a maximum of A different shifts may be assigned to the employees. The second constraint models that indeed only shifts from this selection are assigned to the employees.

A problem like this can be solved by a software package like AIMMS. We now have all the ingredients to actually generate a workforce schedule. We will discuss some results in the following section.

6 Scheduling results

In this section we will generate a workforce schedule for a typical workday of our call center. The values of the inbound call rate during such a day can be found in Appendix A as is mentioned in chapter 4. In chapter 4 we also determined the corresponding values for s_t. Before we can now use the MIP we just discussed, we first need to choose which shifts we have and which shifts can be assigned to the same agent. We will work with shifts of three different lengths: 7, 7.5 and 8 hours. These shifts can start every 30 minutes, which gives us a total of 144 different shifts. We assume that out of these 144 shifts a maximum of 6 different shifts may be assigned to the agents. This means that we will need to use the last MIP that was discussed in the previous chapter. Furthermore we assume that the call center has 30 employees and they are paid by the hour, so the shift of 8 hours costs $\frac{8}{7}$ times as much as the shift of 7 hours.
If we now look again at the last MIP discussed in the previous chapter, the just defined scheduling problem translates itself in the following way. Since we have a total of 144 different shifts and 30 employees we have $K = 144$ and $E = 30$. The maximum number of different shifts that may be assigned is 6, so $A = 6$. Furthermore because employees are paid by the hour, we can define c_k to be 7, 7.5 or 8 depending on the length of shift k. Also like discussed in the last chapter the parameter a_{tk} equals 1 if shift k falls in interval t and 0 otherwise. So for example if shift k is of length 7 hours and starts at 6:00am we have that $a_{tk} = 1$ for $13 \leq t \leq 27$ and 0 otherwise. Since we do not have any contractual agreements concerning the number of agents that can or must work certain shifts, we do not need the fourth constraint. We could however demand that no agent does more than 1 shift on a single day. The fifth constraint models this if we define $N = 1$, $S_1 = \{1, ..., 144\}$, $o_{1e} = 0$ and $b_{1e} = 1$.
If we run our MIP with these shifts and constraints it outputs a schedule within a few minutes. The shifts that are assigned and how many times they are assigned can be found in Table 3.

Shift	Shifttime	#hours	#planned
1	05:00-12:00	7	4
2	08:00-15:00	7	4
3	12:00-19:30	7,5	3
4	15:00-22:00	7	4
5	16:00-23:00	7	1
6	22:00-05:00	7	3

Table 3: Assigned shifts (Call blending policy)

We see that the schedule needs 19 different agents, assuming that every agent can only work one shift per day. The total amount of working hours that are assigned is 134,5, this is simply the total number of scheduled hours of work. This can be compared with the total amount of working hours we actually need to reach our performance targets, by this we mean $\sum s_t \cdot 0.5$ hours, which is 122. This means we have a gap of 10.2%. In Figure 10 the total number of scheduled agents and the number of needed agents are plotted for every half an hour.
We can again compare the call blending policy with the alternative policy. The shifts that are assigned in the optimal schedule when using the alternative policy can be found in Table 4.
Notice that the chosen shifts are almost identical to the shifts that are used when making a schedule for the call blending policy. This schedule 25 different agents, which is 6 more than

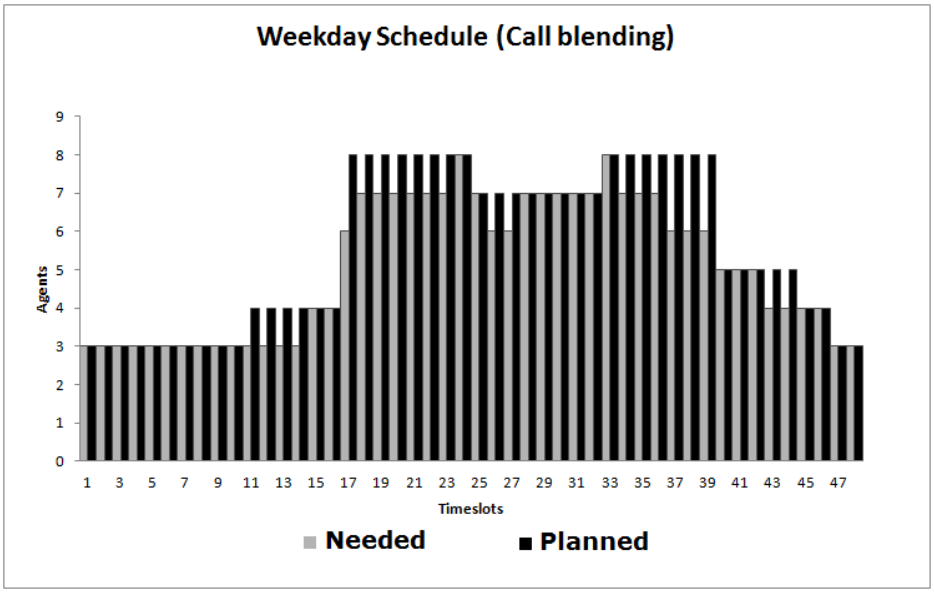

Figure 10: The number of scheduled and needed agents (Call blending policy)

Shift	Shifttime	#hours	#planned
1	05:00-12:00	7	5
2	07:30-15:00	7,5	5
3	12:00-19:30	7,5	4
4	15:00-22:00	7	6
5	16:00-23:00	7	1
6	22:00-05:00	7	4

Table 4: Assigned shifts (Alternative policy)

when using the call blending policy. The total amount of working hours that are assigned is 179,5 and this 33% more than the schedule with call blending. This can again be compared with the total amount of working hours we actually need, which in this case is 158. This means we now have a gap of 13.6%. In Figure 11 the total number of scheduled agents and the number of needed agents are plotted for every half an hour.

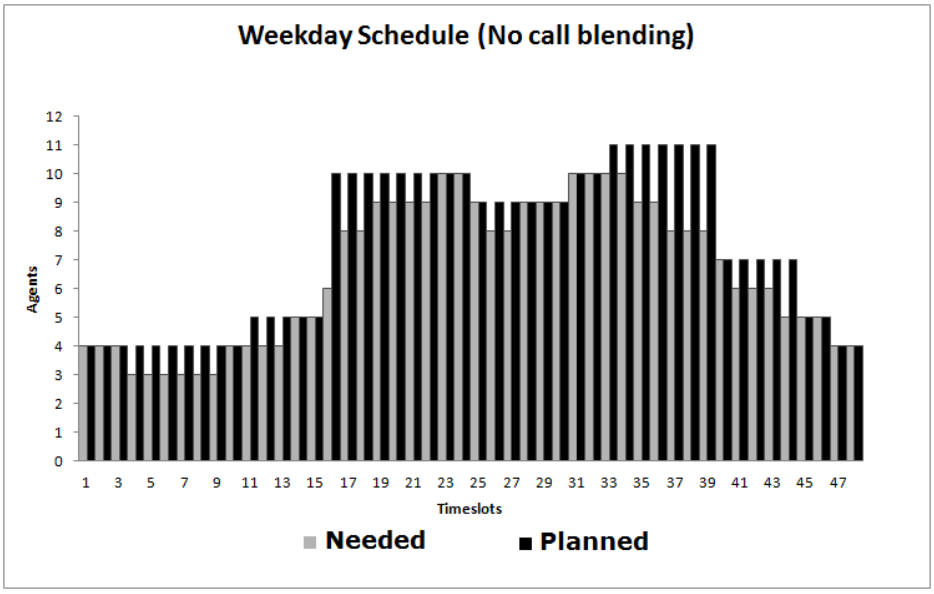

Figure 11: The number of scheduled and needed agents (Alternative policy)

7 Conclusion

The goal of this project was to make a workforce schedule for a typical working day of a call center. This had to be done as cheap as possible while still meeting the following performance targets:

- The percentage of abandoned calls should be smaller than 1.5%.
- The number of calls served within 25 seconds should be bigger than 95%.
- The average speed of answer should be smaller than 10 seconds.

An important aspect of the call center was that besides having to serve incoming calls, also outbound calls had to be made. We have compared two different policies for dealing with these outbound calls. The first policy assumes that whenever an outbound call has to be made, that this is done immediately. The second policy is the so called call blending policy, where an outbound call is made by an agent only when a predetermined number of agents is idle. Both policies are modeled by an extension of the widely used $M/M/c$ queueing model. When comparing these two policies for different values of the inbound call rate λ, the call blending policy clearly is cheaper than the alternative policy. For $\lambda = 100$ the call blending policy needs 9 agents, while the alternative policy needs 12.
A sensitivity analysis of the model that includes call blending shows that the service rate of outbound calls has the most influence on the number of needed agents. This influence becomes bigger when λ increases. The average patience of customers as well as the percentage of people that are willing to wait in the queue are both of relatively low importance.
Numerical results show us that the performance target dealing with the average speed of answer is easily met compared to the other two performance targets which seem to be of equal importance. However when using the call blending policy the requirement that enough outbound calls are made appears to be the most difficult to meet.

The actual workforce schedules are made by solving a mixed integer program. If we work with shifts of three different lengths: 7, 7.5 and 8 hours which can start every 30 minutes and we assume that out of these shifts a maximum of 6 different shifts may be assigned to the agents, we find a day schedule that schedules a total of 134,5 working hours when call blending is used. The total amount of needed working hours is 122. This means there is a gap of 10,2%. When using the alternative policy a total amount of 179,5 hours are scheduled, while 158 hours are needed. This is a gap of 13,6%. When comparing the two polices the alternative policy needs to schedule 33% more hours than the call blending policy. We can conclude that call blending is a very efficient way to make cheap schedules for call centers that have to deal with outgoing calls.

Literature

1 Koole, G. 2007. *Call Center Mathematics: A scientific method for understanding and improving contact centers.* http://www.cs.vu.nl/ koole/ccmath/book.pdf

2 Deslauriers, A., L'Ecuyer, P., Pichitlamken, J., Ingolfsson, A. and Avramidis, A. N. 2007. *Markov chain models of a telephone call center with call blending.* http://apps.business.ualberta.ca/aingolfsson/documents/PDF/ctmc1.pdf

3 Koole, G. and Bhulai, S. 2003. *A queueing model for call blending in call centers.* IEEE Transactions on Automatic Control 48:1434-1438.

4 Riordan, J. 1962. *Stochastic Service Systems (SIAM Series in Applied Mathematics).* Wiley, New York.

5 Adan, I. and Resing, J. 2002. *Queueing Theory.* Lecture notes, Eindhoven University of Technology. http://www.win.tue.nl/ iadan/queueing.pdf

A [Table of incoming call rates during the week]

t	λ	t	λ
1	13	25	63
2	12	26	54
3	11	27	57
4	10	28	60
5	9	29	67
6	9	30	70
7	9	31	71
8	9	32	72
9	10	33	74
10	11	34	71
11	13	35	68
12	16	36	63
13	19	37	59
14	21	38	54
15	24	39	50
16	30	40	44
17	49	41	38
18	59	42	34
19	64	43	31
20	68	44	27
21	69	45	24
22	70	46	20
23	71	47	17
24	74	48	16

B [Table of number of needed agents and threshold]

λ	c	c^*	λ	c	c^*	λ	c	c^*	λ	c	c^*
1	2	0	26	4	1	51	6	3	76	8	4
2	2	0	27	4	1	52	6	3	77	8	5
3	2	0	28	4	1	53	6	3	78	8	5
4	2	0	29	4	1	54	6	3	79	8	5
5	3	0	30	4	1	55	6	3	80	8	5
6	3	0	31	4	1	56	6	3	81	8	5
7	3	0	32	4	1	57	6	3	82	8	5
8	3	0	33	4	1	58	6	3	83	8	5
9	3	0	34	5	2	59	6	3	84	8	5
10	3	0	35	5	2	60	7	3	85	8	5
11	3	0	36	5	2	61	7	3	86	8	5
12	3	0	37	5	2	62	7	3	87	8	5
13	3	0	38	5	2	63	7	4	88	9	5
14	3	0	39	5	2	64	7	4	89	9	5
15	3	0	40	5	2	65	7	4	90	9	5
16	3	0	41	5	2	66	7	4	91	9	6
17	3	0	42	5	2	67	7	4	92	9	6
18	3	0	43	5	2	68	7	4	93	9	6
19	4	1	44	5	2	69	7	4	94	9	6
20	4	1	45	5	2	70	7	4	95	9	6
21	4	1	46	6	2	71	7	4	96	9	6
22	4	1	47	6	2	72	7	4	97	9	6
23	4	1	48	6	3	73	7	4	98	9	6
24	4	1	49	6	3	74	8	4	99	9	6
25	4	1	50	6	3	75	8	4	100	9	6

C [Numerical results of determining c and c^*]

Inbound call rate $\lambda = 40$

Percentage of abandoned calls				
$c\backslash c^*$	0	1	2	3
2	22,02			
3	7,07	10,00		
4	1,92	2,46	5,33	
5	**0,45**	**0,52**	**1,05**	3,27

Effective service rate μ				
$c\backslash c^*$	0	1	2	3
2	26,85			
3	26,46	30,00		
4	26,34	29,68	**32,33**	
5	26,31	29,61	**32,15**	**33,90**

Average speed of answer(sec)				
$c\backslash c^*$	0	1	2	3
2	57,40			
3	16,00	21,71		
4	**3,98**	**4,90**	10,37	
5	**0,88**	**0,97**	**1,91**	**5,90**

Percentage served within 25 seconds				
$c\backslash c^*$	0	1	2	3
2	47,09			
3	17,69	26,03		
4	5,21	6,77	14,75	
5	**1,25**	**1,43**	**2,85**	8,79

Inbound call rate $\lambda = 80$

Percentage of abandoned calls							
$c \backslash c^*$	0	1	2	3	4	5	6
2	45,63						
3	25,75	27,26					
4	12,95	13,36	15,95				
5	5,84	5,84	6,66	9,71			
6	2,38	2,30	2,49	3,45	6,32		
7	**0,88**	**0,82**	**0,84**	**1,10**	1,93	4,39	
8	**0,30**	**0,26**	**0,26**	**0,32**	**0,53**	**1,17**	3,23

Effective service rate μ							
$c \backslash c^*$	0	1	2	3	4	5	6
2	24,61						
3	24,47	25,89					
4	24,42	25,64	27,58				
5	24,39	25,52	27,30	29,28			
6	24,37	25,47	27,18	29,04	30,76		
7	24,37	25,45	27,14	28,95	30,57	31,96	
8	24,37	25,44	27,12	28,92	30,51	**31,82**	**32,92**

Average speed of answer(sec)							
$c \backslash c^*$	0	1	2	3	4	5	6
2	137,12						
3	65,36	67,67					
4	29,49	29,88	34,78				
5	12,30	12,12	13,52	19,25			
6	**4,72**	**4,49**	**4,77**	**6,47**	11,65		
7	**1,66**	**1,52**	**1,53**	**1,97**	**3,40**	**7,63**	
8	**0,54**	**0,47**	**0,45**	**0,55**	**0,90**	**1,96**	**5,33**

Percentage served within 25 seconds							
$c \backslash c^*$	0	1	2	3	4	5	6
2	81,31						
3	56,14	61,13					
4	32,31	33,99	41,71				
5	15,76	15,95	18,45	27,14			
6	6,64	6,44	6,99	9,64	17,50		
7	**2,46**	**2,28**	**2,31**	**2,97**	5,10	11,38	
8	**0,81**	**0,71**	**0,67**	**0,81**	**1,31**	**2,79**	7,49

Inbound call rate $\lambda = 80$ and $\mu_i = \mu_o = \mu = 32$

Percentage of abandoned calls					
$c \backslash c^*$	0	1	2	3	4
2	36,30				
3	16,97	21,13			
4	6,94	8,74	13,17		
5	2,53	3,20	4,86	8,82	
6	**0,83**	**1,05**	1,60	2,92	6,28

Average speed of answer(sec)					
$c \backslash c^*$	0	1	2	3	4
2	98,94				
3	39,40	49,07			
4	14,57	18,34	27,63		
5	**4,94**	**6,24**	**9,49**	17,20	
6	**1,53**	**1,94**	**2,95**	**5,39**	11,57

Percentage served within 25 seconds					
$c \backslash c^*$	0	1	2	3	
2	72,43				
3	41,35	51,51			
4	18,80	23,68	35,65		
5	7,11	8,99	13,65	24,75	
6	**2,30**	**2,91**	**4,44**	8,09	17,38

D [Mathematica script]

```
η = 20;
γ = 90 / 100;
μi = 24;
μ0 = 40;
max = 30;
For[j = 1, j ≤ 100, j++, lambda[j] = j;]
For[m = 1, m ≤ 100, m++,
  check = 0;
  c = 2;
  λ = lambda[m]
    While[check == 0,
     For[cstar = 0, cstar < c - 1, cstar++,
      Clear[μ];
      Table[θ[n] = 0, {n, 0, cstar}];
      θ[cstar + 1] = 1;
      If[cstar < c - 2, Table[θ[n] = (cstar + 1)! (λ/μ)^(n-(cstar+1)), {n, cstar + 2, c - 1}]];
      Table[θ[n] = (cstar + 1)! (λ/μ)^(c-(cstar+1)) Product_{k=1}^{n-c} γ λ / (c μ + k η), {n, c, max}];
      Table[pi[n] = 0, {n, 0, cstar}];
      pi[cstar + 1] = (Sum_{n=cstar+1}^{max} θ[n])^-1;
      Table[pi[n] = θ[n] * pi[cstar + 1];, {n, cstar + 2, max}];
      Ra = λ (1 - γ) (1 - Sum_{n=cstar+1}^{c-1} pi[n]) + η Sum_{n=c}^{max} n pi[n] - η c (1 - Sum_{n=cstar+1}^{c-1} pi[n]);
      Rs = c μ (1 - Sum_{n=cstar+1}^{c-1} pi[n]) + Sum_{n=cstar+1}^{c-1} n μ pi[n];
      Ro = Rs - (λ - Ra);
      p[μ_] = 1 - Ro/Rs;
      h[μ_] = p[μ]/μi + (1 - p[μ])/μ0 - 1/μ;
      W[t_] = Exp[-c μ t] * (Sum_{n=0}^{max-c} (pi[n + c] * Sum_{k=0}^{n} ((Product_{j=0}^{k-1} (c μ/η + j)) (1 - Exp[-η t])^k / k!)));
      dW[t_] = ∂_t (1 - W[t]);
      μ = μ /. FindRoot[h[μ], {μ, (μ0 + μi)/2}, WorkingPrecision → 60][[1]];
      If[μ > (4/(9 μi) + 5/(9 μ0))^-1 && Ra < 15 λ/1000 && ((∫_0^∞ W[t] dt) * 3600) < 10 && W[25/3600] < 5/100 && check == 0,
       check++;
       AGENTSthis = c;
       THRESHOLDthis = cstar;
      ];
     ];
     c++];
  AGENTS[m] = AGENTSthis;
  THRESHOLD[m] = TRESHOLDthis;
```