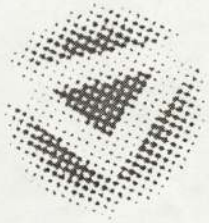
HUMAN RESOURCE ALLOCATION IN A CALL CENTRE

ANICET E. OUDET

Master of Science by Research in Pattern Analysis and Neural Networks

Supervisor: Dr Ian T. Nabney



ASTON UNIVERSITY

September 1998

THE UNIVERSITY OF ASTON IN BIRMINGHAM

HUMAN RESOURCE ALLOCATION IN A CALL CENTRE

ANICET E. OUDET

Master of Science by Research in Pattern Analysis and Neural Networks, 1998

Thesis Summary

Last year G. Lecorroller proposed a new approach to the problem of staff allocation in a telephone call centre, based on the work of Prof D. Lowe and Dr I. Nabney, and using a multi-skill workforce. He developed a two stage process, in order to compute the number of agents required to achieve a particular service level. In the first stage, thanks to a stochastic model of the queues, a minimum number of operators was predicted, in the next stage those were allocated in the different pools using a minimisation algorithm. The results were quite encouraging, 25% fewer agents predicted with this new approach compared to the traditional and current one.

The main purpose of this thesis is to improve this new model.
At first we tested it whilst considering an unlimited pool of people. A tricky problem to solve was the increase of the predicted number of agents with an increasing number of pools given a call volume. The empirical method we proposed to deal with this issue is detailed.

The second part of our project explored the effect of different service time distributions. A new distribution based on mixture models has been developed which fits much better the service time distribution from any given call centre. In terms of agents cost less than 2% more were required compared to the new method.

Next the effects of abandoned calls have been taken into account by revisiting some approximation formulae. However, the predicted increase of agents, necessary to achieve the same service level as before, remains less than 2% compared to the new model which added to the previous improvement correspond still to 20% fewer agents than with the traditional forecast. Our model is also more precise than the Erlang C model used traditionally.

Ultimately we are able to validate the new model with our improvement using both simulation program and new data. All the results are presented in careful details.

Keywords: call centre, mixture models, abandoned calls

Acknowledgements

I would like to thank Dr I.T. Nabney for all his help and his useful advice during this nine month project.

I would also like to thank Steve Grant and John Burton at Callscan for their help and for their kindness during the meetings we had.

I am also thankful to Callscan for their financial support.

Finally I am grateful to all students at the NCRG for their help.

Contents

List of Figures

Chapter 1

Introduction

1.1 Brief presentation of the problem

The problem on which we have been working is human resource allocation in a call centre. Last year G. Lecorroller tackled it and proposed two approaches to deal with it: a people-based approach and a significant improvement of the traditional approach. The first gave disappointing results and therefore we gave up to lead further investigations. The results obtained with the second were far more interesting. That is the reason why we have worked exclusively on it in our thesis.

In this model a call centre can be described in term of queues and skill pools, see Figure (1.1). Arrival calls, whose type is determined from the number dialled, are routed to different queues in which they will wait until a server qualified to answer that type of call is free. The waiting time depends on the intensity of the traffic and on the number of operators qualified to deal with the call. With regard to the agents, these are gathered in pools of same skills profile *i.e* with the same abilities to answer a call. In practice most of them are multi-skilled and thus are able to answer different types of call.

In this way not only is the waiting time for each call minimised but also fewer people are required to provide the same service level, *i.e.* the percentage of answered calls within a given delay, compared to a traditional call centre where there are as many pools as there are queues.

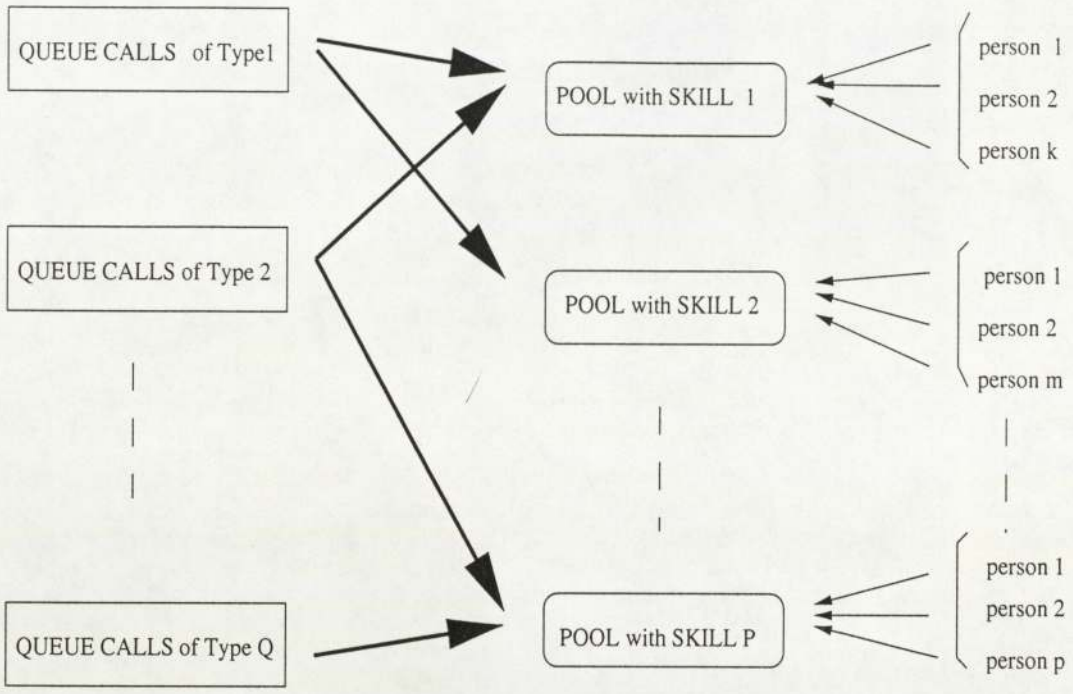


Figure 1.1: Schematic diagram of an improvement approach. There are Q queues and P pools in our system. The agents are gathered in the P pools whose agents can answer calls coming from different queues if having the good skills profile.

Another quite significant point which is now taken into account, and which wasn't in the traditional method, is the quality of the service (see constraints in the next chapter). We can both measure and specify the service level and the quality of how well the different calls are answered by agents with suitable skills. The number of agents in the system will increase or decrease if we modify these parameters.

The aim of this thesis is as follows:

1. to lead investigations on the properties of this model;

2. to explore the distributions of the inter-arrival time and the service time;

3. to investigate the effect of people abandoning calls before they are answered.

1.2 Data used for the forecasting

To forecast the number of agents in a particular call centre, it is essential to be able to describe its daily activity.

The data used last year corresponded to several periods, each of them of half an hour collected during a week. Amongst all the information, two main parameters were useful:

1. the call volume *i.e.* the total number of incoming calls per period;
2. the call duration: the average duration of a call which is the average service time.

The drawback with using only these parameters is: these are just mean values and therefore do not fit perfectly the real outcome, and as strong assumptions had been made to build the model of forecasting, see Chapter 2 and Chapter 4, the only way to improve and to validate it was to use other sample data.

That is what we have done using data collected from a single queue call centre during a week. For each incoming call, the exact arrival time and duration of the call were known. Thus deriving the inter-arrival time distribution or the service time distribution was feasible. I thank also Callscan for supplying the data.

At least by combining these two sets of data we could improve the exactness of the forecasting method to be discussed now.

1.3 A two step process

In this section we shall briefly describe the way the number of people in the system are forecasted. The computation of the number of agents in the different pools is a two step process.

Initially we compute the minimum number of people required to achieve a certain service level, which is the percentage of incoming calls answered within a certain period, for example 95% of calls have to be answered within 30 seconds, for each queue

and for each period. We use a revised stochastic modelling of the queue and the two previous files of data to calculate them. Since the main parts of this model have been laid by G. Lecorroller, we shall briefly present them again in the next chapter to obtain a better understanding our own new development, described itself in chapter 4.

In the second step we calculate the number of people in the pools, considering different constraints: *e.g.* that the number of agents in the system should be at least greater than the one computed in the previous stage or by requiring that the skills of the agents have to correspond as well as possible to the types of calls they answer. By being more or less demanding in these requirements the predicted number of operators rises or decreases.

Mathematically we use the scaled conjugate gradient (scg) algorithm to find the minimum of this optimisation problem under constraints since convergence is both quick and reliable.

It is worth noting that in all the investigations we have carried we have assumed that the pools of people were uncapped: in each pool as many people as necessary could be used.

1.4 Thesis overview

Chapter 2 summarises the theoretical model proposed by G. Lecorroller. It is all the more important to introduce it as our work depends on it.

Chapter 3 presents the different experiments we have carried out on the model: we observe the behaviour of the model by changing the Lagrange multipliers, increasing the number of pools and the effect of multi-skilling amongst others.

Chapter 4 describes the validation done on the stochastic modelling of the queue and the mixture model used to fit more accurately the service time.

Chapter 5 presents the way that abandoned calls have been taken into account in our model.

In Chapter 6 we validate our model using other data.

The last chapter, chapter 7 presents the conclusions.

The appendices detail algorithms and methods:

- In appendix A, the computation of the coefficients of the EM algorithm is detailed for a exponential mixture model.
- In appendix B, we calculate the first and the second order moments of an exponential mixture distribution and some various other elements.
- In appendix C, the code of different algorithms is given with some comments.

Chapter 2

Overview of earlier work

2.1 Introduction

To solve the problem faced by the management, namely the allocation in different pools of agents to achieve a specified service level while still providing a good quality of service, G. Lecorroller proposed a two step scheduling method. The objective of this chapter is to describe it.

First we present in a detailed way the stochastic model of the queues used to compute the required total number of agents in the system in order to achieve a particular service level.

Once this number has been calculated, we have to allocate agents to different pools. Therefore we define, in the second part, a cost function and some constraints.

The practical resolution is the subject of the third part: we also transform our problem of minimisation under non-linear constraints to an unconstrained minimisation problem. The optimum provides a solution to the agent allocation problem.

Finally we present results obtained last year.

2.2 Stochastic modelling of the queue

2.2.1 Observations

In the simplified approach proposed by Prof D. Lowe and Dr Ian T. Nabney, calls were supposed to be uniformly distributed across period (In the method of forecasting we used to work on data collected during a week divided in periods of half hour. The call volume, the average number of incoming calls in an half hour period and the mean call duration were the main parameters). This assumption of course was a bit unrealistic as there might be busy periods inside a half hour and as well as the service time couldn't be constant, some very long calls needing much more time to be answered than others.

For all these reasons, G. Lecorroller used basic elements of queueing theory to take into account fluctuations in the call density and the random nature of the service time distribution. His idea was all the more interesting and practical as it was possible to compute the number of agents to achieve a certain service level thanks to approximation formula.

General notions about queueing are presented here, for more detail see (Allen, 1978) or (Kleinrock, 1975).

2.2.2 Describing a queueing system

Practically, incoming calls enter a queueing system. They are dealt by servers, *i.e* agents, unless all of them are busy. In this case they have to join the queue and wait until a server becomes free.

We can characterise such a queueing system by its inter-arrival distribution, its service time distribution, the number of agents present and the time a call spends in the system, which is equal to the time spent in the queue and the time spent being answered.

In the following sections we are going to describe these parameters.

It should be noted that a lot of queueing theory considers a single queue and a single service time distribution. As our model, see Chapter 1, was made up of Q queues, Lecorroller worked on merging them in one.

2.2.3 Inter-arrival distribution

We assumed that the inter-arrival distribution, for each queue q, *i.e.* the distribution which represents the time between two arrival calls, was exponentially distributed with parameter λ_q. The arrival rate λ_q, was supposed to be equal to the average number of incoming calls in an half period: that is to say to $N_q/(30 \times 60)$.

Moreover we made another assumption: as an agent could in theory answer calls from any queue it was pertinent to merge the different queues in a single one. The inter-arrival distribution of this single queue was also exponential with parameter λ, equal to the sum of each queues parameter λ_q: $\lambda = \sum_{q=1}^{Q} \lambda_q$.

2.2.4 Service time distribution

As far as the service time was concerned we assumed it was exponential for each type of call with parameter $\mu_q = 1/T_q$, different for each queue, where T_q denoted the mean duration of a call in a half hour period.

As the Q queues were merged in a single queue, we used, to model the service time, a Q stage hyper-exponential distribution given by:

$$f_s(t) = \sum_{q=1}^{Q} \alpha_q \mu_q e^{-\mu_q t} \quad \text{with} \quad \alpha_q = \frac{\lambda_q}{\sum_{q=1}^{Q} \lambda_q}, \tag{2.1}$$

where α_q corresponded to the probability that an agent answered a call of type q.

If it was quite easy to give a sound justification for the exponential nature of inter-arrival call distribution, it was far more difficult to give a valid reason why the service time should be exponential. One argument for was that the duration of each answer should vary a lot amongst the time and therefore that the standard deviation will be

very large relative to the mean value: this is one of the characteristics of an hyper-exponential distribution. Because of this uncertainty, we will investigate this issue in Chapter 4 using new data.

2.2.5 Required number of servers

Assuming our model, described with Kendall notation, was an $M/H_q/c$ model, *i.e.* exponential inter-arrival time distribution, a q-stage hyper-exponential service time distribution and c servers (agents), Lecorroller proved it was quite easy to derive the number of agents required to achieve a certain service level by combining two approximation formulae: Martin's estimate and the heavy traffic approximation.

Before describing this calculation we have to introduce some useful notation. The total time spent in the system by a call, described by the random variable w, is equal to the time spent in the queue, defined by the random variable d, plus the service time s, the random variable representing the time to answer the call. To define the service level we need to introduce too the r^{th} percentile value of d, $\pi_d(r)$, defined by:

$$P(d \leq \pi_d(r)) = \frac{r}{100}$$

Heavy traffic approximation

If we suppose that the average utilisation of a server approaches 1 then the heavy traffic approximation states that the distribution of the queueing time, d, approaches an exponential distribution. And so to compute the r^{th} percentile value of d, we use the following formula:

$$\pi_d(r) = W_d \times \ln\left(\frac{100}{100 - r}\right)$$

where $W_d = E[d]$ is the average waiting time in the queue. The justification of this formula and the following one can be found, of course, in G. Lecorroller's thesis.

In fact, as in practice we specify a certain service level to be achieved, the only unknown in this formula is W_d.

Martin's estimate

Martin's estimate has the following form:

$$W_d = \frac{C(c,a) \times E[s]}{c-a} \times \frac{1+C_s^2}{2} \tag{2.2}$$

where

- s is the random variable describing the service time,
- $E[s]$ is the expected service time,
- $a = \lambda \times E[s]$ and λ is the arrival rate,
- $C(c,a)$ is the Erlang's C function,
- $C_s^2 = \frac{Var[s]}{E[s]^2}$ the squared coefficient of variation of s.

With this formula we can link the average waiting time in the system W_q to the required number of servers c. And as all the parameters in our model were fixed, the service time, the arrival rate, also $E[s]$, a, $C(c,a)$ and C_s^2 were computable:

- $E[s] = \sum_{q=1}^{Q} \frac{\alpha_q}{\mu_q}$
- $E[s^2] = 2\sum_{q=1}^{Q} \frac{\alpha_q}{\mu_q^2}$
- $C_s^2 = \frac{Var[s]}{E[s]^2} = \frac{E[s^2]}{E[s]^2} - 1$
- $C(c,a)$, the Erlang's C function, can be derived thanks to an iterative formula.

More detail about this calculations can be found in appendix C of (Lecoroller, 1997). Eventually the only unknown was c. So the required number of servers c can be defined as the smallest positive integer so that

$$c \geqslant a + \frac{C(c,a) \times E[s]}{\pi_d(r)} \times \frac{1+C_s^2}{2} \tag{2.3}$$

In practice the computation of c is slightly different: we are going to explain it thanks to the following example.

Example

The service time distribution and the queueing time known, we want to compute the required number of people so that 90% of the calls are answered within 30 seconds.

That means, as 90 is the percentile value, that $\pi_d(90)$ has to equal to 30 seconds, in other words that $P(d \leq 30) = 0.90$.
By using the heavy traffic approximation we can deduce that $\pi_d(90) = W_d \times \ln(\frac{100}{10}) \approx 2.3 \times W_d$ and so that W_d has to be equal or less that $\frac{30}{2.3} \approx 13$ seconds.

Now using Martin's estimate, we can derive the required number of servers c, which is the value so that

$$\frac{C(c,a) \times E[s]}{c-a} \times \frac{1+C_s^2}{2} \leqslant 13 \leqslant \frac{C((c-1),a) \times E[s]}{(c-1)-a} \times \frac{1+C_s^2}{2}$$

2.3 Formalising the optimisation problem

In this section we formalise the problem of allocation of agents in a call centre. In the following paragraphs we will introduce some mathematical notations.

2.3.1 Variables and mathematical notations

Our system is composed of P pools, in which agents with similar skill profiles are gathered, and Q queues.

Since we aim at computing a required number of agents for each scheduling period throughout a whole week, we have chosen the connectivities between the channel and the skills pools as the major unknowns. The variable (c_{pq}) denotes a link between a pool p and a queue q. The connectivity matrix C is of size $(P \times Q)$. In fact there is a separate copy of C for each scheduling period.

To characterise the workforce we define then another matrix S, $(P \times Q)$. Each element s_{pq} denotes the ability of a person belonging to skills mix pool p to answer a call from the queue q. The values lie between zero and one. For example, if s_{pq} is equal

to 1 that means that people in the pool p have a 100% competency to answer arrival calls from queue q.

We defined n_q, the predicted number of agents required to answer the calls from queue q, by the product of the total number of servers in the system and the active calls in the queue q divided by the total number of active calls in the system in half hour period:

$$n_q = \frac{N_q T_q}{\sum_{q=1}^{Q} N_q T_q} \times c$$

where N_q is the number of incoming calls in queue q during an half hour period, T_q is the average call duration in queue q.

Two remarks can be made:

- the sum $\sum_{p=1}^{P} c_{pq}$ can be regarded as the number of agents used to answer calls from queue q.
- the sum $\sum_{q=1}^{Q} c_{pq}$ may be seen as the number of agent in the pool p.

2.3.2 The cost function

The allocation problem is not one of the simple unconstrained optimisation since there are different aspects we need to take into account.Each of the constraints corresponds to a specific term in the cost function. To reflect the relative importance of the different parameters we will use Lagrange multipliers.

Now let us briefly present the different cost terms. There are:

1. a measure of loss in quality:

$$E_0 = \frac{1}{Q} \sum_{q=1}^{Q} \left(\sum_{p=1}^{P} c_{pq} s_{pq} - n_q \right)^2$$

2. a term enforcing the sum on each queue to equal the number of calls:

$$E_1 = \frac{1}{Q} \sum_{q=1}^{Q} \left(\sum_{p=1}^{P} c_{pq} - n_q \right)^2$$

3. a term corresponding to the minimisation of the total number of staff in general:

$$E_2 = \frac{1}{Q} \sum_{q=1}^{Q} \sum_{p=1}^{P} (c_{pq})^2$$

4. a term corresponding to the minimisation of the deviation of the number of staff from the theoretically required number of agents c:

$$E_3 = \frac{1}{Q} \left(\sum_{q=1}^{Q} \sum_{p=1}^{P} c_{pq} - c \right)^2$$

We derive the final cost function as a combination of all previous terms:

$$\begin{aligned} f: \quad & \left(\mathbb{R}^+\right)^{P \times Q} \longrightarrow \left(\mathbb{R}^+\right) \\ & ((c_{pq})) \quad \longmapsto E_0 + \alpha E_1 + \beta E_2 + \delta E_3 \end{aligned}$$

2.3.3 Constraints

Once the cost function is defined, there are still some other constraints, which don't correspond to quality measures but to some physical requirement such as:

- all the connectivities should be positive: $c_{pq} \geqslant 0$
- no call should be answered by people completely unqualified:

$$\text{If} \quad S_{pq} = 0 \quad \text{then} \quad c_{pq} = 0$$

- the number of agents in the system should be greater than c, the minimum number of agents required to achieve the specified service level:

$$\sum_{q=1}^{Q} \sum_{p=1}^{P} c_{pq} \geqslant c$$

- an optional constraint can be added, if it is necessary, to cap the number of people in some skill pools:

$$\sum_{q=1}^{Q} c_{pq} \leqslant A_p$$

As this worked successfully last year, we concentrated our investigations on other aspects of the algorithm, and all experiments were uncapped pools.

2.4 Transformation to an unconstrained problem

Minimising a function subject to non linear constraints is much harder than looking for the solution of an unconstrained problem. That is why we used the penalty function method to find the optimum for our problem.

2.4.1 Penalty method function and the scg algorithm

In this section we discuss the principle and the way it was implemented.

Consider a function $f(x)$ we want to minimise, subject to constraints $c_i(x) = 0$ or $c_i(x) \geq 0 \quad i = 1, ..., n$. We can construct a quadratic penalty function $\frac{\delta}{2}\, \widehat{c}\, \widehat{c}^T$ that we add to the former function in order to obtain a new function defined by:

$$P_Q(x, \delta) = f(x) + \frac{\delta}{2}\, \widehat{c}\, \widehat{c}^T$$

where $\widehat{c}$ contains only those constraints that are violated at x and δ is called the penalty parameter. The problem becomes unconstrained.

If $x^*(\delta)$ denotes the unconstrained minimum of $P_Q(x, \delta)$, under mild conditions, $\lim_{\delta \to \infty} x^*(\delta) = x^*$, where x^* is the true minimum with constraints. Consequently, we have found a way to compute the constrained optimum of f:

a. We minimise $P_Q(x, \delta)$ using the scg algorithm for a fixed value of δ. According to the direction in which we move, we will have to add more or less quadratic terms to the function.

b. We increase the value of δ.

c. We go to the step a. until convergence.

The effect of the penalty term is to create a local minimum near to the constrained minimum of f, for sufficiently large values of δ. For more details, see (Nabney, 1997).

2.4.2 Application

Implementing the constraints using the penalty method is quite straightforward. For example:

- The fact that no connectivity should be negative corresponds to the following statement

$$\text{If} \quad c_{pq} < 0, \quad \text{then} \quad E_{new} = E_{old} + \frac{\delta}{2} c_{pq}^2$$

- the fact that no call should be answered by completely unqualified people:

$$\text{If} \quad S_{pq} = 0 \quad \text{and} \quad c_{pq} > 0, \quad \text{then} \quad E_{new} = E_{old} + \frac{\delta}{2} c_{pq}^2$$

- the number of agents should be greater than c:

$$\text{If} \quad \sum_{q=1}^{Q} \sum_{p=1}^{P} c_{pq} < c, \quad \text{then} \quad E_{new} = E_{old} + \frac{\delta}{2} \left(\sum_{q=1}^{Q} \sum_{p=1}^{P} c_{pq} - c \right)^2$$

Eventually as the solutions *i.e.* the final value of the c_{pq} are real positive numbers, the number of people in each pool p is the nearest integer rounded up from $\sum_{p=1}^{P} c_{pq}$.

2.5 Conclusion

The results computed with such an algorithm are quite good, as shown in Figure (2.1). The total reduction in the workforce was quite high: 25% fewer agents required in a week than with the traditional method.

In a practical sense, the convergence was quite quick and straightforward and we did not need more than 30 minutes to schedule a full week consisting of 120 periods.

Anyway some issues haven't yet been taken into account as:

1. the validation of the hyper-exponential distribution. In a first place this distribution seems too simple to model the true one and a Gaussian distribution may fit it much better. We will check this intuition in Chapter 4.

2. the abandoned calls whose integration could lead to an decrease of the number of staff. Chapter 5 deals with this issue.

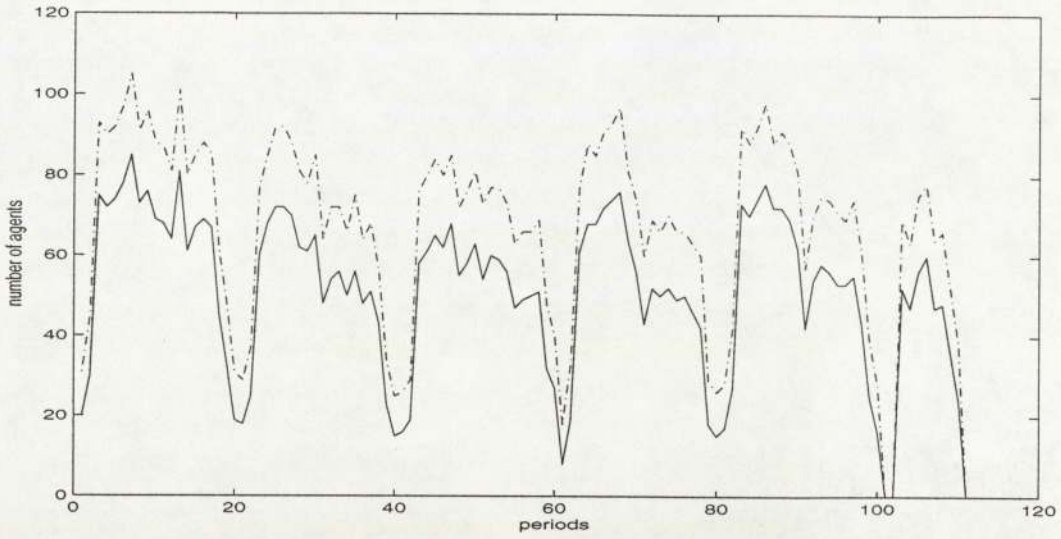


Figure 2.1: Comparison between the number of agents used in the traditional model (the dotted line) and the one used in the stochastic model of the queues (solid line).

But before tackling those problems, investigating the behaviour of the scheduling program seems suitable.

Chapter 3

Investigating algorithm properties

3.1 Introduction

3.1.1 Presentation of the issue

In this chapter we discuss the experience we gained in testing the different programs written last year and exploring the effects of parameter variations on the optimisation.

First we ran the simulation algorithm and laid out some results in detail. Then for the scheduling algorithm, we have tested different values for the Lagrange multipliers. Next we have generated new data by scaling up the existing ones and observing the effect of increasing the call volume, the mean call duration, and the number of pools on the predicted number of agents in our model.
Third as we have noticed that the number of agents increases with the number of pools we have tried to cap it.
Finally we have tested the effect of the multi-skilling.
In the last section we have set out another approach to solving our optimisation problem.

3.1.2 Fixed parameters

To keep the problem realistic and to be able to make good comparisons with Lecorroller's results we have decided:

- not to cap the number of agents in each pool *i.e.* to work with unlimited pool of people in each skills mix.
- to utilise the "usual" skills matrix (generated by Callscan) most of the time (see Table 3.1)

Skills Mix Matrix (unscaled)

Queues	1	2	3	4	5	6
Pool 1	35	100	100	74	0	0
Pool 2	36	0	0	0	100	44
Pool 3	60	0	60	0	0	0
Pool 4	80	0	51	30	100	0
Pool 5	23	0	100	20	77	100
Pool 6	60	0	60	0	0	0
Pool 7	0	100	0	37	0	0
Pool 8	27	58	0	0	69	100
Pool 9	0	100	74	0	59	0
Pool 10	22	0	42	80	51	0
Pool 11	0	0	69	50	0	0

Table 3.1: The eleven skills mixes.

3.2 New simulations

A simulation algorithm was developed last year to model the activity of a call centre. With the predicted number of agents computed in the minimisation algorithm, we try to estimate the service level or the over-waiting time through a week composed of 120 periods of half hour. The over-waiting time can be defined as the percentage of calls which experience a delay superior to this specified by the service level. Because we are going to use it later, we need to define also the quality of service provided to the customers: it is the average competence with which calls have been dealt. A value of 1 will signify that all calls have been answered by fully competent people.

More explanations about this algorithm can be found in G. Lecorroller Thesis.

Our system was composed of 6 queues and 4 pools corresponding to the first four pools of the Table 3.1.

By running the simulation again but with different random files, we didn't find exactly the same results as Lecorroller even using the same data set. However to be able to make relevant comparisons, in the following chapters of our thesis, we are going to present our results.

We used both the agent data file, coming from optimisation procedure and computed so that 95% of calls could be answered within 30 seconds and 38 different random data files, which are using to simulate incoming calls. All the results are a bit worse than those predicted last year. Nevertheless they still remain quite acceptable. With regards to the service level: on average 4.40% of calls are answered after 30 seconds, as shown on Figure 3.1; it is a bit less than twice as much as 2.75% predicted last year. On average now, the service level is not reached in more than 38 half periods amongst the 120 periods as shown on Figure 3.2. However the fact that for some simulations in more than 80 periods also in two thirds of the whole periods the service level is insufficient could seem very worrying but we should bear in mind several things:

- What really matters is the computation of the service level for the whole week and not for every period. Let us consider the following example in order to have a better understanding. Indeed if during 60 periods just two calls are answered after 30 seconds, and if during 30 other periods on average 6 calls are not answered with an acceptable delay, in the first case the number of over waiting period will be much higher than if the second case. But for the managing staff in the second case 180 customers could be considered as dissatisfied compared to the 120 angry clients of the first case and that is the most important thing.

- A last remark can be added: the 80 periods in which the service level is not satisfied, only correspond to 11% of the calls.

In order to explain the bad simulations and the big standard deviation between the data it would be interesting to calculate the real duration of the calls which experiences

delays greater than the acceptable one.

Until now we have focused on the bad cases, but it worth noting that in more than 2/3 of the simulations the average service level is respected.

As expected the quality of service, in Figure 3.3, is not as high as computed last year, but the average skill of an agent answering a call is still equal to 76.6%. The good point is that it remains near constant through the simulation, the fluctuations never exceeding 5%.

In conclusion even though the results obtained by running several simulations were sometimes quite disappointing we are going to use them as a standard in order to compare the validity of further improved models in the next sections.

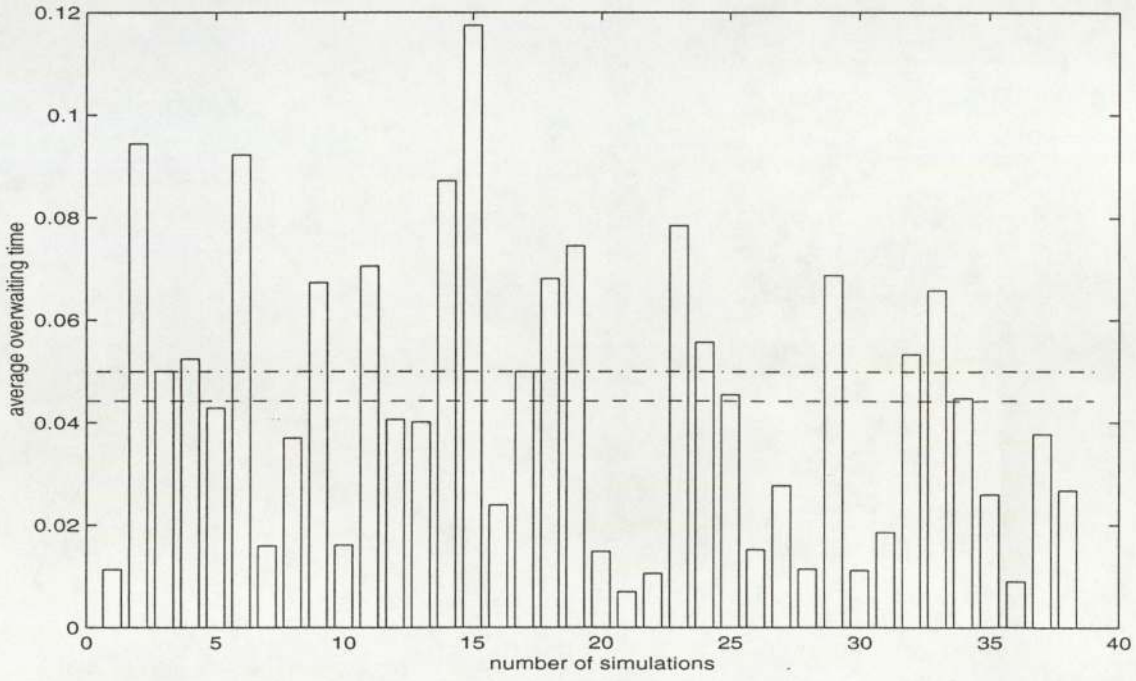


Figure 3.1: Histogram of the average percentage of calls, which experience a delay greater than 30 seconds before being answered throughout a whole week. Each interval corresponds to a simulation. The dotted line represents the mean of the data, where the dotted-dashed symbolises the 5% bar.

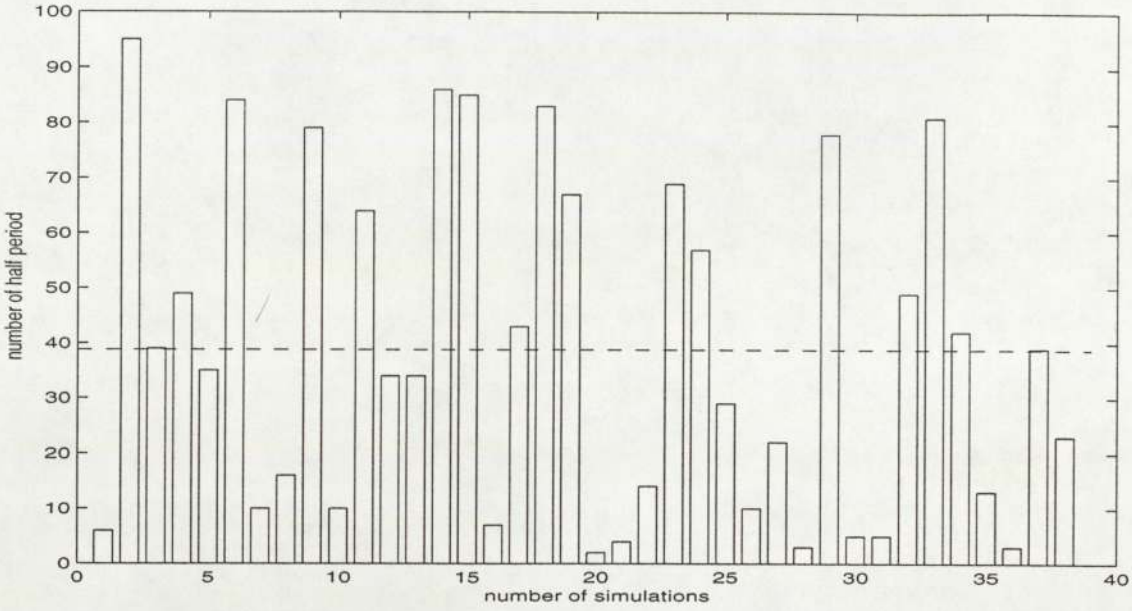


Figure 3.2: Histogram of the number of periods per week in which the service level is not reached. Each interval corresponds to a simulation. The mean number of periods is represented by the dotted line.

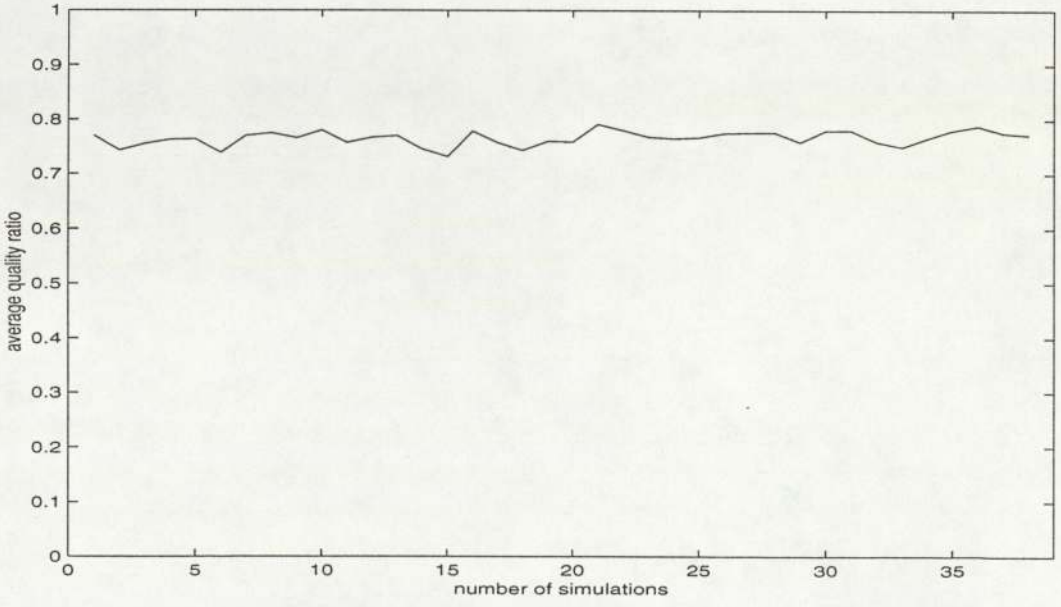


Figure 3.3: Service quality provided against experiments

3.3 Testing different values for the Lagrange multipliers

Recall that the cost function can be written as:

$$E = E_0 + \beta\ E_1 + \gamma\ E_2 + \delta\ E_3 \tag{3.1}$$

where β, γ, δ are Lagrange multipliers.
Considering 4 pools and 6 queues we varied the coefficients amongst 3 values $\{0.1;1;10\}$ and noticed that:

- the quality ratio doesn't change significantly, only 1.3%. To be precise it fluctuates in the interval $[0.6157; 0.6243]$ (1 indicating perfect quality).
- the number of required agents fluctuates less than 4%.

Since these changes do not affect the results significantly, and as modifying the value of these coefficients will change the relative weight of the parameters which make up our problem and also the meaning of the problem itself, we have kept the coefficients chosen last year: $\beta = 10$, $\gamma = 1$, $\delta = 1$ (see Lecorroller Appendix D p 114).

3.4 Increasing the traffic charge

Last year, some experiments were carried out with a fixed number of pools:

- increasing the call volume;
- increasing the average call duration;
- increasing the call volume and the average call duration.

The reduction in the number of agents computed with the stochastic model compared to the Erlang C model decreased as a percentage, whereas it increased in term of agents.

We decided to investigate in more detail. First we have increased the initial call volume by 2, 5 then the mean call duration by 2, 5 and then both by 2, 5. In each

case, we have raised the number of pools from 4 to 11. But unfortunately no linear relationship appeared.

We have also carried out the same experiments but for a given number of pools, 6, by changing the coefficients of the skills matrix: the results remained disappointing.

3.5 Increasing the number of pools

Even if the experiments on the traffic charge didn't give convincing results, we noticed that the number of agents increases with the number of pools. Therefore we have decided to work out this problem. For practical reasons and to be quite realistic we have considered an increase from 4 to 11 pools with all other factors held constant.

400 more people were predicted with 11 pools than with 4 pools with a given call volume and a given mean call duration as we can see from Figure 3.4.

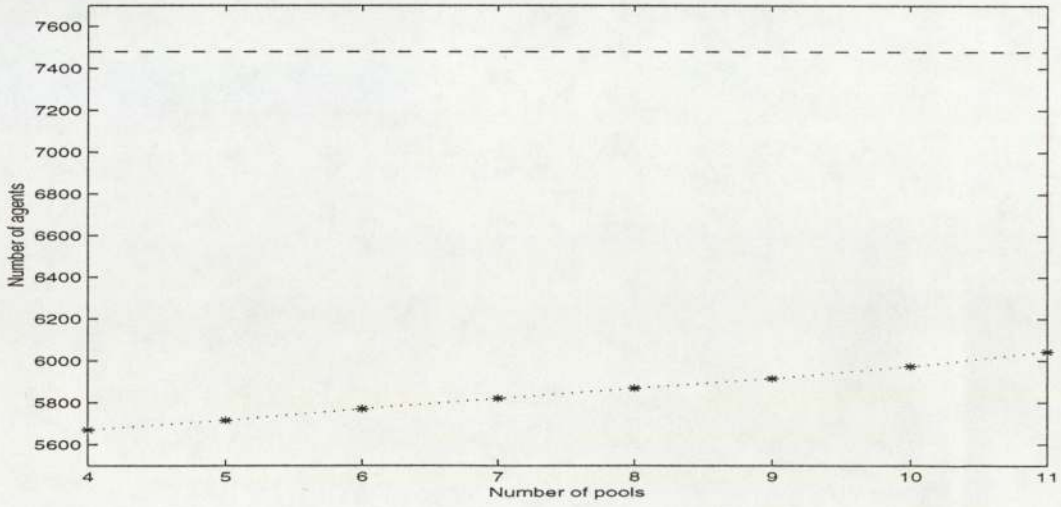


Figure 3.4: Evolution of the number of agents used in the traditional Erlang C model (dashed line) and the number of agents used in our approach (stars) with the increase of pools.

With regard to the quality ratio, as shown in Figure 3.5, after a gain of 10% with 5 pools, we denote a stabilisation. We conclude that in this case increasing the number of pools used more staff without any increase in quality of service.

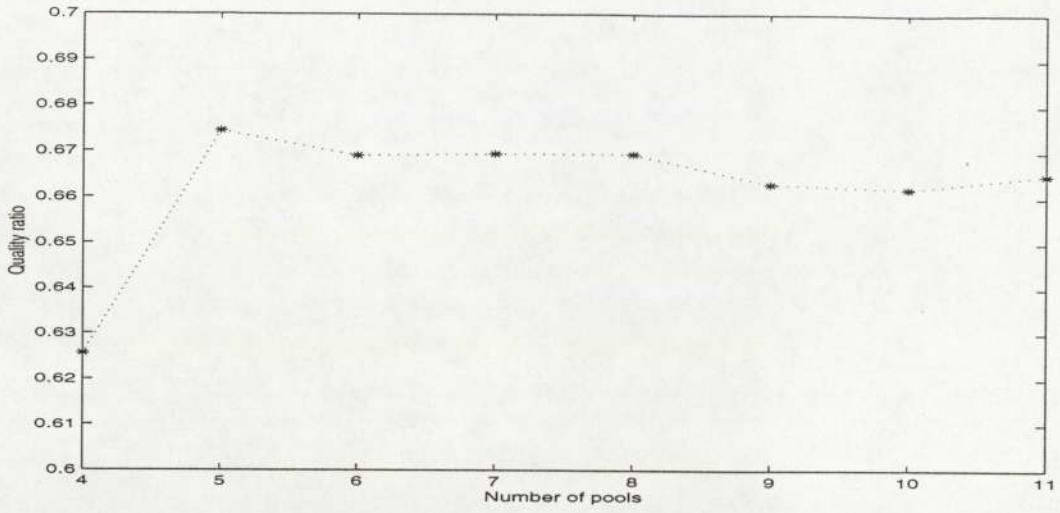


Figure 3.5: Evolution of the quality service with the increase of the number of the pools

3.5.1 A rounding problem

The explanation of this problem is quite simple: it is a rounding problem.

We have already explained the two steps of the algorithm:

1. First we compute the minimum number of agents required in each queue for each period in order to achieve a satisfactory service level. In fact as the different queues are merged into one, thanks to the approximation formulae, we derive an integer number for each period which is completely independent of the number of pools.

2. In the second stage by using the scg algorithm we share this number between the different pools and for each period. In most cases real numbers are obtained for the coefficients c_{pq} and rounded up to the nearest greater integer.

Consider an example taken from the results above.

Let us consider two cases: in the first case there are 4 pools in the system, and in the second 11. Now if we suppose that the minimum number of agents required to achieve the correct service level is *8* for one period, we have to share them in the different pools, using the scg method. In the first case after rounding, *10* people will be necessary, whereas for 11 pools *13* agents will be predicted, as shown in Table 3.2 and in

Table 3.3. Thus just for one period and for the same call volume, 3 more people will be required with 11 pools than with 4 pools.

The way to solve this issue poses a difficult problem for us. Finally after many unsuccessful attempts which we will explain later, we managed to find an empirical solution which nevertheless gives quite good results. It is the subject of the next section.

3.5.2 Solution to the rounding problem

In the following we will always use two kinds of rounding: the rounding up towards the nearest greater integer (for example 1.4 will be rounded to 2), and the rounding towards the nearest integer (still for the same example, 1.4 will be rounded to 1).

In the optimisation algorithm so far all numbers were rounded toward the greater integer. So the main issue is now *not to round all* the numbers to the nearest integers towards infinity but only a few and therefore to round the others towards the nearest integers whilst preserving the quality factor.

Let us explain the principle of our solution. We run the optimisation algorithm twice. The first time, we compute the agents required for a determined number of pools, we will consider as a reference, (in our example the standard will be 4 pools). And we store them without any rounding in a first matrix. Then we repeat the operation, for the desired number of pools (still in our example those were the 11 pools) and store them in a second matrix without carrying out any rounding. At this stage in our algorithm there are no rounding errors.

Then for each period we round the numbers of the second matrix, whose quality factors are the greatest- see Lecorroller scheduling algorithm code- toward the nearest greater integers and the other numbers simply toward the nearest integers. In this way we manage to keep a good quality factor.

	True numbers	Rounding numbers
Pool 1	2.33	3
Pool 2	3.33	4
Pool 3	1.34	2
Pool 4	1	1
Total	8	**10**

Table 3.2: Results with 4 pools before rounding (true numbers) and after rounding (rounding numbers)

	True numbers	Rounding numbers
Pool 1	0.6	1
Pool 2	1.4	2
Pool 3	1.2	2
Pool 4	0.2	1
Pool 5	0.6	1
Pool 6	0.6	1
Pool 7	0	0
Pool 8	0	0
Pool 9	0.4	1
Pool 10	1.5	2
Pool 11	1.5	2
Total	8	**13**

Table 3.3: Results with 11 pools before rounding (true numbers) and after rounding (rounding numbers).

Practically, the way to handle each period can be described:

- we classify the real numbers of the second matrix, each corresponding to one pool, in ascending order according to the value of their quality factor.
- we round the numbers (real) of the second matrix, the ones after the others, toward the nearest greater integers, and we stop roughly when the sum, of all the numbers is superior to the sum, rounded to the nearest integer, of the numbers belonging to the first matrix of reference.
- if there are numbers of the second matrix left which haven't been rounded towards the nearest greater integer, we round them, but towards the nearest integer this time.

Thus we manage to cap the number of people in each period even with lots of pools.

3.5.3 Results and validation

The results are depicted in Table 3.4. We have represented the number of people required according to the new method against the one computed with the former algorithm. With this new approach and throughout a week only 32 more people are predicted with 11 pools than with 4 pools. The capping is also quite good because 400 more people were necessary with 11 pools than with 4 pools before.

To validate more carefully our new approach we have run the algorithm that simulates the activity of a call centre for a whole week. These results are the average of 10 representative simulations each run with a different random seed.

These results are quite good, see in Table 3.5. We noticed that there is a gap, both for the over waiting time and the quality ratio between the 4 pools system and the other systems component of more pools. We can easily explain this phenomenon by the presence of both more agents and agents 100% competent to deal with calls coming from the queue 6, as shown on Table 3.1.
In a nutshell we won't draw any sort of general conclusion between the pools themselves in particular, because it is a question of skills competencies. So we will content

	Staffs with the new method	Staffs with the old method
4 pools	5670	5670
5 pools	5671	5716
6 pools	5680	5770
7 pools	5698	5822
8 pools	5699	5870
9 pools	5708	5916
10 pools	5714	5974
11 pools	5704	6044

Table 3.4: Predicted number of people with the new and the old approach for 4 until 11 pools

	Over waiting time	Quality ratio
4 pools	4.6	76.4
5 pools	3.4	83.9
6 pools	3.8	83.0
7 pools	3.7	82.6
8 pools	3.0	82.1
9 pools	3.0	81.8
10 pools	3.1	82.5
11 pools	3.4	81.8

Table 3.5: Average over waiting time, *i.e* the percentage of the calls which experiment a delay superior to this specified by the service level and average quality ratio for different number of pools.

with noticing that the service level is satisfied in all the cases, and that the quality ratio slightly improves with the increase of the number of pools.

Some remarks need to made. First, even if the capping seems quite good, other experiments should be carried out with different skills matrices to really ensure that the method works in every case. Second we can't assert that our solution is completely rigorous because we need in the first stage to compute the workforce required for a determined number of pools, which will use as a reference later. Shall you take 3, 4 or 5 pools in the first step as reference pool?

3.5.4 Other approaches which do not work

To resolve the problem, the first idea we tried, was to integrate a new constraint in the optimisation algorithm under the form of a penalty term:

$$\text{If} \quad \sum_{q=1}^{Q} c_{pq} \notin \mathbb{N}, \quad \text{then} \quad E_{new} = E_{old} + \frac{\delta}{2}\left(\sum_{q=1}^{Q} c_{pq} - \left[\sum_{q=1}^{Q} c_{pq}\right]\right)^2$$

where [x] is the integer part of x.

But unfortunately despite the running of hundred simulations, we got no convincing results. Even by using different penalty functions, let them move, by combining in the algorithm both new and older constraints and using several loops, no significant decrease occurs.

Eventually we had to give up and lost a lot of time, because each simulation was very costly in terms of time.

3.6 Testing the effect of multi-skill pools

We have carried out experiments to investigate the behaviour of the multi-skilling, *i.e.* when each pool could answer several queues. We wanted to confirm the results obtained last year by Lecorroller. He noticed, by using a modified skills matrix (see (Lecoroller, 1997), p83) that is to say, when each incoming call could be allocated at least to two pools of agents, that the fraction of over waiting calls decreases compared

to the ones computed with the usual skills matrix see in Table 3.1. By considering 6 pools and 6 queues, we ran the "simulation" program 6 times, when each pool could answer from 1 to 6 queues: in the first simulation we considered that each pool had the full competency to answer 1 queue, then in the second simulation that each pool could answer two queues, one with a full competency and the other one with half competency et cetera.

We noticed that both the quality ratio and the over waiting time decreased with the multi-skilling. Unfortunately in spite of those observations nothing could be derived, because other simulations with different random seed and a different number of pools gave contradictory results. The dependency of the data skills on the fluctuation of the service level and of the quality ratio seemed obvious. Therefore no conclusion as for the best number of queues that a pool has to answer can be drawn. All the more it depends on the importance we give either on the quality ratio or to the over waiting time. We can just confirm that it seems preferable that a queue can be answered at least by two pools.

3.7 A quadratic programming problem

In the previous chapter, we showed how to solve the problem of human allocation resource by using the scg algorithm and the penalty method.

As the cost function contains quadratic terms and as all constraints except one are linear, we have investigated whether we could transform the problem into a quadratic programming problem, with just linear constraints. That is to say to be able to rewrite the problem in the following form:

$$\begin{cases} \text{Min} \quad \frac{1}{2}\, x^t H x + c^t x \\ \text{under constraints} \quad Ax \leq b \end{cases} \tag{3.2}$$

where x, b are vectors and H, A are matrices.

The function to minimise modulo a multiplier factor $1/Q$, see section (2.3.3), can be defined by:

$$f: \quad \left(\mathbb{R}^+\right)^{P\times Q} \longrightarrow \left(\mathbb{R}^+\right)$$
$$(c_{pq}) \longmapsto \frac{1}{Q}\sum_{q=1}^{Q}\left(\sum_{p=1}^{P} c_{pq}S_{pq} - n_q\right)^2 + \beta\sum_{q=1}^{Q}\left(\sum_{p=1}^{P} c_{pq} - n_q\right)^2 + \gamma\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq}^2 + \delta\left(\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq} - c\right)^2$$

It can easily be rewritten, modulo a multiplier factor, within the required form:

$$\underbrace{\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq}^2\left(S_{pq}^2 + \beta + \gamma + \delta\right) + 2\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq}\left(\sum_{p'\neq p} c_{p'q}\left(S_{p'q}S_{pq} + \beta\right) + \sum_{q'\neq q}\sum_{p'\neq p} c_{p'q'}\right)}_{\frac{1}{2}x^tHx}$$
$$\underbrace{+2\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq}\left(-S_{pq}n_{pq} - \beta n_q - \delta c\right)}_{c^tx} + \underbrace{\left(\sum_{q=1}^{Q} n_q^2\left(1+\beta\right) + \delta c^2\right)}_{constant}$$

We can also make two remarks. First we can identify the c_{pq} variables with the x variables in the previous equation. Secondly, as a constant doesn't change anything in a minimisation problem, this equation has the correct form.

With regard to the constraints, all are linear except one:

- the fact that all connectivity should be positive:

$$c_{pq} \geq 0 \quad \forall (p,q)\epsilon[1,..,P]\times[1,..,Q]$$

- the fact that the number of agents in the system should be greater than c:

$$\sum_{q=1}^{Q}\sum_{p=1}^{P} c_{pq} \geq c$$

- even the optional constraint to cap, if necessary, the number of people in some skills mixes is linear:

$$\sum_{q=1}^{Q} c_{pq} \leq A_p$$

The non linear constraint is that no call should be answered by people completely unqualified, *i.e.* if $S_{pq} = 0$ then $c_{pq} = 0$, can be suppressed by removing the variables c_{pq} whose associated S_{pq} are zero.

In this way we can transform the problem into a quadratic programming problem with linear constraints. We have not implemented it in this fashion because quadratic programming algorithms are complex, and using a general purpose optimisation algorithm (scg) with the penalty function method was adequate for all our experiments.

Chapter 4

Refinement of the stochastic model

4.1 Introduction

In Chapter 2 we have presented briefly the stochastic model of the queues. Strong assumptions have been made that we are going to test thanks to the availability of much more detailed data. Firstly we will be interested in the inter-arrival time distribution and secondly in the service time distribution.

Although the data provided by Callscan is from a single queue and comes from a different centre than the one used last year, we can perform some relevant investigations. Indeed the above-mentioned distributions have general characteristics independent of the call centre, that we can show by exploiting several types of information for each call:

- the nature of the call, either normal calls or calls abandoned in the ring or in the queue;
- the time the call occurred;
- the time the call waited before being answered or the caller abandoned;
- the call duration for answered calls;
- the group which was intended to service the call, -1 indicates the group was unknown.

From this information, we have computed the time between each arrival *i.e.* the inter-arrival time, and the service time. An anomaly was detected. Too many service times equalled zero, at least one third of all the data. So we have recomputed a realistic number of zeros by using the mean of the data. In fact this computation seems all the more justified since the presence of excessive zeros can only be considered as some 'noise' or error in recording the data. Moreover answered call whose duration is null, requires no agents and can also be partially removed from the sample data.

4.2 The inter-arrival distribution

4.2.1 Review

Last year, we assumed that there were several queues indexed by q, each one described by the arrival rate λ_q. We also decided to model the arrivals of the calls from the different queues as a single Poisson process whose arrival rate parameter λ was equal to $\lambda = \sum_{q=1}^{Q} \lambda_q$ and for obvious reasons we also modelled the inter-arrival time of calls as an exponential distribution.

Thanks to the new data we are going now to investigate the inter-arrival service time distribution.

4.2.2 Results

As expected, an exponential distribution with mean computed from data is an excellent fit, as shown by Figure (4.1). Hence there is no need to change our model for the inter-arrival distribution.

4.3 The service time distribution

4.3.1 Review

In our model the service time distribution for each call was exponentially distributed with different parameters for each queue q:

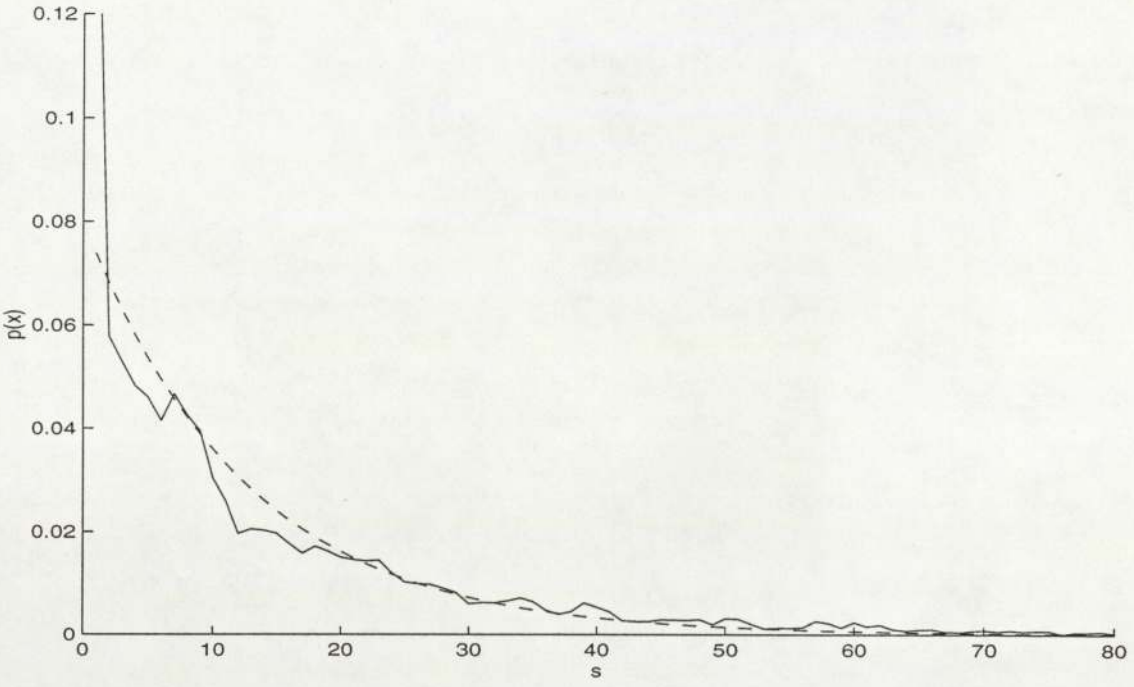


Figure 4.1: Real inter-arrival time distribution (solid line) against exponential distribution (dotted line) with mean computed from the data.

- the service rate μ_q
- and the probability an agent answers a call of type q, $\alpha_q = \frac{\lambda_q}{\sum_{q=1}^{Q} \lambda_q}$.

We also assumed that all the queues could be merged in one, using a Q-stage hyper-exponential distribution, the density function of this hyper-exponential service time distribution being equal to

$$f_s(t) = \sum_{q=1}^{Q} \alpha_q \mu_q e^{-\mu_q t} \quad \text{with} \quad \alpha_q = \frac{\lambda_q}{\sum_{q=1}^{Q} \lambda_q} \tag{4.1}$$

4.3.2 First results

If we were quite confident with regard to the inter-arrival time distribution, there was less reason to expect from the first principles that the service times should have an exponential distribution. Therefore we investigated other parametric distributions such as the Gamma distribution, Erlang-k distribution, and Gaussian distribution to see if

they were more appropriate to fit the service time. Their analytical definition is given in Appendix B.3.4.

But manifestly, see Figure 4.2, if it is not really an exponential distribution, (see Figure 4.3(a)), it is less likely to be these others, as shown in Figures 4.3(b), 4.3(c) and 4.3(d).

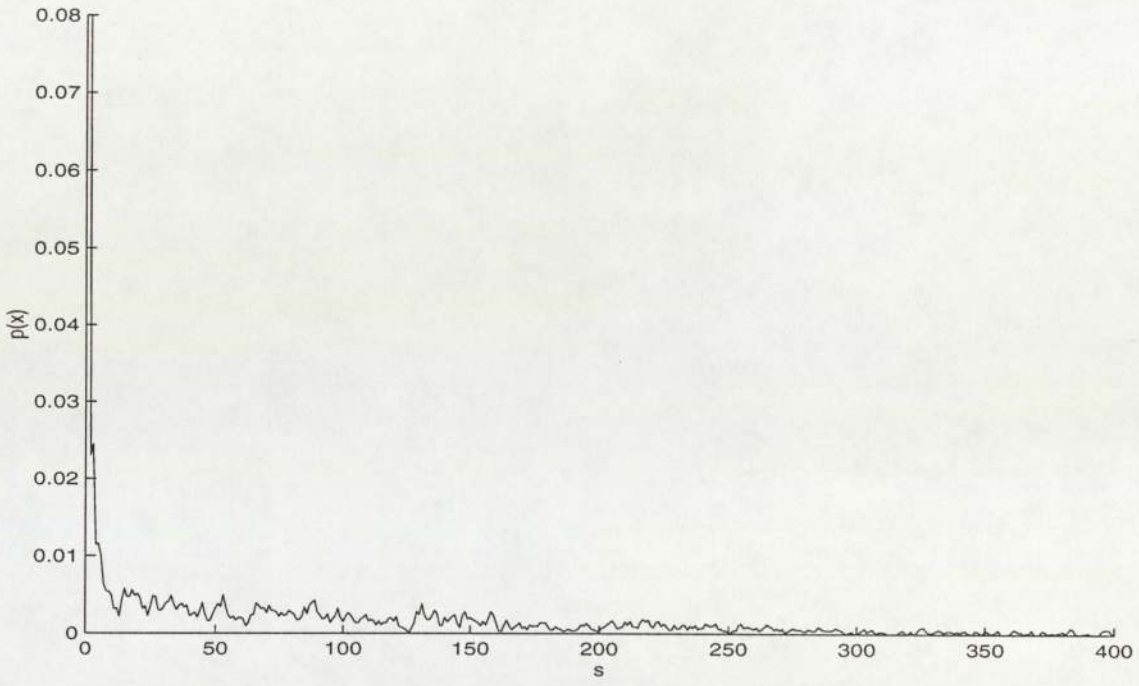
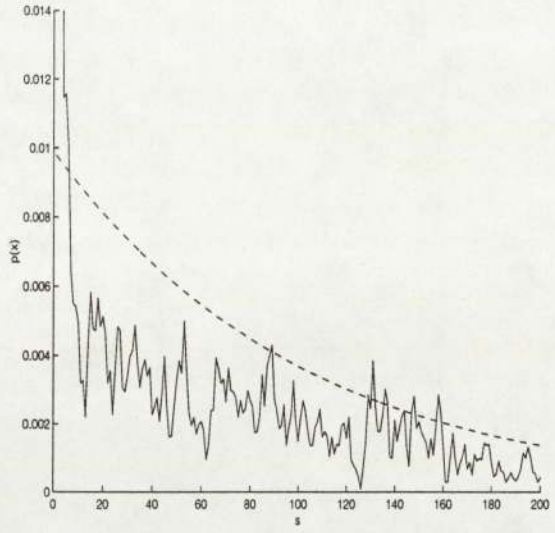


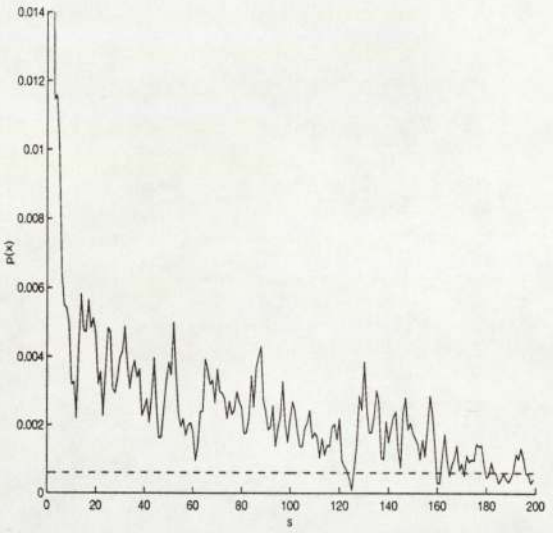
Figure 4.2: Service time distribution.

As the sample data of the service time distribution has a large standard deviation relative to the mean value, $standard/mean = 6.5$ the hyper-exponential distribution seemed the most likely near to the service time distribution. Therefore we decided to try to fit this by a mixture of exponentials.

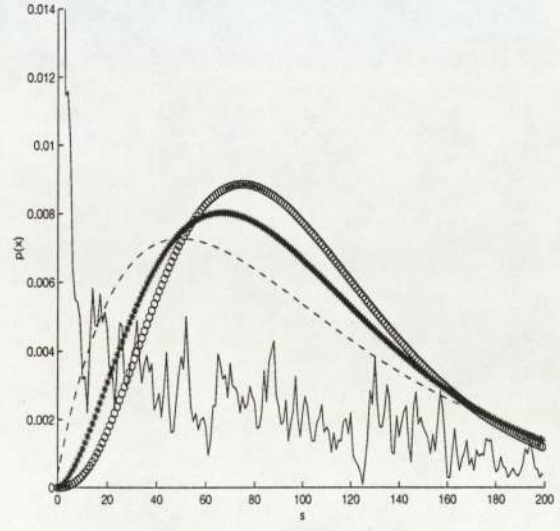
Thus in the next section we will introduce the theory of the mixture model and describe how those parameters can be estimated by the EM algorithm in the case of a mixture of exponential distributions. In the following section after we will present the new model we had to create to take the new service time distribution into consideration in the optimisation procedure and the results which ensue.



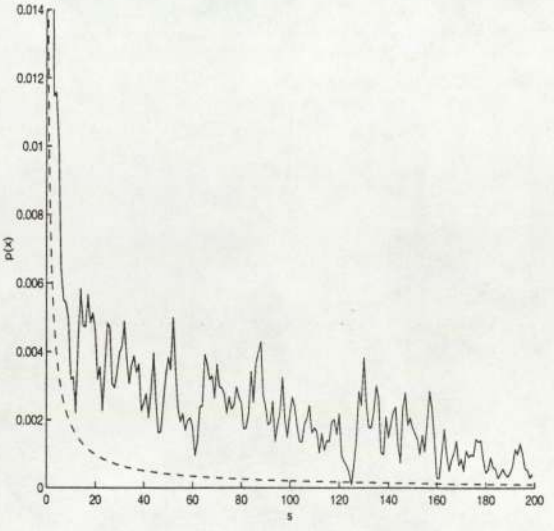
(a) Exponential distribution (dotted line) against true values (solid line).



(b) Gaussian distribution (dotted line) against true values (solid line).



(c) Erlang-2 (dotted line), Erlang-3 (stars line), Erlang-4 (circle line) against true value (solid line).



(d) Gamma distribution (dotted line) against true value (solid line).

Figure 4.3: Parametric fits to the service time distribution.

4.3.3 Mixture models and EM algorithm

Mixture models

A mixture model represents, see (Bishop, 1995), an underlying density function as a combination of M simpler density functions:

$$p(\mathrm{x}) = \sum_{j=1}^{M} P(j)p(\mathrm{x}|j) \tag{4.2}$$

where the $P(j)$'s are the *mixing coefficients* or *priors*, which satisfiy the following constraints:

$$\begin{cases} 0 \leqslant P(j) \leqslant 1 \\ \sum_{j=1}^{M} P(j) = 1 \end{cases} \tag{4.3}$$

and where $p(\mathrm{x}|j)$ is the probability that the random value x is generated by the component j. This is also normalised: $\int p(\mathrm{x}|j)\, d\mathrm{x} = 1 \quad \forall j,\, j = 1, \ldots, M$.

In our case, to model the service time distribution by a mixture of exponentials we choose the density functions $p(\mathrm{x}|j)$ so that $p(\mathrm{x}|j) = \lambda_j.e^{-\lambda_j \mathrm{x}}$, λ_j is the inverse of the mean of the component j.

Then fitting the model to data is a question of estimating the unknown parameters *i.e.* $\{P(j), \lambda_j,\ j = 1, \ldots, M\}$. We can do this using the Expectation-Maximisation algorithm.

Expectation-Maximisation algorithm

The EM algorithm, see (Bishop, 1995), is an iterative algorithm used to determine the parameters of a mixture model.

Consider a data set of N values $(x_1, x_2, \ldots, x_N)$. We want to find the mixture model, see equation (4.2), which describes the distribution of this data set. To determine the parameters of this exponential mixture model, making the usual i.i.d. assumption, we can maximise the likelihood defined by: $\mathcal{L} = \prod_{n=1}^{N} p(x_n)$. In fact it is

equivalent to minimising the negative likelihood:

$$E = -\ln \mathcal{L} = -\sum_{n=1}^{N} \ln \Big\{ \sum_{j=1}^{M} P(j)p(\mathrm{x}_n|j) \Big\} \tag{4.4}$$

To simplify the notation, we let $\theta = \{\theta_j,\ j = 1, \ldots, M\} = \{P(j), \lambda_j,\ j = 1, \ldots, M\}$

Principle

We have talked about an iterative algorithm: during the optimisation procedure, we shall adapt the parameters in order to increase the likelihood $\mathcal{L}$. So we define the current value of the parameter vector θ, to be θ^{old} and its new value θ^{new} after modification.

We have therefore to minimise a new quantity:

$$\mathcal{E}^{comp}(\theta^{new}) = -\sum_{n=1}^{N}\sum_{j=1}^{M} P(j|x_n, \theta_j^{old}) ln\{P(x_n|j, \theta_j^{new}) P^{new}(j)\} \tag{4.5}$$

The justification of this new formula is given in Appendix A.

As for the principle of the EM algorithm, it alternates two steps: the E-step and the M-step (We will later discuss its initialisation).

The **E-Step** or "expectation" step computes the expected complete-data log-likelihood $\mathcal{E}^{comp}\ (\theta^{new})$ with respect to the posterior probabilities $P(j|\mathrm{x}_n)$ for every data point x_n and every component j, using the current, fixed, values of the parameters λ_j^{old}, $P^{old}(j)$:

$$P(j|\mathrm{x}_n) = \frac{p(\mathrm{x}_n|\lambda_j^{old}) P^{old}(j)}{\sum_{j=1}^{M} p(\mathrm{x}_n|\lambda_j^{old}) P^{old}(j)} \tag{4.6}$$

The **M-Step** or "maximisation" step determines new, re-estimated, values of the parameters θ^{new} by maximising the value of $\mathcal{E}^{comp}\ (\theta^{new})$ computed in the E-step. In the case of an exponential mixture model the updating relations are:

$$\lambda_j^{new} = \frac{\sum_{n=1}^{N} P(j|x_n, \theta_j^{old})}{\sum_{n=1}^{N} P(x_n|j, \theta_j^{new}) x_n} \tag{4.7}$$

$$P^{new}(j) = \frac{1}{N} \sum_{n=1}^{N} P(j|x_n, \theta_j^{old}) \tag{4.8}$$

Proofs and details can be found in Appendix A.

Then we repeat the E-step and the M-step until our algorithm converges. It is worth noting that both steps guarantee the increase of the likelihood, *i.e.* the decrease of the negative log-likelihood unless it is already at the maximum. We can use it with confidence although sometimes we find sub-optimal local maxima or degenerate cases (when the variance of a component shrinks to zero).

Initialisation

The optimum parameters depend on the starting points of the equations (4.7) and (4.8). As we haven't any information *a priori*, we initialise the priors P(j) to $\frac{1}{M}$ for all j, and choose randomly the λ_j in the interval [0,1].

4.3.4 Data's problem

The practical implementation of this algorithm is quite easy and after a period of tests ensuring our model was working with basic examples, we have tried to compute the priors and the lambda parameters from our sample of data. But it didn't work at all. After a few steps at least one lambda parameter tended to infinity. This was due to the presence of too many zeros, one third of all the data values, among the data of the service time distribution. After discussion with CallScan we decided that most of these zeros were erroreously recorded.

To resolve this issue we computed a more realistic number of zeros by assuming that the service time distribution followed an exponential distribution with parameter the inverse of the mean of the data *i.e.* fitted an exponential to calculate an approximate frequency of zeros.

In practice we just had to solve the following equation:

$$P(0 \leq X \leq 1) = \frac{\text{number of calls whose duration equals to zeros}}{\text{total number of calls}}$$

i.e.
$$\int_0^1 \lambda e^{-\lambda t} dt = \frac{n_0}{n_0 + n_1}$$

i.e.
$$1 - e^{-\frac{(n_0+n_1)}{\sum_i X_i}} = \frac{n_0}{n_0 + n_1}$$

The definitions are listed below:

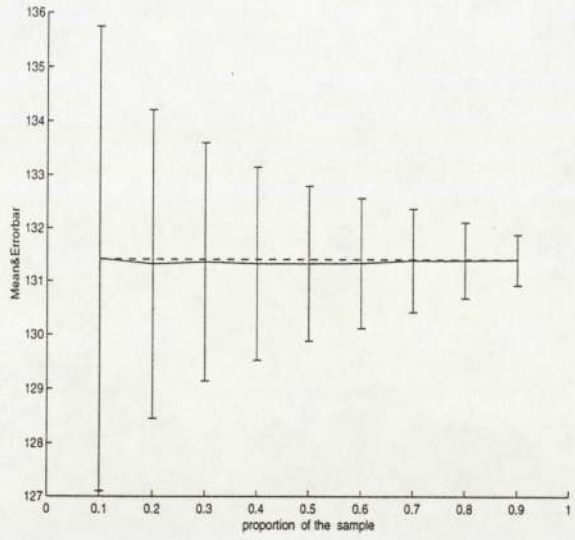
- X, X_i are the random variables corresponding to the service time duration
- n_0 the number of calls whose service time is equals to 0
- n_1 the number of calls whose service time is $\geqslant 1$
- λ the parameter of the service time distribution is equal to the inverse of the mean m, defined by : $\sum_i X_i/(n_0 + n_1)$

And as result the predicted number of zeros is now 132 compared to 8300 from the original data. In the same time, we removed data with a very large service time, that is to say, calls lasting more than 2 hours: they are very few (0.17% of the total sample) indeed and are erroreous records.

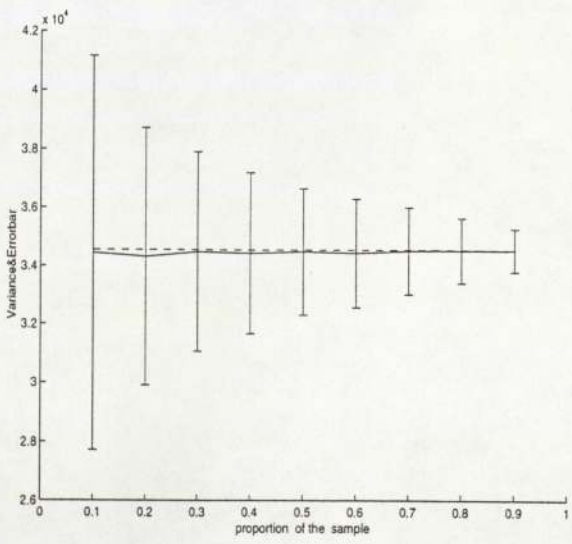
An interesting point to investigate now is the consistency of the data: it is all the more important to check that the mean and the variance of the data are stable as lots of theorems in the queueing theory rely on them.
Thus we have computed, see Figures 4.4(a) and 4.4(b), the error bar of 10%, 20%,..., 90% our sample. The results, in each case, are the average of 1000 samples randomly chosen among the whole data and with the required proportion. It is obviously that the average values of the mean and the variance for each sub-sample fit very well the mean and the variance of the whole data. Therefore we can conclude that our sample is consistent.

4.3.5 Results

We computed the coefficients of the mixture model for the new data. We also had to determine the complexity of the model, *i.e.* the number of mixture components.



(a) Mean error bar. The dash line represents the mean computed with the whole data.



(b) Variance error bar. The dash line represents the variance computed with the whole data.

Figure 4.4: Representation of the data error bars for the mean a), for the variance b). Each error bar has been computed with 1000 sub-samples. The solid line joints the average of the value which range over the error bars.

Choice of the number of components M

To choose the required number of mixture components, see equation (4.2), we computed the error per point for different numbers of parameters: this is simply the negative log-likelihood per data, see Figure (4.5).

The kink in the graph at 3 components indicates that this is a reasonable choice, in that adding more components does not significantly affect the data likelihood.

Fit the service time distribution

The fit of the service time distribution with a mixture model of three exponentials is shown in Figure (4.6(a)). There is no doubt that it fits much better the real service time data, particularly at small values, than a single exponential as shown as in Figure (4.6(b)).

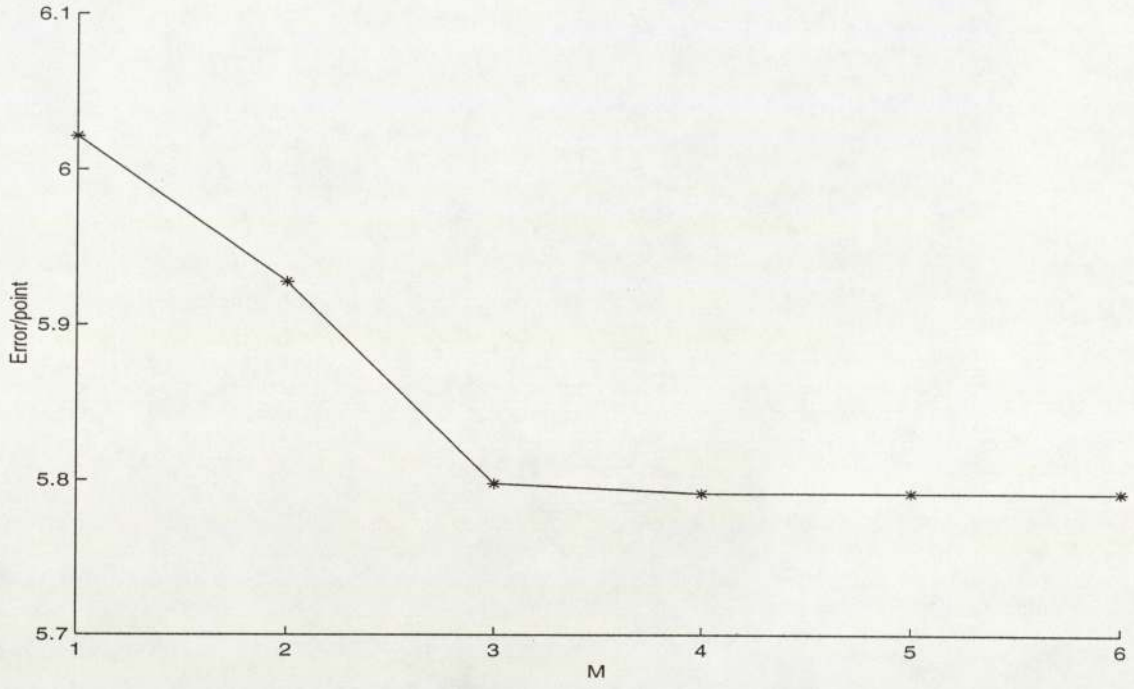


Figure 4.5: Number of components vs Log-likelihood per point

4.4 New queueing model

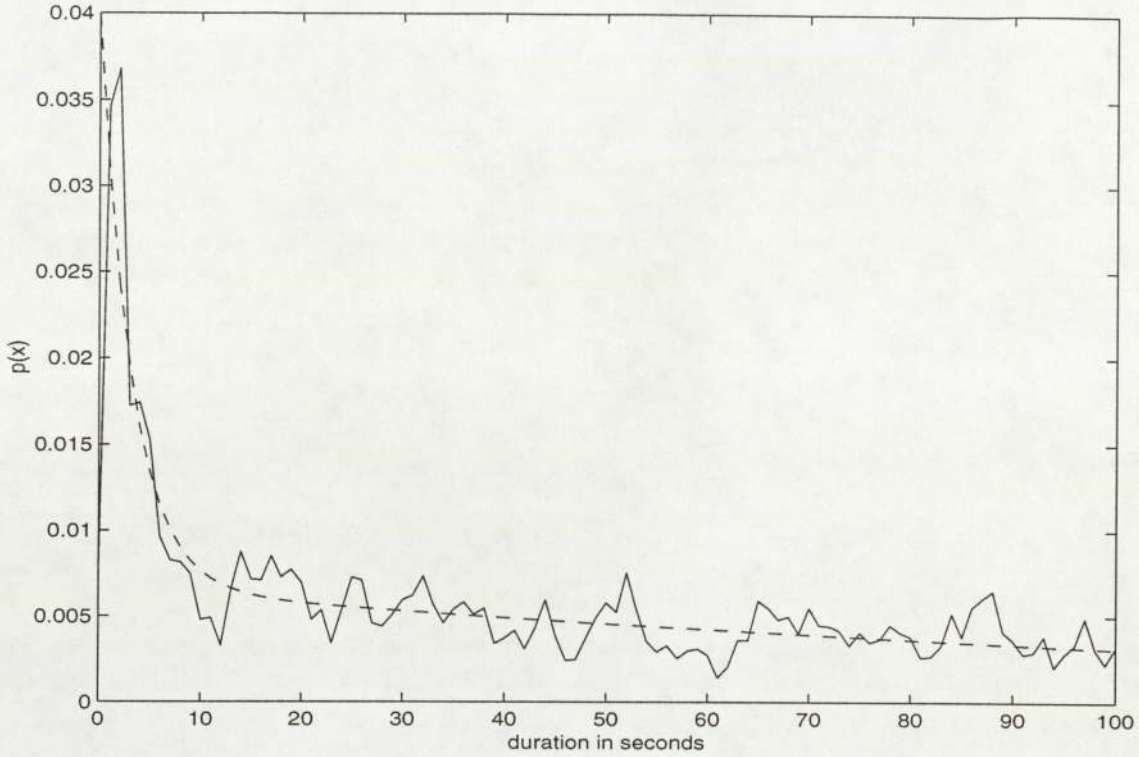
4.4.1 New service time distribution

In this section we shall integrate the new service time distribution in our stochastic model of the queues. In the last paragraph we showed with new data from a single queue, that the service time distribution could be approximated well by a mixture of exponentials.

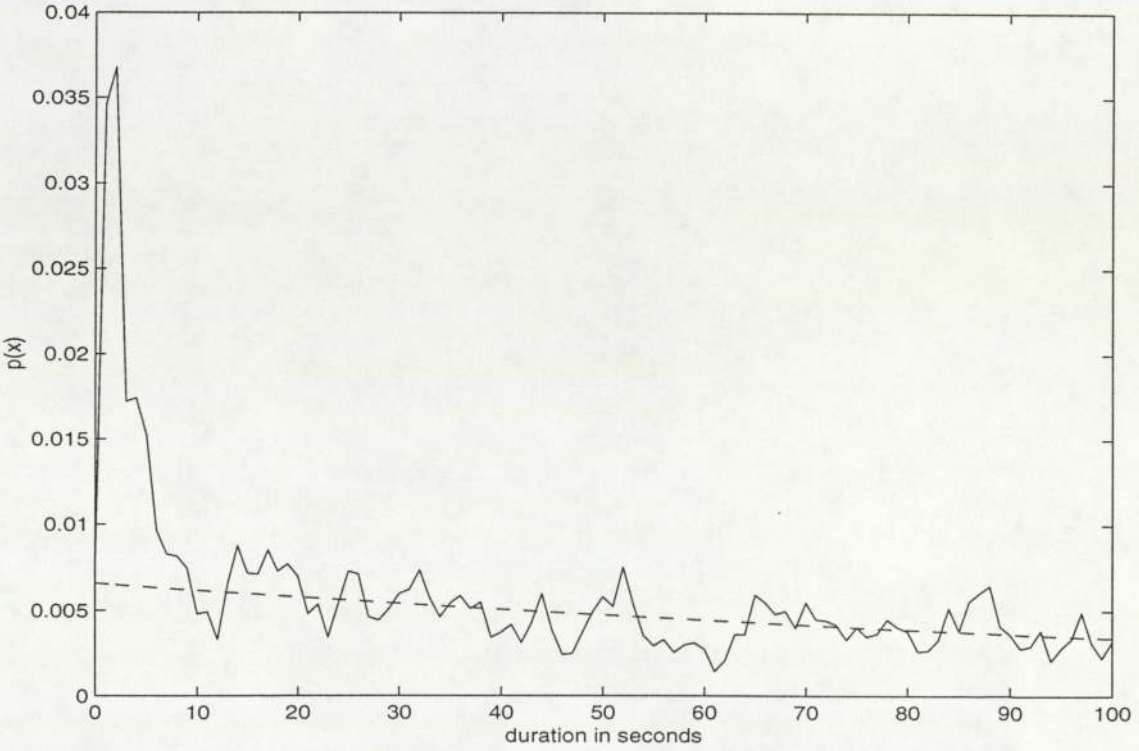
Since there are several queues, we have to make an assumption: we suppose that each queue has a service time distribution whose density function follows the mixture density of exponentials we have computed earlier, but with suitable scaling.

Let us explain in more detail what it means. We have just seen that the density function of a mixture model of exponentials was given by:

$$\tilde{f}(t) = \sum_{j=1}^{M} \tilde{P}(j)\tilde{\mu}_j e^{-\tilde{\mu}_j t} \tag{4.9}$$



(a) Service time distribution (solid line) fitted with a mixture of 3 exponentials (dotted line).



(b) Service time distribution (solid line) vs single exponential (dotted line).

and so we can derive the mean of this distribution, m:

$$m = E[X] = \sum_{j=1}^{M} \frac{\tilde{P}(j)}{\tilde{\mu}_j} \tag{4.10}$$

We now substitute, for each queue q, the former service rate μ_q $(= \frac{1}{T_q})$, equal to the inverse of the mean duration of a call in a period, see review and equation (4.1), by a M-vector $(\mu_{q1}, \ldots, \mu_{qM})$ where the jth element μ_{qj} is equal to:

$$\mu_{qj} = a_q \, \tilde{\mu}_j = \frac{m}{T_q} \, \tilde{\mu}_j \tag{4.11}$$

Thus we can derive a new service time distribution with mixture model for our Q queues system. The density function is then equal to:

$$f_s(t) = \sum_{q=1}^{Q} \alpha_q \left(\sum_{j=1}^{M} \tilde{P}(j) \tilde{\mu}_{jq} e^{-\tilde{\mu}_{jq} t} \right) \tag{4.12}$$

with $\alpha_q = \frac{\lambda q}{\sum_{q=1}^{Q} \lambda_q}$.

This shows an advantage of using a mixture of exponentials, as the density function is still hyper-exponential. Let us come back to the formula (4.11). Previously we only had the mean of the service time distribution T_q for each half hour period and each queue q. Now thanks to the new data, we have represented the service time distribution amongst a week, and the mean $m = \sum_{j=1}^{M} \frac{\tilde{P}(j)}{\tilde{\mu}_j}$ in case of a single queue. We want to scale the mean of the mixture of exponentials so that the mean of the mixture model is the same as μ_q. Thus if we suppose that the service time for each period and for each queue has the same distribution that for a week modulo a multiplier coefficient a_q, we can deduce that

- in terms of mean:

$$T_q = \frac{m}{a_q} \qquad \text{i.e.} \qquad a_q = \frac{m}{T_q}$$

- in terms of service rate :

$a_q \, (\tilde{\mu}_1, \ldots, \tilde{\mu}_M)$ (instead of μ_q) is the new service rate for each queue q

It is worth noting that the mean duration of calls for each queue in our previous model did not vary too much throughout the week as we can see on Figure 4.6. The periods in which the mean duration is zero, have no worth because there are then no calls.

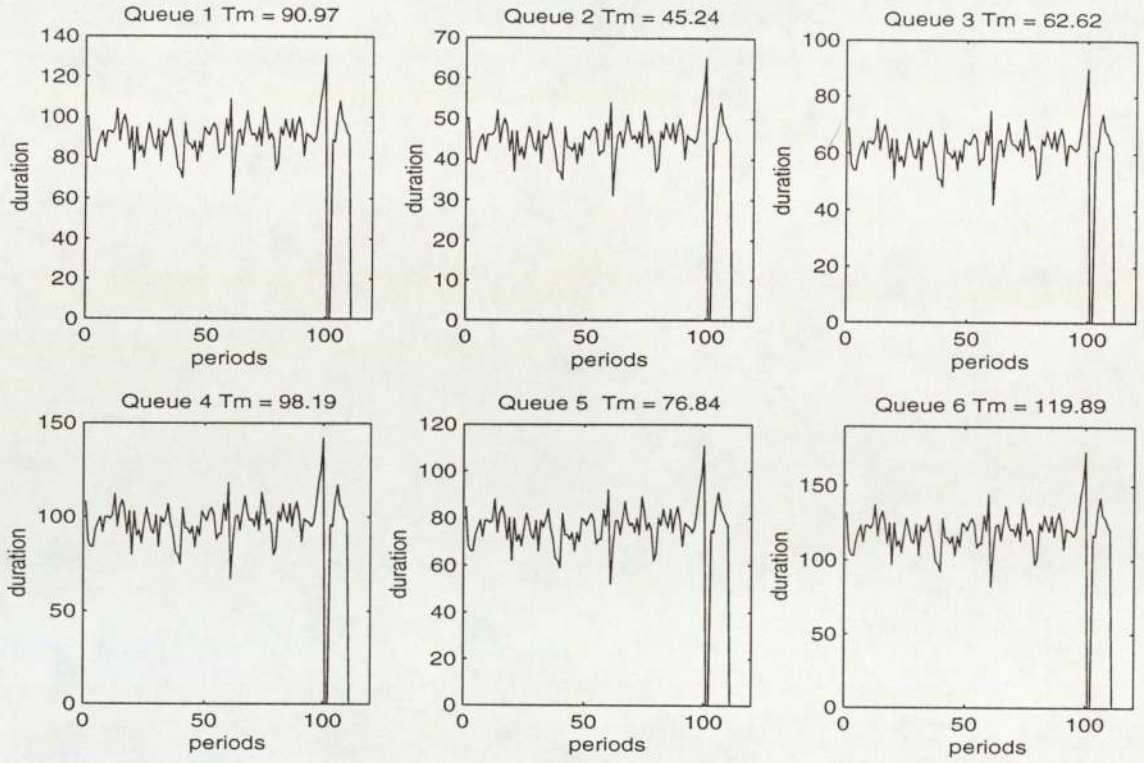


Figure 4.6: Mean duration of a call in the six queues during a week.

Having found a better model for our system, we are going to derive the number of servers required to achieve a given service level. This is done in the next section.

4.4.2 The minimum number of servers required

As we recalled in Chapter 2, to compute the required number of agents in the system we applied two approximation formulae available for $M/G/c$ model, *i.e.* for a random Poisson arrival pattern and a random general service time distribution and c servers.

As the inter-arrival distribution is still exponential, that is to say the arrival pattern is still a Poisson process and even if the service time distribution has changed, we can use those previous formulae.

In fact the service time distribution appeared only in Martin's estimate:

$$W_d = \frac{C(c,a) \times E[s]}{c-a} \times \frac{1+C_s^2}{2}$$

where

- s is the random variable describing the service time
- $E[s]$ is the expected service time
- $a = \lambda \times E[s]$ and λ is the arrival rate
- $C(c,a)$ is the Erlang's C function
- $C_s^2 = \frac{Var[s]}{E[s]^2}$ is called the squared coefficient of variation of s

As we know the mixture hyper-exponential service time distribution, see equation (4.12), the different parts can be computed:

- $E[s] = \sum_{q=1}^{Q} \alpha_q \, T_q$
- $E[s^2] = 2 \sum_{q=1}^{Q} \frac{\alpha_q \, {T_q}^2}{m^2} \left(\sum_{j=1}^{M} \frac{P(j)}{\tilde{\mu}_j^2} \right)$
- $C_s^2 = \frac{Var[s]}{E[s]^2} = \frac{E[s^2]}{E[s]^2} - 1$
- $C(c,a)$, the Erlang's C function, can be derived thanks to an iterative formula

For more details see appendix B. Thus we can compute the minimum number of servers in the system for each half hour *i.e.* the minimum number of servers to achieve a specified service level, and the remainder of the procedure is as before.

4.4.3 Results

As expected, the minimum number of people, corresponding to the first step our process see paragraph 1.3, computed with the new service time distribution increases compared to the one computed with the old stochastic model of the queue. But it's a very small increase: less than 1.6%.

And if we also run the algorithm to compute the number of people we have to share between the different pools, the increase is still smaller : less than 1.5% compared to

the old one. In Figure (4.7), we can barely see the difference, so we have represented it in Figure (4.8).

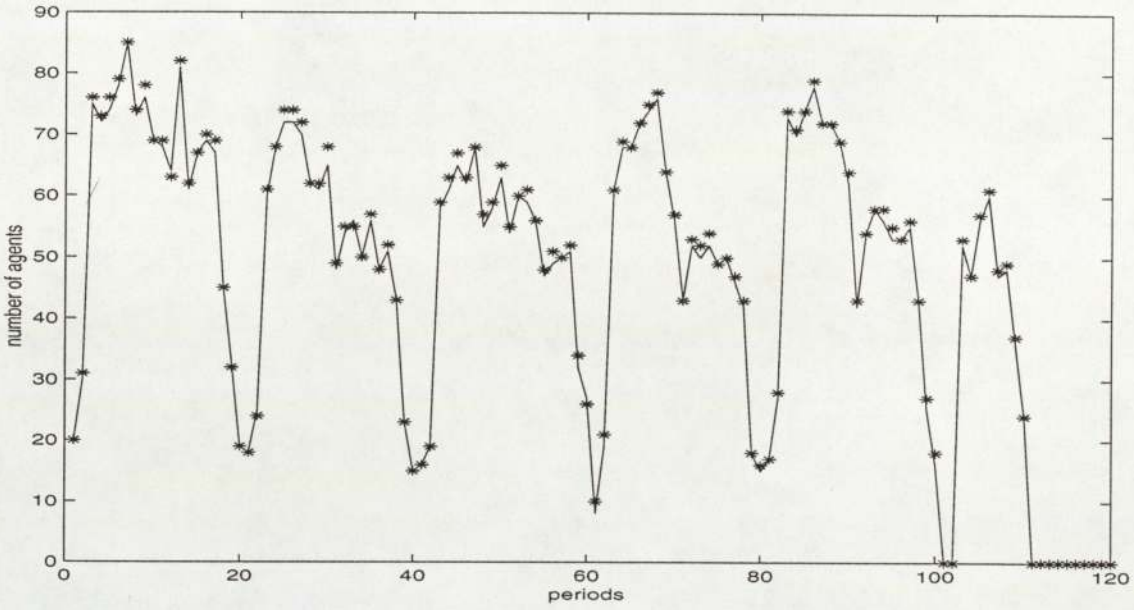


Figure 4.7: Comparison between the number of agents required with the stochastic model(solid line) and with the new model (stars).

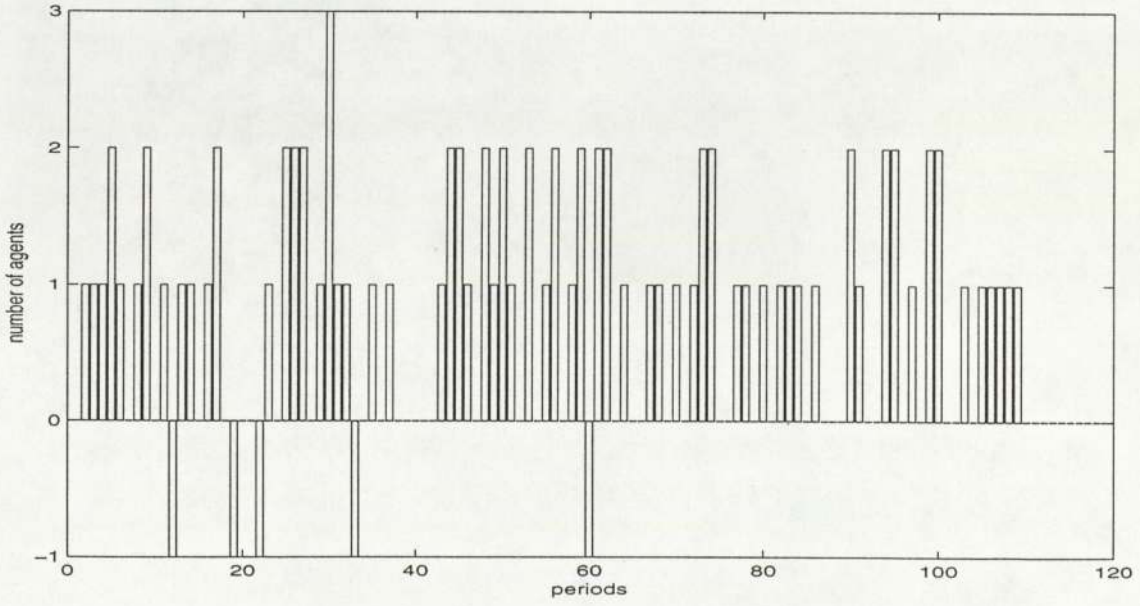


Figure 4.8: Difference between the number of agents required with the new model and the old stochastic model through periods.

The topic of the next section is to validate our model using the simulation program.

4.4.4 Validation

The improvement carried out in the stochastic model accompanied by an increase of the predicted number of agents in the system. Normally it should lead to an improvement of the service level. One of the way to check this assumption is to run the simulation program.

It is actually the case, as shown in Figure 4.9, now, only 3.58% of calls are not answered within 30 seconds, compared to the 4.42% with the previous stochastic model. The most appreciable difference takes place in the simulations which gave very bad service level: the decrease of the overwaiting time is significant between 1.5% and 2%. It is worth noting that the service time is reached now in 71% of the simulations against 61% before. With regards to the average number of periods in which some calls experience an unwished delay, we observe a more significant drop in Figure 4.10(a): from 38 periods to 30 periods. That may confirm the assumptions that in lots of periods there are just a few number of calls which are waiting for a too long a time. And the presence of one, two or three people more, see Figure 4.8, is enough to cope with this.

Figure 4.10(b) depicts the quality of service. We should not be misled by the scale of the graph, indeed it still remains quite stable, and we can notice a very slightly improvement from 76.6% to 77.0%. It is easily understood: some calls will be answered by agents more qualified, who were occupied before answering other calls for which they were less qualified. As the increase of agents in each pools is limited, the increase of the quality will be moderate too.

4.5 Conclusion

Even if the stochastic model of the queue developed last year was quite satisfactory, some doubts remained as to some assumptions made. Thanks to the new data file, provided by Callscan, some weaknesses of the old model have been spotted: in particular the non hyper-exponential nature of the service time distribution. To cope with this

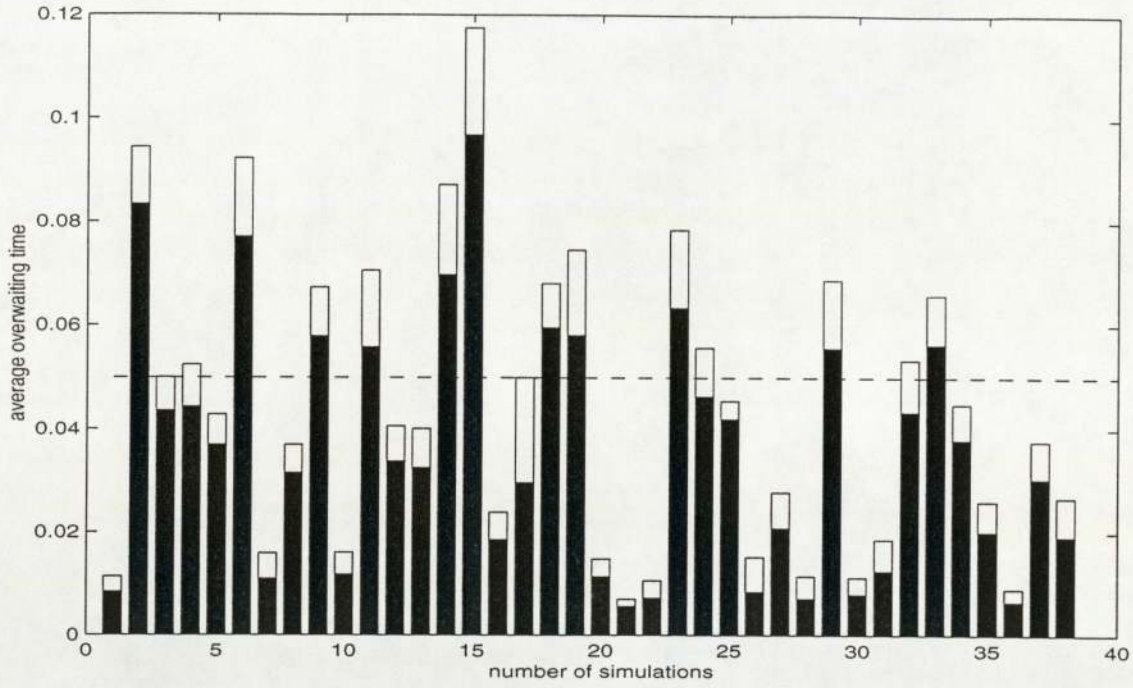


Figure 4.9: Histogram of the average percentage of calls, which experiment a delay superior to 30 seconds before being answered against the simulations. The white bars represent the percentage obtained with the stochastic model where as the black ones have been computed thanks to the new model. The dashed line marks the 5% service level.

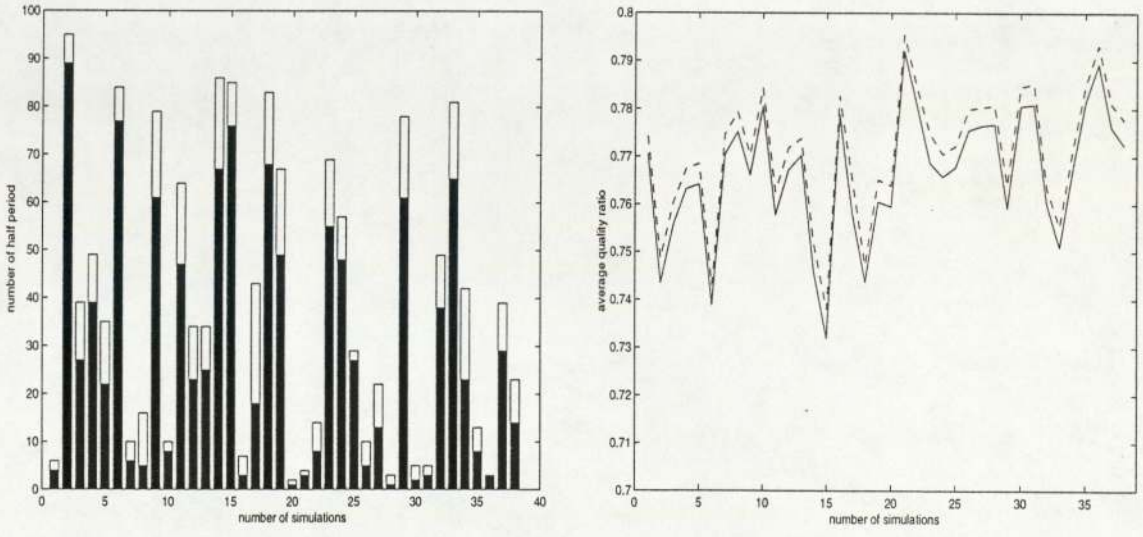


Figure 4.10: a) Histogram of the number of periods per week in which the service time is not reached. The black bars correspond to the stochastic model whereas the white ones to the new model.
b) Quality service provided for the stochastic model (solid line) and for the new model (dotted line) against several simulations.

issue, a new model based on mixture model has been developed which fits now much better the real phenomenon, the service time.

To validate the new model, two approaches can be proposed:

- running some simulations. That has been done, see previous paragraph and the results were quite promising since the increase of agents goes with the increase of the service level.
- testing with another set of data: the results will be presented in Chapter 6.

Chapter 5

Abandoned Calls

5.1 Introduction

So far we have not taken abandoned calls into account: this is the objective of this chapter. Intuitively we may think that their integration in our model may decrease the number of staff required for the same service level as previous but this is not the case as we shall see.

After a short study of the data and of the queueing distribution, we will propose a method to tackle this problem.

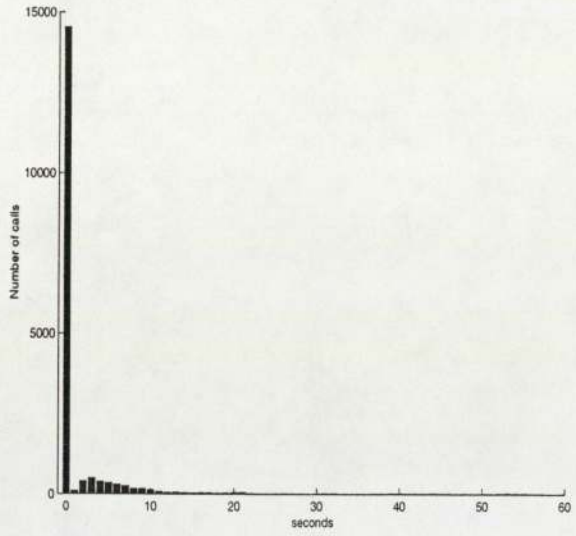
5.2 Data analysis

From the data provided by Callscan, already used in the previous chapter, we can compute the queueing time *i.e.* the duration a call spent in the queue. Of course we will remove from the sample, the data whose service time duration is equal to zeros. A priori, a call whose service time is equal to zero cannot be considered as significant. Moreover as it does not require any agents, it has not to be taken into account in the long process of the computation of the number of agents.

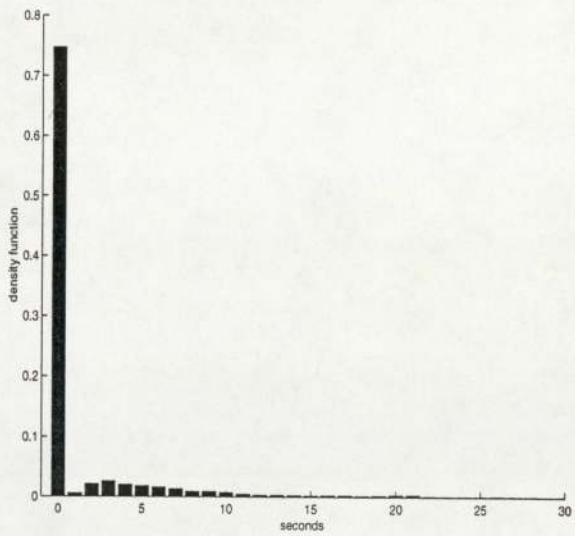
In Figures 5.1(a) and 5.1(b) we have represented the general queueing distribution, *i.e* the duration of incoming calls (answered and abandoned) spent in the queue, and the density function. The obvious point about these figures is that more than 2/3 of

calls, 77%, do not spend any time in the queue. They are immediatly routed forward to agents or abandoned.

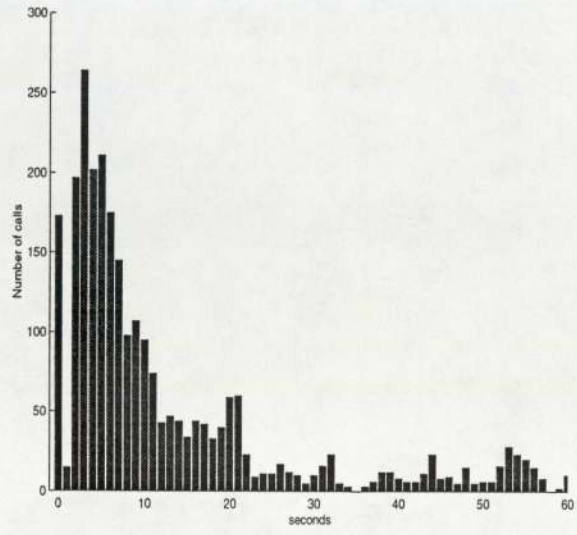
With regard to abandoned calls, the calls are more spread, as shown in Figures 5.1(c) and 5.1(d).



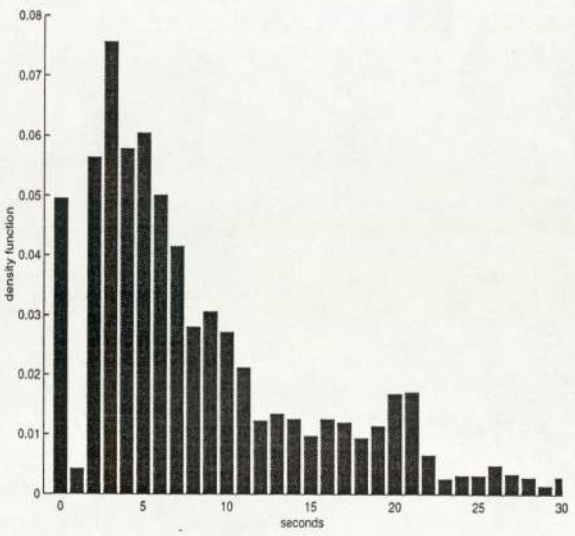
(a) general queueing time distribution



(b) general queueing p.d.f. in first 30 s



(c) queueing distribution of abandoned calls



(d) p.d.f. of abandoned calls in first 30 s

Figure 5.1: Representation of the queuing time for incoming calls, a) and b), and for the abandoned calls, c) and d).

It is worth noting that the proportion of abandoned calls is 18%. This is due to the fact that we didn't count the calls with service time duration equal to zero. Otherwise this proportion would have gone down to 11%. In fact it is not as important as we could think because we won't use this proportion directly as we will see in the next paragraph. It just outlines the importance of the phenomenon and the necessity of dealing with it.

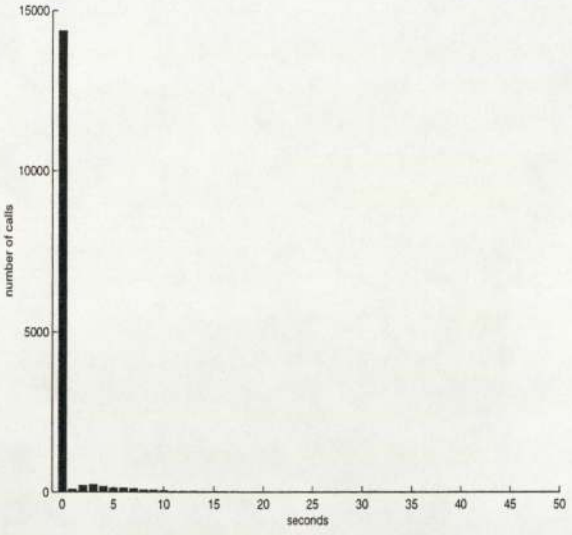
Another interesting point that needs to be examined is the queueing distribution of answered calls, that is to say, calls which will be dealt with by operators.
Figures 5.2(a) 5.2(b) provide good representations of the distribution.

In the former model, strong assumptions were made: the Heavy traffic approximation implies that the distribution of the queueing time approaches that of an exponential. The random variable describing the queueing time d, see paragraph 2.2.5 concerned all incoming calls. But as we now take abandoned calls into consideration, we will restrict the validity of the Martin's Estimate and Heavy traffic approximation just to the answered calls. Furthermore it is interesting to check the practical implication of the Heavy traffic approximation on the queueing distribution of the calls being answered. Although the probability density function (p.d.f.) of those calls is not exactly fitted by an exponential distribution as shown in Figures 5.2(c) and 5.2(d), it is much better that other classical distributions such as the gamma Erlang. Therefore in the following we will still go on working with this assumption.

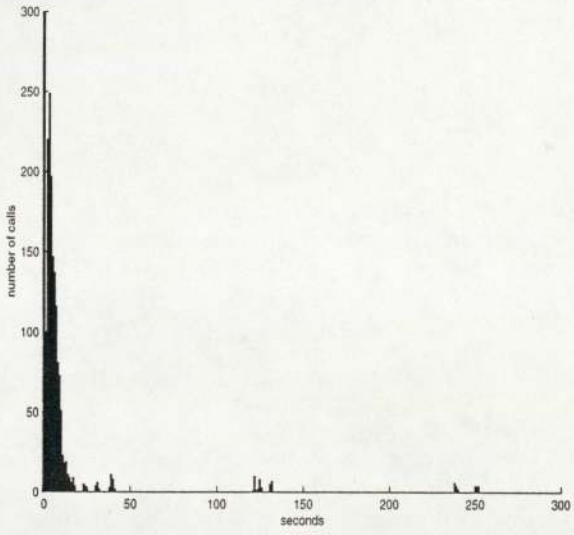
5.3 New model

5.3.1 Observation

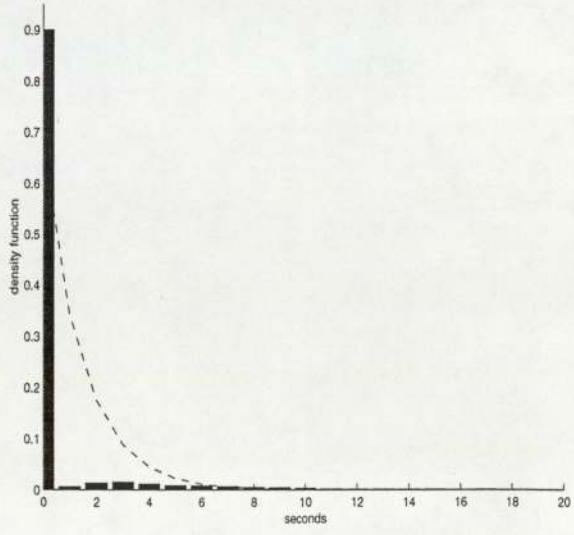
Abandoned calls may affect the predicted number of agents in the system. Paradoxically more agents will be required to respect the same service level. This is because abandoned calls are treated as unanswered: we have therefore to ensure that a greater proportion of the remaining calls is answered within the time limit. Thereby more calls will have to be answered during the same time so that the average delay of waiting for answered calls will decrease. Consequently the required number of people will increase



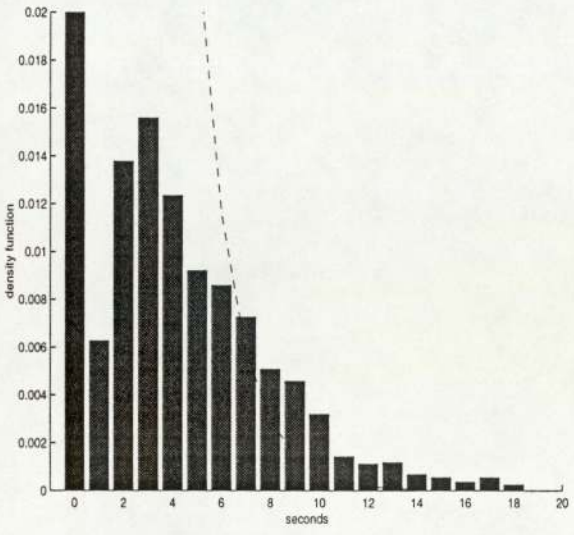
(a) queuing distribution of answered calls



(b) queueing distribution of answered calls (different scale)



(c) p.d.f. of answered calls (vertical bars) against exponential distribution(dotted line)



(d) p.d.f. of answered calls (vertical bars) against exponential distribution(dotted line) other scale

Figure 5.2: Representations of the queueing time of the abandoned calls. In the two last graphs, the parameter of the exponential distribution is computed thanks to the data.

according to Martin's approximation formula.
The definition of the service time will slightly change: if it was seen before as the percentage of incoming calls within a period, it will be regarded now as the percentage of non abandoned calls which are going to be answered.

5.3.2 Revisiting the Heavy Traffic approximation

After this practical justification we focuse on revisiting the two approximation formulae. Let us define $\tilde{d}$, the random value describing the average waiting time spent in the queue by a call which will be answered. The expected value of the time spent in the queue, considered as the average waiting time in queue will be denoted as $W_{\tilde{d}} = E[\tilde{d}]$ and will still be equal to Martin's estimate.

With regards to the Heavy Traffic Approximation, as the definition of the service level has slightly changed, the usual formula $\pi_d(r) = W_d \times \ln\left(\frac{100}{100-r}\right)$ doesn't apply for the r^{th} percentile value of $\tilde{d}$, $\pi_{\tilde{d}}(r)$. All the more as the abandoning calls should have to be integrated. We let d still denote the random value describing the time spent in the queue, but by all the calls, *i.e.* incoming calls = abandoned calls + answered calls.

Let us introduce some useful notation:

- N: the number of incoming calls.
- r: the proportion of incoming calls which have to be answered in order to achieve the former service level, in fact r/100 is a percentage.
- a_r: the proportion of abandoned calls, $a_r N$ is the number of abandoned calls. Consequently the number of calls effectively answered is $N - a_r N$.

For a specified service level, given a number of incoming calls, we can express the number of calls which have to be answered within a certain period. In fact, the considered period in both cases for all incoming calls or non-abandoned incoming calls, will be the same, then $\pi_d(r) = \pi_{\tilde{d}}(r)$.

Therefore the first expression is equal to the number of incoming calls multiplied by the probability that the duration of incoming calls in the queue be less that $\pi_d(r)$. But it can be seen too as the number of answered calls multiplied by the probability that the duration of answered calls be less than $\pi_{\tilde{d}}$. With our previous notations:

$$P(\tilde{d} \leq \pi_{\tilde{d}}(r)) \times (N - a_r N) = P(d \leq \pi_d(r)) \times N$$

And as $P(d \leq \pi_d(r)) = \frac{r}{100}$ we deduce that the r^{th} percentile value of $\tilde{d}$ is defined by:

$$P(\tilde{d} \leq \pi_{\tilde{d}}(r)) = \frac{r}{1 - a_r} \times \frac{1}{100}$$

Still assuming the Heavy traffic Approximation, and that the queueing time distribution for answered calls can still be approximated by an exponential distribution, see 5.2, we can use the following formula:

$$\pi_{\tilde{d}}(r) = W_{\tilde{d}} \times \ln\left(\frac{100}{100 - r/(1 - a_r)}\right) \tag{5.1}$$

Details can be found in appendix B.3.4.

In this relation, the expected queueing time, $W_{\tilde{d}}$, the r^{th} percentile value and the proportion of abandoned calls a_r are linked. To solve our problem of computing the required number of agents, we have also to calculate $W_{\tilde{d}}$. But this can only done if we are able to evaluate a_r. This is the aim of the following section.

5.3.3 Fraction of abandoned calls

As abandoned calls affect the service level, we need to be able to compute the fraction of abandoned calls in some time interval.

This can be done by using a convolution. If $A(t)$ denotes the probability density of the fraction of abandoned calls, then by convolving $A(t)$ with the relevant exponential distribution for non abandoned calls, denoted $w(t)$, it also possible to calculate the fraction of abandoned calls in the interval $[0, \pi_d(r)]$:

$$a_r = \int_0^{\pi_d(r)} A(t)w(t)dt \tag{5.2}$$

where:

- the exponential distribution of the queueing distribution given by the heavy traffic approximation is equal to:

$$w(t) = \frac{1}{W_{\tilde{d}}} \times e^{-\frac{1}{W_{\tilde{d}}} \times t}$$

- and the probability density of the fraction of abandoned calls is defined by

$$A(t) = \frac{\text{number of abandoned calls during any time interval } [t, t+\delta t]}{\text{number of (incoming) calls queueing during any time interval } [t, t+\delta t]}$$

In Figure 5.3 we have represented this fraction using the current data.

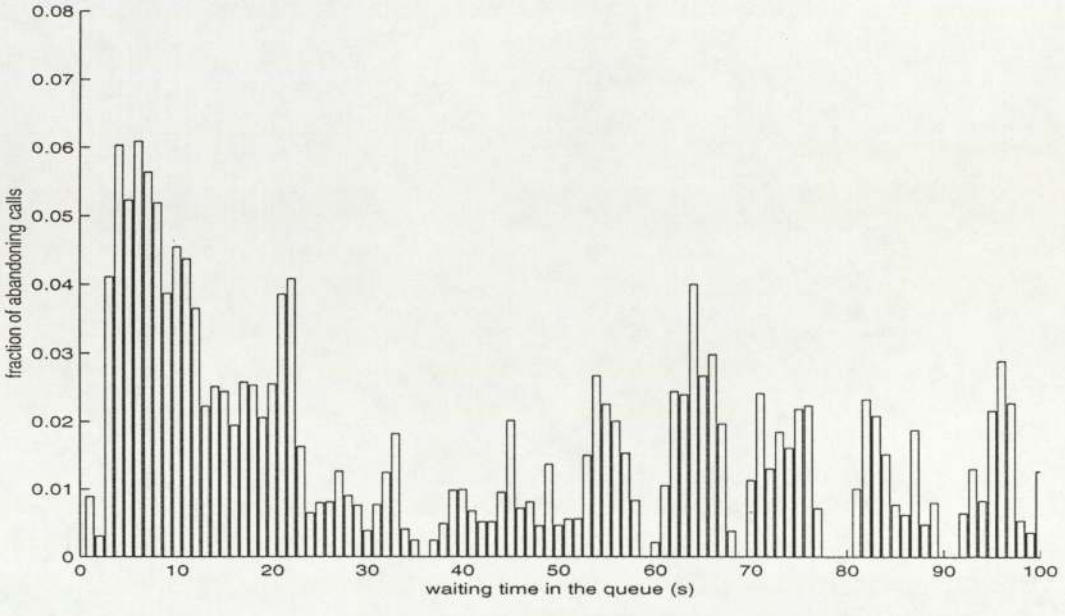


Figure 5.3: Representation of the fraction of abandoned calls against the time.

5.3.4 Algorithm

To compute the new required number of agents in the system we need to evaluate $W_{\tilde{d}}$, defined in the equation (5.1), but $W_{\tilde{d}}$ depends on a_r, and a_r on $W_{\tilde{d}}$, so we can't compute in a one step those quantities. An iterative procedure looking for a fixed point solution has to be used:

1. Compute $W_{\tilde{d}}$ as in the current procedure (i.e set $a_r = 0$ initially):

$$W_{\tilde{d}} = \frac{1}{\ln\left(\frac{100}{100-r}\right)} \times \pi_{\tilde{d}}(r))$$

2. Compute a_r from (5.2)

3. Compute $W_{\tilde{d}}$ from (5.1)

4. If a_r and $W_{\tilde{d}}$ have not converged, go to 2.

Since a_r is quite small in practice this procedure should converge relatively rapidly. Once $W_{\tilde{d}}$ fixed, the integer c is determined too.

We can now justify why more operators are necessary in the system, while taking the abandoned calls into account for a given service level.

And since a_r is a positive number and since $\pi_{\tilde{d}}(r) = \pi_d(r)$, $W_{\tilde{d}} \leqslant W_d$, consequently more agents are now required than before. (In both case Martin's estimate is be used to derive those numbers)

5.4 Results

In this section we present the results obtained whilst using both the new data set to compute the fraction of abandoned calls and the older data to run the algorithm through a whole week. The model used has also the new service time distribution.

As expected, in order to achieve the right service level 95% of the calls answered within 30 seconds, and as shown in Figure 5.4, the required number increases slightly compared to the previous stochastic model: 2.9% but that is still 22% fewer agents than with the Erlang'C model, which does neither take into account service time distribution or abandoned calls.

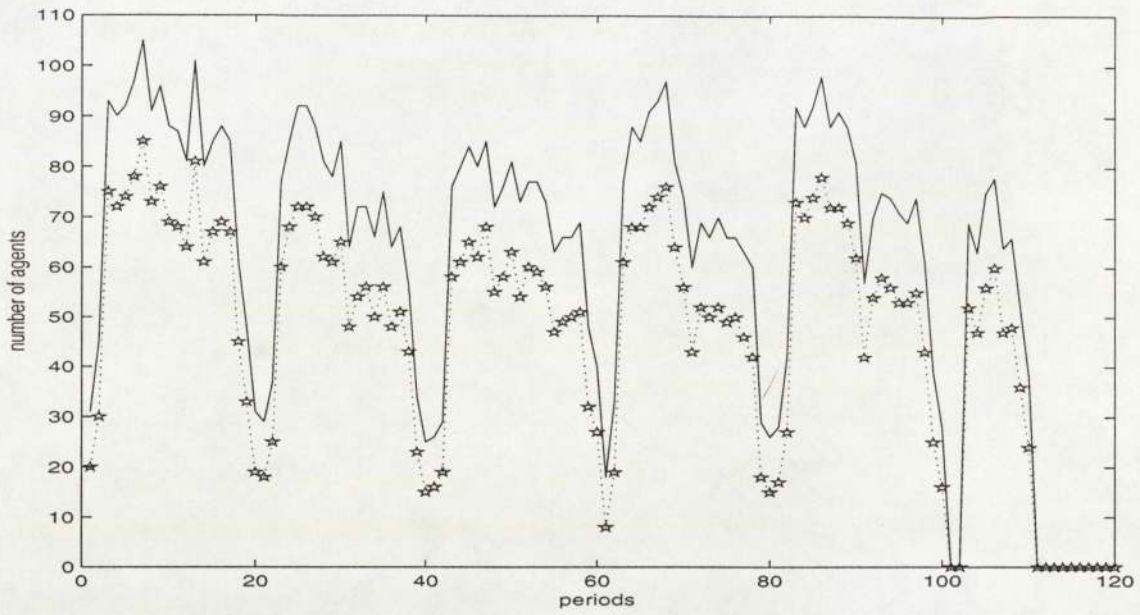


Figure 5.4: Comparison between the number of agents used in the traditional model (the solid line), the stochastic model (the dotted line) and the new model, with a new service distribution and a procedure to include abandoned calls, (stars).

With regards to this new model compared to the mixture model, the increase is about 1.4% of more agents. Figure 5.5 brings some remarks: the increase is neither uniform against the week neither dependent on the traffic as the representation of agents in background shows.

5.4.1 Validation

To validate our model, we have run the simulation program. The only major drawback of this approach comes from the fact that abandoned calls were not simulated.

Failing anything better, we have also run it. For this reason the conclusions we could draw will be limited. As more agents are predicted in the system, an increase of the service level and of the quality ratio is expected.

Finally after running it several times, the results are in keeping with our expectation. Compared to the mixture model, as shown in Figures 5.6(a) and 5.6(b) all the characteristic parameters are improved:

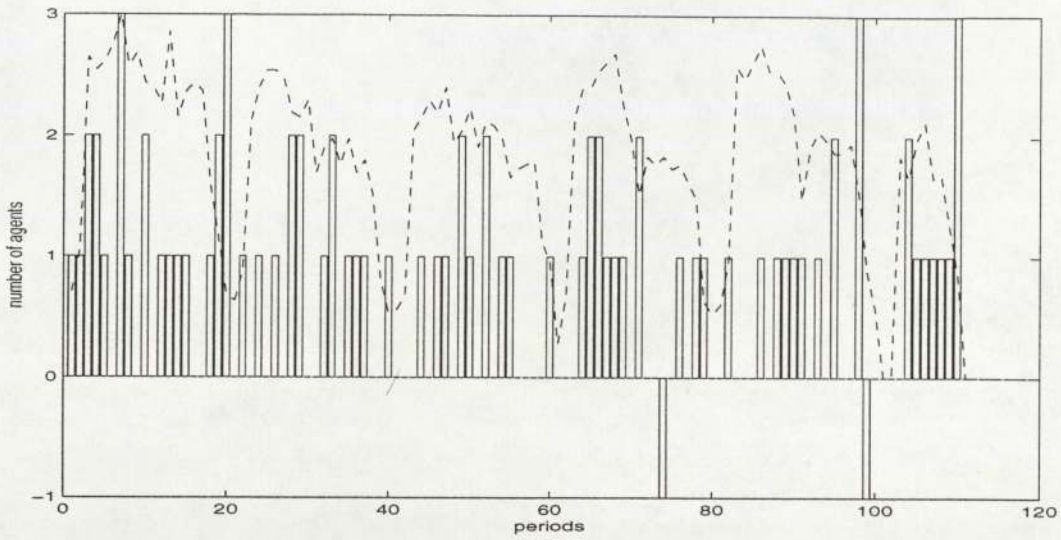


Figure 5.5: Histogram of the difference between the number of agents required with the "mixture model" and those with the new model through a 120 periods week. In background the distribution of agents of the new model has been represented in an other scale (dashed line).

- the number of calls experiments a delay superior of 30 seconds before being answered decreases.

- whereas the quality ratio increases.

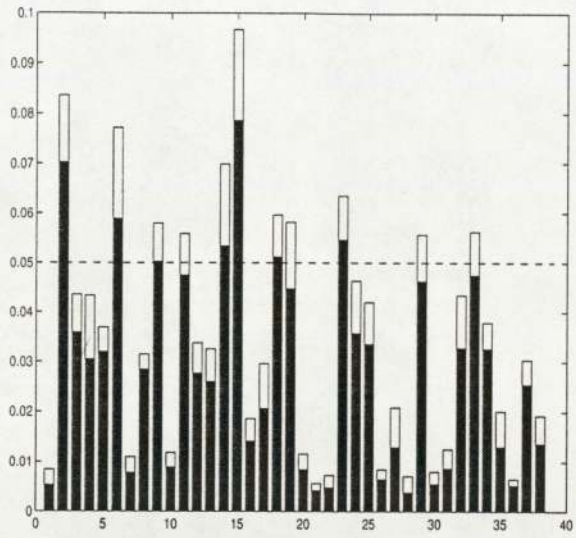
At least to show the improvement brought to the model we have also summarised all the results, from this chapter and from the previous chapter in Table 5.1.

	New Model	Mixt. Model	Stoch. model
Over waiting time [a]	2.86	3.58	4.42
Number of periods [b]	21	30	38
Quality ratio	77.5	77.0	76.6
Number of agents	5839	5758	5675

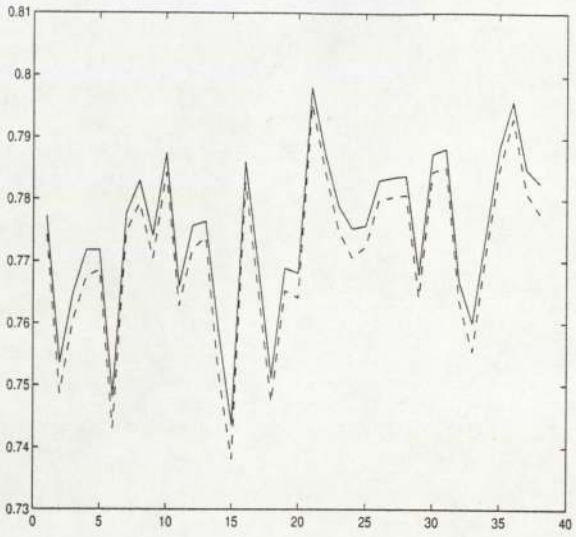
[a] the overwaiting time is the average percentage of calls across a week which are not answered within 30 seconds

[b] this is the average number of period across a week in which the service time is not achieved

Table 5.1: Average results obtained with 38 simulations.



(a) Histogram of the average percentage of calls, which have experienced a delay greater than 30 seconds before being answered. The white bars represent the percentage obtained with the "abandoning model" where as the black ones with the "mixture model". The dashed line symbolises the service level which should be respected.



(b) Quality service provided for the mixture model (dashed line) and for the new model (solid line).

Figure 5.6: Simulations'graphs

5.5 Conclusion

To integrate abandoned calls in our previous model we have revisited the approximation formulae relating to the service level and used some ideas like convolution product, directly based on the signal theory.

Theoretically to respect the same service level, more people were required. We checked this assumption using two sets of data coming from two different call centres.

In order to validate more thoroughly our new approach we will use an other data sample next chapter.

Chapter 6

Validation

6.1 Changing the service level

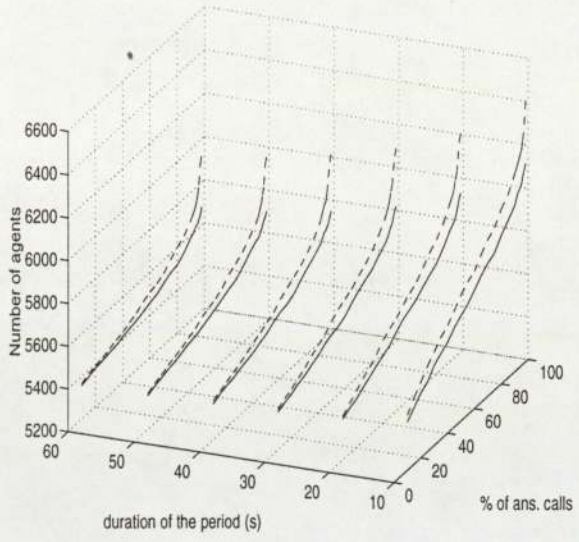
So far we have carried out all experiments with the same service level: 95% of the calls have to be answered within 30 seconds. Therefore it could be interesting to explore the behaviour of our model compared to the previous stochastic one, by now moving the service level: both the percentage of the calls to be answered within a period, and the period itself.

Then, we varied the required percentage from 10% to 99% and the specified period amongst the interval [10,...,60] (in second).
All the results are presented in Figures 6.1(c) and 6.1(d). As expected in every case the predicted number of people with our new model is bigger than with the stochastic model. Although on average the quality ratio of our model is higher than that of the old one, it is worth noting that sometimes the reverse happens. It is due to the skills of the agents.

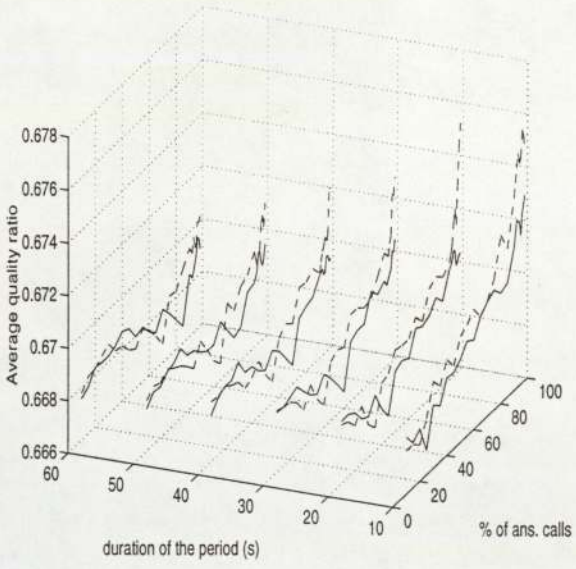
Finally some remarks can be made:

- for a specified period, the higher the percentage of calls to be answered is, the more agents they are in the system, and usually the quality ratio is higher.
- for a given required percentage of calls, the bigger the time to answer calls is, the less agents they are in the system, which is quite logical and the smaller the

quality ratio is, which is less logical. Nevertheless it can be explained by the presence of less qualified agents in the system.



(a) Predicted number of people with the stochastic model (solid line) and with the new model (dashed line).



(b) Quality of service.

In conclusion, we have checked that our new algorithm was reliable.

6.1.1 Presentation

In the previous chapter we have proposed some improvement of the stochastic model. Most of the validation was made with the simulation program which was not always appropriate.

Using another set of data from another call center, composed of several recordings with main parameters:

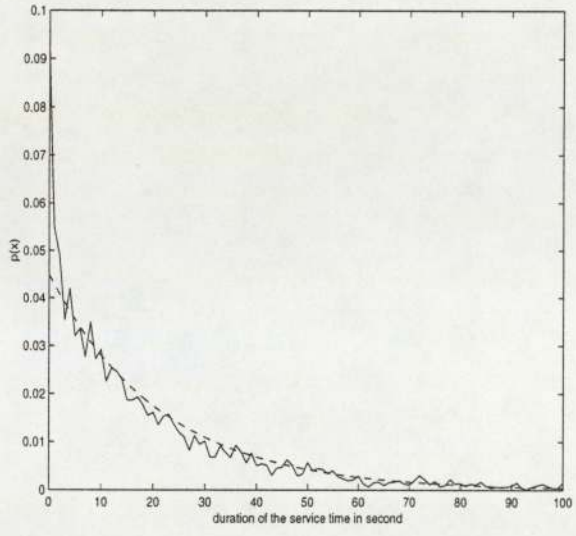
- the arrival pattern
- the service time distribution

we have checked that the refinement brought to the model was still relevant. Effectively the arrival pattern follows a Poisson process and also the inter-arrival service time

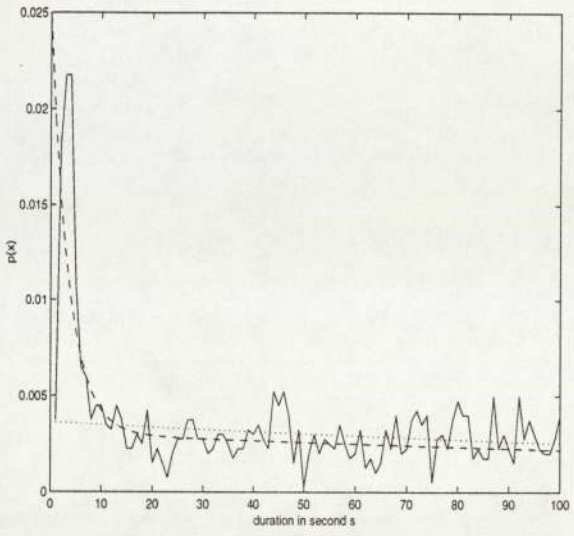
distribution is still exponential, as shown in Figure 6.1(c), whereas the recomputed service time distribution can be fitted by a mixture of exponentials as shown in Figure 6.1(d).

In this stage, by running the scheduling algorithm, to achieve the usual service level, 95% of calls answered within 30 seconds, a bit more than 0.2% of people more are expected than with the previous scheduling algorithm.

As the data comes from two different call centres, what really matters is not the percentage but the tendency.



(c) inter-arrival service time distribution against exponential p.d.f (dashed line).



(d) Representation of the recomputed service time distribution (solid line) against exponential p.d.f (dotted line) and mixture of exponentials p.d.f.

As far as the abandoning calls are concerned we could think, in the first place reading raw data, that the percentage of abandonment will increase significantly the number of agents in the system, as shown in the Table 6.1.

In fact that would mean ignoring that the abandoning calls, roughly 30% of the sample, are taken into account in a convolution product, which smooths their relative importance: the significant number in our present problem is the number of abandoning calls within 30 seconds, just 4.73% of the queueing calls. Compared to the previous

	Abandoning calls	Anwering calls	Total
Number of calls	1706	5586	7292
Percentage	23.23%	77.77%	100%
	Abandoning calls	Answering calls [a]	Total
Number of calls	1706	4000	5706
Percentage	29.89%	70.11%	100%

[a] A realistic number of calls whose service time duration was equal to zero has been recomputed with the method presented in Appendix C.3.7

Table 6.1: Data statistics

data, see Chapter 5, where the percentage was from 14.6%, it is really weak. No wonder also if the required number of people by running the algorithm is steady, just a minuscule increase. And then, the quality ratio is still slightly better.

6.1.2 Conclusion

We proved that we could use a method with data coming from another call center and that the number of people doesn't change too much. In accordance with the theory we noticed an increase of the number of staff compared to the older stochastic model.

Chapter 7

Conclusions

The aim of this project was to go further in the investigations of the model developed last year by G. Lecorroller on the basis of the feasibility study made by Prof D. Lowe and Dr I. Nabney.

The first point was concerned with reviewing the algorithm proposed to tackle this problem of human resource allocation. By combining the stochastic model of the queue to take into account the fluctuations of the incoming calls and the service time and a minimisation method based on a penalty function, the number of agents to achieve a proper level in our system could be derived. The results were quite encouraging as the number of staff remained smaller than those computed in the standard method, a lot of assumptions needed to be questioned and properties to be investigated.

Secondly we have tested extensively the algorithm to discover some properties and highlighted the major weaknesses in order to improve it. We varied the coefficients of the cost functions, the Lagrangian parameters. Since there was no significant effect on the results, we left their values fixed in subsequent work. Thus by exploring a possible correlation or linear relationship between the increase of the pool and those of the traffic rate, we noticed that while using more pools for a steady traffic rate more people were necessary. Proposing an empirical method we managed to control this phenomenon at least. From the several other investigations, we recalled that it was preferable that each queue could be answered by two pools.

Using new data sets we tested the assumptions the stochastic model was relied on. The modelling of the arrival pattern with a Poisson process seemed quite reasonable, on the other hand the exponential distribution gave a very poor fit to the service time distribution. Therefore we have developed a new distribution, still hyper-exponential but with some mixture density coefficients. By running the algorithm, we noticed that more agents were predicted in the system, but the increase was still limited.

The last issue discussed was abandoned calls. Intuitively we might think that taking them into account will lead to less agents. But that idea was wrong since the definition of the service level had changed: the correct definition was the percentage of non abandoned calls which would have to be answered within a certain time. So since the abandoned calls affected the service level, it implied some modification of the approximation formulae. Running also the algorithm with the new data confirmed the increase of the number agents in the system.

To test the reliability our model and the non dependency of the data, we trained it on new sample: we obtained a new confirmation our previous assumptions *i.e.* still more agents were predicted in the system but the rise remained reasonable less than 3% and still 22% fewer that with the traditional approach *i.e.* the Erlang C model.

However this new model was all the more satisfying as it seemed closer to the real behaviour of the queues.

The ultimate step would be the implementation in a call centre to assess the real life performance of our model.

Appendix A

Coefficients of the EM for the exponential mixture model

In this appendix, we are going to demonstrate the updating formulae of the parameters λ_j^{new} and $P^{new}(j)$ we have used in chapter 4.

A.1 Presentation

To compute the new coefficients of the parameters astated above in the E-steps of the EM Algorithm we have to minimise the function $\mathcal{E}^{comp}(\theta^{new})$ under the condition $\sum_{n=1}^{N} P^{new}(j) = 1$ where:

- $\mathcal{E}^{comp}(\theta^{new}) = -\sum_{n=1}^{N}\sum_{j=1}^{M} P(j|x_n, \theta_j^{old}) ln(P(x_n|j, \theta_j^{new}) P^{new}(j))$
- $\theta^{new} = \{\lambda_j^{new}, P^{new}(j), j = 1, .., M\}$

Indeed given a data set of N values $(x_1, x_2, \ldots, x_n)$ and given the mixture model which should describe it, $p(\mathrm{x}) = \sum_{j=1}^{M} P(j)p(\mathrm{x}|j)$, we consider those data to be incomplete because we do not know which component j generated at a given data point n. We also introduce a variable z_n, n varying from 1 to N, denoting the unkown generating component.

The complete data log-likelihood is:

$$\begin{aligned} L^{comp}(\theta) &= \sum_{n=1}^{N} ln(p(x_n, z_n|\theta)) \\ &= \sum_{n=1}^{N} ln\{p(x_n|z_n, \theta)P(z_n|\theta)\} \end{aligned} \tag{A.1}$$

Now we take its expectation with respect to the distribution

$$P(\mathbf{z}) = \prod_{n=1}^{N} P(z_n|x_n, \theta^{old})$$

And since z is a discrete variable, the expectation over all z_n is a combination of N sums:

$$\begin{aligned} \mathcal{E}^{comp}(\theta) &= \sum_{n=1}^{N}\left[\sum_{z_1=1}^{M}\sum_{z_2=1}^{M}\cdots\sum_{z_N=1}^{M}\prod_{m=1}^{N} P(z_m|x_m, \theta^{old}) ln\{p(x_n|z_n, \theta)P(z_n|\theta)\}\right] \\ &= \sum_{n=1}^{N}\left[\sum_{z_1=1}^{M}\cdots\sum_{z_{n-1}=1}^{M}\sum_{z_{n+1}=1}^{M}\cdots\sum_{z_N=1}^{M}\prod_{m\neq n}^{N} P(z_m|x_m, \theta^{old})\right] \\ &\times \left[\sum_{z_n=1}^{M} P(z_m|x_m, \theta^{old}) ln\{p(x_n|z_n, \theta)P(z_n|\theta)\}\right] \\ &= \sum_{n=1}^{N}\sum_{z_n=1}^{M} P(z_m|x_m, \theta^{old}) ln\{p(x_n|z_n, \theta)P(z_n|\theta)\} \end{aligned} \tag{A.2}$$

since the first square-bracketened term in (A.2) evaluates to unity as each of the individual sum $\sum_{z_m=1}^{M} P(z_m|x_m, \theta^{old}) = 1$ because the probability that x_m has been generated by the set $(z_1, z_2, \ldots, z_n)$ is equal to 1.

Our previous notation for a mixture model was $p(\mathrm{x}) = \sum_{j=1}^{M} P(j)p(\mathrm{x}|j)$ which equals to $\sum_{j=1}^{M} P(j)p(\mathrm{x}|j, \theta_j)$. And so we may rewrite $\mathcal{E}^{comp}(\theta^{new})$ equivently as:

$$\mathcal{E}^{comp}(\theta^{new}) = -\sum_{n=1}^{N}\sum_{j=1}^{M} P(j|x_n, \theta_j^{old})\ ln(P(x_n|j, \theta_j^{new})P^{new}(j)) \tag{A.3}$$

A.2 Computation

Thus to minimise the function $\mathcal{E}^{comp}\ (\theta^{new})$ under constraints, see above, we can use Lagrange multipliers.

Define a function g writing down by: $\theta^{new} \mapsto g(\theta^{new}) = \sum_{n=1}^{N} P^{new}(j) - 1$.

The minimum can be determined by solving the simultanous equations:

$$\frac{\partial \mathcal{E}^{comp}}{\partial \theta_j^{new}} - \beta \cdot \frac{\partial g}{\partial \theta_j^{new}} = 0 \tag{A.4}$$

$$g(\theta_j^{new}) = 0 \tag{A.5}$$

for all $j = 1, \ldots, M$

.

A. By differentiating equation (A.4) with respect to the parameters $P^{new}(j)$ we get the following equations for all $j \epsilon [1, ..., M]$:

$$-\frac{\partial \mathcal{E}^{comp}}{\partial P^{new}(j)} - \beta \cdot \frac{\partial g}{\partial P^{new}(j)} = 0 \tag{A.6}$$

We expand the lefthand side:

$$\begin{aligned} 0 &= -\frac{\partial \mathcal{E}^{comp}}{\partial P^{new}(j)} - \beta \cdot \frac{\partial g}{\partial P^{new}(j)} \\ &= -\frac{\partial \left(\sum_{n=1}^{N} \sum_{j=1}^{M} P(j|x_n, \theta_j^{old}) ln(P(x_n|j, \theta_j^{new}) P^{new}(j)) \right)}{\partial P^{new}(j)} \\ &\quad - \beta \cdot \frac{\partial \left(\sum_{j=1}^{M} (P^{new}(j) - 1) \right)}{\partial P^{new}(j)} \\ &= \frac{\sum_{n=1}^{N} P(j|x_n, \theta_j^{old})}{P^{new}(j)} - \beta \end{aligned} \tag{A.7}$$

We conclude that:

$$P^{new}(j) = \frac{1}{\beta} \cdot \sum_{n=1}^{N} P(j|x_n, \theta_j^{old}) \tag{A.8}$$

As this expression is available for all $j = 1, \ldots, M$ we can sum over the j, then

$$\underbrace{\sum_{n=1}^{N} \sum_{j=1}^{M} P(j|x_n, \theta_j^{old})}_{=1} = \beta \cdot \underbrace{\sum_{j=1}^{M} P^{new}(j)}_{=1}$$

i.e. $$N = \beta$$

So by substituting β for N in the equation (A.8) we obtain:

$$\boxed{P^{new}(j) = \frac{1}{N}\sum_{n=1}^{N} P(j|x_n, \theta_j^{old})}$$

B. If we differentiate (A.4) with respect to the parameters λ_j^{new}, given that the density function is: $P(x_n|j, \theta_j^{new}) = \lambda_j^{new} e^{-\lambda_j^{new} \cdot x_n}$ we get:

$$-\frac{\partial \mathcal{E}^{comp}}{\partial \lambda_j^{new}} - \beta \cdot \frac{\partial g}{\partial \lambda_j^{new}} = 0 \quad \forall\, j \,\epsilon\, [1, \ldots, M] \tag{A.9}$$

So after having substituted $\mathcal{E}^{comp}$ for its expression and noticed that g does not depend on λ_j^{new} we obtain:

$$-\frac{\partial \left(\sum_{n=1}^{N} \sum_{j=1}^{M} P(j|x_n, \theta_j^{old}) \left(ln(\lambda_j^{new}) - \lambda_j^{new} \cdot x_n + ln(P^{new}(j)) \right) \right)}{\partial \lambda_j^{new}} = 0 \tag{A.10}$$

i.e.

$$\sum_{n=1}^{N} P(j|x_n, \theta_j^{old}) \left(\frac{1}{\lambda_j^{new}} - x_n \right) = 0 \tag{A.11}$$

Finally, from this we conclude:

$$\boxed{\lambda_j^{new} = \frac{\sum_{n=1}^{N} P(j|x_n, \theta_j^{old})}{\sum_{n=1}^{N} P(j|x_n, \theta_j^{old})\, x_n}}$$

Appendix B

Stochastic modelling of the queue

B.1 Expected value of an hyper exponential distributed random value with mixture coefficients

We shall consider a random variable X with a density function given by:

$$f_x(t) = \begin{cases} 0 & \text{if t} \leqslant 0, \\ \sum_{n=1}^{N} \alpha_n \left(\sum_{j=1}^{M} P(j)\mu_{jn}\, e^{-\mu_{jn}t} \right) & \text{if t} > 0. \end{cases} \tag{B.1}$$

with:

- $\mu_{qn} = a_q\ \mu_n = \frac{m}{T_q}\ \mu_n$
- $m = \sum_{j=1}^{M} \frac{P(j)}{\mu_j}$
- $\sum_{n=1}^{N} \alpha_n = 1$

The expected value of X is defined as follow:

$$\begin{aligned} E[X] &= \int_{-\infty}^{\infty} t\ f_x(t)dt = \int_{0}^{\infty} t\ \sum_{n=1}^{N} \alpha_n \left(\sum_{j=1}^{M} P(j)\mu_{jn}\, e^{-\mu_{jn}t} \right) dt \\ &= \sum_{n=1}^{N} \alpha_n \sum_{j=1}^{M} P(j) \int_{0}^{\infty} t\ \mu_{jn}\, e^{-\mu_{jn}t} dt = \sum_{n=1}^{N} \alpha_n \sum_{j=1}^{M} \frac{P(j)}{\mu_{jn}} \\ &= \sum_{n=1}^{N} \alpha_n \left(\sum_{j=1}^{M} \frac{P(j)}{\frac{m}{T_n}\ \mu_j} \right) = \sum_{n=1}^{N} \alpha_n T_n \end{aligned}$$

So the expected value of s, the service time, is equal to:

$$E[s] = \sum_{q=1}^{Q} \alpha_q T_q$$

We also have to compute the moment with second order:

$$\begin{aligned} E[X^2] &= \int_{-\infty}^{\infty} t^2 \; f_x(t)dt = \sum_{n=1}^{N} \alpha_n \sum_{j=1}^{M} P(j) \int_0^{\infty} t^2 \; \mu_{jn} \; e^{-\mu_{jn} t} dt \\ &= 2 \; \sum_{n=1}^{N} \alpha_n \sum_{j=1}^{M} \frac{P(j)}{\mu_{jn}^2} = 2 \; \sum_{n=1}^{N} \alpha_n \sum_{j=1}^{M} \frac{P(j)}{(\frac{m}{T_n} \; \mu_j)^2} \\ &= 2 \; \sum_{n=1}^{N} \frac{\alpha_n \; {T_n}^2}{m^2} \; \Big(\sum_{j=1}^{M} \frac{P(j)}{\mu_j^2} \Big) \end{aligned}$$

And so we can conclude that the second order of the value s is:

$$E[s^2] = 2 \; \sum_{q=1}^{Q} \frac{\alpha_q \; {T_q}^2}{m^2} \; \Big(\sum_{j=1}^{M} \frac{P(j)}{\mu_j^2} \Big)$$

B.2 Calculation of the percentile value

In this section we are going to justify the formula used in Chapter 5:

$$\pi_d(r) = W_d \times \ln \left(\frac{100}{100 - r/(1 - a_r)} \right) = E[d] \times \ln \left(\frac{100}{100 - r/(1 - a_r)} \right)$$

where r^{th} is the percentile value of d the queueing.

Considering an exponentially distributed random variable X, and $\pi(r)$, its r^{th} percentile value, defined by:

$$P(X \leq \pi(r)) = \frac{r}{1 - a_r} \times \frac{1}{100}$$

since X is an exponentially distributed random variable, whose parameter is denoted λ, we can write:

$$P(X \leq \pi(r)) = 1 - e^{-\lambda \pi(r)} = 1 - e^{-\frac{\pi(r)}{E[X]}}$$

In deed for an exponential distribution, $E[X] = \frac{1}{\lambda}$.

Having two expressions of the probability $P(X \leq \pi(r))$, we deduce that:

$$1 - e^{-\frac{\pi(r)}{E[X]}} = \frac{r}{1-a_r} \times \frac{1}{100} \quad \Longleftrightarrow \quad e^{-\frac{\pi(r)}{E[X]}} = \frac{1}{100} \times \left(100 - \frac{r}{(1-a_r)}\right)$$

And eventually that:

$$\pi(r) = E[X] \times \ln\left(\frac{100}{100 - r/(1-a_r)}\right)$$

B.3 Some distributions

B.3.1 Exponential distribution

A continuous random variable X is said to have an exponential distribution with parameter $\lambda > 0$ if its density function f is given by:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0 \\ 0, & x \leqslant 0. \end{cases}$$

Some properties:

$E[X] = \frac{1}{\lambda} \qquad Var[X] = \frac{1}{\lambda^2}$

B.3.2 Normal distribution

A continuous random variable X has a gaussian or normal distribution with parameters m and $\sigma > 0$ if its density function f is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} \qquad x \in \mathbb{R}$$

Some properties:

$E[X] = m \qquad Var[X] = \sigma^2$

B.3.3 Gamma distribution

A continuous random variable X has a gamma distribution with parameters $\alpha > 0$ $\lambda > 0$ if its density function f is given by:

$$f(x) = \begin{cases} \frac{\lambda}{\Gamma(\alpha)} (\lambda x)^{\alpha-1} e^{-\lambda x}, & x > 0 \\ 0, & x \leqslant 0. \end{cases}$$

where the Γ function is defined by

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx \qquad t > 0$$

Some properties:

$E[X] = \frac{\alpha}{\lambda}$ $\qquad$ $Var[X] = \frac{\alpha}{\lambda}^2$

B.3.4 Erlang k distribution

A special class of gamma random variables is the Erlang k random variable with parameter λ. Its density function is given by:

$$f(x) = \begin{cases} \frac{\lambda k}{(k-1)!} (\lambda k x)^{k-1} e^{-\lambda k x}, & x > 0 \\ 0, & x \leqslant 0. \end{cases}$$

Some properties:

$E[X] = \frac{1}{\lambda}$ $\qquad$ $Var[X] = \frac{1}{k \lambda^2}$

Appendix C

Outline algorithm

This appendix displays the full algorithm which was used to produce the figures obtained in this thesis. The code is written in the interpreted matrix manipulation language MATLAB.

C.1 Rounding program

```
% Rounding algorithm

% Read in the staff matrix

% Load the standard matrix used to cap the number of agents in each pools
% It is worth noting that this file contains real numbers and had been
% created by running the old version of the scheduling algorithm:
% to save the staff we use ''staff(i,:) = sum(con') instead of
% staff(i,:) = ceil(floor(sum(con')*100)/100);
load sta4_un.dat
staf0 = sta4_un;

load sta11_un.dat
staf1 = sta11_un;

% Read in the quality matrix, the coefficient correspond now to the pools
% This file was created still running the old version of the
% scheduling algorithm: to save it we have used the following instruction
% quality(i,:) = sum(skills'.*con') instead of
% quality(i,:) = sum(skills.*con);
load qual111_un.dat
qual = qual111_un;

num_staf = size(staf1);

% Initialize the matrix which  will contain the rounded number of people
% in each pool. This number will be an integer.
staff = zeros(size(staf1));

for i = 1:num_staf(1)

  % Number of required agents for one period
```

```
  c__i = sum(ceil(floor(staf0(i,:)*100)/100));

  % We sort the quality matrix in function of the quality factor
  [qual1 ind1] = sort(qual(i,:));

  ind0 = length(ind1);
  while  (((sum(staf1(i,:))) < c__i)&(ind0>0))
    staf1(i,ind1(1,ind0)) = ceil(floor(staf1(i,ind1(1,ind0))*100)/100);
    ind0 = ind0 - 1;
  end

  staff(i,:) = round(staf1(i,:));

end
%sum(sum(staff))

save 'sta11_r.dat' staff -ascii;
```

C.2 Code of EM Algorithm for exponential mixture model

C.2.1 Creation of good data file

```
% Program fx.m
% Create a realistic data file

% Read the file
% The first column indicates the nature of the  call 1->N, 2->Q, 3->R
% The second column can be ignored
% The next following columns contain the arrival time the col 3: h
% the col. 4 : mn, the col. 5: ss .
% The column 6 : 60*mn + ss
% In the column 7: The duration of wait.
% In the column 8: The duration of calls.
% In the column 9: The number of the pool which has answered the call.
% In the column 10: The number of periods.

load calls10.dat

len = length(calls10);

% We keep only the answering calls which duration is >0 and <2 hours

% we are going to compute the length of the futur file, without the
% zeros

len1 = 0;
for i = 1:1:len
  if ((calls10(i,1) == 1) & (calls10(i,8)>0) & (calls10(i,8) < 3600))
    len1 = len1 + 1;
  end
end

% We are going to create the file
file1 = zeros(len1,1);
k = 1;
for j = 1:1:len
  if ((calls10(j,1) == 1) & (calls10(j,8)>0) & (calls10(j,8) < 3600))
     file1(k,1) = calls10(j,8);
    k = k + 1;
  end
end


% We compute the number of zeros thanks to the mean
m = mean(file1);
nzeros = ceil(fzero('fx',1,[],[],len1,sum(file1)));
azeros = zeros(nzeros,1);

% Finally we can create the realistic file
file2 = [azeros; file1];

fid = fopen('data.dat','w');
for i=1:1:length(file2)
```

```
  fprintf(fid,'%d\n',file2(i));
end

fclose(fid);


% Function fx
function y = fx(x,n,sum1)
y = n*(exp((x+n)/sum1)-1) - x;
```

C.2.2 Mixture coefficients

```
%The following code is used to compute the mixture coefficients
load data.dat

% The log likelihood
enew = -100000;
eold = 0;

% the number of exponential distribution
i = 0;

while (abs(enew - eold)>1e-2)

  i = i + 1
  eold  = enew;
  % Initialisation
  [ncentres, priors, lambda] = emm(i);

  options = foptions;
  options(1) = 1;         % Display error values
  options(14) = 10000;    % Number of training cycles
  option(3) = 1e-8;

  [priors, lambda, options] = emmem(ncentres, priors, lambda,
  data,options);
  enew = options(8)

end

% We compute the mixture coefficient for a model with (i-1) components
[ncentres, priors, lambda] = emm(i-1);
  options = foptions;
  options(1) = 1;         % Display error values
  options(14) = 10000;    % Number of training cycles
  option(3) = 1e-8;
  [priors, lambda, options] = emmem(ncentres, priors, lambda, data,...
options);

 vect = zeros(i-1,2);
 vect(:,1) = priors';
 vect(:,2) = lambda;

save 'mixcoef.dat' vect -ascii
```

C.2.3 Exponential mixture model

```
function [ncentres, priors, lambda] = emm(ncent)
%EMM    Creates a exponential mixture model with specified architecture.
%
%  Description
%  MIX = EMM(NCENTRES) takes the dimension of the
%  space DIM, the number of centres in the mixture model and the type of
%  the mixture model, and returns a data structure MIX.
%
%  The priors are initialised to equal values summing to one, and the
%  covariances are all the identity matrix (or equivalent).  The centres
%  are initialised randomly from a zero mean unit variance Gaussian.
%  This makes use of the MATLAB function RANDN and so the seed for the
%  random weight initialisation can be set using RANDN('STATE', S) where
%  S is the state value.
%
%  The fields in MIX are
%
%  ncentres = number of mixture components
%  priors = mixing coefficients
%  lambda = Lambda parameters: stored as rows of a matrix
%

ncentres = ncent;

% Initialise priors p(j), j between[1...ncenters] to be equal and
% summing to one
priors = ones(1,ncentres) ./ ncentres;

% Initialise centres: lambda parameters
lambda = abs(randn(ncentres, 1))*0.1;
```

C.2.4 Expectation Maximisation

```
function [priors, lambda, options, errlog] = emmem(ncentres, priors,
 lambda, x, options)
%EMMEM  EM algorithm for Exponential mixture model.
%
%     Description
%       [MIX, OPTIONS, ERRLOG] = EMMEM(MIX, X, OPTIONS) uses the
%       Expectation
%       Maximization algorithm of Dempster et al. to estimate the
%       parameters of a Gaussian mixture model defined by a data
%       structure MIX. The matrix X represents the data whose
%       expectation is maximized, with each row corresponding to a
%       vector. The optional parameters have the following
%       interpretations.
%
%       OPTIONS(1) is set to 1 to display error values; also logs error
%       values in the return argument ERRLOG. If OPTIONS(1) is set to 0,
%       then only warning messages are displayed.  If OPTIONS(1) is -1,
%       then nothing is displayed.
%
%       OPTIONS(3) is a measure of the absolute precision required of the
%       error function at the solution. If the change in log likelihood
%       between two steps of the EM algorithm is less than this value,
%       then the function terminates.
%
%       OPTIONS(14) is the maximum number of iterations; default 100.
%
%       The optional return value OPTIONS contains the final error value
%       (i.e. data log likelihood) in OPTIONS(8).
%
%       See also
%       EMM, EMMINIT
%

[ndata, xdim] = size(x);

% Sort out the options
if (options(14))
  niters = options(14);
else
  niters = 100;
end

display = options(1);
store = 0;
if (nargout > 2)
  store = 1;    % Store the error values to return them
  errlog = zeros(1, niters);
end
test = 0;
if options(3) > 0.0
  test = 1;     % Test log likelihood for termination
end
eold = -100000;
```

```
% Main loop of algorithm
for n = 1:niters

  % E-step
  % Calculate posteriors based on old parameters
  [post, act] = emmpost( ncentres, priors, lambda, x);

   % Calculate error value if needed
  if (display | store | test)
    prob = act*(priors)';
    % Error value is negative log likelihood of data
    e = - sum(log(prob));
    if store
      errlog(n) = e;
    end
    if display > 0
      fprintf(1, 'Cycle %4d  Error %11.6f\n', n, e);
    %end
 end
    if test
      if (n > 1 & abs(e - eold) < options(3))
        options(8) = e;
        return;
      else
   eold = e;
      end
    end
  end

  % M-step
  % Adjust the new estimates for the parameters
  % the matlab5 instruction was new_pr = sum(post,2)
  new_pr = sum(post');
  new_pr = new_pr';
  new_c = post * x;

  % Now move new estimates to old parameter vectors
  priors = new_pr' ./ndata;
  lambda = new_pr./new_c;

end

% see Tipping 161, we now compute the negative log-likelihood
% E = -sum(n)(log(P(xn))

 foptions(8)  = -sum(log(emmprob(ncentres, priors, lambda, x)));
 fprintf(1, 'Last Cycle %4d  Error %11.6f\n', n, foptions(8));


 if (display >= 0)
  disp('Warning: Maximum number of iterations has been exceeded');
 end
```

C.2.5 Computation of the activation

```
function a = emmactiv(ncentres, priors, lambda, x)
%EMMACTIV Computes the activation of an exponential mixture model.
%
%     Description
%       This function computes the activations A (i.e. the  probability
%       P(X|J) of the data conditioned on each component density)  for a
%       exponetial mixture model.   The data structure MIX defines the
%       mixture model, while the matrix X contains the data vectors.
%       Each row of X represents a single vector.
%
%       See also
%       EMM, EMMPOST, EMMPROB
%


% we compute the number of data, n, in our problem
  ndata = size(x, 1);

% Calculate squared norm matrix, of dimension (ndata, ncentres)

%create the n*m matrix [x1..x1, x2..x2, .., xn..xn]
  ax = x*ones(1,ncentres);

% create the m*m matrix [l1 0..0,0 l2 0..0, ..,0..0 lm]
  al = diag(lambda);


% create the n*m matrix [exp(-l1x1)..exp(-lmx1), .., exp(-l1xn)..
%exp(-lmxn)]
  ae = exp(-ax*al);

% create the N*M matrix [l1*exp(-l1x1)..lm*exp(-lmx1),..,];
  a = ae*al;
```

C.2.6 Posterior probabilities computation

```
function [post, a] = emmpost(ncentres, priors, lambda, x)
%EMMPOST Computes the class posterior probabilities of an exponential
%  mixture model.
%
%     Description
%       This function computes the posteriors POST (i.e. the
%       probability of each component conditioned on the data P(J|X))
%       for a exponential mixture model. The data structure MIX
%       defines the mixture model, while the matrix X contains the
%       data vectors.  Each row of X represents a single vector.
%
%       See also
%       EMM, EMMACTIV, EMMPROB
%

ndata = size(x, 1);

% Compute the ((P(Xn|J) )) of the data conditioned on each component
% density we obtain a N*M matrix
a = emmactiv(ncentres, priors, lambda, x);

% We compute  the following matrix, M*N, [P(1)P(x1|1)..P(1)P(xn|1),..,
% P(m)P(x1|1)..P(xn|m)P(m)]
 post = diag(priors)*a';

% we have the following matrix, n*1, [sum(k,1,m)P(k)P(x1,1k)..
% sum(k,1,m)P(k)P(xn,1k)        (see Tipping (206))

if ( min(size(post)) == 1) % we have to correct this line
  s = post;
 else
  s = sum(post);
 end

% Set any zeros to one before dividing
s = s + (s==0);
sum(s)

% ((P(J|Xn))) it's an M*N matrix
post = post./(ones(ncentres,1)*s);
```

C.2.7 Data probability computation

```
function prob = emmprob(ncentres, priors, lambda, x)
%EMMPROB Computes the data probability for a exponential mixture model.
%
%     Description
%       This function computes the unconditional data density P(X) for a
%       Exponential mixture model.  The data structure MIX defines
%       the mixture model, while the matrix X contains the data
%       vectors.  Each row of X represents a single vector.
%
%       See also
%       EMM, EMMPOST, EMMACTIV
%


% Compute activations:((P(X|J)))
a = emmactiv(ncentres, priors, lambda, x);

% Form dot product with priors P(X) = sum(k) P(X|k)P(k)
prob = a * (priors)';
```

C.3 Revisiting optimisation program

Last year's code has been used as the basis of the following top-level program. As lots of functions haven't even changed, we won't including their code again, we are just going to talk about the Hmm_nagents.m function and its following which remplace the old H_agents.m function.

C.3.1 Optimisation of staff scheduling

```
load skills;
skills = skills;
% Rescale to range 0-1.
skills = skills ./100.0;
% Only use the first four skills profiles to make it interesting
% In the people-based approach this line should be removed
skills = skills(1:4, :);

% Read in the data matrix describing the activity of the queues
load volume;

% time_steps_a_day is the number of half hour periods in a day
time_steps_a_day = 20;

% Read the mixture coefficient matrix
load mixcoef.dat
prior = mixcoef(:,1);
lambda = mixcoef(:,2);

% Load the queuing calls
load qu_data_frac1.dat;

qu_data_frac = qu_data_frac1;
% Compute what should be the number of agents answering calls
% from each queue, according to the stochastic modelling of the queues
H_q = zeros(120,8);
%H_q = Hmm_ratio(prior,lambda,volume,time_steps_a_day);
H_q = Hmm_nratio(qu_data_frac,prior,lambda,volume,time_steps_a_day);
% queue contains the required number of agents in each queue
....
% The rest is also the same
```

C.3.2 Number of agents in each queue

```
function H = Hmm_nratio(frac,prior,lambda,volume,time_steps_a_day)

% This function calculates the number of servers required
% if we merge all the queues(M/M/c system) in one system
% considered as a M/H6/c system
% It also computes the normalized number of agents
% required in each queue

% volume provided all the data describing the six queues :
% number of incoming calls per queue and per half-hour period
% mean service time per queue and per half-hour period
% number of server required in each queue considered separately

% time_steps_a_day is the number of half hour periods in a day
%
% nb contains the number of abandoned calls a t = 0,...,nb
% queue, the number of calls in the queue a t=0,...,nb

% number of queues in our system
num_q = size(volume,1)/time_steps_a_day;
% number of days considered in the week
num_day = size(volume,2)/3;
% total number of half hour periods for the whole week
num_time_steps = num_day*time_steps_a_day;

% queue contains the numbers of calls active at an instant in each queue
queue = zeros(1,num_q);

% the required number of agents in each queue for each half hour period
q = zeros(num_time_steps,num_q);
% the total number of servers, i.e. agents required in the system
c = zeros(num_time_steps,1);
% the server utilisation
ro = zeros(num_time_steps,1);

% First line: call volume in each queue
% Second line: average call duration in each queue
param = zeros(2,num_q);

for n=1:1:num_time_steps

  % Search for the call volume in the matrix volume
  for i = 0:1:num_q-1
    param(1,i+1) =
  volume(rem(n-1,time_steps_a_day)+1+time_steps_a_day*i
,... 3*ceil(n/time_steps_a_day)-2);
  end

  % Search for the average call duration in the matrix volume
  for i = 0:1:num_q-1
    param(2,i+1) =
  volume(rem(n-1,time_steps_a_day)+1+time_steps_a_day*i
...3*ceil(n/time_steps_a_day)-1);
  end
```

```
  total = 0;

  % Count the total number of calls active at an instant
  for i = 1:1:num_q
    queue(1,i) = param(1,i)*param(2,i)/(30*60);
    total = total + queue(1,i);
  end

  % Compute the total number of agents required in the system
  % to answer 95% of all calls within 30 seconds
  if total > 0  w = zeros(1,2);
        w = Hmm_nagents(frac,prior,lambda,param,95,30);
        r = w(1)/total;
        c(n,1) =  w(1);
                ro(n,1) = w(2)/w(1);
           else r = 1; ro(n,1) = 1; c(n,1) = 0;
  end

  % Compute the number of agents required in each queue
  for m = 1:1:num_q
    q(n,m) = queue(1,m)*r;
  end
end

% The function returns q, c and ro
H = [ q c ro ];
```

C.3.3 Number of agents to achieve the right service level

```
function H = Hmm_nagents(frac,prior,mu,param,p,t)

ar = 0;
arold = -100000;

wd = 0;
wdold = -100000;

while ( (((wd - wdold) >1e-4)|((ar - arold)>1e-4))&(ar<0.050))

   % we compute amongst other things the average queuing time
  P = Proc_wd(prior, mu, param, ar, p, t);

  % wd is the average waiting time in the queue
  wdold = wd;

  wd = P(1);
  n_agents = P(3);
  arold = ar;
  ar = 0;
  for i =1:1:t
    ar = quad('AW',i-1,i - 1e-4,[1e-4 1e-4],[],wd,frac) + ar;
  end

end


% a is the offered load in erlangs of the system
a = P(2);

% the function returns n_agents and a
H = [ n_agents, a];
```

C.3.4 Computation of agents

```
function P = Proc_wd(prior, mu, param,ar,p,t)

% Proc_wd calculates the minimum number of agents in order to
% answer p/(1-ar)% of all the calls within t seconds

c = 1;
w = zeros(1,2);
w = Wmm_q(prior,mu,param,c);

while w(1) > (t/log(100/(100-p/(1-ar))))
  c = c+1;
  w = Wmm_q(prior,mu,param,c);
end

P = [w c];
```

C.3.5 Computation of the p.d.f. of the fraction of abandoned calls

```
function y = AW(x,wd,frac);

W = lexp(1/wd,x,0);

% this function is null except between [0 ,length(anb)-1]
A = zeros(length(x),1);

% we have to take into account the fact that the first
% element of the file is zero

for i =1:1:length(x)
  if ((x(i) >=0)& (x(i) <=(length(frac)-1)))
    A(i,1) = frac(floor(x(i))+1);
  end
end

y = W.*A;
```

C.3.6 Exponential function

```
function y = lexp(lambda,x,x0)

% The function lexp evaluates the exponential function

   y = ((x>=x0).*lambda.*exp(-lambda*(x-x0)))';
```

C.3.7 Average queueing time

```
function w = Wmm_q(prior,mu,param,c)

% This function Wmm_q calculates the average queueing time
% for a M/H6/c system:

% Exponentially distributed arrival times

% The service time distribution is a 6 stages hyperexponential
% distribution
% (the parameters describing this two distributions are in param

% c servers

% The priors are the mixture coefficient
% The lambda are the coefficient of the exponential distribution

% lambda is the total arrival rate, that is equal to the sum of
% the six single arrival rates of each queue

          num_q = size(param,2);

          lambda = 0;
          for i = 1:1:num_q
```

```
            lambda = lambda + param(1,i);
   end
   lambda = lambda/1800;

% E_s is the expected value of s, the service time
          E_s = 0;
          for i = 1:1:num_q
            E_s = E_s + param(1,i)*param(2,i)/1800;
   end
   E_s = E_s/lambda;

% a is the offered load of the system
   a = lambda*E_s;

% the offered load must be strictly less than the number of servers
% else the number of calls waiting grows without limit
   if (c-a) < 0  q = 1000000;
   else
        E_s2 = 0;
                for i = 1:1:num_q
% E_s2 is equal to (E_s)^2
  E_s2 = E_s2 + param(1,i)*param(2,i)^2/1800;
        end
        E_s2 = 2*E_s2/lambda;
                M = 0;
S = 0;
for j = 1:1:length(prior)
                  M = M + prior(j,1)/mu(j,1);
  S = S + prior(j,1)/(mu(j,1)^2);
end
E_s2 = E_s2*S/M^2;
       C = Erlang_C(c,a);
% q is the average waiting time in the system
q = C/(c-a)*E_s2/(2*E_s);
    end

% The function returns q, the average queueing time in the system,
% and a, the offered load of the system
    w = [ q a ];
```

Bibliography

Allen, A. O. (1978). *Probability, Statistics and Queueing Theory with computer science applications.* Academic Press, Inc.

Beckmann, P. (1968). *Introduction to Elementary Queueing Theory and Telephone Traffic.* The Golem Press.

Bishop, C. M. (1995). *Neural Networks and Pattern Recognition.* Oxford University Press.

Kleinrock, L. (1975). *Queueing Systems Volume I: Theory.* John Wiley and Sons Inc.

Kleinrock, L. (1976). *Queueing Systems Volume II: Computer Applications.* John Wiley and Sons Inc.

Lecoroller, G. (1997). *MSc Thesis: Human resource allocation in a call centre.* Aston University.

Lee, A. M. (1966). *Applied Queueing Theory.* Mac Millan.

Lowe, D. and Nabney, I. (1996). Human resource allocation in a call centre: Feasibilty study report.

Nabney, I. T. (1997). *Algorithms and Computational Mathematics, Lecture notes.*

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press.

Saaty, T. L. (1961). *Elements of Queueing Theory with Applications.* McGraw-Hill Book Company.

Tipping, M. (1997). *Statistical Pattern Analysis, Lecture notes.*