632 | 2010

Jean-Christophe Van den Schrieck

Multi-skill Queueing Models for Call Centers

Multi-skill Queueing Models for Call Centers: Approximations and Performance Optimization

JEAN-CHRISTOPHE VAN DEN SCHRIECK

SEPTEMBRE 2010

Thèse présentée en vue de l'obtention du grade
de docteur en sciences économiques
et de gestion

Faculté des sciences
économiques, sociales, politiques et
de communication

Université catholique de Louvain

UCL

MULTI-SKILL QUEUEING MODELS FOR CALL CENTERS: APPROXIMATIONS AND PERFORMANCE OPTIMIZATION

JEAN-CHRISTOPHE VAN DEN SCHRIECK

Thèse présentée en vue de l'obtention du titre de docteur en sciences économiques et de gestion

PROMOTEUR:

Philippe Chevalier

JURY:

Per Agrell, Université catholique de Louvain

Zeynep Akşin, Koç University

Ger Koole, Vrije Universiteit Amsterdam

Nico Vandaele, Katholieke Universiteit Leuven

Laurence Wolsey, Université catholique de Louvain

Septembre 2010

Jean-Christophe Van den Schrieck: *Multi-Skill Queueing Models for Call Centers: Approximations and Performance Optimization*, Thèse présentée en vue de l'obtention du titre de docteur en sciences économiques et de gestion, © Septembre 2010

Les enfants sont des énigmes lumineuses.

— Daniel Pennac

A Arthur, ce sacré petit bonhomme qui me fait rire en toutes circonstances.

ABSTRACT

Our main objective in this thesis is to support the management of call centers by developing tools and methods to decide on the number of operators required in a call center. We use existing queueing theory models to analyze call centers with several types of calls. In this context the goal is to find the most efficient configurations, i.e. configurations that achieve the best performance at a minimum cost by combining different sorts of operators. Some can only answer calls of one type, others are skilled to treat calls of different types, but at a greater cost.

We propose a Branch and Bound method to find the best combinations of operators while keeping the integrality constraints on the number of operators. The latter constraints matter in small-size call centers.

Looking for efficient configurations requires assessing the performance in the models. We propose new methods to estimate the waiting probability, the average waiting time or the service level. These methods exploit the similarities observed in single-skill models between systems with no queue and the same systems with a queue of infinite size. Using Hayward's approximation, that permits to approximate the probability for a call to not be handled in a multi-skill system with no queues, we show how to estimate the performance of the same multi-skill system when blocked calls are put on hold.

In the process we also present the peakedness functional. It is a measure of variability of the stochastic flows. We explain how to extract the peakedness from real-life data and to use it in our models. Different applications are proposed.

RÉSUMÉ

La motivation première de cette thèse est d'aider à la gestion de centres d'appels en développant des outils et des méthodes pour décider du nombre d'agents nécessaires dans le centre d'appels. Concrètement, nous travaillons avec les modèles de théorie des files d'attente existants pour analyser les centres d'appels qui reçoivent des appels de plusieurs types. Dans ce cadre, l'objectif est de trouver les configurations qui permettent d'atteindre la meilleure performance à un moindre coût en combinant des operateurs de plusieurs sortes: certains ne traitent qu'un seul type d'appels, d'autres sont capables de répondre à des appels de plusieurs types.

Nous proposons une méthode d'optimisation "Branch and Bound" qui conserve l'intégralité de la solution, contrainte importante pour les centres de petite taille.

La recherche de configurations efficaces requiert d'être capable de mesurer la performance atteinte dans ces modèles. Nous proposons de nouvelles méthodes pour évaluer les mesures de performance classiques, comme la probabilité de devoir attendre, le temps moyen d'attente ou le niveau de service. Ces méthodes exploitent les similitudes observées dans les modèles avec un seul type d'appels entre les systèmes où il n'y a pas de file d'attente et les systèmes avec une file d'attente de longueur infinie. Grâce à l'approximation d'Hayward, qui permet d'estimer la probabilité qu'un appel ne reçoive pas de réponse quand il n'y a pas de file dans un système avec plusieurs types d'appels, il est possible d'estimer la performance du même système quand les appels bloqués sont mis en attente.

Nous présentons aussi la peakedness. Il s'agit d'une mesure de variabilité des processus d'arrivée. Dans cette thèse, nous expliquons comment la peakedness peut être déterminée à partir d'un échantillon d'arrivées et nous montrons plusieurs usages qu'il peut être fait de cette mesure.

PUBLICATIONS

Some of the ideas presented in this thesis have appeared previously in the following publications:

- Chevalier, P. and Van den Schrieck, J.-C. (2008). Optimizing the staffing and routing of small-size hierarchical call centers. *Production and Operations Management*, 17(3):306 – 319
- Chevalier, P. and Van den Schrieck, J.-C. (2008). Approximating multiple class queueing models with loss models. *Core Discussion Papers*, 21

ACKNOWLEDGMENTS

A thesis is never the work of a single person. It is a long process that one cannot achieve without the intellectual and moral support of many.

First and foremost I am grateful to Philippe Chevalier, my PhD advisor, who has been present from the beginning to the end to provide advice, new orientations, encouragements and useful comments. Without his support, I would not have been able to achieve this thesis.

The second person I want to thank is Zeynep Akşin for her involvement in my work. In particular, I am grateful to her for inviting me to come and work for three months at Koç Univeristy. This stay in Istanbul was intellectually and culturally rich.

Besides the two of them, I am grateful to the other members of my Jury for their interest in my work and for the time they spent reviewing it. I value their comments and meaningful suggestions a lot.

My gratitude also goes to my whole family: my parents, who have always been supportive to me, trusting me in all my decisions; my grand-parents whose succesful lifes and experiences have always been an inspiration to me; my sister and her husband, whose dynamism and successes I admire much, the greatest to date being their little Arthur; my brother, so different, so close; my godparents and the remaining of the family.

I'd like also to mention my friends, especially Jean-François, Jérôme, Gilles, Xavier and Yaya. I am grateful for their constant support and for their efforts to push my head out of the thesis from time to time. I value their friendship dearly.

I also acknoledge Bénédicte for her constant support during these precious years. I will never forget this.

There is a large bunch of people at LSM and Core I also would like to mention here. Alltogether, they created and maintained an atmosphere that contributed much to this achievement: the sound emulation as well as the entertainement they brought has been valuable since the beginning. In particular, I am grateful to: the one with whom I spent so many hours helping students in case studies; the one who speaks of his native Germany with such an irony; the one who does not hesitate to ask questions, most of which quite interesting; the one who likes to play the axis, which is good as I prefear to play the allies; the one with whom I had so many uncontributing but entertaining netmeetings; the really funny one who speaks and laugh so loudly; the

one who walks barefoot at the office; the one with an airsoft gun in her desk and who pretends not to like children (she is expecting her second one); the one who is always complaining; the one with whom I can speak for hours about the battle of Waterloo; the one who came from China to discover the peakedness; the one who is always out of office; all the other ones.

I also would like to thank the whole Koç team for welcoming me so nicely in Turkey. In addition to Zeynep, my gratitude goes to Selim, Taylan, Aytüg, Fikri, Lerzan...

Finally thanks to my secret source of motivation in these long lasting months of writing.

Thanks to all of you and the ones I forgot to mention here: without you I could not have made it.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

GENERAL INTRODUCTION

The focus in this thesis is on call centers. Call centers have been studied for at least two decades now from various perspectives, all equally important and interesting. If you look at a call center from a human resource perspective, the desired objective is to decrease the turnover. From a customer point of view, it is to answer the requests in the best possible conditions. From an operations management point of view, the goal is to minimize the operating costs while ensuring a given level of service. In this thesis we take this last perspective. More specifically, our final goal is to provide managers with tools to help achieving a certain level of performance in the call center at a minimum cost.

We use the framework offered by queueing theory to develop our tools and applications. Models of queueing theory represent a call center as a set of agents (operators, or servers) that face uncertain external demand (the arriving calls) and deal with them in uncertain processing times (the service time), if possible. Within this general framework many models have been developed.

In particular we focus on multi-skill call centers. These are call centers where calls of different types arrive and are answered by the operators. This is for example the case of a call center that receives calls in several languages.

Two of the known advantages of multi-skill call centers as compared to call centers with homogeneous calls are the synergies created by centralizing the fixed costs on the one hand and the economies of scale brought with the aggregation of the variability on the other hand. Proportionally, the total variability decreases when we aggregate two independent flows of uncertain demand. We focus on this latter effect in our analysis.

In the mean time, centralizing demands of several types poses difficulties. It is often difficult to find operators skilled to answer all types of calls and they are often more expensive to hire than operators with one unique skill. In practice call centers use operators with different combinations of skills and this permits them to capture

most of the benefits of aggregating demand. However the presence of operators with different skills makes operations management more complex. Questions such as "how many operators do we need and with which set of skills?" or "how to allocate calls to the different operators?" are difficult to solve in this context. Providing some answers to these questions is the ultimate objective of our research.

Answering such questions requires to be able to assess the performance of the system. Our secondary goal is to find methods to accurately estimate the performance attained with a configuration in our models. In some conditions, this can be done analytically but it requires making strong assumptions about the behaviour of the elements in the model. Simulations are an alternative way of computing the performance. This is probably the way of determining the performance with the widest range of application but it might require a lot of resources and/or time to obtain a reliable measure of performance. A third approach is to use approximations in the model. Approximations permit one to evaluate the performance in a relatively accurate way based on inexact or empirical assumptions that do not impact much the quality of the measurement. Good approximations should balance accuracy and ease of use. Note that the two first approaches are approximations as well, in the sense that they permit one to determine exactly or with great accuracy the performance in models which are approximations of real-life situations.

We do not consider a call center in particular in this thesis. We work at a different level by using existing models as a framework for the development of our tools. The results need to be adapted to real-life situations but we consider this as beyond the scope of this work.

The thesis is structured as follows. In the first chapter we describe the research context. This includes reviewing the literature on call centers, describing the queueing models we are working on and presenting the general methodology. The following chapters are divided into three parts.

In the first part, Chapters 3 to 5, we introduce an existing approximation procedure to measure the performance in multi-skill loss systems, i.e. systems where blocked calls cannot wait, and adapt it to the specific features of our models. In Chapter 3, we present a measure of variability of flows that is well suited for queueuing models: the peakedness. We also show how to compute the peakedness from real-life data. In Chapter 4, we describe Hayward's approximation. It permits us to compute the loss probability in loss systems where the incoming flow is not a Poisson process. It uses the peakedness as a second order parameter for the incoming flow. We also show how Hayward's approximation can be used in the context of the outsourcing of flows in call centers. Chapter 5 explains how Hayward's approximation can be adapted to

compute the performance of multi-skill loss systems. Adaptations are also proposed to take into account different sets of assumptions in the models.

In the second part, i.e. Chapter 6, we propose a procedure to find the best configurations of operators in small-size multi-skill loss systems, subject to performance constraints. In such systems, the integrality of the decision variables is a much more important issue than for larger call centers. In this context, we search for the minimum cost configuration subject to service level constraints using a Branch and Bound algorithm. What is at stake is to find the right balance between gains resulting from the economies of scale of pooling and the higher cost of cross-trained agents. We show that in most cases our method allows us to significantly decrease the staffing cost compared to configurations with only cross-trained operators or only dedicated operators. This chapter actually reproduces the content published in [Chevalier and Van den Schrieck, 2008].

In the third part, Chapters 7 to 10, we develop approximation methods to extrapolate the performance obtained with a configuration in multi-skill queueing systems from the performance observed with the same configuration in the equivalent multi-skill loss model. In Chapter 7, we analyze the relationship between queueing and loss models in a single-skill environment. In particular, we provide an interpretation for the well-known algebraic connection between the Erlang-B and the Erlang-C formula. In Chapter 8, we exploit the results of Chapter 7 to build approximations for the waiting probability, i.e. the probability that a call must wait, in a multi-skill queueing model based on the loss probability found in the equivalent loss models. Chapter 9 proposes an alternative approximation for the waiting probability. The approximation is extended to evaluate other performance measures too. Finally, we compare the two methods in Chapter 10.

We summarize our main results and list some of the possible extensions of this thesis in Chapter 11.

2

RESEARCH CONTEXT

In this chapter, we describe the model we work with throughout the thesis. First we assess its pertinence with a description of the literature. The review focuses on the general context of the research. More specific reviews of the topics tackled along the thesis are presented in their related chapters. In Section 2.2, we describe the models of queueing theory we are using in the thesis. We first introduce a few concepts of queueing theory and describe them in single-skill models. After that, we present the multi-skill models we use as our framework, and the related assumptions. This section is followed by a brief discussion of the limitations of our models. In Section 2.4 we present a general description of the methodology. Finally we present the datasets we are going to use in our simulations experiments.

2.1 GENERAL CONTEXT

The development of telecommunications in the last two decades has been phenomenal. The possibilities offered by new telecommunication technologies to provide services remotely have led to the development of a lot of new business opportunities. Demand for such services has boomed in the last decades in the business to consumer (B2C) sector as well as in the business to business (B2B) sector.

An important way of providing these types of services is through call centers, where clients call to obtain some kind of service (e.g. after sales, helpdesk, financial,...). The growing importance of call centers for operations management in services has led to a growing interest of researchers in this topic. [Gans et al., 2003] and [Akşin et al., 2007a] present very well documented surveys of the research in the field of call centers. [Mandelbaum, 2006] presents an impressive list of publications, with their abstracts, that complements the two above-mentioned survey papers.

With the growing importance of call center services, new businesses have appeared. Nowadays some firms specialize in call center operations and provide services to

other firms that choose to outsource the call center part of their own businesses. We address some specificities of call center outsourcing in Section 4.3. A review of the literature on call center outsourcing is presented in Section 4.3.2.

The main incentive for firms to either outsource or specialize in call center operations is the opportunity of benefiting from the possible economies of scales.

Such economies of scale can be attained in two possible ways. They both aim at increasing the size of the call center and could be used simultaneously. The first option is to increase the size of the external demand, i.e. the calls, by centralizing call center operations. For example, instead of operating five call centers in five different regions, operating one common call center for the five regions permits to smooth the demand and to reduce the relative variability. Large call centers that operate on the whole North-American territory try to benefit from this effect. Centralization however has its limitations: a call center cannot grow larger than the size of the demand[1] and, the purpose of the call center being very often to be an after sales service, there is no incentive to generate additional demand. Fluid approximations are often used to model the behaviour of large-size call centers, where high utilization is often sought. [Harrison and Zeevi, 2005] presents a method for staffing large-scale call centers based on fluid approximations. It applies to systems where most of the randomness is due to the non-stationary arrival process. [Whitt, 2006b] describes a multi-class deterministic fluid model for a multi-skill call center used to approximate systems with large arrival rates and high numbers of servers. Other works by Whitt on large-scale call centers include [Halfin and Whitt, 1981], [Whitt, 2005], [Whitt, 2006a]. [Jiménez and Koole, 2004] review the different fluid methods used in models with overloads, compare them and improve some of them to propose new approximations.

The second option to increase the size of a call center is through diversification. Instead of operating one call center for each of two similar services, operating one call center with operators skilled in answering both types of requests permits one to benefit from economies of scale. Flexibility is here the key driver to the economies of scale: it is through the employment of cross-trained staff, i.e. employees that have some expertise in more than one field, that companies can benefit from the possible economies of scale. Although cross-training brings more flexibility, it is more expensive than hiring single-skilled employees. Such cross-trained employees are scarce too, making it often impractical to have all employees capable of handling each task. Nevertheless research in this field shows that "a little flexibility goes a long way". In other words, it is possible to reap most of the benefits of a fully cross-trained workforce with much less cross-training. On this topic, the reader might be interested by [Wallace and Whitt, 2005], [Akşin et al., 2005] or [Chevalier et al., 2005]. The first article

1 As it will become clear later on, we only consider "passive" call centers, that receive calls. Call centers that actively contact potential customers are not in the scope of this thesis.

illustrates that if all operators are cross-trained, hiring double skilled agents permits to capture most of the flexibility. The second paper investigates the value of resource flexibility and flexibility of the structure. In the third paper, the authors find out that if only a fraction of operators are cross-trained, a good practice would be to dedicate twenty percent of a staffing budget on flexible agents. Our focus in this thesis is on such cases where the call center has operators with different skills and faces different types of demand.

Queueing theory serves as the main theoretical background in call center analysis. This theory is widely known and taught (see [Cooper, 1972], [Khintchine, 1960], [Kleinrock, 1975], [Wolff, 1989], [Gross and Harris, 1998] among others for books on this theory) but it primarily applies to situations in which only one arrival stream, the calls in the context of call centers, is serviced by only one group of operators.

The two main research questions related to multi-skill call centers are "How to staff the call center?" and "How to route the calls within the call center?". [Koole and Pot, 2006] review the existing literature on these two research questions. [Akşin et al., 2007b] presents a complete review of the existing research on cross-training in call centers. More specifically, [Pot et al., 2008], [Cezik and L'Ecuyer, 2008], [Avramidis et al., 2009] among others all address cross-training in different ways. Articles on the routing of calls include [Örmeci, 2004], [Bhulai, 2009], [Borst and Seri, 2000], [Sisselman and Whitt, 2007a].

Other works on multi-skill call center models include [Maglaras, 1999], [Armony and Bambos, 2003], [Bambos and Walrand, 1993] and [Gans and van Ryzin, 1997].

[Maglaras, 1999] and [Armony and Bambos, 2003] present models similar to those we use in this thesis.

[Bambos and Walrand, 1993] treat another kind of systems, with different types of calls (referred to as jobs). Jobs require a combination of different operators/processors that, together, will simultaneously process them. The focus is also on the best way to schedule the different types of jobs.

[Gans and van Ryzin, 1997] present another type of multi-skill system. In their model, each pool of operators has its own queue in front of it and jobs need to be processed successively in different pools. They propose routing policies to process the different jobs in the shortest possible time.

2.2 MODEL DESCRIPTION

In this section we present the different models that we analyze in this thesis. In Section 2.2.1, we briefly review single-skill models and present the basic elements and

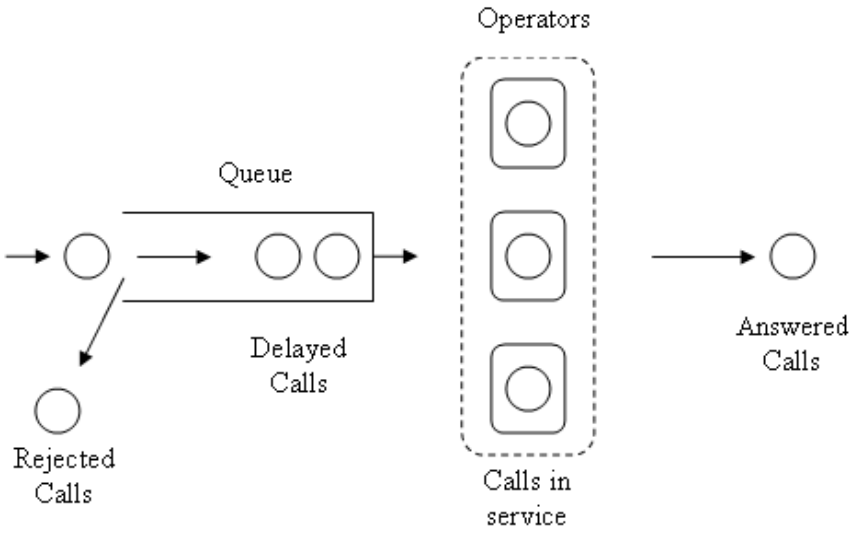

Figure 2.1: Main features of queueing models.

important concepts of a queueing model. After that, in Section 2.2.2, we describe the multi-skill queueing models.

2.2.1 *Single-skill queueing models*

Any queueing model consists of three main elements: arrivals, operators[2] and a queue. There are different kinds of queueing models, each with their own characteristics. Kendall's notation classifies the models with a three- or four-factor notation. The first factor describes the arrival process, the second the distribution of the service times, the third the number of operators and the fourth the size of the system. If the size of the system is not limited, the fourth factor is omited in the notation.

Figure 2.1 depicts a typical queueing model.

The arrivals, the calls actually, are represented as bullets in Figure 2.1. They come from an external source and are inputs to the model. Upon its arrival, a call is either accepted or rejected by the system. If accepted it enters the system and goes to the queue where it is put on hold. After some delay (its might be a zero-delay) it is serviced and upon the completion of the service, it leaves the system. Arrivals are modeled as a stochastic flow. Throughout this thesis we usually assume that the arrival streams are

2 We use "operators" or "servers" indifferently to designate the agents that serve the arriving calls.

Poisson processes of rate λ. Other types of processes are possible though. In Kendall's notation, a M-symbol for the first factor indicates Poisson arrivals and a G-symbol a general distribution of interarrival times.

The three boxes inside the dashed rectangle in Figure 2.1 represent operators. Each operator answers calls one at a time in a random time. This time is called the service time. The distribution of the service time might be of any type but we usually assume it to be exponentially distributed with parameter $\mu = 1$. In Kendall's notation a M in the second factor is used to indicate exponentially distributed service times. s in the third factor indicates the number of operators.

The open box in front of the operators represents a queue. Depending on the model, the queue may have different sizes. In the M/M/s queueing model, the size of the queue is assumed to be infinite, In a the M/M/s/s loss model, the size of the queue is equal to zero, meaning that a call cannot enter the system if all operators are busy. The call is then rejected. The size of the system is thus limited to the s sites for the calls being serviced. This justifies the use of a s for the fourth element in Kendall's notation. In between those two extreme cases are the M/M/s/s+m models where the length of the queue is limited to m calls. This means that an arriving call can enter the system if there are less than m calls waiting. When there are m waiting calls, i.e. the queue is full, then an arriving call is rejected. A call waits in the queue until an operator becomes available. The first call in the queue then leaves the queue and is serviced by the operator.

We assume in this thesis that the discipline in the queue is First Come First Served (FCFS) but other disciplines can be considered in queueing models.

Another feature that can be represented in a queueing model but that we do not consider in this thesis is the abandonments, i.e. calls that become impatient and that leave the queue before being answered. The literature ([Subba Rao, 1965], [Whitt, 1998], [Jouini et al., 2009], [Jouini et al., 2010] and references therein) usually recognizes two types of abandonments : calls that leave the system upon realizing they need to wait (this is called "balking") and calls that abandon after waiting a while (this is referred as "reneging").

Note the difference between the queue, the service and the system. Calls in the queue are all the calls that are waiting before being serviced. Calls in service are the calls that are associated to an operator and are effectively in process. The system comprises the queue and the service. The time spent by a call in the queue is the waiting time. It does not includes the service time. The total time spent by a call in the system is the sum of its waiting time and its service time.

We call a blocked call any call that does not find an available operator upon its arrival. Depending on the model considered, blocked calls are either rejected, i.e. they

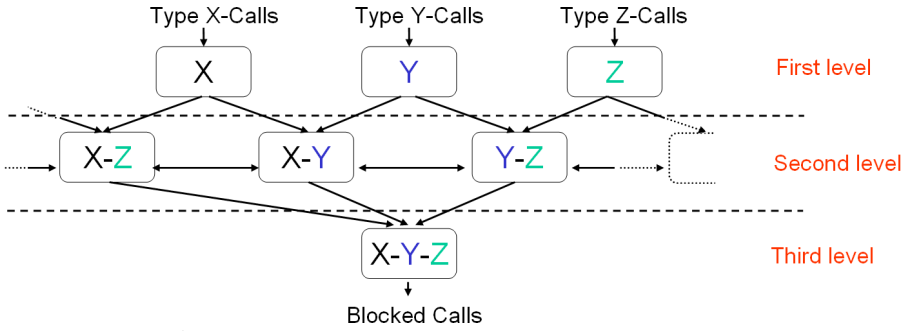

Figure 2.2: A multi-skill queueing system with three types of demands.

exit the system without being answered and are considered lost, or put on hold, i.e. they wait until an adequate operator becomes available. We talk in the former case about a loss model and in the latter case about a queueing model.

2.2.2 *Description of the multi-skill models*

Here, we present the multi-skill models that are used throughout thesis. Our models fall within the more general framework used in [Maglaras, 1999] and [Armony and Bambos, 2003]. Both articles consider systems with one queue for each type of call and different routing policies. They focus on the problem of scheduling calls to the different operators (or, equivalently, of assigning operators to waiting calls when they become available), according to their skills and the current state of the system. Both give particular attention to the stability conditions of multi-skill systems under their routing policies. We work with similar systems but restrict our attention to a FCFS routing policy within each queue.

In a multi-skill model, the system receives arrivals of different types. Each type of call requires a different skill in order to be served. We assume that each arrival stream is Poisson distributed. The call center has different kinds of operators that answer the arriving calls. Each operator has a set of skills which determines the types of calls that he can answer. An operator with n different skills is called a n cross-trained operator. Operators with a same skill set are grouped into an homogeneous pool., i.e. operators in one pool are equally qualified. We classify the pools according to the number of skills of their operators. We call the n-th level pool the set of pools with operators having exactly n different skills.

Example 2.1 *Figure 2.2 illustrates a pool with three types of arrivals, x, y and z. There are seven different pools to answer the calls. Three dedicated pools x, y and z form the first level.*

The three 2-skilled pools xy, yz and xz, that each accept calls of two types, form the second level. The third level only consists of pool xyz with fully polyvalent operators.

An arriving call is associated to an available operator with the adequate skill. We say that the operator serves the call. If such an operator does not exist, the call is either rejected or put on hold, depending on the model. If there is more than one skilled operator available, the call is associated to one of them according to a routing policy. Many routing policies are possible. For example, a call may select an operator randomly, or the operator who has been idle for the longest time will be selected, or the operator will be chosen according to the skills he has. In this paper, the favoured routing policy is the commonly named hierarchical routing. Under this routing policy, an arriving call is prioritarily associated to an available operator with the lowest level of skills. For example it can be an operator of its first-level pool, i.e. an operator that can answer calls of this type only. If there is no dedicated operator available, then the call is routed to a pool of the second level, where it is checked whether an operator is available there. And so on. The pool of operators that can answer all types of calls is visited last. The principle of the hierarchical routing is to keep more flexible operators available for future uncertain demand by routing arriving calls to the group with an available agent that has the smallest level of cross-training. Such a policy is often used in the literature: [Franx et al., 2006], [Pot et al., 2008], [Koole and Talim, 2000], [Bhulai, 2009], [Borst and Seri, 2000]. The sequence of pools visited by an arriving call before it is blocked forms an overflow path. When there is more than one pool with skilled operators to answer calls of a given type in a level, there might be more than one possible overflow path.

Example 2.2 *In Figure 2.2, the arrows represent flows in the system. From the configuration we deduce that the systems works with a hierarchical routing policy. The flow from pool x to pool xy is made of all x-calls that did not find an available operator in pool x and that are routed to pool xy. Answered calls are not represented in the figure. The sequence $\{x, xy, xz, xyz\}$ form an overflow path for the x-calls. It means that pool x is visited before pool xy that is visited before pool xz and that pool xyz is the last visited pool. Here, there is more than one overflow path for each type of calls. A type-x call that is not answered by an operator of pool x can be routed to either pool xy or pool xz. A second overflow path for type-x calls is thus $\{x, xz, xy, xyz\}$.*

In this context one option is to consider that all calls use only one of the possible overflow paths. This assumption is appropriate when in practice the dispatching rule routes arriving calls in one level to the same available operators in priority. Another possibility is to assume that a call chooses one of the overflow paths randomly. This approach corresponds to situations where the call center management wants to bal-

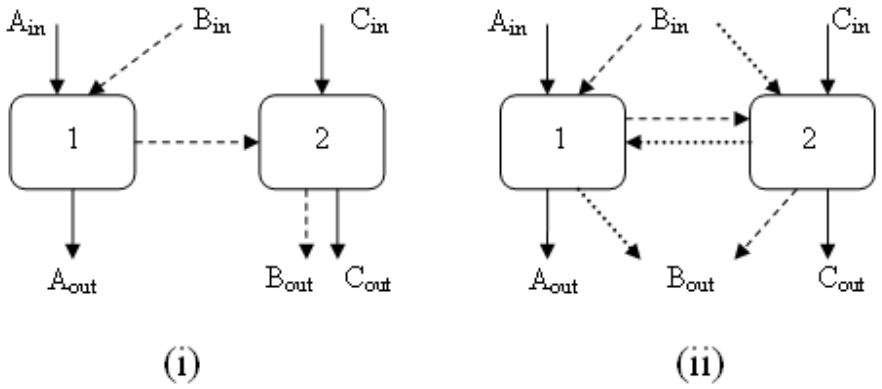

Figure 2.3: Two routing policies. In (i) there is only one overflow path for B-calls whereas in (ii) two overflow paths are considered.

ance the occupation of its operators at one level so that there is capacity for available for calls of other types. We illustrate this in the following example.

Example 2.3 *Consider a system that receives call of three different types, A, B and C, such as illustrated Figure 2.3. There are two types of operators to answer these calls. Pool 1 has operators that can answer calls of types A and B and pool 2 serves calls of types B and C.*

In Figure 2.3 (i), calls of type B are all first routed to pool 1 and the B-calls that cannot find an available operator are routed to pool 2. This routing policy increases the occupation of the operators of pool 1. The capacity available for type-A calls will decrease and this will therefore increase the loss probability for these calls. We have the opposite effect at pool 2.

The situation presented in Figure 2.3 (ii) differs as here B-calls can first visit either of the two pools. This will tend to balance the occupation of operators in both pools and this could possibly result in a smaller difference in loss probability between calls of type A and calls of type C.

We assume that the service times are independent of both the call type and the operator that serves the call and that they are exponentially distributed. The i.i.d. assumption is not such a bad approximation. Pooling calls that would have very different processing time distributions would increase the variability and this could offset the benefits of the economies of scale. [van Dijk, 2004] and [van Dijk and van der Sluis, 2008] investigate the effect of pooling calls with different service time distributions. This is especially true true in call centers that receive calls of different languages and provide a same service to them. If differences in pattern between languages, we believe that this would not impact the average service time much and that this could be neglected.

The assumption that dedicated and cross-trained operators are equally efficient needs to be further commented as well. First it seems to neglect learning effecs in the staff: a dedicated agent is expected to acquire more experience by working full time on one skill than a polyvalent operator that shares its time between several skill. [Pinker and Shumsky, 2000] investigate the trade-off between flexibility brought by cross-trained operators and the loss in quality expected with such less specialized workers. Simulation tests in [Agnihothri et al., 2003] also show this tradeoff: the proportion of cross-trained decreases dramatically with a decrease of their efficiency. [Gans and Zhou, 2002] study the problem of staffing and capacity utilization when operators are not homogeneous because of learning and turnover effects. They model these effects with a Markov Decision Process and test different staffing policies, and identify optimal hiring policies. Here we suppose that it is possible to hire polyvalent agents equally efficient as dedicated agents by rewarding them adequately, i.e. by having a sufficiently large premium for extra skills. This premium is seen here as a reward for the additional skills but it can also be seen as a cost for training agents for new skills. Examples of training costs are provided in [Agnihothri et al., 2003]. When the skills are different languages, finding such equally efficient agents does not raise any difficulty. In the case of more technical skills, this assumption might be reconsidered.

The exponential distribution of service times is a simplifying assumption but it is a reasonable assumption as, in practice many call centers still use the Erlang-C formula, that requires making the same assumption. See [Gans et al., 2003] or [Koole et al., 2003] for discussions on this. Exponential service times is a common assumption in queueing theory in general and in call centers applications more particularly.

The arriving calls of one type form a flow of arrivals. We call it an incoming flow as the calls come from an external source. We assume that each incoming flow is a Poisson process. Within the system, the calls that could not find an available operator in one pool and that are routed to another pool form a flow as well. This type of flow is called an overflow as it consists in calls overfling from a pool. It is easy to see that overflows are not Poisson processes, as the time between two successive calls in an overflow is not always exponentially distributed. Example 2.4 shows with a sequence of events how distributions of interarrival times become different in the overflow than in the incoming flow.

Example 2.4 *Figure 2.4 illustrates a situation in which an overflow is created. A pool with three operators receives calls of a Poisson process. If no operator is available upon its arrival, a call is rejected.*

Layers represent the occupation of operators over time. Red areas indicate that an operator serves a call and the number of red layers gives the number of busy operators. A drop represents

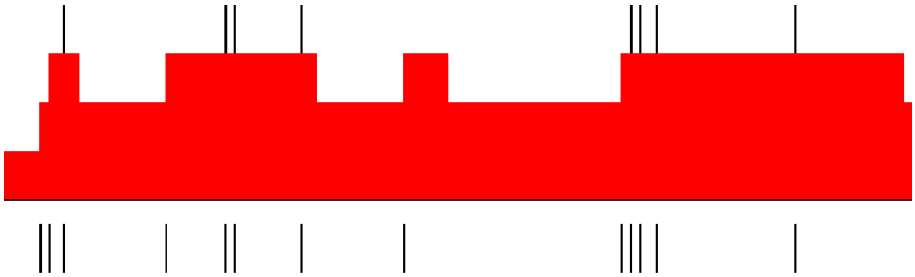

Figure 2.4: A situation in which an overflow is created. Incoming calls are represented by the black bars below the chart and the overflow is represented by the black bars above the chart.

an end of service and a jump, that only occurs at an arrival time, represents the start of the service of an arrivig call.

At the beginning of the time horizon, one of the operators is busy serving a call. The following sequence of events occur. 1&2. Two successive arrivals are immediately serviced by the two available operators. 3. A new call arrives. It is rejected and is therefore part of the overflow. 4. One operator finishes servicing a call. 5. An arrival occurs and the call is serviced by the available operator. 6. A new arrival occurs whereas no operator is available. It is rejected.

The times between the arrival in 3. and the arrival in 5. and between the arrival in 5. and the one in 6. are exponentially distributed. Arrivals in 3. and in 6. are two successive arrivals in the overflow. As their interarrival time is the sum of two exponentially distributed random variables, this time is distributed according to an Erlang distribution.

The absence of the exponential distribution of interarrival times in overflow processes is the major complication that arises from working with multi-skill systems as compared to single-skill systems. It is addressed in Chapter 5.

2.3 MODEL LIMITATIONS

Models cannot fully represent the complexity of reality. Many aspects of call center management and operations are not modeled here. Excluding abandonments, retrials or time-varying arrival rates are probably the most retricting assumptions of the models we use here. This is a compromise we had to make between completeness and tractability of the model and including them is amongst the main possible extensions to our results.

One should never forget that call center operators are not machines and the models presented in this thesis are inherently very limited in their representation of the behavior of operators. For example, although hierarchical routing and the principle

of keeping flexible operators available for future calls makes a lot of sense, in practice call center managers try to balance to some extent the workload between agents and may decide to dispatch calls to cross-trained operators during periods of lower activity to decrease the burden on the dedicated agents. More generally, operators would expect that the dispatching rules take into account some "fairness" considerations in sharing the workload. How to find a good trade-off between fairness for the operators and overall efficiency to provide the customers a good service is a broad question that will not be treated in this thesis.

As the following real-life example shows, there are aspects of the management of the staff of operators that are even more difficult to include in a queueing model. A company operated a call center below the level of performance it wished to reach: the service level was too long. The management hired additional staff to increase the capacity with the hope that more customers would receive an immediate answer. This had indeed a positive effect in the short run but after a while, the performance fell back to its previous level. An investigation by the management unveiled the unanticipated phenomenon at stake: hiring additional staff decreased the workload per agent. Upon realizing that, operators decided to exploit their increased idle time to improve the quality of the service to customers and they started taking more time to better answer calls. The resulting decrease in effective capacity cancelled the benefits brought by the additional operators, missing the objective of improving the service level, although it improved the quality of service.

2.4 METHODOLOGY

For each method or approximation developed in this thesis, a three-step procedure is used. We first describe the problem and show why it is important to find a solution to it. We then propose a solution and, finally, we validate the solution.

2.4.1 *Step 1. Problem description*

As said earlier on, our research relies on existing multi-skill queueing models. As these models form the framework of our research, most of the problems we address in this thesis are problems encountered within such models. They often correspond to real-life problems but they are primarily encountered in the model. There is one noticable exception: this is the problem addressed in Section 3.2. The problem of computing the peakedness from real-life data is basically a modelling problem: we want to translate real-life information into data for the models.

2.4.2 *Step 2. Resolution*

There is not a specific method to solve the problems we observe. Most of the time we adapt and use existing tools when they are available. They either come from other disciplines - this is the case when we use optimization methods in Chapter 6 - or they come from the single-skill queueing theory and one adapted to the multi-skill situations.

2.4.3 *Step 3. Validation*

Validation is the final step. It permits is to test the proposed method for solving the problem and evaluate its range of application. Because there is little theory related to multi-skill models, the favoured way of validating our results is by comparing the results obtained with the proposed method to simulation results.

2.4.3.1 *Validation by simulation*

Each of our main formulae is validated by comparing it with simulations results. The main reason is the lack of existing theory for multi-skill systems as compared to the classical queueing theory that applies to single-skill systems. A second reason is that validation through simulations permits in the mean time to identify the conditions under which the approximations work best.

Simulations consist in implementing a model using a simulation software[3]. The software is then run according to the assumptions and configuration and after a significantly long time horizon results are collected. Each simulation experiment consists in several simulation runs of a set of configurations of the model that form a dataset. The datasets are described in Section 2.5.

Depending on the model we work with we collect different kinds of performance measures from the simulations.

- In multi-skill loss systems the main performance measure is the loss probability. It is the probability for a call to be rejected by the system upon its arrival, because no skilled operator is available. In simulation experiments we estimate this probability by recording the number of rejected calls and divide it by the total number of arrivals.
- The main perfomance measure in multi-skill queueing systems is the waiting probability. This is the probability that a call has to wait before being served by

3 We developed our own simulation tools in Java. They are available upon request at jc.vandenschrieck@uclouvain.be

a skilled operator. In simulation, this probability is evaluated by recording the number of calls that had to wait and dividing it by the number of arrivals.

- The average waiting time is another performance measure of multi-skill queueing systems. This is the time that any call is expected to wait on average before being served. We compute this measure in simulation by recording the total number of arrivals and the cumulated waiting time and divide the latter by the former.

- Another common performance measure is the service level. The service level measures the proportion of callers that have to wait less than some time limit. This measure is important as regulations exist in some countries that impose minimum performance criteria in terms of service level and as many contracts in the call center industry use service level as the performance measure. See [Avramidis et al., 2009] or [Hasija et al., 2008] for some applications involving service level measurements. In the thesis when we work with service levels we fix the time limit as a function of the average service time. Typically we fix service level limits at 5%, 10%, 25%, 50% and 100% of the average service time. In simulation this performance measure is computed by recording all calls that have to wait less than the time limit (including those who do not wait at all) and divide the results by the total number of calls observed in the simulation run.

The simulation results then serve as benchmarks to which our approximations are compared. To evaluate the effectiveness of the approximation we typically compute the percentage error in absolute value between the simulation results and the results obtained with the approximation. When the investigated performance measure is a proportion or a probability, such as the loss probability, the waiting probability or a service level, the error is the difference between the simulation result and the approximation result. This difference is not scaled by dividing by the simulation result because the evaluated performance measure is a proportion itself and is comprised between 0 and 1. As an average waiting time might take any positive real value we divide the difference in absolute value by the simulation result and thus measure the relative error.

When we want to evaluate whether approximations present distortions compared to the simulation results we compute the difference between simulation results and approximations results but do not apply the absolute value transform. As a convention we fix that the approximation result is substracted from the simulation result: a positive value means that the approximation underestimates the actual performance and a negative value indicates an overestimation.

For the sake of fluidity in the text we are going to use acronyms for each type of evaluator.

- The mean absolute error in absolute value is denoted Mean AAbsErr.
- The mean relative error in absolute value is denoted Mean RAbsErr.
- The mean absolute error is denoted Mean AErr.
- The mean relative error is denoted Mean RErr.

We replace "Mean" by "Min" or "Max" in the acronym when we denote the minimum or maximum errors, respectively.

2.4.3.2 *Validation from existing theory*

As stated previously, a theory for multi-skill queueing system does not really exist to the same extent as the single-skill queueing theory. This is the reason why simulations are required for the validation. However some approximations presented in this thesis have been obtained by extending the theory of single-skill queueing systems to multi-skill situations. When it is the case we present the theoretical background in detail. This is a form of ex-ante validation but simulation experiments are performed anyway.

2.5 DESCRIPTION OF THE SIMULATION DATASETS

Here we briefly describe the datasets we use for our simulation experiments. They are displayed in Appendix A.

A first dataset is presented in Table A.1. It consists of small-size systems (between 5 and 20 operators) with two call types. There are six combinations of demands. They are built such as to vary the total load of the system as well as the imbalance between both arrival rates. Each combination of arrival streams is combined with 15 different pool size combinations to obtain a set of 90 examples. The dataset is built so that we can test more or less exhaustively all combinations with utilizations varying from 0.7 to 0.95. Without loss of generality, we set the service rate, μ, to 1 in all examples.

A second dataset is used to validate the approximations on configurations where the two flows of demands are larger than the ones in the dataset of Table A.1, i.e. to investigate the quality of the approximation when the size of the system increases. We adjust the number of operators accordingly to keep realistic performance results. It is presented in Table A.2. Experiments with this dataset are usually compared to the experiments made with the first dataset.

A third dataset consists in twelve configurations with three types of calls and seven different pools, like the one illustrated in Figure 2.2. This dataset is used for validating secondary or intermediary results.

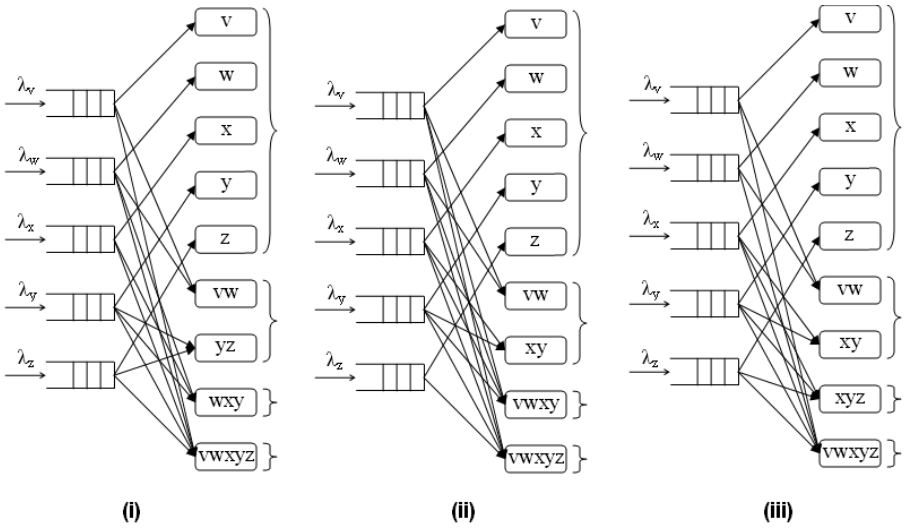

Figure 2.5: Structure of the systems with five types of calls. Pool 8 is different for each configuration. Braces group pools according to the number of skills.

The fourth and fifth datasets used for validation consist of systems with five types of calls. Our objective with these two datasets is to test how the complexity of the system affects the quality of our approximations. The configurations in the fourth dataset are made of nine pools that are ordered as in Figure 2.5: five pools are single skilled, two are double skilled, one pool has operators qualified to answer calls of either three of four different types and the ninth pool is a fully skilled pool. The related data is presented in Table A.4. Configurations 1 to 9 have low utilization: the waiting probability never exceeded twenty percent in simulation. In the remaining ones, the utilization is higher. The fifth dataset presents a few configurations with a higher number of pools (14) than in the dataset of Table A.4. Although not complete (With 5 types of calls it is possible to have up to 21 different pools), we believe that the number of pools in these configurations already exceeds what is considered in real-life call centers: it might not be easy to manage configurations with such complexity. Figure 2.6 gives the structure of the systems and Table A.5 presents the rates of the incoming flows and the staffing of the different pools.

For each configuration we conducted 15 different simulation runs. The simulation length comprised a warm-up period of 1000 time units and guaranteed at least 1 million arrivals for each type of call and for each simulation run. For each set of

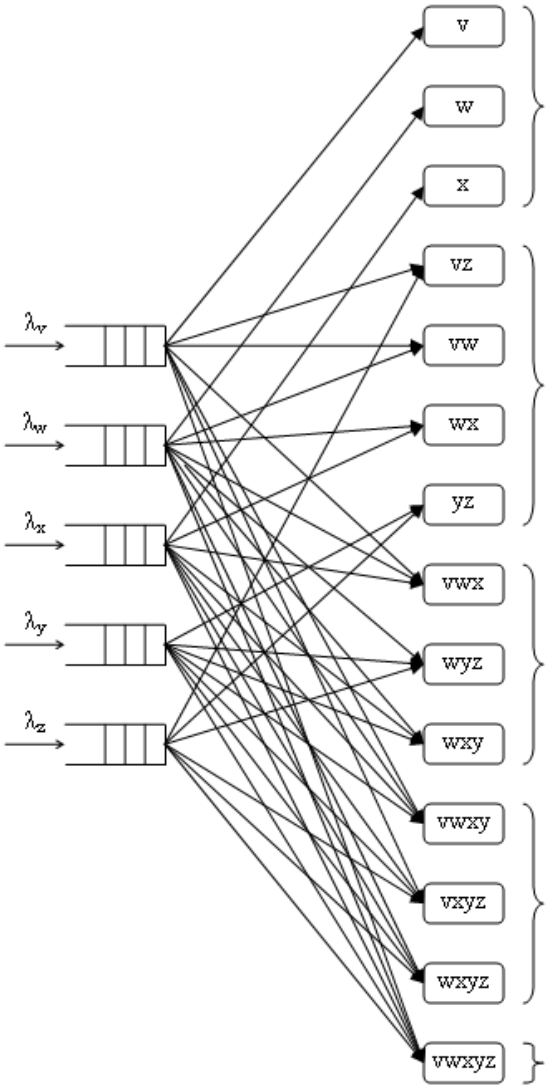

Figure 2.6: structure of the systems with 5 types of calls and 14 pools.

simulations, we computed confidence intervals (assuming a Student distribution of the observations) for the waiting probability and average waiting time. For the waiting probability, the confidence interval is never above $\pm$ 0.01. it is around $\pm$0.001 in most cases. The relative error for the average waiting time is less than 1% in all cases and for each measurement, lying in most cases around 0.1 or 0.2%. The results are therefore highly reliable.

Note that all the examples of the datasets are built such that there exists a way of allocating the capacity that would guarantee the stability of the whole system (i.e. any queue in the system is stable in the long run, see [Armony, 1999] for a discussion on the stability of multi-skill systems). In all cases, the system remains stable when simulated with the FCFS priority rule. Determining the conditions for stability when the priority policy is FCFS is beyond the scope of this thesis. Note however that [Armony, 1999] and [Armony and Bambos, 2003] provide interesting and probably useful elements for this task.

3

THE PEAKEDNESS FUNCTIONAL

In this chapter we focus on a measure of variability that has been initially used in telecommunications. It is now also used in queueing theory as a way of characterizing variability of flows when they are not Poisson distributed. This chapter explains how to compute the peakedness of a flow. In Section 3.1, we present a formal description of the peakedness functional and explain how to compute the peakedness of a flow when the distribution of the interarrival times is known. In Section 3.2, we show how to compute the peakedness from real-life data, i.e. from a collection of arrival times or cumulative arrivals over time intervals. After, that, we evaluate the impact of the service time distribution on the peakedness with a simulation study. Concluding remarks end the chapter.

3.1 DESCRIPTION OF THE PEAKEDNESS

The observation that arrival flows or overflows are often more variable than a Poisson process has been reported by many authors ([Wilkinson, 1956], [Whitt, 1995], [Jongbloed and Koole, 2000], [Tabordon, 2002], [Macq, 2005], or [Brown et al., 2005] for an analysis of arrivals to a call center). The variability of arrivals has often been attributed to non-constant arrival rates. The nature of the arrival process is another vector for a higher variability. To account for the variability of the interarrival times the most popular metric in queueing theory is the variance of the interarrival times. A related alternative is the coefficient of variation. This is the ratio of the standard deviation of the interarrival times over their mean value.

The peakedness is another way of measuring the variability. It is a measure that does not make any assumption on the interarrival times. It was first derived in the telecommunications literature to study overflows in telecommunication networks. It is in fact an indirect measure of variability as it measures the number of servers that would be busy if the arrival stream feeds an infinite server pool. More precisely, the

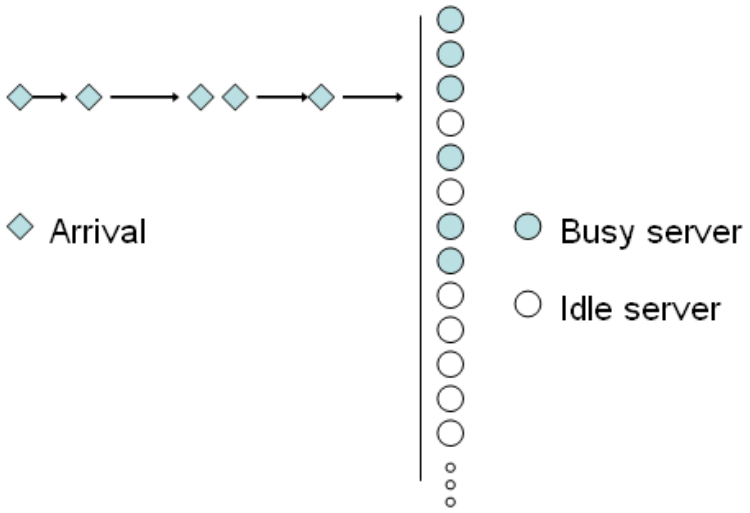

Figure 3.1: A schematic representation of the arrival stream and the number of busy servers.

peakedness is the ratio of the variance divided by the mean of the number of busy servers from this infinite pool. Figure 3.1 shows the fictitious infinite server pool that is used to evaluate the peakedness.

Let X be a point process with rate λ_X. Suppose that all arrivals of X go to a queue with an infinite number of servers. Their service time cumulative distribution function is denoted by $\mathsf{G}_\mu(t)$, where μ is the service rate. We observe $S_X(t)$, the number of busy servers, and define the *peakedness* of X with service time distribution function $\mathsf{G}_\mu(t)$, noted $z_X(\mathsf{G}_\mu)$, as

$$z_X(\mathsf{G}_\mu) = \frac{\mathrm{Var}[S_X(t)]}{E[S_X(t)]}. \tag{3.1}$$

Specifically, when $\mathsf{G}_\mu(t)$ is an exponential distribution with service rate μ, $0 < \mu < \infty$, we denote the peakedness by $z_X(\mathsf{M}_\mu)$; and $z_X(\mathsf{D}_\mu)$ when $\mathsf{G}_\mu(t)$ is deterministic with (constant) service time $1/\mu$. The peakedness measures the total variability of the order flow. To illustrate this, consider two deterministic arrival process X_1 and X_2. For X_1, one order arrives every day, and for X_2, 10 orders arrive at the first day, and no order arrivals for the next consecutive nine days. As a result, the arrival rates for both deterministic arrival processes are equal to 1. Suppose the service time is a constant such as one day. For X_1, the number of busy servers $S(t) = \{1, 1, ..., 1\}$

and the peakedness is obviously 0; while for X_2, $S(t) = \{10, 0, ..., 0\}$, the peakedness is larger than zero. Therefore, the peakedness can differentiate between these two deterministic arrival processes.

An extensive introduction to the notion of peakedness and the related formulae is presented in [Jagerman et al., 1996], in [Wolff, 1989] and in [Macq, 2005]. In particular, they explain how to compute analytically the peakedness from X and $\mathsf{G}_\mu(t)$. We now reproduce the main steps and formulae required for the analytical computation of $z_X(\mathsf{G}_\mu)$.

Let $N_X(t)$ denote the number of arrivals in the interval $(0, t]$. Let also $\tilde{N}_X(t, t + \Delta t) = N_X(t + \Delta t) - N_X(t)$ be the number of arrivals in $(t, t + \Delta t]$. Let us finally define k_X as the covariance density function of $X(t)$. The latter function satisfies:

$$\text{Cov}\left[\tilde{N}_X(t_1, t_1 + \Delta t_1), \tilde{N}_X(t_2, t_2 + \Delta t_2)\right] =$$

$$\int_{t_1}^{t_1+\Delta t_1} \int_{t_2}^{t_2+\Delta t_2} k_X(\tau_1 - \tau_2) d\tau_1 d\tau_2, \quad \forall t_1, \Delta t_1, t_2, \Delta t_2 \in \mathbb{R}. \tag{3.2}$$

We now define $\rho_{\mathsf{G}^c_\mu}(t)$ the autocorrelation function of $\mathsf{G}^c_\mu(t)$, where $\mathsf{G}^c_\mu = 1 - G_\mu(t)$ is the complementary distribution function of $\mathsf{G}_\mu(t)$, as

$$\rho_{\mathsf{G}^c_\mu}(t) = \int_{\max(0,-t)}^{+\infty} \mathsf{G}^c_\mu(\tau)\mathsf{G}^c_\mu(\tau + t) d\tau.$$

In particular, one can show that for an exponential distribution, $\rho_{\mathsf{G}^c_\mu}(t) = \frac{1}{2\mu} e^{-\mu|t|}$.

The following proposition presents the general formula which permits one to compute the peakedness from the covariance density function of the arrival process and the autocorrelation function of the complementary service time distribution.

Proposition 3.1 *Let X be a stationary point process. Its peakedness with service time distribution is given by:*

$$z_X(\mathsf{G}_\mu) \quad = \quad 1 + \frac{\mu}{\lambda_X} \int_{-\infty}^{\infty} (k_X(t) - \lambda_X \delta(t)) \rho_{\mathsf{G}^c_\mu}(t) dt, \tag{3.3}$$

where $\delta(t)$ is the Dirac delta function.

Proposition 3.1 is proven in [Eckberg, 1976] and in [Eckberg, 1983]. The proof is reproduced in Appendix B.

Property 1 *If $X(t)$ is a Poisson process, its peakedness is 1.*

Proof. [Macq, 2005] shows that if X is a Poisson process or rate λ_X, $k_X(t) = \lambda_X \delta(t)$. Introducing the latter in (3.3) completes the proof. ■

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
0.0416	0.0288	0.0717	0.1296	0.169	0.1945	0.1501	0.1054	0.0616	0.0212	0.0189	0.0048	0.0027

Table 3.1: Proportion of time each state is reached over the time horizon of the 400 arrivals of Table D.1.

This property is a direct consequence of the fact that the number of busy servers is geometrically distributed for an $M/G/\infty$ system whatever the service time distribution. This is handy, as this gives an easy benchmark for the variability of any process. Moreover this benchmark coincides with the more familiar coefficient of variation.

3.2 APPLICATION: DETERMINING THE PEAKEDNESS FROM A SET OF ARRIVALS

As a first illustration of the notion of peakedness, we explain how to estimate it for any sequence of observed arrival times. In particular, we focus on the case of exponentially distributed service times for the fictitious operators. As for the remainder of this section we work with exponentially distributed service times with rate μ most of the time, we omit G_μ in our notation when the service time is exponential.

3.2.1 *Computing the peakedness for a sample of arrival times*

Recall that the peakedness z is the ratio of the variance to the mean of the number of busy servers if the arrivals were handled by infinitely many servers. One immediate way of measuring the peakedness is to simulate this infinite pool of servers, using the historical arrival times and simulated random service times.

Example 3.1 *We illustrate this approach with an example. Table D.1 in Appendix D gives the arrival times of 400 arrivals of a Poisson process of rate $\lambda = 10/hour$. In Table D.2 we show the service times for these arrivals in the fictitious pool with an infinite number of operators. Service times are exponentially distributed with parameter $\mu = 2/hour$. Figure 3.2 shows $S(t)$ for the first 15 time units of the time horizon. Note that we fixed $S(0) = 5$.*

There are different approaches to compute $Var[S(t)]$ and $E[S(t)]$. A first method is to pick sample values of $S(t)$ at random times. A second approach, which is the one used here, consists in computing for all $i \in \mathbb{N}$ the proportion of time when $S(t) = i$. We then compute $E[S(t)] = \sum_{i=0}^{\infty} i\, p_i$ and $Var[S(t)] = E[S(t)^2] - E[S(t)]^2$. In the example we observe that the maximum value $S(t)$ takes over the time horizon of 41.39 [1] is equal to 12. Table 3.1 presents the values of p_i for $i \in \{0, 1, .., 12\}$.

1 41.39 is the time of the 400-th arrival.

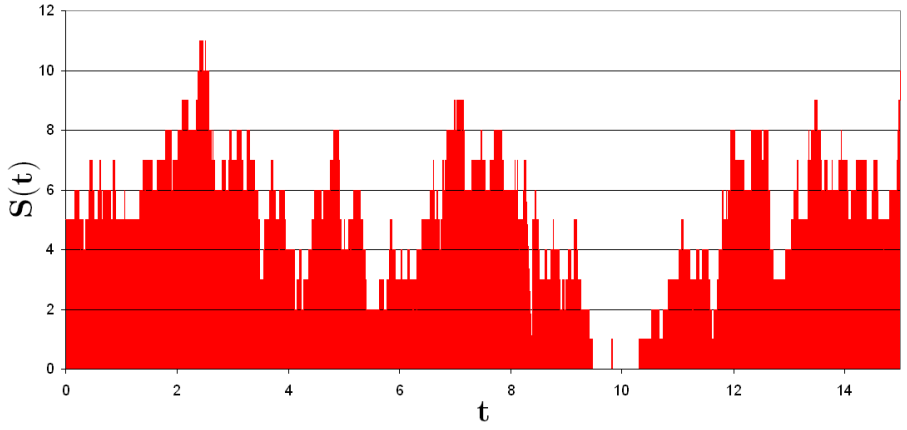

Figure 3.2: S(t) as observed with the 400 arrivals and service times of Tables D.1 and D.2 over the first 15 time units of the time horizon.

We find that $E[S(t)] = 4.81$, $E[S(t)^2] = 28.14$ *and* $Var[S(t)] = 5.04$. z *is thus equal to 1.05, which is not so far from the theoretical vale of* $z = 1$ *for a Poisson process.*

3.2.2 *Computing the fluid peakedness*

The major drawback of the first method is that the randomness of the service times will add some noise to the estimator of the peakedness.

In order to circumvent this difficulty, [Macq, 2005] defines a fluid version of the peakedness. In this case an arrival is virtually divided into infinitesimal requests with processing times that follow the service time distribution. This means that an arrival at time τ will still be "using" $G^c_\mu(t-\tau) = 1 - G_\mu(t-\tau)$ servers at time t (for $t \geq \tau$). Thus the number of busy servers does not include any variability that is due to the service process. We define the fluid peakedness $\bar{z}$ as the ratio of the variance to the mean of the number of busy servers when service is "fluid". [Macq, 2005] shows that there is a direct relationship between the fluid peakedness $\bar{z}$ and the peakedness z. This relationship is given in the following proposition

Proposition 3.2

$$\bar{z}_X(G_\mu) = z_X(G_\mu) - 1 + \mu\rho_{G^c_\mu}(0).$$

The proof of this proposition presented in [Macq, 2005] is reproduced in Appendix C. When the service time distribution is exponential this relationship simplifies to:

$$\bar{z} = z - \frac{1}{2} \tag{3.4}$$

Computing the number of busy fluid servers, $\bar{S}(t)$, is particularly easy in this case. Between two arrivals at times t_1 and t_2 the number of busy servers will decrease exponentially such that at time t_2^-, i.e. just before the arrival in t_2, $\bar{S}(t_2^-) = \bar{S}(t_1)e^{-\mu(t_2^- - t_1)}$. $\bar{S}(t)$ is increased by one in t_2 and at every arrival epoch. Given the observed arrival moments, it is thus easy to deduce the number of busy fluid servers at any time. We can then compute the time average and variance in order to estimate the fluid peakedness. Finally we add $1/2$ to obtain the peakedness.

Example 3.2 *We continue our illustrative example and estimate the fluid peakedness for the sample of arrivals presented in Table D.1 of Appendix D. This time, Table D.2 of the services times is not required.* $\bar{S}(t)$ *suffers an exponential decay of parameter* $\mu = 2$*. Figure 3.3 shows the evolution of* $\bar{S}(t)$ *over the first 15 time units of the time horizon. We start out by computing the number of busy operators for each arrival time. Denote by* t_i *the arrival time of the* i^{th} *arrival,* $\bar{S}(t_i)$ *is computed using the property that* $\bar{S}(t_i) = \bar{S}(t_{i-1})e^{-\mu(t_i - t_{i-1})} + 1$*. To limit the bias at the beginning of the time horizon we set* $\bar{S}(0) = \lambda/\mu = 5$*. We randomly choose 50 time points on the time horizon and for each of these 50 epochs* t^* *we compute the value of* $\bar{S}(t^*)$*. We thus have 50 measurements of* $\bar{S}(t)$ *at random times and we compute the mean and variance of this sample. We obtain* $E[\bar{S}(t)] = 4.98$ *and* $Var[\bar{S}(t)] = 2.52$*, resulting in* $\bar{z} = 0.51$*. Adding 1/2, according to (3.4), yields a value of 1.01. This is close to the theroretical value of 1 for the peakedness.*

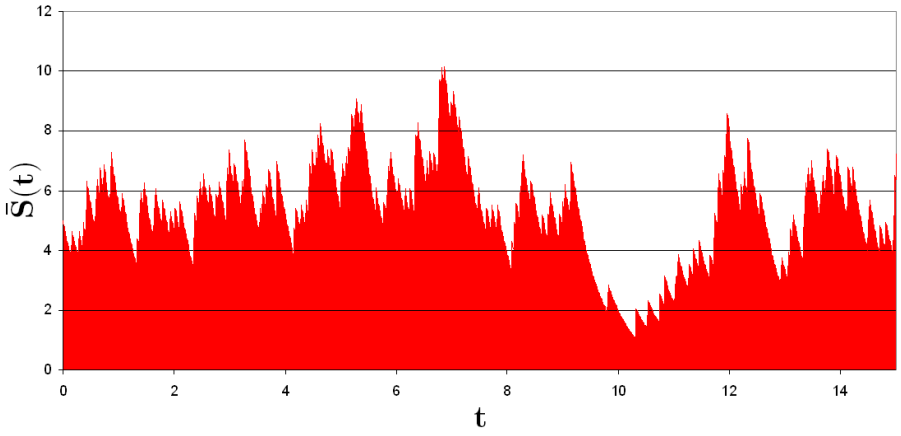

Figure 3.3: $\bar{S}(t)$ as observed with the 400 arrivals of Table D.1 for the first 15 hours of the considered time horizon.

3.2.3 *Computing the fluid peakedness for aggregated data*

In practice most call centers keep reports of past arrivals as aggregated demand over time intervals. For example reporting software may collect the number of calls over 15-minute time intervals. Although some information is lost when aggregating the data, it is possible to compute an accurate estimate of the peakedness based on $\bar{z}_b$, the fluid peakedness obtained from the aggregated data. For a Poisson process, the exact value of $\bar{z}$ can be derived from $\bar{z}_b$. Proposition 3.3 gives the formula. It is presented after the following two lemmas. For other types of arrival processes, an approximation is required. It is described after Proposition 3.3.

Lemma 3.1 *Let D be a deterministic process with interarrival time $T = 1/\lambda_D$, then its peakedness is given by*

$$z_D = 1 - \frac{1}{\mu T} + \frac{e^{-\mu T}}{1 - e^{-\mu T}} \tag{3.5}$$

$$\bar{z}_D = \frac{1}{2} - \frac{1}{\mu T} + \frac{e^{-\mu T}}{1 - e^{-\mu T}}. \tag{3.6}$$

All elements for proving this proposition can be found in [Macq, 2005]. We reproduce them here to introduce and illustrate the method we use to demonstrate Lemma 3.2.

Proof. Take any two epochs t_1 and t_2 and consider the two infinitesimal intervals $[t_1, t_1 + dt)$ and $[t_2, t_2 + dt)$. We immediately see that

$$\tilde{N}_D(t_i, t_i + dt) = \begin{cases} 1 & \text{with } p = \frac{dt}{T} \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1, 2$. $\tilde{N}_D(t_i, t_i + dt)$ is a bernouilli variable, so $E[\tilde{N}_D(t_i, t_i + dt)] = \frac{dt}{T}$. Two situations are possible: either $t_2 - t_1 = \Delta t$ is a multiple of T or it is not.

- In the former case $\tilde{N}_D(t_1, t_1 + dt) = \tilde{N}_1$ and $\tilde{N}_D(t_2, t_2 + dt) = \tilde{N}_2$ take identical values. Therefore we have

$$\begin{aligned} E[\tilde{N}_1 \tilde{N}_2 | \Delta t = nT\,,\, n \in \mathbb{N}] &= \sum_{i=0}^{1} i^2 P[\tilde{N}_1 = i, \tilde{N}_2 = i | \Delta t = nT\,,\, n \in \mathbb{N}] \\ &= P[\tilde{N}_1 = 1, \tilde{N}_2 = 1 | \Delta t = nT\,,\, n \in \mathbb{N}] \\ &= P[\tilde{N}_1 = 1] = \frac{dt}{T}. \end{aligned}$$

Remembering that $\mathrm{Cov}[\tilde{N}_1, \tilde{N}_2] = E[\tilde{N}_1\,\tilde{N}_2] - E[\tilde{N}_1]E[\tilde{N}_2]$, we find

$$\mathrm{Cov}[\tilde{N}_1, \tilde{N}_2 | \Delta t = nT\,,\, n \in \mathbb{N}] \quad = \quad \frac{dt}{T} - \left(\frac{dt}{T}\right)^2.$$

- In the latter case, $\tilde{N}_1$ and $\tilde{N}_2$ cannot be equal to 1 at the same time. Therefore

$$\mathrm{Cov}[\tilde{N}_1, \tilde{N}_2 | \Delta t \neq nT\,,\, n \in \mathbb{N}] = -\left(\frac{dt}{T}\right)^2.$$

Using (3.2), it is easy to verify that the covariance density function of $X(t)$ is given by

$$k_X(t) = -\frac{1}{T^2} + \sum_{n=-\infty}^{\infty} \frac{\delta(t-nT)}{T}.$$

Since we know that if G_μ is an exponential distribution function with parameter μ, $\rho_{G^c_\mu}(t) = \frac{1}{2\mu} e^{-\mu|t|}$, we find with (3.3) that

$$\begin{aligned} z_D &= 1 + \mu T \int_{-\infty}^{\infty} \left(-\frac{1}{T^2} + \sum_{n=-\infty}^{\infty} \frac{\delta(t-nT)}{T} - \frac{\delta(t)}{T} \right) \frac{1}{2\mu} e^{-\mu|t|} dt \\ &= 1 - \frac{1}{\mu T} \int_0^{\infty} \mu e^{-\mu t} dt + \sum_{n=1}^{\infty} e^{-\mu n T} \\ &= 1 - \frac{1}{\mu T} + \frac{e^{-\mu T}}{1 - e^{-\mu T}}. \end{aligned}$$

This proves (3.5). We immediately deduce (3.6) from (3.4). ■

Lemma 3.2 *Let D be a deterministic point process with interarrival time $T = 1/\lambda_D$. We construct a point process Y in such a way that at each arrival in D a batch of arrivals occurs in Y. The batch sizes are assumed independent and identically distributed and have the same distribution as a given random variable B. The rate and peakedness of Y are given by*

$$E[Y] = E[B]/T \tag{3.7}$$

$$z_Y = E[B] z_D - \frac{E[B]}{2} + \frac{1}{2} + \frac{1}{2} \frac{Var[B]}{E[B]} \tag{3.8}$$

$$\bar{z}_Y = E[B] \bar{z}_D + \frac{1}{2} \frac{Var[B]}{E[B]}. \tag{3.9}$$

Proof. The proof is similar to the proof of Lemma 3.1. Take any two epochs t_1 and t_2 and consider the two infinitesimal intervals $[t_1, t_1 + dt)$ and $[t_2, t_2 + dt)$. The distribution of $\tilde{N}_Y(t_i, t_i + dt)$ is:

$$\tilde{N}_Y(t_i, t_i + dt) = \begin{cases} B_k & \text{with } p = \frac{dt}{T} \\ 0 & \text{otherwise,} \end{cases}$$

where B_k is the random variable associated to the k^{th} batch, with $k \in \mathbb{N}$. As the batch size is independent of the batch arrival time, we immediately see that $E[\tilde{N}_Y] = E[B]\frac{dt}{T}$. Therefore, $E[Y] = E[B]/T$.

Three situations might occur for t_1 and t_2. First, $t_2 - t_1 = \Delta t$ can be a non-zero multiple of T. Second, Δt might not be a multiple of T and, third, we could have $t_2 = t_1$.

1. In the first case if there is a batch arrival in $\tilde{N}_i(t_1, t_1 + dt) = \tilde{N}_{i,1}$, $i \in \{D, Y\}$ then there is a batch arrival in $\tilde{N}_i(t_2, t_2 + dt) = \tilde{N}_{i,2}$, $i \in \{D, Y\}$ and inversely. This means that

$$\begin{aligned} &E[\tilde{N}_{Y,1}\tilde{N}_{Y,2}|\Delta t = nT\,,\, n \in \mathbb{N}_0] \\ &\quad= \sum_{i_1=0}^{\infty}\sum_{i_2=0}^{\infty} i_1\, i_2 P[\tilde{N}_{Y,1} = i_1, \tilde{N}_{Y,2} = i_2|\Delta t = nT\,,\, n \in \mathbb{N}_0] \\ &\quad= \sum_{i_1=0}^{\infty}\sum_{i_2=0}^{\infty} i_1\, i_2 P[\tilde{N}_{Y,1} = i_1, \tilde{N}_{Y,2} = i_2|\Delta t = nT\,,\, n \in \mathbb{N}_0, \tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1] \\ &\qquad\qquad \cdot P[\tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1|\Delta t = nT\,,\, n \in \mathbb{N}_0] \\ &\quad= \sum_{i_1=0}^{\infty}\sum_{i_2=0}^{\infty} i_1\, i_2 P[\tilde{N}_{Y,1} = i_1|\Delta t = nT\,,\, n \in \mathbb{N}_0, \tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1] \\ &\qquad\qquad \cdot P[\tilde{N}_{Y,2} = i_2|\Delta t = nT\,,\, n \in \mathbb{N}_0, \tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1]\frac{dt}{T} \\ &\quad= E[B]^2\frac{dt}{T}, \end{aligned}$$

   and

$$\mathrm{Cov}[\tilde{N}_1, \tilde{N}_2|\Delta t = nT\,\forall n \in \mathbb{N}_0] \quad = \quad E[B]^2\frac{dt}{T} - \left(E[B]\frac{dt}{T}\right)^2.$$

2. In the second case, $\tilde{N}_Y(t_1, t_1 + dt)$ and $\tilde{N}_Y(t_2, t_2 + dt)$ cannot be both strictly positive. Therefore, $E[\tilde{N}_{Y,1}\tilde{N}_{Y,2}|t_2 - t_1 \neq nT\,,\, n \in \mathbb{N}] = 0$ and

$$\mathrm{Cov}[\tilde{N}_{Y,1}\tilde{N}_{Y,2}|t_2 - t_1 \neq nT\,,\, n \in \mathbb{N}] = -\left(E[B]\frac{dt}{T}\right)^2.$$

3. The third case, with $t_1 = t_2$, proceeds as follows:

$$\begin{aligned} E[\tilde{N}_{Y,1}, \tilde{N}_{Y,2} | t_2 = t_1] &= \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} i_1 \, i_2 P[\tilde{N}_{Y,1} = i_1, \tilde{N}_{Y,2} = i_2 | t_2 = t_1] \\ &= \sum_{i=0}^{\infty} i^2 P[\tilde{N}_{Y,1} = i, \tilde{N}_{Y,2} = i | t_2 = t_1, \tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1] \\ &\qquad \cdot P[\tilde{N}_{D,1} = 1, \tilde{N}_{D,2} = 1 | t_2 = t_1] \\ &= \sum_{i=0}^{\infty} (i)^2 P[\tilde{N}_{Y,1} = i | \tilde{N}_{D,1} = 1] \frac{dt}{T} \\ &= E[B^2] \frac{dt}{T}, \end{aligned}$$

and

$$\mathrm{Cov}[\tilde{N}_{Y,1}, \tilde{N}_{Y,2} | t_2 = t_1] = E[B^2] \frac{dt}{T} - \left(E[B] \frac{dt}{T} \right)^2.$$

In summary the covariance density function of $Y(t)$ is given by

$$\begin{aligned} k_Y(t) &= -\frac{E[B]^2}{T^2} + E[B]^2 \sum_{n=-\infty}^{\infty} \frac{\delta(t - nT)}{T} - E[B]^2 \frac{\delta(t)}{T} + E[B^2] \frac{\delta(t)}{T} \\ &= -\frac{E[B]^2}{T^2} + E[B]^2 \sum_{n=-\infty}^{\infty} \frac{\delta(t - nT)}{T} + \mathrm{Var}[B] \frac{\delta(t)}{T}. \end{aligned}$$

Recalling that $\rho_{G_\mu^c}(t) = \frac{1}{2\mu} e^{-\mu|t|}$ and using (3.3) with $\lambda_Y = \frac{E[B]}{T}$, we find that

$$\begin{aligned} z_Y &= 1 + \frac{\mu T}{E[B]} \int_{-\infty}^{\infty} \left(-\frac{E[B]^2}{T^2} + E[B]^2 \sum_{n=-\infty}^{\infty} \frac{\delta(t - nT)}{T} + \mathrm{Var}[B] \frac{\delta(t)}{T} - E[B] \frac{\delta(t)}{T} \right) \frac{1}{2\mu} e^{-\mu|t|} dt \\ &= 1 - \frac{E[B]}{\mu T} \int_0^{\infty} \mu e^{-\mu t} dt + E[B] \sum_{n=1}^{\infty} e^{-\mu n T} + \frac{E[B]}{2} + \frac{1}{2} \frac{\mathrm{Var}[B]}{E[B]} - \frac{1}{2} \\ &= E[B] - \frac{E[B]}{\mu T} + E[B] \frac{e^{-\mu T}}{1 - e^{-\mu T}} + \frac{1}{2} - \frac{E[B]}{2} + \frac{1}{2} \frac{\mathrm{Var}[B]}{E[B]} \\ &= E[B] z_D + \frac{1}{2} - \frac{E[B]}{2} + \frac{1}{2} \frac{\mathrm{Var}[B]}{E[B]}. \end{aligned}$$

This gives (3.8). We then obtain (3.9) using (3.4) in (3.8). ■

The property we use to build the approximation is a direct consequence of Lemma 3.2.

Proposition 3.3 *Let D be a deterministic point process with interarrival time T. Let X be a Poisson process with arrival rate λ_X. We construct a batch process Y in such a way that there is a new batch arrival every time an arrival occurs in D. The batch size of the n^{th} arrival is equal*

to the cumulated number of arrivals in X between $[(n-1)T; nT)$*, i.e.* $\tilde{N}_X((n-1)T, nT)$*. The fluid peakedness of Y is given by*

$$\bar{z}_Y \quad = \quad \lambda_X T \bar{z}_D + \bar{z}_X. \tag{3.10}$$

Proof. Note first that $\tilde{N}_X((n-1)T, nT)$ and $\tilde{N}_X((m-1)T, mT)$ never overlap for $m \neq n \in \mathbb{N}_0$ and are therefore independent. The conditions are fulfilled for equation (3.9) to be valid. We have $E[B] = E[\tilde{N}_X((n-1)T, nT)] = \lambda_X T$ and $\mathrm{Var}[B] = E[B] = \lambda_X T$ so $\frac{\mathrm{Var}[B]}{E[B]} = 1$. From (3.4) and because $z_X = 1$, we know that $\bar{z}_X = \frac{1}{2}$. Proposition 3.3 is thus proven. ■

Equation (3.10) holds only for Poisson Arrivals. We propose to use it for other arrival processes as well, as an approximation. So we assume:

$$\bar{z}_b \simeq \lambda T \bar{z}_D + \bar{z}, \tag{3.11}$$

where $\bar{z}_b$ is the peakedness of the batch and $\bar{z}_D$ is the peakedness of the deterministic point process D, with intearrival time T. An interpretation for this approximation is that a portion of the variability of the batch process comes from the initial arrival process and another part is due to the aggregation of the arrivals in batches, that depends on the frequency of the batch arrivals. In Appendix E we evaluate this approximation with simulations of different types of arrival processes.

We can have a good estimate of $\bar{z}$ once we know $\bar{z}_b$. $\bar{z}_b$ is easier to compute than $\bar{z}$ because it does not require any sampling of the number of busy operators. In the process that gives the number of fluid servers related to the batch arrival process, $\bar{S}_{b,T}(t)$, we can exploit the memoryless property of the exponential distribution to compute the fluid peakedness easily. We actually only need to know the values of $\bar{S}_{b,T}(nT)$, the number of busy operators at the end/beginning of each time interval, to determine the value of $\bar{S}_{b,T}(t)$ for any t. Proposition 3.4 provides the relationship between the ratio of the variance and the mean value obtained from the values of $\bar{S}_{b,T}(nT)$ only and the fluid peakedness of the batch arrival process.

Proposition 3.4 *Take the process Y defined in Lemma 3.2. Let* $\bar{z}_{Y,nT}$ *be the ratio of the variance over the mean value of all values of* $\bar{S}_Y(nT)$*, the number of busy operators at times* nT*,* $\forall n \in \mathbb{N}_0$*, i.e. at each batch arrival. Then*

$$\bar{z}_Y \quad = \quad \bar{z}_{Y,nT} \frac{1 + e^{-\mu T}}{2} + \frac{E[B]}{2} \frac{1 + e^{-\mu T}}{1 - e^{-\mu T}} - \frac{E[B]}{\mu T}, \tag{3.12}$$

and, if Y is defined as in Proposition 3.3,

$$\bar{z}_Y = \bar{z}_{Y,nT}\frac{1+e^{-\mu T}}{2} + \frac{\lambda T}{2}\frac{1+e^{-\mu T}}{1-e^{-\mu T}} - \frac{\lambda}{\mu}. \tag{3.13}$$

Proof. Let $E[\bar{S}_Y(nT)]$ be the expected value of $\bar{S}_Y(nT)$, $\forall n \in \mathbb{N}$. As $\bar{S}_Y(nT) = \bar{S}_Y((n-1)T)e^{-\mu T} + \tilde{N}((n-1)T, nT)$, we have

$$E[\bar{S}_Y(nT)] = \frac{E[B]}{1-e^{-\mu T}}. \tag{3.14}$$

We can write $\bar{S}_Y(t)$ as a function of $\bar{S}_Y(nT)$, where $nT \leq t < (n+1)T$ in the following way:

$$\bar{S}_Y(t) = \bar{S}_Y(nT)e^{-\mu\tau},$$

with $\tau = t - nT$. In fact $\bar{S}_Y(t)$ suffers an exponential decay and is "fed" again every T time units. We immediately find that:

$$E[\bar{S}_Y(nT+\tau)] = e^{-\mu\tau}E[\bar{S}_Y(nT)]$$
$$E[\bar{S}_Y^2(nT+\tau)] = e^{-2\mu\tau}E[\bar{S}_Y^2(nT)].$$

Note that the last two expressions are restricted to the time points situated τ time units after nT, $n \in \mathbb{N}_0$. The first two moments of $\bar{S}_Y(t)$ computed on all values of t are given by:

$$\begin{aligned} E[\bar{S}_Y(t)] &= \frac{1}{T}\int_0^T E[\bar{S}_Y(nT)]e^{-\mu\tau}d\tau \\ &= \frac{1-e^{-\mu T}}{\mu T}E[\bar{S}_Y(nT)] \\ E[\bar{S}_Y^2(t)] &= \frac{1}{T}\int_0^T E[\bar{S}_Y^2(nT)]e^{-2\mu\tau}d\tau \\ &= \frac{1-e^{-2\mu T}}{2\mu T}E[\bar{S}_Y^2(nT)]. \end{aligned}$$

Therefore we have

$$\begin{aligned} \mathrm{Var}[\bar{S}_Y(t)] &= \frac{1-e^{-2\mu T}}{2\mu T}E[\bar{S}_Y^2(nT)] - \frac{(1-e^{-\mu T})^2}{(\mu T)^2}E^2[\bar{S}_Y(nT)] \\ &= \frac{1-e^{-2\mu T}}{2\mu T}(\mathrm{Var}[\bar{S}_Y(nT)] + E^2[\bar{S}_Y(nT)]) - \frac{(1-e^{-\mu T})^2}{(\mu T)^2}E^2[\bar{S}_Y(nT)]. \end{aligned}$$

Dividing this last result by $E[\bar{S}_Y(t)]$ yields

$$\begin{aligned}\bar{z}_Y &= \frac{\frac{1-e^{-2\mu T}}{2\mu T}\mathrm{Var}[\bar{S}_Y(t)]}{\frac{1-e^{-\mu T}}{\mu T}E[\bar{S}_Y(nT)]} + \frac{\frac{1-e^{-2\mu T}}{2\mu T}E^2[\bar{S}_Y(nT)]}{\frac{1-e^{-\mu T}}{\mu T}E[\bar{S}_Y(nT)]} - \frac{\frac{(1-e^{-\mu T})^2}{(\mu T)^2}E^2[\bar{S}_Y(nT)]}{\frac{1-e^{-\mu T}}{\mu T}E[\bar{S}_Y(nT)]} \\ &= \bar{z}_{Y,nT}\frac{1+e^{-\mu T}}{2} + \frac{1+e^{-\mu T}}{2}E[\bar{S}_Y(nT)] - \frac{1-e^{-\mu T}}{\mu T}E[\bar{S}_Y(nT)].\end{aligned}$$

Replacing $E[\bar{S}_Y(nT)]$ with (3.14) and simplifying gives Equation 3.12 . As E[B] is equal to λT, (3.13) follows immediately. ■

Introducing Equation (3.12) into (3.9) and replacing $\bar{z}_D$ with (3.6) gives the following remarkably simple formula for computing $\bar{z}_{Y,nT}$:

$$\bar{z}_{Y,nT} = \frac{\mathrm{Var}[B]}{E[B](1+e^{-\mu T})}. \tag{3.15}$$

Similarly, putting (3.13) in (3.11) with $\bar{z}_D$ replaced by (3.6) yields the following simple approximation to estimate the peakedness of the unaggregated arrival process from the aggregated data:

$$\bar{z} \simeq \bar{z}_{b,nT}\left(\frac{1+e^{-\mu T}}{2}\right). \tag{3.16}$$

This relation is exact when the arrivals are uniformly distributed within aggregation periods.

Example 3.3 *Going back to the illustrative example we aggregate the data in time intervals of $T = 0.25$, corresponding to 15 minute time intervals. The numbers of arrivals in each interval, $\tilde{N}((n-1)T, nT)$ are displayed in Table D.3. Note that the last interval considered in this example goes from 41 to 41.25. The two arrivals after 41.25 are neglected. We define $barS_{b,T}(t)$ as the number of busy operators at time t associated to the batch process with period T.*

For each interval n we compute

$$\bar{S}_{b,T}(nT) = \bar{S}_{b,T}((n-1)T)e^{-\mu T} + \tilde{N}((n-1)T, nT) \quad \forall n > 0.$$

To decrease the bias brought by warm-up effects we assume $\bar{S}(0) = \frac{\lambda T}{\mu}\frac{e^{-\mu T}}{1-e^{-\mu T}} = 3.85$. Figure 3.4 depicts $\bar{S}_{b,nT}(t)$, $0 \le t \le 15$.

From the computed values of $\bar{S}_{b,T}(nT)$ we find $E[\bar{S}_{b,T}(nT)] = 6.09$ and $Var[\bar{S}_{b,T}(nT)] = 3.79$. $\bar{z}_{b,nT}$ is equal to 0.62. Using (3.16), we find $\bar{z} = 0.5$.

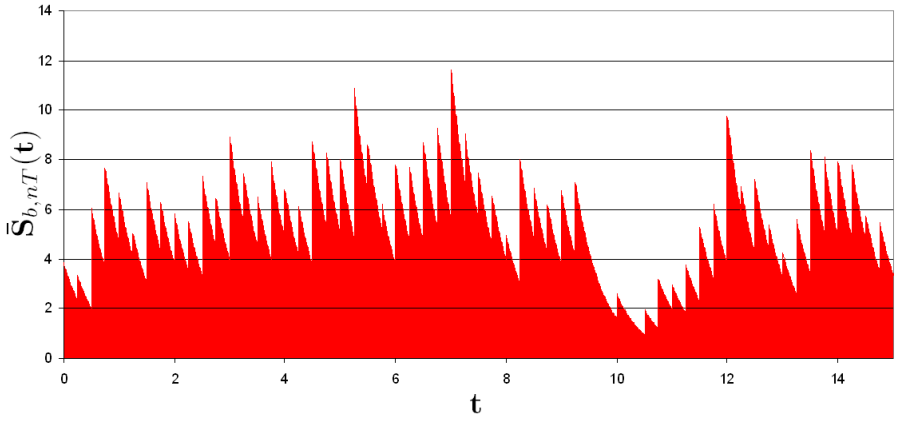

Figure 3.4: $\bar{S}_{b,nT}(t)$ as observed with the 400 arrivals of Table D.1 for the first 15 time units of the considered time horizon.

This procedure is designed for exponential service time distributions. In section 3.3 we evaluate the impact of changing the service time distribution on the value of the peakedness.

3.3 EVALUATING THE IMPACT OF THE SERVICE TIME DISTRIBUTION ON THE PEAKEDNESS

For a given arrival process, we can compute the peakedness using any service time distribution for the servers. Usually, in the context of call centers the chosen service time distribution and its parameters are the ones that are observed in practice when serving the arrivals (this is for instance the case for Hayward's approximation, which is explained in Section 4.1). Most of the formulae and methods presented in Section 3.2 (with the noticable exception of the part presented in Section 3.2.1) only apply to situations with exponential service time distributions. Other popular service time distributions do not have the memoryless property that is key to the particularly easy computation of $\bar{z}$ detailed in Sections 3.2.2 and 3.2.3.

This section aims at showing that the service time distribution of the fictitious operators and its variability do not affect the value of the peakedness much. We present simulation experiments in which the peakedness of a flow is computed with different service time distributions and variabilities. There are many types of processes for which we can compute the peakedness. Here, we only consider some of them, that we think are more relevant for this thesis.

We know that the peakedness of a homogeneous Poisson process is always equal to one but this is not the case for other more general Poisson processes. In Section 3.3.1 we compute the peakedness of two non-homogeneous stationary Poisson processes.

In Section 3.3.2, we focus on the peakedness of different kinds of overflow processes. For each flow we compute the peakedness with lognormally distributed service times of variance equal to 0.25, 1 and 4 and we compare them to the peakedness with exponentially distributed service times. All service times have an expected value of one. We restrict our simulation study to lognormal service time distributions as this distribution is widely used in queueing models and fits well to service times observed in practice ([Brown et al., 2005], [Bolotin, 1994]). [Ulrich and Miller, 1993] and [Breukelen, 1995] present mechanisms that could generate lognormally distributed service times.

In each experiment, we generate arrivals of some process and associate each of them to an operator of a fictitious pool. The operators serves the call according to the service time distribution. The peakedness is computed using the method presented in Section 3.2.1. The results displayed in the Tables are the simulated peakedness in each experiment and the mean relative difference in absolute value between the lognormal peakedness and the exponential peakedness.

3.3.1 *Non-homogeneous stationary Poisson processes*

Table 3.2 displays the peakedness observed in simulations when the flow is a Poisson process with sinusoidal arrival rate equal to

$$\lambda(t) = 1 + \lambda(1 + \sin(\omega t)),$$

where λ ranges from 4 to 100 and ω is the angular frequency of the sinusoid. We choose ω such that the period of the sinusoid is equal to $1/10$, 1 and 10 time units, i.e. $\omega = 20\Pi$, 2Π or $\Pi/5$, respectively.

In Table 3.3 we present the peakedness of non-homogeneous Poisson processes with arrival rates that change from one period to another. The length of one period is equal to one time unit. The general formula for $\lambda[n]$, the arrival rate in the n^{th} time interval is

$$\lambda[n] = \alpha\lambda[n-1] + (1-\alpha)X,$$

where n is the n^{th} period of the time horizon, α is a correlation factor and X is a random variable uniformly distributed over $[0, 2\lambda]$. To keep stationarity over time, we impose $\lambda[n] \in [0, 2\lambda]$.

ω	λ	Exp	LN(0.25)	LN (1)	LN(4)
$\omega = \Pi/5$	4	1.5656	1.8122	1.6166	1.3977
	8	2.1581	2.6608	2.2423	1.8248
	12	2.7139	3.459	2.8367	2.2341
	16	3.2509	4.2086	3.4447	2.6142
	20	3.7618	4.9233	4.0029	2.9711
	40	6.3087	8.4812	6.844	4.806
	60	8.7784	11.8184	9.2841	6.5646
	80	11.0266	14.8998	11.7845	8.3289
	100	13.0595	18.1882	14.6118	9.7933
Relative Errors			0.3003	0.0634	0.2033
$\omega = 2\Pi$	4	2.2230	2.5510	2.2015	1.7224
	8	3.7342	4.4534	3.69	2.6219
	12	5.2316	6.3777	5.1961	3.5299
	16	6.7676	8.3268	6.7013	4.4078
	20	8.2736	10.2352	8.1621	5.3345
	40	15.889	19.7982	15.6708	9.7791
	60	23.3802	29.3376	23.034	14.3169
	80	30.9686	38.7757	30.3661	18.7819
	100	38.4603	47.8707	37.9163	23.106
Relative Errors			0.2249	0.0126	0.3464
$\omega = 20\Pi$	4	1.4736	1.6752	1.5025	1.3343
	8	1.8977	2.2903	1.9522	1.625
	12	2.2519	2.8235	2.3679	1.8988
	16	2.5946	3.3014	2.7018	2.1172
	20	2.93	3.7811	3.054	2.3431
	40	4.4684	5.9915	4.7114	3.4548
	60	6.1101	8.4254	6.5633	4.6249
	80	7.6859	10.7209	8.2463	5.7008
	100	9.2996	13.1102	10.1182	6.7747
Relative Errors			0.2983	0.0526	0.1977

Table 3.2: Peakedness computed for arrivals processes with sinusoid arrival rates and various service time distributions.

We see in both tables that the exponential peakedness is close to the lognormal peakedness with coefficient of variation equal to 1. It is not as close to the lognormal peakedness with other coefficient of variations, even though the difference is never large. We conclude that the coefficient of variation of the service times impacts the peakedness more than the service time distribution and that approximating a peakedness by the exponential peakedness is fine as long as the coefficient of variation is close to one.

α	λ	Exp	LN(0.25)	LN (1)	LN(4)
$\alpha = 0$	4	1.4912	1.7133	1.5207	1.3399
	8	1.9743	2.4192	2.0298	1.6636
	12	2.4846	3.1447	2.5527	1.9921
	16	2.9445	3.8377	3.0868	2.3348
	20	3.4341	4.5642	3.6203	2.6543
	40	5.9956	8.0918	6.1774	4.3545
	60	8.4016	11.63	8.9004	5.9963
	80	10.8218	15.4192	11.4622	7.8388
	100	13.4294	18.5232	13.6843	9.2306
Relative Errors			0.3123	0.0384	0.2266
$\alpha = 0.25$	4	1.3875	1.5123	1.3966	1.2677
	8	1.7605	2.0265	1.7907	1.5287
	12	2.1298	2.532	2.2088	1.7907
	16	2.5251	3.0107	2.5762	2.0612
	20	2.909	3.5565	2.9992	2.3438
	40	4.7926	6.0849	5.0564	3.6855
	60	6.8032	8.6485	6.9629	4.9957
	80	8.6333	11.1354	8.8687	6.3553
	100	10.4587	13.7692	10.9669	7.7729
Relative Errors			0.2213	0.0296	0.197
$\alpha = 0.5$	4	1.2775	1.3341	1.277	1.197
	8	1.5450	1.665	1.5562	1.3877
	12	1.8109	2.0002	1.8284	1.5719
	16	2.1044	2.3096	2.0982	1.7939
	20	2.3544	2.6864	2.3511	1.9506
	40	3.7286	4.3449	3.7054	2.9636
	60	5.0422	5.8783	5.0438	3.8738
	80	6.5108	7.5486	6.5597	4.8467
	100	7.6261	9.211	7.8249	5.8778
Relative Errors			0.1293	0.0069	0.1709
$\alpha = 0.75$	4	1.1469	1.1653	1.1473	1.1156
	8	1.2996	1.3307	1.2932	1.2259
	12	1.4453	1.4886	1.4423	1.339
	16	1.5876	1.6724	1.6024	1.4753
	20	1.7425	1.8337	1.7456	1.5815
	40	2.467	2.6243	2.4889	2.1468
	60	3.2332	3.5062	3.2314	2.6971
	80	3.9836	4.3562	3.9078	3.4697
	100	4.7473	5.0861	4.9102	3.8152
Relative Errors			0.0543	0.009	0.1046

Table 3.3: Peakedness computed for arrivals processes with correlated arrival rates and various service time distributions.

3.3.2 *Overflow processes*

Tables 3.4 present and 3.5 a set of combinations of arrival rates, noted λ, for a Poisson process and of servers, noted s. Each combination was simulated under twelve dif-

ferent conditions. In each experiment the simulated Poisson process is sent to a pool with the number of operators specified in the configuration. The pool is a loss system and this creates the overflow process for which we wish to compute the peakedness.

In a first experiment each configuration is simulated with exponential service time distributions at both pools. This series is the benchmark for the other experiments. In the next three experiments exponential service time distributions are considered at the first pool but the distribution is switched to a lognormal at the infinite pool, with a variance of respectively 0.25, 1 and 4 for each experiment. In experiments 5 to 7, all service times, at the first pool as well as at the infinite pool, follow lognormal distributions with identical variances, 0.25, 1 and 4. In the three following experiments the distribution of the service times at the first pool is lognormal whereas it is exponential at the infinite pool. In the two last experiments service times at both pools are lognormally distributed but with different variances.

The results are displayed in Tables 3.4 and 3.5 alongside the configurations.

We observe here again that the the variance of the service time distribution affects the value of the peakedness quite significantly: the mean relative error is lower when the variance of the lognormal distribution is equal to the variance of the exponential distribution, i.e. when the coefficient of variation is equal to one, as shown by the results of Experiments 3, 6 and 9. This suggests once again that the service time distribution has a limited impact on the peakedness.

The second observation, taken from experiments 5, 6 and 7, is that peakedness variations are small compared to the benchmark when the service time distribution is the same at both the first and the infinite pool. Furthermore, when looking at the results of experiments 11 and 12, we observe that having different variances at both pools increases the difference with the benchmark, suggesting that keeping the same coefficient of variation for the service times at the two pools would

In conclusion approximating the peakedness computed with the service time distribution observed in the system by an exponential peakedness provides good results when the coefficient of variation of the service time distribution is close to 1.

3.4 CONCLUDING REMARKS

Although it is probably a less intuitive measure of variability than the well-known coefficient of variation, the peakedness presents many interesting features that make it richer than other more popular measures of variability. Firstly it does not require assuming independent interarrival times. On the contrary, it permits to include correlation between arrivals in the measure of variability. Secondly it is a particularly flexible metric as one has the opportunity to fix the service time distribution, and

the related parameters, of the fictitious servers that best fits to the context where the peakedness is used. We also discovered that the distribution has less impact on the value of the peakedness compared to the service rate and the variance of the service time distribution. We have also presented methods to compute the peakedness from real-life data. The computation is particularly amenable when service times at the fictitious pool are exponentially distributed.

		Experiments											
		1	2	3	4	5	6	7	8	9	10	11	12
λ	s	Exp Exp	Exp LN(0.25)	Exp LN(1)	Exp LN(4)	LN(0.25) LN(0.25)	LN(1) LN(1)	LN(4) LN(4)	LN(0.25) Exp	LN(1) Exp	LN(4) Exp	LN(0.25) LN(4)	LN(4) LN(0.25)
4	1	1.1314	1.2006	1.1471	1.0913	1.1463	1.1344	1.123	1.0949	1.1196	1.1694	1.071	1.2358
4	2	1.2787	1.4093	1.2937	1.1879	1.3125	1.282	1.2519	1.2054	1.2416	1.3478	1.1414	1.4689
4	3	1.4145	1.6032	1.4558	1.2932	1.4668	1.4195	1.3725	1.3044	1.3861	1.5226	1.2289	1.7036
4	4	1.5361	1.7866	1.5869	1.3743	1.6176	1.5385	1.471	1.4165	1.4964	1.6662	1.2861	1.8872
4	5	1.6344	1.9066	1.6807	1.4457	1.737	1.6338	1.5557	1.4936	1.589	1.7669	1.3486	2.0191
8	2	1.1915	1.2704	1.208	1.1322	1.2059	1.1911	1.1833	1.1424	1.1768	1.2462	1.0991	1.3429
8	4	1.4013	1.6122	1.4469	1.295	1.4519	1.411	1.3674	1.2848	1.3824	1.5276	1.2088	1.7305
8	6	1.6601	1.966	1.7174	1.4567	1.7401	1.6603	1.5913	1.4741	1.6182	1.8322	1.3558	2.1241
8	8	1.8928	2.3007	1.9543	1.6315	2.0206	1.8877	1.7922	1.6796	1.8273	2.0917	1.4792	2.4696
8	10	2.0439	2.4833	2.1158	1.7457	2.2193	2.0325	1.8976	1.7837	1.9627	2.2593	1.5656	2.6892
12	3	1.2116	1.3172	1.241	1.1683	1.2364	1.2196	1.2093	1.1509	1.1948	1.2768	1.1185	1.403
12	6	1.4984	1.7482	1.5603	1.3545	1.5416	1.5062	1.4732	1.3576	1.4626	1.6433	1.258	1.8941
12	9	1.8187	2.2211	1.9179	1.5698	1.9274	1.8631	1.7613	1.6282	1.7678	2.0672	1.4371	2.4706
12	12	2.143	2.7015	2.2658	1.8205	2.3357	2.1884	2.0385	1.8658	2.0586	2.4316	1.6421	2.9399
12	15	2.3556	2.9832	2.4547	1.9434	2.5971	2.3452	2.2175	2.0777	2.258	2.6506	1.7533	3.2091
16	4	1.2287	1.3469	1.26	1.165	1.2554	1.2379	1.2223	1.162	1.2186	1.3153	1.1371	1.4432
16	8	1.5715	1.8445	1.6227	1.3901	1.6127	1.5752	1.5364	1.378	1.521	1.7262	1.2908	2.0211
16	12	1.9583	2.431	2.0671	1.6923	2.0735	1.9757	1.9001	1.7055	1.9012	2.2437	1.5231	2.7031
16	16	2.3968	3.0531	2.5277	1.9653	2.5666	2.3924	2.2784	2.0848	2.3053	2.6896	1.7697	3.3483
16	20	2.6441	3.2734	2.753	2.1368	2.8976	2.6179	2.4104	2.2466	2.4956	2.9428	1.924	3.6262
20	5	1.2468	1.3802	1.2837	1.1919	1.2513	1.2565	1.2287	1.1622	1.2211	1.3522	1.1442	1.4583
20	10	1.6205	1.893	1.6698	1.428	1.6496	1.6105	1.5734	1.4411	1.56	1.7783	1.3201	2.089
20	15	2.0933	2.6532	2.2055	1.8009	2.2109	2.1131	2.0177	1.7932	1.9955	2.4188	1.5821	2.9051
20	20	2.599	3.3476	2.7278	2.1453	2.8653	2.5855	2.4422	2.2198	2.4967	2.9921	1.8585	3.6954
20	25	2.8397	3.7056	3.0065	2.3105	3.1488	2.8798	2.5512	2.4808	2.7227	3.241	2.0325	3.9921

Table 3.4: Peakedness computed for different combinations of arrivals ranging from 4 to 20 and operators and with varying service time distributions.

		Experiments											
		1	2	3	4	5	6	7	8	9	10	11	12
λ	s	Exp Exp	Exp LN(0.25)	Exp LN(1)	Exp LN(4)	LN(0.25) LN(0.25)	LN(1) LN(1)	LN(4) LN(4)	LN(0.25) Exp	LN(1) Exp	LN(4) Exp	LN(0.25) LN(4)	LN(4) LN(0.25)
40	10	1.3046	1.3977	1.3049	1.1876	1.2847	1.2777	1.2923	1.1901	1.2814	1.3772	1.1556	1.5357
40	20	1.7378	2.1085	1.8221	1.5427	1.7959	1.7538	1.7157	1.5289	1.6767	1.9648	1.3976	2.4096
40	30	2.4816	3.2259	2.6501	2.0771	2.5759	2.4761	2.3919	2.0584	2.361	2.9211	1.781	3.6611
40	40	3.4308	4.5294	3.6384	2.7167	3.7533	3.3657	3.2311	2.8182	3.3212	3.9951	2.3174	5.0304
40	50	3.6523	4.7223	3.9903	2.8126	4.05	3.6687	3.2504	3.1074	3.4066	3.9438	2.479	5.295
60	15	1.3042	1.4419	1.3071	1.2288	1.3116	1.2822	1.2946	1.1857	1.2803	1.394	1.1352	1.5779
60	30	1.8027	2.1973	1.851	1.6092	1.8384	1.848	1.8037	1.57	1.7574	2.067	1.4131	2.5298
60	45	2.7333	3.5926	2.9553	2.2513	2.828	2.7323	2.6018	2.2412	2.576	3.2282	1.8927	4.1102
60	60	4.0166	5.4419	4.3537	3.1527	4.4938	4.0738	3.7887	3.3172	3.8749	4.7732	2.6411	6.2407
60	75	4.2677	5.3547	4.3567	3.3257	4.9429	3.9649	3.8918	3.4766	3.8894	4.7298	2.9273	5.955
80	20	1.3168	1.4496	1.3394	1.2147	1.2991	1.2889	1.3063	1.2041	1.2971	1.433	1.1677	1.5884
80	40	1.8613	2.2697	1.9005	1.6061	1.864	1.8394	1.8677	1.6107	1.7673	2.1437	1.4317	2.5899
80	60	2.8966	3.7604	3.0604	2.3582	2.9948	2.8564	2.8126	2.2602	2.8107	3.3798	2.0264	4.4629
80	80	4.5723	6.2952	4.9278	3.5179	5.1688	4.6304	4.2431	3.7096	4.3227	5.496	2.9576	6.8504
80	100	4.4737	6.0399	4.6486	3.361	5.1184	4.8053	3.5506	3.9587	4.2286	5.0087	3.0651	6.2258
100	25	1.307	1.4691	1.3261	1.2516	1.3175	1.3236	1.2999	1.2147	1.2638	1.4366	1.1634	1.571
100	50	1.8802	2.3061	1.955	1.6049	1.8573	1.9024	1.854	1.5915	1.8061	2.1583	1.4369	2.6462
100	75	3.0039	4.0302	3.2262	2.419	3.1267	3.0132	2.9622	2.4252	2.9139	3.6635	2.0751	4.7606
100	100	5.0368	7.0055	5.54	3.9443	5.6441	5.1275	4.8203	4.0211	4.7994	6.1227	3.1961	7.9971
100	125	4.9524	5.8782	4.9816	3.6542	5.9009	4.5326	3.7946	3.8916	4.4717	5.2642	3.112	6.8972
Relative Errors			0.2151	0.041	0.1378	0.0584	0.0135	0.0444	0.1228	0.0358	0.1174	0.2169	0.3376

Table 3.5: Peakedness computed for different combinations of arrivals ranging from 40 to 100 and operators and with varying service time distributions.

4

HAYWARD'S APPROXIMATION

In the previous chapter we described the peakedness functional. We presented the peakedness as an indirect measure of variability of flows. We showed how it can be computed or estimated either from a sample of arrival times or from cumulative arrivals over time intervals. Assuming that it has been computed from arrival data, we now present applications of the peakedness to Queueing models. In Section 4.1, we present Hayward's approximation used to approximate the loss probability when the incoming flow is not Poisson distributed. In Section 4.2, we briefly mention some of the existing alternatives to Hayward's approximation. Finally, in Section 4.3 we apply the principles of Hayward's approximation to solve a call center outsourcing issue.

4.1 DESCRIPTION OF HAYWARD'S APPROXIMATION

G/G/s/s models are queueing models with a buffer space equal to zero. Calls that find all operators busy upon their arrival are rejected from the system and considered lost[1]. When the arrival flow is a Poisson process it is possible to find the exact value of the probability for an arriving call to be lost. This value is given by the Erlang-B formula:

$$B_E(s,a) = \frac{\frac{a^s}{s!}}{\sum_{k=0}^{s} \frac{a^k}{k!}}, \tag{4.1}$$

where a is the offered load to the system, i.e. the arrival rate λ divided by the service rate μ and s is the number of operators. The Erlang-B formula was first presented in [Erlang, 1917]. [Erlang, 1948] surveys the works of A.K. Erlang. The formula has been widely used and analyzed ever since then. See among others [Jagerman, 1974],

1 These models are sometimes refered as loss models

[Harel, 1990], [Berezner et al., 1998]. In particular, [Harel, 1988] proposes lower and upper quadratic bounds as well as quadratic approximations for this formula. The best bounds to date are presented in [Adelman, 2008].

Formula (4.1) does not apply to non-Markovian arrival processes. There exists however an approximation formula that is both simple and accurate: the Hayward approximation. This approximation was first presented in [Fredericks, 1980] and it was extended to call centers in [Chevalier and Tabordon, 2003] and [Franx et al., 2006]. The idea is to take the volatility of the overflow into account by working with an additional parameter: the peakedness. Practically, the Hayward approximation consists in introducing the peakedness in the Erlang-B formula by dividing both input parameters, namely the offered load and the number of operators, by the value of the peakedness.

As one can observe from (4.1) a M/G/s/s system is subject to economies of scale: if one doubles both the size of the system and the arrival rate, then the proportion of lost calls decreases. As described in [Fredericks, 1980], the principle of the approximation is to rescale the system, by dividing the rate of arrivals and the number of servers by the peakedness, the intuition being that the variability of the incoming flow offsets some of the economies of scale of the system.

To summarize, with Hayward's approximation, the blocking probability is estimated in the following way:

$$\mathrm{LP}(s, a, z) \approx B_E(s/z, a/z), \tag{4.2}$$

where LP(.) is the loss probability and z is the peakedness of the incoming flow as computed with the same service time distribution of the operators in the system. Note that s/z might not be an integer. This prevents using (4.1) directly. To overcome this difficulty one can compute the loss probability with $\lfloor s/z \rfloor$ and then with $\lceil s/z \rceil$ and use a linear interpolation method to evaluate the loss probability for s/z, which lies in between. Another possibility is to use an interpolation of the Erlang-B function, noted $B(s, a)$, mentioned and discussed in [Jagerman, 1984] and in [Chevalier et al., 2005], among others:

$$B(s, a) = \left[a^{-s} e^{a} \int_{a}^{\infty} e^{-x} x^{s} dx \right]^{-1}. \tag{4.3}$$

Note that in our notation $B_E(.)$, $B(.)$ and LP(.) are three slightly different notions. $B_E(.)$ is the loss probability that results from using (4.1), $B(.)$ is the loss probability obtained with (4.3) and LP(.) is the actual loss probability in a system. The first two

are equal to each other for integer values of s. Moreover, when the arrival process is Poisson, they are also equal to the latter.

The peakedness for an overflow can be computed analytically (see [Kosten, 1973] for a proof of this formula) when the incoming flow is Poisson and the service time distribution is exponential with the following formula:

$$z = 1 - aB_E(s,a) + \frac{a}{s+1+aB_E(s,a)-a}. \tag{4.4}$$

If the input is not Poisson, the peakedness can be approximated by the formula proposed in [Fredericks, 1980]:

$$z_{out} \simeq z_{in}\left(1 - \frac{a}{z_{in}}\mathrm{LP}(s,a,z_{in}) + \frac{a}{s+z_{in}+a\mathrm{LP}(s,a,z_{in})-a}\right), \tag{4.5}$$

where z_{in} is the peakedness of the input flow and z_{out} is the peakedness of the overflow. It is possible to show that for an overflow the peakedness is larger than one, reflecting the "bursty" nature of this type of flow.

We concluded in Section 3.3 that the distribution of the service times at the fictitious operators has a limited impact on the value of the peakedness. Therefore we believe that Formulae (4.4) and (4.5) provide good approximations for the peakedness when the service time is not exponential.

[Fredericks, 1980], [Wolff, 1989] and [Tabordon, 2002] note that Hayward's approximation underestimates the loss probability. This can be illustrated with a very simple experiment.

Example 4.1 *Assume a flow of Poisson arrivals with rate $\lambda = 5$ and a pool with 6 operators. The service time distribution is exponential with $\mu = 1$. With (4.1), we find that LP $= 0.19$. Divide now the pool into two subpools each staffed with 3 operators. The flow of arrival is first sent to the first pool and the overflown calls are then dispatched to the second subpool. The loss probability at the first pool is computed with (4.1) and is equal to 0.53. The overflow rate and peakedness are respectively equal to 2.65 and 1.39. When computing the loss probability at the second pool with (4.2), we find 0.35. The overall loss probability is obtained by multiplying the two partial loss probabilities, which yields LP $= 0.18$.*

To our knowledge no efficient correction for the bias has ever been proposed. The underestimation is however limited and can be neglected for most applications of Hayward's formula.

4.2 OTHER EXISTING METHODS

Hayward's approximation is one of many methods that have been proposed to address the problem of non-Markovian arrival processes in general and of overflow processes in particular. We hereafter list and briefly describe three of them. All are designed to compute the performance in a pool whose incoming flow is an overflow from a previous pool, similar to the one represented in Figure 4.2 (i).

THE INTERRUPTED POISSON PROCESS METHOD. The Interrupted Poisson Process (IPP) method was first proposed in [Kuczura, 1973]. It was extended in [Van Muylder, 2001] and in [Tabordon, 2002]. They both consider a system similar to the one displayed in Figure 4.2 (i), with two pools put in series. This method relies on the observation that in this model the incoming flow of the second pool can be divided in two phases. When some operators are available at the first pool, the second pool does not receive any call. This is what they call the OFF phase. The ON phase is the period of time when the incoming flow is routed to the second pool, because all operators in the first pool are busy. This phase starts each time the last available operator in the first pool starts serving a call. It ends each time an operator of the first pool completes a service. Figure 4.1 illustrates a succession of ON and OFF phases. Based on the previous work by [Kuczura, 1973], they assume that the overflow process can be approximated by an IPP. The interarrival times of such a process have an hyperexponential distribution of the second order.

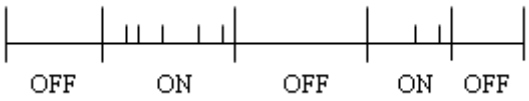

Figure 4.1: A sequence on ON and OFF phases. Big bars represent change of phase and small bars arrivals in the ON phase.

The principle of the IPP method is to assume that, even when the incoming flow to the first pool is not a Poisson process, the resulting overflow process is an IPP. More details, including a description of how to obtain the parameters of the approximating IPP, can be found in [Van Muylder, 2001].

THE HYPER-EXPONENTIAL METHOD. This method is presented in [Franx et al., 2006]. It uses the property that the interarrival times of the overflow of an M/M/s/s

system have an hyper-exponential distribution of order s+1, with probability density function

$$f(t) = \sum_{i=0}^{s+1} p_i \gamma_i e^{-\gamma_i t},$$

where γ_i is the parameter, i.e. the rate, of the i^{th} exponential distribution in the hyper-exponential distribution and p_i is the probability that the random variable has the same distribution as the i^{th} exponential distribution.

The method consists in approximating the distribution of the interarrival times of the overflow by an hyper-exponential distribution of order $\bar{s} \leq s$. The procedure for finding the parameters of this hyper-exponential distribution of order $\bar{s}$ can be divided in three steps:

1. determine the parameters of the distribution of the interrival times in the overflow process, given the incoming flow, the service rate and the number of operators. This requires solving a rather complicated set of equations.
2. Compute the first $2\bar{s} - 1$ moments of this distribution, knowing that its n^{th} moment is given by
$$M^{(n)} = \sum_{i=0}^{2\bar{s}-1} n! \frac{p_i}{\gamma_i^n}.$$
3. Deduce the coefficients of the hyper-exponential distribution of order $\bar{s}$ from the moments computed in the second step.

More details on these three steps, including the formulae used in the third step, are presented in [Franx et al., 2006], pp.7-8.

THE EQUIVALENT RANDOM METHOD. The equivalent random method (ERM) was first developed in [Wilkinson, 1956]. The principle relies on the observation that the loss probability for arrivals of a process that are serviced by s operators must be equal to the loss probability for arrivals of the same process when it is serviced by the same number of operators divided into two successive pools, as described in Figure 4.2.

Suppose an arriving flow is characterized by a rate λ and a peakedness z. It is required to compute LP(s,a,z), the loss probability when the flow is serviced by a pool of s operators. Based on the observation mentioned in the previous paragraph, about the performance fo two successive pools, ERM assumes that the arriving flow is the overflow from a Poisson process of rate λ_p that was first

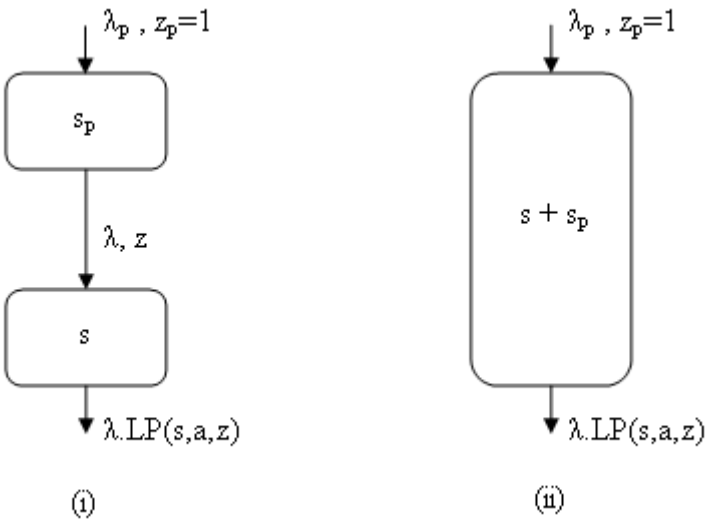

Figure 4.2: A system with two successive pools of operators (i) and the same system with one pool (ii).

sent to a pool with s_p operators. This means that the following two equalities hold:

$$\lambda = \lambda_p B(s_p, a_p) \tag{4.6}$$

$$z = 1 - a_p B(s_p, a_p) + \frac{a_p}{s_p + 1 + a_p B(s_p, a_p) - a_p}, \tag{4.7}$$

where $a_p = \lambda_p/\mu$. By solving this set of equations, we find the values of λ_p and s_p. A new system is then built in which the incoming flow is the Poisson process of rate λ_p and where the number of operators in the pool is equal to $s_p + s$, as represented in Figure 4.2 (ii). LP($s_p + s$,a_p,1) is computed with (4.1) (or (4.3), as s_p might not be an integer). As the The loss probability in the first pool, LP(s_p,a_p,1) is obtained from (4.6) and as the two systems must have the same overall loss probability, LP(s,a,z) immediately follows:

$$LP(s, a, z) = \frac{LP(s_p + s, a_p, 1)}{LP(s_p, a_p, 1)}.$$

The major difficulty is to determine λ_p and s_p from λ and z, because it requires solving $B(s_p, a_p)$ for a_p or s_p. To overcome this difficulty [Tabordon, 2002] suggests the following approximation, from [Rapp, 1964]:

$$a_p = a_p B(s_p, a_p)z + 3z(z-1).$$

An alternative recursive approach for solving the system (4.6)-(4.7) is proposed in [Jagerman et al., 1996].

Comparative studies of all these approximations in [Tabordon, 2002] and [Franx et al., 2006] show that Hayward's approximation performs in general best. Its simplicity of use makes it our approximation of choice throughout this thesis. Besides, the principle of dividing the load and the number of operators by the peakedness has potentially many more applications than Hayward's approximation only. One is presented in the next section.

4.3 AN APPLICATION OF HAYWARD'S APPROXIMATION: STAFFING FROM THE OUTSOURCER'S PERPECTIVE IN A CO-SOURCING SETTING

4.3.1 *Introduction*

In call center outsourcing settings, one of the biggest challenges that the vendor faces is the variability of arrivals to be handled. Even when the arrival rate is contractually determined, many operational decisions from the outsourcing company can have a significant impact on the short term variability. These decisions can range from call routing or sharing mechanisms to management of promotional activities. The effect of short term variability can be quite dramatic on the performance of the vendor. In a very simple example we show that depending on the variability, the performance levels change by an order of magnitude at least. In this section we propose a way, based on Hayward's Approximation to measure this short term variability from arrivals data and develop staffing methods that take this variability into account.

Call center outsourcing, the practice of sharing some or all calls at a client company's (outsourcer) call center with a vendor company's (service provider) call center, is a large industry worldwide. There is a growing literature in service operations management, emphasizing the growth in service outsourcing, particularly call center outsourcing ([Akşin et al., 2007a]; [Gans and Zhou, 2007]; [Akşin et al., 2008]; [Ren and Zhou, 2008]; [Hasija et al., 2008]; [Ren and Zhang, 2009]; [Akan et al., 2007]; [Keblis and Chen, 2006]; [Allon and Federgruen, 2005]).

In this section, we focus on the staffing problem faced by a vendor firm, in an outsourcing setting where calls are shared between a client firm and a vendor firm. Many firms prefer to share their calls with vendors, a practice known as co-sharing (or co-sourcing). According to a survey conducted by ICMI ([Dawson, 2007]), call centers that use outsourcing, predominantly rely on such partial sharing of their calls: Out of the outsourcing contact centers surveyed 42.9% outsource only 1%-20% of customer contacts the company receives, 12.7% outsource 21%-40% of the workload, 19% outsource 41%-60% of total volume, 17.5% outsource 61%-80% of contacts; and 7.9% report using a vendor to handle most or all (81%-100%) of the customer contacts received by the company. This highlights the prevalence of co-sourcing agreements and the associated need to understand call center operations under such agreements. Note that outsourcing may take many different forms. For example a company might answer all calls during working hours and close its operations overnight and outsource all its night calls. Here, we suppose in our analysis that outsourcing occurs continuously over the whole day, although our conclusions might extend to other kinds of outsourcing.

In many co-sharing settings (see for example [Gans and Zhou, 2007]; [Akşin et al., 2008]; [Keblis and Chen, 2006]), the client firm will serve incoming calls, that can be modeled as Poisson arrivals, in-house up to a given threshold determined by its own internal capacity, and will route any calls in excess of this threshold to the service provider. This type of overflow handling by outsourcer firms is quite common. According to [Dawson, 2007] overflow contacts are among the most common outsourced customer contacts (25.4%). Among the key reasons for outsourcing, handling the overflow is cited by 41.3%. In such settings the vendor call center will experience an overflow type arrival process from each client, or a combination of overflow arrival streams from multiple clients in the presence of pooled capacity. Short term variability of this type of call stream will be high. Capacity optimization at the vendor should take this variability into account.

It is natural to expect large call centers with uncertain demand to outsource overflow contacts, possibly to multiple smaller vendors. While partnerships between client-vendor firms may exist, a client firm operating in a high uncertainty environment dealing with several vendors having a cost minimization objective in mind, will typically treat the vendors as standard subcontractors. In the latter situation, information sharing between the two parties is minimal: the client does not share details of its own operations with the vendor, such as its own demand forecast or its own service capacity. Instead the client announces an average call volume that the vendor can expect to receive, coupled with required service levels and pricing information that will be stipulated in a contract. Some co-sourcing contracts may also impose minimum call

volumes that have to be outsourced to a vendor. Even with a client that shares more information, vendors will find it hard to deduce the variability of the arrival stream they will receive based on this information, due to the high volatility in such environments as well as the fact that calls may be dynamically shared with multiple vendors. In environments where promotional activities at the client firm can affect the burstiness of call volumes, a similar problem exist for the vendor who will typically not be informed of these activities. The vendor's capacity optimization is performed in such settings with restricted information, typically ignoring the additional variability of the arrivals.

4.3.2 *Literature review*

The literature on call center outsourcing is relatively young. Papers that focus on contracting problems constitute an important part of this literature ([Akşin et al., 2008], [Ren and Zhou, 2008], [Hasija et al., 2008], [Ren and Zhang, 2009], [Akan et al., 2007]). The contracts studied in [Akşin et al., 2008] are the only ones that allow for co-sharing of calls, which is where the overflow arrival issue being studied here will be relevant. Such contracts are also mentioned by [Keblis and Chen, 2006]. The focus in [Hasija et al., 2008], [Ren and Zhang, 2009], and in [Akan et al., 2007] is on contracts for complete outsourcing (i.e. all calls are outsourced to a vendor) under various forms of information asymmetry between outsourcers and service providers. The lack of information for the vendor regarding service capacity and demand at the client as considered herein, can be thought of as a combination of an operational information asymmetry found respectively in the form of asymmetric information regarding service productivity and cost parameters in [Hasija et al., 2008] and [Ren and Zhang, 2009], and demand information asymmetry as in [Akan et al., 2007].

In a related stream of literature, operational problems like staffing and call routing that arise from various outsourcing settings and contracts are analyzed ([Keblis and Chen, 2006], [Gans and Zhou, 2007], [Milner and Olsen, 2008], [Baron and Milner, 2009]). The capacity optimization problem of a client company, Amazon.com, is considered by [Keblis and Chen, 2006], where some calls are shared with vendors in different co-sourcing agreements. The co-sourcers are used to handle overflows, and the overall optimization imposes constraints on the fraction of calls that can be allocated to them. This suggests that the co-sourcing call centers face a similar problem to the one studied here.

Both [Gans and Zhou, 2007] and [Milner and Olsen, 2008] consider settings with two types of customers, a low and a high type or a contract versus non-contract type. It is shown in [Milner and Olsen, 2008] how certain types of service level agreements

might lead to inferior delay performance for contract type customers. This aspect of the problem is not considered in the current paper, where we focus on the simplest case with a single client. In [Gans and Zhou, 2007], high type calls are taken in-house, while the low type customers can be outsourced to a vendor. Minimization of the variable costs for outsourcing implies that the client firm will want to take low type calls in-house whenever its capacity allows it to do so. Different call sharing and routing schemes of low type calls to the vendor are analyzed. This is the only other paper in the literature that considers a co-sourcing setting as well as the operational problem of staffing at the vendor that may receive overflow type calls. Unlike the setting we consider, all information is symmetric. A queueing model with an interrupted Poisson process as arrivals is used to analyze the staffing problem. Instead, we consider staffing using the peakedness and Hayward's principle. A vendor firm's staffing problem under different service level agreements is analyzed in [Baron and Milner, 2009].

4.3.3 *Motivating example*

There are many ways for clients to share the load with vendors, in some circumstances the sharing mechanism might be very complex such as described in [Keblis and Chen, 2006]. In order to develop some understanding about the implications this might have for the vendor, we will analyze a simple example where the sharing mechanism is based on overflow when all internal operators are busy.

Figure 4.3 illustrates a schematic representation of the setting being considered in this section, for the simplest case with a single client and a vendor with a single pool of servers. We assume that the client firm receives calls arriving according to a Poisson process with rate λ_c. Calls are served in-house as long as one of the s_c servers are available, and overflow to the vendor otherwise. Assuming exponential service times, the client level of this system operates as an Erlang-B system. Motivated by the case of a real call center, we assume in this example that the vendor has no information regarding the client's arrival rate or number of servers. The client only communicates an average call volume λ_v to the vendor and requires a service level. The client's staffing problem can be interpreted as finding the capacity level that ensures the desired or announced call volume. The vendor will determine a staffing level s_v in response to the announced call volume λ_v.

In our illustrative example, we set the desired overflow volume to be $\lambda_v = 20$. We consider different cases where the client call center may experience different call volumes $\lambda_c = \{20, 60, 100, 140\}$. These correspond to cases where a decreasing percentage of calls are outsourced by the client, ranging from hundred percent to fourteen percent as we go from $\lambda_c = 20$ to $\lambda_c = 140$. Based on the survey results quoted in

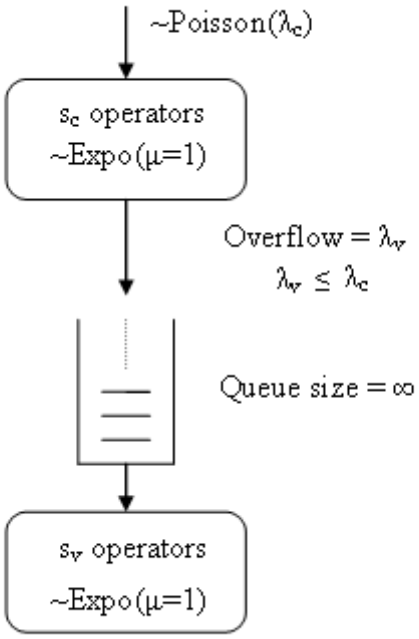

Figure 4.3: The system with one client and one vendor.

Section 4.3.1, we can state that the proportion of calls outsourced in these examples is in line with what we expect to find in practice. The varying λ_c values can be interpreted as variations in call arrivals experienced by a single client at different points in time. Interpreted this way, we would have call volumes reaching 140 at the peak. Another way of interpreting the λ_c values would be as different clients of the same vendor, each having a different call volume, but all promising the same call volume of 20 to the vendor. In such a multi-client setting, despite differences between the clients, the vendor would perceive them to be identical clients due to the common overflow volume $\lambda_v = 20$.

In the example, it is assumed that the client finds the corresponding staffing levels for each λ_c value to ensure an overflow with an average rate of 20. Service times are assumed to be exponential at both the client and vendor. To simplify the exposition, and without loss of generality, we set the service rates equal to one. Without going into the details of the staffing procedure at the vendor let us exogenously set $s_v = 30$ in the example. Note that given $\lambda_v = 20$, this capacity choice would ensure high service levels if arrivals did not have an overflow characteristic. Table 4.1 tabulates the waiting probability (WP) the waiting time (WT) and the service level (SL) at level 0.05

λ_c	20	60.2	99.6	140.07
s_c	0	42	83	125
λ_v	20	20	20	20
s_v	30	30	30	30
WP	0.025	0.157	0.256	0.327
WT	0.002	0.034	0.082	0.137
SL at 0.05	0.985	0.875	0.781	0.71

Table 4.1: The case where the vendor has no information about the client level.

experienced at the vendor as a result of this staffing level, obtained via simulation. As a reminder, a service level at level 0.05 is defined as the proportion of calls that have to wait less than 0.05 time units before being served. We observe that despite the fact that the arrival rate is constant at the vendor, the different service level measures show large variability as we go from $\lambda_c = 20$ to $\lambda_c = 140$, with more than an order of magnitude of difference between the extreme cases for the waiting probability, expected waiting time or probability to wait more than 0.05 of the service time.

We next consider the same example, but rather than assuming a staff level for the vendor, we determine via simulation the minimum number of servers that ensure a particular service level. The staffing levels are determined repectively to achieve a constant waiting probability (WP) of 0.05, constant waiting time (WT) of 0.05, or constant service level (SL) of 0.9. This setting can be seen as a theoretical benchmark or as representing the case where the vendor has perfect information about the client operations. The results are tabulated in Table 4.2. Irrespective of which service level measure is chosen as the objective, we see a big difference in the number of servers required at the vendor. This difference can be as high as forty percent of the number of servers in the Poisson arrivals case.

The examples in this section illustrate the impact short term variability can have on the operations of the service provider that has to comply with a service level agreement. In the following sections, we explore this situation further. We first consider a manager that uses square root staffing, a well known rule of thumb for staffing call centers. Using the same example from this section, we replace our exogenously specified $s_v = 30$ from before by the s_v obtained through square root staffing. We then consider the possibility of measuring the burstiness of the arrival stream when information about the first tier is not available.

4.3.4 *Square root staffing at the vendor*

A well known rule of thumb for staffing a call center with a single homogeneous pool of agents is known as the square root safety staffing rule. First attributed to [Erlang, 1948], it was formalized in the seminal paper by [Halfin and Whitt, 1981] for an Erlang-C system and shown to perform well for an Erlang-A system, i.e. a system in which each customer's patience is exponentially distributed ([Palm, 1957]), in [Garnett et al., 2002]. Other related references can be found in the survey articles [Gans et al., 2003] and [Akşin et al., 2007a].

Given an offered load λ/μ and a function β of the desired service level the square root staffing rule states that the number of servers should be set as

$$s^* = \frac{\lambda}{\mu} + \beta\sqrt{\frac{\lambda}{\mu}},$$

where s^* is the number of operators obtained with the square root staffing rule. This rule will ensure high utilization levels for the servers and low waiting times for the customers as long as the offered load is high.

Let us reconsider the example from the previous section in a setting where the manager at the vendor call center only knows its own average call volume and uses square root safety staffing. The offered load is 20 and depending on the magnitude of β chosen, the manager will determine an s_v^* value, i.e. the number of operators at the vendor as determined with the square root staffing rule. For example for $\beta = 2.24$ the manager will set $s_v^* = 30$ as assumed before. Our previous example illustrates that this choice of staffing will not obtain the desired service levels as the fraction of outsourced calls at the client is reduced. Since the square root safety rule adds a safety capacity that is proportional to the offered load, it fails to recognize the difference between the four systems due to the lack of information about the client's operations. This suggests that the square root safety rule might perform better if the client offered load is also considered.

When the client and vendor have a partnership relationship, the vendor will know the client's arrival rate and staffing level. Albeit not common, if this is the case, the square root staffing rule can be applied for the offered load of the entire system. In addition to the staff levels that have been obtained via simulation when a constant WP, WT, or SL was targeted, Table 4.2 also displays the corresponding square root staffing levels. The β used in these examples is obtained using the relationship

$$\beta = \sqrt{\frac{\mu}{\lambda}}(s_c + s_v)\left(1 - \frac{\lambda}{(s_c + s_v)\mu}\right)$$

λ_c	20	60.2	99.6	140.07
s_c	0	42	83	125
λ_v	20	20	20	20
$s_v \vert \text{WP} = 0.05$	29 (0.04)	35 (0.043)	39 (0.044)	42 (0.046)
Square root $\beta = 2.01$, s_v^*	29	33.81	36.68	38.88
$s_v \vert \text{WT} = 0.05$	25 (0.042)	29 (0.048)	32 (0.046)	35 (0.039)
Square root $\beta = 1.12$, s_v^*	25	26.87	27.76	28.3
$s_v \vert \text{SL at } 0.05 = 0.9$	27 (0.932)	31 (0.905)	34 (0.901)	37 (0.91)
Square root $\beta = 1.37$, s_v^*	27	30.34	32.22	33.59

Table 4.2: Comparaison of the square root staffing rule with simulations in the case where the vendor has perfect information about the client level.

applied to the first case where $\lambda_c = 20$, $s_c = 0$, ensuring Poisson arrivals at the vendor. The s_v in this expression is set to the simulated staff level for the corresponding system. Using this β in the three cases where $\lambda_c = \{60, 100, 140\}$, we obtain the number of servers for the informed vendor as follows:

$$s_v^* = \frac{\lambda_c}{\mu} + \beta\sqrt{\frac{\lambda_c}{\mu}} - s_c.$$

An application of square root staffing that takes the global offered load into account is clearly superior to one where only the offered load at the vendor is considered. However, we still find that as the fraction of outsourced calls is reduced, the gap between the staffing level obtained via square root staffing and one that is obtained via simulation is growing. Handling the same offered load in two separate systems (the client and the vendor) creates an efficiency loss due to variability, which is not taken into account by the square root staffing rule. This example suggests that standard staffing approaches, even when applied in sophisticated settings with information sharing, will not solve the vendor's capacity optimization problem in environments with call sharing.

4.3.5 *Adjusting the square root staffing rule*

[Whitt, 1992] shows how, in a system with Poisson arrivals and general service time distribution, the square root staffing rule can be obtained from the analysis of an equivalent (i.e. with the same arrival process and the same service time distribution as the initial model) model with an infinite number of servers (IS). Drawing upon the relationship with an IS model, the author then derives a square root staffing rule

in the case of general interarrival times.This leads to the following adjustment of the original square root staffing rule for a setting with arrivals having peakedness z:

$$s_H^* = \frac{\lambda}{\mu} + \beta\sqrt{z}\sqrt{\frac{\lambda}{\mu}}, \tag{4.8}$$

where s_H^* is the number of operators.

It is interesting to note that this expression can be obtained by applying Hayward's idea of scaling the number of servers and the load by the peakedness in the original square root staffing rule.

4.3.6 *Applying the Hayward approximation to queueing systems*

In this section we propose to apply Hayward's scaling idea to the standard queueing formulae. The peakedness acts here again as a sort of "dis-economies of scale" factor. We propose thus to use the following approximation for the Waiting Probability (WP) :

$$\mathrm{WP}(s, a, z) \approx C_E(s/z, a/z), \tag{4.9}$$

where $C_E(s, a)$ is the standard Erlang-C formula. It gives the probability for an arriving call to wait in a M/M/s system. Its equation is given by

$$C(s, a) = \frac{\dfrac{a^s}{[(s-1)!(s-a)]}}{\displaystyle\sum_{i=0}^{s-1} \frac{a^i}{i!} + \frac{a^s}{(s-1)!(s-a)}}. \tag{4.10}$$

In order to compute the Erlang-C for non integer values of the number of servers, it is convenient to use the algebraic relationship between the Erlang-B and Erlang-C formulae.

$$C(s, a) = \frac{sB(s, a)}{s - a(1 - B(s, a))}. \tag{4.11}$$

Note that we make the same distinction between $C_E(.)$, C(.) and WP(.) as the distinction we made between $B_E(.)$, B(.) and LP(.). This idea of using Hayward's approximation for a queueing system is also mentioned by [Whitt, 1992] as an implication of the infinite server approach to generalize the classical square root staffing rule to general arrival processes.

In order to estimate the waiting time distribution we use the fact that for a queueing system with a general arrival process (including overflow arrival processes) and an exponential service time distribution, the waiting time distribution of a call, pro-

vided that the call has to wait, is exponential (see [Kleinrock, 1975], Chapter 6, [Wolff, 1989], Chapter 7 , [Khintchine, 1960] or [Gross and Harris, 1998]). The expectation of the exponential waiting time distribution depends on the distribution of the interarrival times. No closed form expression exists for overflow type processes. We propose to once again continue the generalization of Hayward's approximation, dividing the load and staff levels by the peakedness of the flow. This yields the following approximations for the Waiting Time (WT):

$$P[\text{WT} \leq t | \text{WT} > 0] \approx 1 - e^{-\frac{s\mu-\lambda}{z}t}. \tag{4.12}$$

$$P[\text{WT} \leq t] \approx 1 - C(s/z, a/z)e^{-\frac{s\mu-\lambda}{z}t}. \tag{4.13}$$

$$E[\text{WT}] \approx \frac{zC(s/z, a/z)}{s\mu - \lambda}. \tag{4.14}$$

As the analysis in the next section shows, these remarkably simple approximations give very good staffing rules. Before that we show the virtues of Formulae (4.11) to (4.14) by testing them on a dataset and comparing the analytically obtained results with those obtained by simulation. The dataset consists of 144 configurations each made of a flow of arrivals and two successive pools. The latter pool receives arrivals only if all operators are busy in the former pool. When both pools are saturated, calls are put on hold. Waiting calls can only be serviced by operators at the second pool. All service times are exponentially distributed with a mean value of 1. The arrival rates and the staffing at each pool are presented in Appendix F. They represent a diverse set of examples with λ values that range between 4, representing small-size systems, and 100, corresponding to larger call centers, and different ways in which the servers are distributed in the two pools. Changing the number of servers in the first pool permits one to have different kind of overflows, leading to z values ranging between 1.13 and 5.10. Varying the number of servers in the second pool allows to test different levels of performance.

In Table 4.3 we present the mean absolute error (Mean AAbsErr), the mean error (Mean AErr), the minimum (Min AAbsErr) and maximum (Max AAbsErr) errors observed when comparing the simulated results with the results obtained with Formula (4.11) on the 144 configurations of the dataset.

We observe that the mean absolute error lies between 0.03 and 0.04. This is higher than what one would obtain when using the Hayward approximation in loss systems. Besides we see that the mean error is identical to the mean absolute error. This indicates that the approximation underestimates the actual waiting probability. A similar

	WP
Mean AAbsErr	0.0346
Mean AErr	0.0346
Min AErr	0.0001
Max AErr	0.1224

Table 4.3: Errors observed with Formula (4.11) in comparison to simulations results for the dataset presented in Appendix F.

	SL 0.05 (A)	SL 0.1 (A)	SL 0.25 (A)	SL 0.5 (A)	SL 1 (A)	$E[WT\|WT>0]$ (R)
Mean AAbsErr/RAbsErr	0.0036	0.0058	0.0087	0.0097	0.009	0.0529
Mean AErr/RErr	0.0029	0.0044	0.0055	0.0041	-0.0002	0.0005
Min AErr/RErr	-0.0023	-0.0044	-0.0101	-0.0183	-0.031	-0.1436
Max AErr/RErr	0.0255	0.0335	0.0489	0.0394	0.0285	0.5799

Table 4.4: Errors observed with Formulae (4.12) and (4.15) for the dataset presented in Appendix F. (A) indicates an absolute error and (R) a relative error.

behavior was observed in loss models by [Tabordon, 2002]. Here, the underestimation is further amplified with Formula (4.11). However, the major cause for the underestimation is the fact that during the epochs when the second pool serves waiting calls, the arrival stream is interrupted when operators in the first pool are available. During such interruptions, the queue is emptied at a faster rate. The likelihood for the second pool to be saturated is higher when the pool is fed with new arrivals than when it is not. Therefore, PASTA (Poisson Arrivals See Time Averages) does not hold and the proportion of arriving calls that are delayed is higher than the proportion of time the second pool is saturated. Despite this underestimation, the approximation still performs well.

We now use the same dataset to verify the accuracy of Formula (4.12). We test whether the parameter $(s\mu - \lambda)/z$ is a good proxy for the real parameter of the exponential distribution. We compute the proportion of the waiting calls that wait less than 5, 10, 25, 50 and 100% of the service time and compare them to what is observed in simulation. The results are presented in Table 4.4. In addition to the results on the service level, we also compare the average waiting time for the waiting calls $E[WT|WT > 0]$. According to our method, it can be estimated with

$$E[WT|WT > 0] \approx \frac{z}{s\mu - \lambda}. \tag{4.15}$$

We obtain excellent results: the mean error in absolute value is less than 0.01 for the 5 levels considered. It is even lower for smaller service levels. As the mean error shows, there is no significant bias. The maximum error is quite low as well. The

conditional waiting time is accurately estimated too, with on average a 5% error and no significant bias in the estimation. The maximum observed is close to 60%. This is large but corresponds to an extreme case where the load at the vendor is about 99.9% of the capacity. There are in fact three cases with extremely high loads for the vendor where absolute errors of about 30% are observed. Aside from these cases in which the system virtually reaches saturation, the approximation performs really well.

Although (4.12) gives impressive results, we must point out that this formula is not as important as Formula (4.13). This is why we repeat the whole experiment with the latter to effectively test its accuracy, despite the similarities of the two metrics. Table 4.5 presents the results in a similar way as Table 4.4.

	SL 0.05 (A)	SL 0.1 (A)	SL 0.25 (A)	SL 0.5 (A)	SL 1 (A)	$E[WT]$ (R)
Mean AAbsErr/RAbsErr	0.0322	0.0303	0.026	0.0212	0.0161	0.0968
Mean AErr/RErr	-0.0322	-0.0302	-0.0256	-0.0208	-0.0157	0.0859
Min AErr/RErr	-0.116	-0.1103	-0.102	-0.091	-0.0795	-0.143
Max AErr/RErr	0.0007	0.0025	0.0029	0.004	0.0044	0.5825

Table 4.5: Errors observed with Formulae (4.13) and (4.14) in comparison to simulations results for the dataset presented in Appendix F. (A) indicates an absolute error and (R) a relative error.

Considering that the method combines the approximations tested in Table 4.3 and in Table 4.4, the results we obtain are as expected. The average absolute error is a bit lower, at 0.02 or 0.03. Here again we observe a tendency to overestimate the service level. The estimation for the average waiting time is not as accurate as the other estimations but remains of good quality with an average error of less than 10%. It tends to underestimate the actual waiting time a little bit.

As a conclusion we believe that Formulae (4.11) to (4.14) yield an excellent compromise between simplicity of utilization and accuracy.

4.3.7 *Experiments and validation*

In this section we propose to compare both methods introduced in the previous sections to the classical Square Root Rule and show that both enable to improve the staffing decision in the presence of the peakedness.

We apply our methods to the examples presented in Section 4.3.4 For each scenario we determine the staffing level obtained using the square root rule with peakedness (z-Square Root) and the Hayward approximation applied to queueing systems method (Ha-Queueing). Because the number of operators must be integer, the simulation results are not exactly equal to the desired performance level: the staffing level has been

λ_c	20	60.2	99.6	140.07
s_c	0	42	83	125
λ_v	20	20	20	20
Simulated, s_v|WP $= 0.05$	29 (0.04)	35 (0.043)	39 (0.044)	42 (0.046)
z-Square root, s_v	29 $\beta = 2.01$	33.54 $\beta = 1.98$	36.27 $\beta = 1.97$	38.05 $\beta = 1.94$
Ha-Queueing, s_v	29	34.58	37.94	40.35
Simulated s_v|WT $= 0.05$	25 (0.042)	29 (0.048)	32 (0.046)	35 (0.039)
z-Square root, s_v	25 $\beta = 1.12$	27.44 $\beta = 1.05$	29.16 $\beta = 1.08$	31.01 $\beta = 1.15$
Ha-Queueing, s_v	25	28.96	31.88	34.8
Simulated s_v|SL at $0.05 = 0.9$	27 (0.93)	31 (0.91)	34 (0.9)	37 (0.91)
z-Square root, s_v	27 $\beta = 1.57$	29.86 $\beta = 1.4$	31.74 $\beta = 1.38$	33.69 $\beta = 1.42$
Ha-Queueing, s_v	27	30.67	33.18	35.7

Table 4.6: Comparison of peakedness methods with full information.

chosen in order to ensure the minimum number of operators that outperforms the desired performance level. Therefore, we do not set the desired performance level as the objective, but the effective performance reached in the simulations.

For example, in the case with an initial load of 60.2 and a desired waiting probability of 0.05, the smallest number of operators achieving this target is 35. With 35 operators the waiting probability observed in the simulation is equal to 0.043. Given that for our two approximation methods (peakedness adjusted square root and Hayward approximation) we are not constrained by the integrality constraint, we will use 0.043 as the target performance for those methods in order to make the results as comparable as possible with simulations.

For the z-Square Root rule, this implies that we compute a different β for each scenario using the procedure of Section 4.3.4 where we adjust s_v such that we obtain, in the base case where $\lambda_c = 20$ and $s_c = 0$, the same performance as the simulated benchmark (the number of operators is found analytically for the base case with exponential arrivals).

Table 4.6 summarizes the twelve scenarios. It is divided into three parts: one for each of the performance measures. In each of the three parts, the first line repeats the simulation results presented in Table 4.2 and gives the observed performance that serves as a target for the two methods. The second and final lines present s_v, respectively with the z-Square root method and the Ha-Queueing method.

We observe that the introduction of the peakedness improves the estimation of the staffing level. It is imteresting to note that while the approximations rely on less information than the square root rule (with perfect information) presented in Table 4.2, they provide better results.

4.3.8 *Peakedness in large-size call centers*

One relevant question for practitioners is whether the effect of peakedness demonstrated in the analysis persists in large systems. In particular, it is well known in call center practice that the effects of supply side variability diminishes with the size of a call center. In this section we illustrate that the peakedness is different and will persist in large systems.

As before, we choose to analyze the effect in a simple setting in order to gain intuition about the effect of peakedness for large size call centers. Consider a call center where the ratio of the incoming load over the number of operators is equal to α, i.e. $a = \alpha s$, the overflow of this call center is sent to the external service provider. We suppose that $\alpha > 1$ as the overflow rate would quickly be negligible for large systems if $\alpha < 1$.

The following Lemma provides an asymptotic expansion for $B_E(s, \alpha s)^{-1}$ when $s \to \infty$:

Lemma 4.1

$$B_E(s, \alpha s)^{-1} \sim \sum_{i=0}^{\infty} \frac{\alpha}{\alpha - 1} s^{-i} \frac{\delta^i}{\delta \alpha^i} \left(\frac{\alpha}{\alpha - 1} \right), \tag{4.16}$$

$s \to \infty$, $s > 0$, $\alpha > 1$.

Proof. See [Jagerman, 1974], Theorem 13. ■

We can exploit this result to find z when s tends to ∞.

Proposition 4.1 *Suppose $\alpha > 1$. Then,*

$$\lim_{s\to\infty} z(s, \alpha s) \lim_{s\to\infty} \left(1 - \alpha s B_E(s, \alpha s) + \frac{\alpha s}{1 + s - \alpha s + \alpha s B_E(s, \alpha s)} \right) == \frac{\alpha}{1 - \alpha}. \tag{4.17}$$

Proof. Replacing $B_E(s, \alpha s)^{-1}$ in (4.4) with (4.16) yields:

$$\begin{aligned}
\lim_{s\to\infty} z(s, \alpha s) &= 1 + \lim_{s\to\infty} \frac{-\alpha s}{\frac{\alpha}{\alpha-1}\left[1 - \frac{s^{-1}}{(\alpha-1)^2} + \frac{(2\alpha+1)s^{-1}}{(\alpha-1)^4} + o(s^{-3})\right]} \\
&\quad + \lim_{s\to\infty} \frac{\frac{\alpha^2 s}{\alpha-1}\left[1 - \frac{s^{-1}}{(\alpha-1)^2} + \frac{(2\alpha+1)s^{-1}}{(\alpha-1)^4} + o(s^{-3})\right]}{\frac{\alpha}{\alpha-1}\left[1 - \frac{s^{-1}}{(\alpha-1)^2} + \frac{(2\alpha+1)s^{-1}}{(\alpha-1)^4} + o(s^{-3})\right](1 + s - \alpha s) + \alpha s} \\
&= 1 + \lim_{s\to\infty} \frac{-s(\alpha-1)}{1 - \frac{s^{-1}}{(\alpha-1)^2} + o(s^{-2})} + \lim_{s\to\infty} \frac{\alpha s - \frac{\alpha}{(\alpha-1)^2} + \frac{2\alpha^2+\alpha}{(\alpha-1)^4} + o(s^{-2})}{\frac{\alpha}{\alpha-1} + \frac{3\alpha s^{-1}}{(\alpha-1)^3} + o(s^{-2})} \\
&= 1 + \lim_{s\to\infty} \frac{-\alpha s + \frac{3\alpha}{(\alpha-1)^2} + \alpha s - \frac{\alpha}{(\alpha-1)^2} - \frac{\alpha}{(\alpha-1)^2} + o(s^{-1})}{\frac{\alpha}{\alpha-1} + o(s^{-1})}
\end{aligned}$$

$$= \quad 1 + \frac{1}{\alpha - 1} = \frac{\alpha}{\alpha - 1}.$$

■

Figure 4.4 shows the evolution of z when s increases for two values of α. It is noticeable that the peakedness of the overflow will always stay higher than 1, this means that the overflow will always exhibit more variability than a Poisson process. Equation (4.17) also confirms the observation of our initial example that the smaller the fraction of calls that is overflown the larger its peakedness will tend to be. Observe also that the asymptotic peakedness increases when α tends to 1. This means that the impact of the peakedness will be maximal in systems that operate close to full capacity regime.

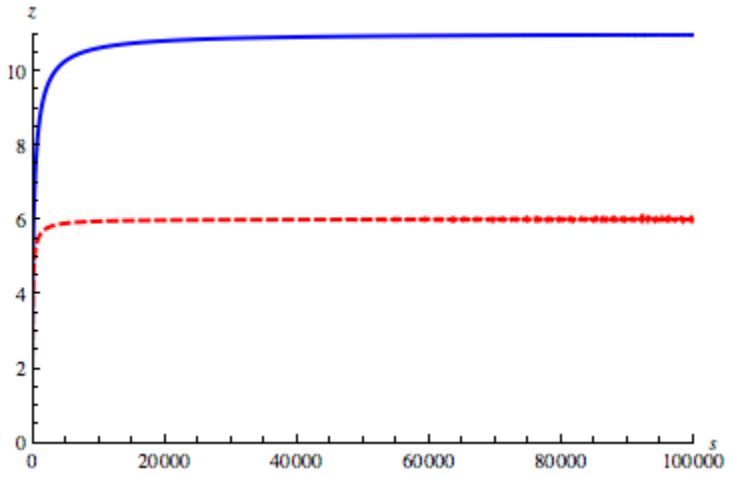

Figure 4.4: The evolution of the peakedness z when s grows, the blue continuous line corresponds to $\alpha = 1.1$, the red dotted line corresponds to $\alpha = 1.2$.

4.3.9 *Managerial implications*

Our results are applicable in co-sourcing relationships where overflow contacts are outsourced. In addition to providing simple quantifications of the effect of overflows on the vendor's operations, we provide a data-based approach to its measurement. Our analysis highlights the variability of arrivals as an important issue to be managed for successful vendor operations.

In the analysis we considered the example of overflow type arrivals at the vendor as a prototypical case of arrivals with high short term variability, characterized by an arrival stream with peakedness greater than one. Overflows are not the only form

of call sharing encountered in such settings. In practice, the client may choose to outsource its calls partially, but based on the type of call rather than a fraction of arriving calls. If this is the case, calls of one type will be entirely treated in-house while calls of another type will be fully outsourced to the vendor. Our approach enables the measurement of the peakedness of the outsourced call stream. If peakedness is found to be different than one (corresponding to non-Poisson arrivals) the proposed staffing approximations will apply. Another possibility, observed in practice, is for the client to outsource a steady stream of calls to the vendor, irrespective of whether it has capacity in-house or not. This type of outsourcing is labeled as *outsourcing the base* in [Akşin et al., 2008]. Other forms of sharing calls between a client and a vendor are analyzed in [Gans and Zhou, 2007]. Depending on how call sharing is actually implemented in real time, the vendor may observe arrival streams with different peakedness for the same base volume objective (or other objective) of the client. Thus, the possibility of measurement is a critical success factor for the vendor, particularly since details of operational decisions at the client will not be visible to the vendor. We next discuss how this issue will affect outsourcing contracts between the client and vendor.

In a setting with co-sourcing, our analysis has shown that the vendor's staffing costs depend on the mean demand as well as the peakedness of the arrival flow. Recalling the peakedness adjusted square root staffing rule, we observe that in addition to the economies of scale in arrival rate, the staffing at the vendor exhibits dis-economies of scale in the peakedness z , albeit as a second order effect. In other words, the vendor firm will prefer a client that shares a higher demand volume with lower peakedness. Recall that the peakedness of the arrival stream is a function of the original call volume at the client company, as well as their internal decisions regarding how to staff (and possibly allocate calls between multiple vendors), which then drives the call volume that will overflow to the vendor. Depending on the relationship between the two parties, the client may promise an average call volume over an agreed upon time interval, or may provide minimum and maximum levels within which call volumes can fluctuate to aid in the vendor's staffing decision. However in a typical setting where operational details are not shared, the vendor will not know the peakedness of the underlying demand, and the client will have an incentive to misrepresent it.

Taking a contracting approach to the problem, the vendor would want to screen clients for their call volumes and peakedness. In a setting with information asymmetry regarding call volumes, [Akan et al., 2007] show that a two-part tariff can screen different types of clients. This result relies on the underlying economies of scale. In a setting with information asymmetry and overflow type call sharing, this screening would have to be done for both the call volume and its peakedness. Screening will be difficult as the peakedness will subdue some of the economies of scale. There will be

two dimensions of information asymmetry and the two dimensions are not perfectly correlated. Furthermore, in practice relationships run over multiple periods where call volumes are uncertain. Distinguishing fluctuations due to misrepresentation of call volume type or peakedness type from fluctuations due to uncertainty will be difficult. Our measurement based approach provides a practical alternative to this difficult screening situation. Instead of trying to screen the clients upfront, vendors may include measurement based peakedness penalties to compensate for the dis-economies of scale these cause in their operations. Contracts that disregard the peakedness issue can be extended to include conditions regarding measured peakedness.

4.3.10 *Concluding remarks*

Throughout Section 4.3 we have analyzed the problem of staffing for a vendor firm that serves the overflow process of its client . The main difficulty for the vendor is that it does not possess the necessary information about the original demand and staffing level at the client which would enable it to understand the variability of its own arrival stream. As a result of the lack of information, if the vendor staffs according to average demand, it will lead to very bad service levels in cases where the client outsources a small fraction of its original call volume to the vendor. Without measuring the variability of the arrival rate the vendor is left to guessing how to staff. To overcome the problem, we propose to measure the variability of the overflow arrivals making use of the peakedness, and use this to staff the second tier call center. We propose two approaches to staffing, one which is an adjustment of the square root staffing rule, and the other that makes use of an extension of Hayward's approximation for the queueing system of the vendor. Both methods give very satisfactory results in order for the vendor to reach the promised service targets.

In modeling the vendor's operations, we have considered an infinite queue with patient customers. In practice, call center customers are known to abandon, so that the vendor's system will also have abandonments in practice. As shown in earlier literature (see for example [Garnett et al., 2002]) abandonments have a smoothing effect on service performance. Abandoning customers free system resources up for non-abandoning ones, thus improving service levels experienced by those who stay. Qualitatively, this implies that with abandonment, the differences in service levels or in staff levels that we have for the different cases in our examples will be smaller. However, based on our observations from the call center that motivated our analysis, clients typically request very low abandonment levels from the vendor, thus bringing the system at the vendor closer to the no abandonments case.

Even though we have focused on the simplest case with a vendor having a single client, the qualitative nature of our results will continue to hold when there are several clients. If it is possible for the vendor to pool calls from several clients in the multi-client case, it will face an arrival stream that is the sum of several overflow streams. Even though the vendor will benefit from pooling on the supply side, this will not necessarily reduce the peakedness. Thus the effects we have shown in the examples throughout the paper will still be present. Moreover, since our methods are based on the measured peakedness, the vendor can directly measure the peakedness after calls have been pooled and implicitly take into account correlations that might exist between different arrival streams. If the vendor cannot pool calls from different clients, this implies that it will be managing several small pools of servers, one for each client. As often with stochastic systems, smaller pools will exacerbate the effect of the variability on performance.

The client is modeled as an Erlang-B system in our analysis. In practice, the client may choose to outsource its calls partially, but based on the type of call rather than a fraction of arriving calls. If this is the case, calls of one type will be entirely treated in-house while calls of another type will be fully outsourced to the vendor, and standard models can be applied. Another possibility, observed in practice, is for the client to outsource a steady stream of calls to the vendor, irrespective of whether it has capacity in-house or not. This type of outsourcing is labeled as *outsourcing the base* in [Akşin et al., 2008]. Other forms of sharing calls between a client and a vendor are analyzed in [Gans and Zhou, 2007]. Interesting problems for future research arise from the analysis of different forms of call sharing when information is not shared between the parties, as studied herein. A related issue concerns determining better forms of call sharing that might mitigate the negative effects of not knowing operational details of the client company. These issues will play a role in contract negotiations for the vendor.

5

HAYWARD'S APPROXIMATION IN MULTI-SKILL LOSS SYSTEMS

After presenting the peakedness functional in Chapter 3, we illustrated in the previous chapter the importance of the peakedness in call center management. We presented Hayward's approximation, that uses the peakedness as a scaling factor in the Erlang-B formula. This permits one to account for the loss of efficiency due to the high variability of the incoming flow in a single-skill loss system. We also illustrated with an application that Hayward's principle of dividing the flow rate and the number of operators is applicable to many call center situations, even relatively simple ones.

These two chapters only considered single-skill models. Hayward's approximation is also useful for solving issues specific to multi-skill models. The main goal of the present chapter is to show how one can exploit Hayward's approximation to estimate the loss probability in a multi-skill loss system. This chapter is critical as from now on until the end of this thesis we are going to focus mainly on multi-skill models and all our applications will be based on this approximation.

Actually, even when all flows arriving to the multi-skill system are Poisson processes, computing the loss probability in a system with several types of calls and different sorts of operators, raises some problems. In Section 5.1 we review the difficulties, one by one. After that, in Sections 5.2 and 5.3 we present two close but different alternatives to compute the loss probability in multi-skill loss systems. The approximation presented in Section 5.2 is simpler but does not take into account some of the features described in Section 5.1. The method proposed in Section 5.3 is more complete. The two approaches are compared in Section 5.4. We end the chapter with a few concluding remarks.

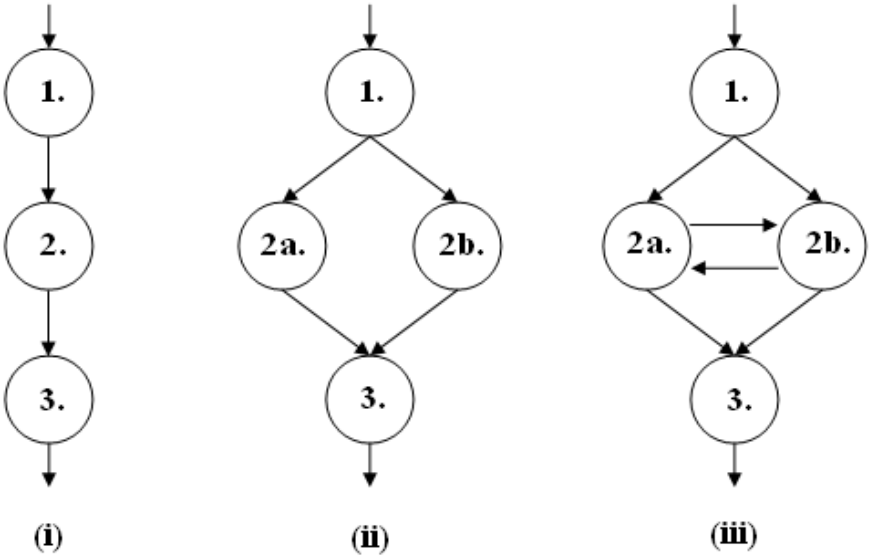

Figure 5.1: Three types of overflow situations one can encounter in a multi-skill system.

5.1 BUILDING A LOSS APPROXIMATION TO MULTI-SKILL SYSTEMS: MAJOR ISSUES

5.1.1 *Routing flows*

To compute the total loss probability in a multi-skill system, one needs to compute the loss probability at each pool. The total loss probability is then obtained by combining the pool loss probabilities. The combination depends on the routing configuration of the pools and different types of situations are possible. Figure 5.1 illustrates the typical situations one could face.

- The simplest situation one could encounter in a multi-skill system is for one type of call to have only one overflow path. This is illustrated in Figure 5.1 (i). In this case the loss probability for calls of this type is the product of the loss probabilities at each pool of the overflow path.

$$\mathrm{LP}_{(i)} = \prod_{j=1}^{3} \mathrm{LP}^{j}, \tag{5.1}$$

where $\mathrm{LP}_{(i)}$ is the loss probability in setting (i) and LP^j is the loss probability at pool $j \in \{1, 2, 3\}$, i.e. the ratio of the overflow over the flow arriving to the pool. We need to compute the loss probability in a downstream order as the overflow rate at the first pool is the inflow rate to the second pool.

- A more common situation is for a type of call to have multiple overflow paths. In Figure 5.1 (ii) there are two possible overflow paths. In this case the loss probability is obtained by first computing the loss probability for each overflow path, like in Figure 5.1 (i) and by then computing an average value of these loss probabilities weighted by the probability for a call to use each overflow path. Here this gives:

$$\mathrm{LP}_{(ii)} = \mathrm{LP}^1 \left(p\mathrm{LP}^{2a} + (1-p)\mathrm{LP}^{2b}\right) \mathrm{LP}^3, \tag{5.2}$$

  where p is the probability of choosing overflow path a. In the example presented here flows are first divided and then merged. Dividing flows and merging flows is adressed in Section 5.1.3. Note here that the two pools at the second level can be solved independently. Both computations require however to first compute the loss probability at the pool in the first level. Solving the pool at the third level requires knowing the loss probability at both second-level pools.

- Figure 5.1 (iii) represents a more complicated situation, where a portion of the overflow of one pool is dispatched to another pool of the same level whereas a portion of the overflow of the latter is next routed to the former pool. We say that there are cross-overflows in the model. This kind of situation occurs when at one level there is more than one pool that is skilled to serve calls of a specific type and when there is more than one overflow path for calls of this type. In our example, some calls will first check whether there is an available operator in pool 2a before visiting pool 2b while others will check pool 2b before going to pool 2a.

  In the model, this situation creates a loop in the computation as the rate of the incoming flow at one of the two pools in the loop depends on the overflow rate of the other pool and conversely. We present different ways of addressing this problem in the following sections.

  This is an issue specific to the model. Not considering the possibility of cross-overflows, like in Figure 5.1 (ii) would induce that calls might be dispatched to a pool of higher level without even checking all pools of one level and some calls might be rejected even though a skilled operator is available. This problem

does not occur if there is only one overflow path. We addressed this issue in Section 2.2.2.

5.1.2 *Computing the loss probability at one pool*

With Hayward's approximation it is possible to evaluate the size of each overflow. Practically we need to consider each pool separately. We compute or estimate the loss probability, LP, at each pool j with Hayward's approximation. The required parameters to use Hayward's approximation are the rate and the peakedness of the incoming flow.

The rate of the incoming flow is deduced from the overflow rate observed at the upstream pools in the overflow paths. An overflow rate at one pool is equal to

$$\lambda_{out}^{j} = \lambda_{in}^{j} \mathrm{LP}, \tag{5.3}$$

where λ_{in}^{j} is the rate of the incoming flow and λ_{out}^{j} the rate of the overflow.

The peakedness of the overflow, is found using (4.4) or (4.5) on page 47, depending on whether the input is a Poisson process or an overflow process.

5.1.3 *Merging and dividing flows*

As illustrated by Figure 5.1 (ii) and (iii) pools may receive more than one flow of calls, each with its own rate and its own peakedness. Using the Hayward approximation at each pool requires us to merge the flows and compute a unique rate, λ_{in} and a unique peakedness, z_{in}. This is done in the following way:

$$\lambda_{in} = \sum_{i=1}^{n} \lambda_i \tag{5.4}$$

$$\lambda_{in} z_{in} = \sum_{k=1}^{n} \lambda_k z_k + \sum_{k=1}^{n} \sum_{l \neq k} \mathrm{Cov}(S_k, S_l), \tag{5.5}$$

where λ_i and z_i are the rate and peakedness of the i^{th} of the n different flows received by the pool. $\mathrm{Cov}(S_k, S_l)$ is the covariance between process S_k and process S_l, S_i being the process associated to flow i, $i \in \{k, l\}$ that gives the number of busy fictitious operators. (5.5) holds because $\lambda_{in} z_{in}$ is a variance. See [Neal, 1971] for additional details on the covariance of flows.

When there are many types of operators, correlation effects exist among flows, even when the incoming calls are independent. Flows that pass through a same pool are

overflown from the pool at common epochs, gaining some correlation in the process. The loss probabilities and, consequently, the rate and peakedness of the overflows are affected by these correlations. This problem is presented in detail in [Neal, 1971] and in [Tabordon, 2002]. In [Neal, 1971] an algorithmic procedure is proposed to compute the covariance of the overflows of two different pools when their respective inflows are correlated overflows of a common third pool. The author also provides a formula for splitting an overflow from a single incoming flow into K subflows and to compute the covariance of the subflows:

$$z_{k,out} = \alpha_{k,out}(z_{out} - 1) + 1, \tag{5.6}$$

where $z_{k,out}$ is the peakedness of subflow $k \in \{1, .., K\}$, z_{out} is the peakedness of the total overflow and $\alpha_{k,out} = a_k/a$ is the proportion of subflow k in the overflow. By replacing z_k in (5.5) with the expression for z_{out} in (5.6) and then by isolating $\text{cov}(S_k, S_l)$, we find that the covariance between two subflows in this context is:

$$\text{Cov}(S_k, S_l) = \alpha_{k,out}\alpha_{l,out}a_{out}(z_{out} - 1). \tag{5.7}$$

5.1.4 *Different loss probabilities for different types of calls at one pool*

Using Hayward's Approximation with λ_{in} and z_{in} yields an overall overflow probability at the pool, LP. The overall overflow rate is equal to $\text{LP}\sum_{i=1}^{K}\lambda_i$, where $\{1, .., K\}$ is the set of subflows. Let LP_k be the overflow probability for subflow k. If all flows have identical distributions, then $\text{LP}_k = \text{LP}$ but if they have different variabilities we expect the LP_k's to be different. Simulation experiments show that the loss probability and peakedness of the overflow of a particular type of call will both be higher if the peakedness of the incoming flow of calls of that type is higher. To the best of our knowledge there has not been any publication about a model for this relation. It is usually assumed that the loss probability at one pool is the same for each subflow. We propose here a simple method to differentiate the loss probability for each subflow as a function of its variability.

Let us first note that in any case the following should remain true

$$\sum_{i=1}^{K}\lambda_i\text{LP}_i = \text{LP}\sum_{i=1}^{K}\lambda_i. \tag{5.8}$$

This equation has many solutions. Obviously, one consists in assuming that all LP_k's are equal, as if the incoming flows had identical distributions. Since we expect that

flows with higher peakedness (which means higher variability) will have higher overflow probabilities, we propose to assume

$$\lambda_k \mathrm{LP}_k = \frac{\lambda_k z_k}{\sum_{i=1}^{K} \lambda_i z_i} \mathrm{LP}. \tag{5.9}$$

This equation verifies (5.8). We test it and compare it in Section 5.4 to the classical approximation, that assumes that all LP_k's at one pool are the same. This simple approximation deserves a deeper analysis but this is left as a future extension.

5.2 COMPUTING THE LOSS PROBABILITY IN MULTI-SKILL SYSTEMS WITH UNCORRELATED OVERFLOWS

Chapter 3 of [Tabordon, 2002] is fully dedicated to adapting Hayward's approximation to the framework of multi-skill call centers. This includes investigating the importance of correlation effects among flows. We here reproduce the procedure she proposes and adapt it to our notation.

The general principles of the method have been presented in the previous sections of this chapter. The loss probability of a multi-skill system is determined by first computing the loss probability in each pool of the system. The resulting loss probabilities are then associated, in accordance with (5.1) and (5.2) to obtain the total loss probability. In this method [Tabordon, 2002] assumes that 1. subflows are independent of each other and 2. the proportion of each subflow in an overflow is the same as the proportion in the incoming flow. Put differently the loss probability at each pool is assumed to be independent of the type of call.

We now focus on pool j and detail how to determine the overflow for each of the incoming subflows. s^j is the number of operators at pool j. We also define $\mathcal{K}$ as the set of all subflows of the system. $\mathcal{K}^j$ is the set of subflows that pass through pool j. We use index k to refer to any element of $\mathcal{K}^j$. The procedure for computing the loss probability at pool j is the following.

1. Identify all incoming flows and merge them to form a unique incoming flow with rate λ^j_{in} and peakedness z^j_{in}. This is done in the following way:

$$\lambda^j_{in} = \sum_{k \in \mathcal{K}^j} \lambda_k \tag{5.10}$$

$$z^j_{in} = \frac{\sum_{k \in \mathcal{K}^j} \lambda_k z_k}{\sum_{k \in \mathcal{K}^j} \lambda_k}. \tag{5.11}$$

2. Compute the loss probability at pool j by introducing λ_{in}^{j} and z_{in}^{j}, along with s^{j} in (4.2).

3. Find the rate and peakedness of the overflow, λ_{out}^{j} and z_{out}^{j} with (5.3) and (4.5).

4. Find the rate, $\lambda_{k,out}$, and peakedness, $z_{k,out}$, of each subflow $k \in \mathcal{K}^{j}$ with:

$$\lambda_{k,out} = \lambda_{out}^{j} \frac{\lambda_k}{\lambda_{in}^{j}} \quad \forall k \in \mathcal{K}^{j} \tag{5.12}$$

$$z_{k,out} = z_{out}^{j} \quad \forall k \in \mathcal{K}^{j}. \tag{5.13}$$

The procedure presented in Chapter 3 of [Tabordon, 2002] is designed for solving models without cross-overflows, like the one represented in Figure 5.1 (iii). It is referred to as the static model. In Chapter 6, the author proposes a simple four-step procedure to account for cross-overflows in so-called dynamic models. We describe the procedure for the situation illustrated in Figure 5.1 (iii).

1. Compute the loss probability at each pool j in the loop as if there were no cross overflows. In our example, this means that only overflows from pool 1 are considered in the incoming flow to pool 2a and the one to pool 2b. Deduce the corresponding overflow rate $\lambda_{out}^{j,(1)}$. Decompose the general overflow to obtain an overflow for each incoming subflow. Here, each overflow consists in only one flow as the only incoming flow comes from pool 1.

2. For each subflow involved in the loop, route the overflow to the next pool of its overflow path and add it to the incoming flow. The incoming flow to pool 2a consists now in an overflow from pool 1 and an overflow from pool 2b.

3. For each pool j in the loop, compute the loss probability with the new incoming flow. The loss probability will be higher as the load of the incoming flow is higher.

4. We find a new general overflow with rate $\lambda_{out}^{j,(2)}$. This overflow mixes different types of flows: in our example, at pool 2a part of the overflow is routed to pool 2b and the remaining overflow goes to pool 3. The rate of the overflow to the next level is computed as $\lambda_{out}^{j,(2)} - \lambda_{out}^{j,(1)}$.

This procedure is relatively simple to implement and easily extended to more complex loops that include more than two pools and more overflows.

[Tabordon, 2002] investigates the importance of correlation effects among subflows by comparing this latter method with a similar method where the correlation between flows is taken into account. Practically she uses Formulae (5.6) and (5.7) instead of

(5.11) and (5.13). She tests both methods on a dataset of configurations with three types of calls and seven different pools (see [Tabordon, 2002], Section 3.6.2) and compares the results to the ones found when simulating the configurations of the dataset.

She concludes that this application of Hayward's approximation to multi-skill systems tends to (slightly) overestimate the loss probability when it is higher than 10%. For lower loss probabilities the procedure usually underestimates the real loss probability. When taking into account correlations effects, the approximation is not improved. It appears that neglecting the correlation actually smoothes the tendency of Hayward's approximation to underestimate the loss probability.

5.3 COMPUTING THE LOSS PROBABILITY IN MULTI-SKILL SYSTEMS WITH CORRELATED OVERFLOWS

In this section we present an alternative method for computing the loss probability in multi-skill call centers. It aims at taking into account the specificity of each overflow and the correlation effects among overflows. The former is accomplished by differentiating the loss probability at each pool according to the call type.

As stated before, computing the loss probability for any particular call type in the system requires one to compute the loss probability at each pool. Therefore we need to determine the incoming flow to each pool. To do that, we first divide flows entering the system into subflows such that there is exactly one subflow for each overflow path and for each call type. The rate of each subflow is equal to the total arrival rate for the calls of the type multiplied by the probability that a call of this type chooses to follow the corresponding overflow path. Along the overflow path, the rate and the peakedness of the subflow change, accounting for the calls that are answered.

Before computing the loss probability in one pool, it is necessary to first determine all the flows that form the incoming flow. This might require computing the loss probability at other pools as, as described with Figure 5.1, overflows from some pools are parts of the incoming flow to other pools. The order in which loss probabilities are computed matters: any pool whose overflow is an incoming flow to another pool must be solved before computing the loss probability at the latter. When the routing is hierarchical, a pool of a lower level is solved first.

The problem of cross-overflows illustrated with Figure 5.1 (iii) is solved by applying the following iterative procedure:

1. Compute the loss probability for each of the pools that share cross-overflows, without including the cross-overflows. Determine the overflows for each of the pools.

2. Include the cross-overflows computed in the previous step in the incoming flows and compute new loss probabilities for each pool. Update the overflows.

3. Repeat step 2 until cross overflows are stabilized, i.e. the difference between the newly computed cross overflows and the previous ones becomes negligible.

At each iteration and at each pool involved in the loop, the incoming flow is increased by the cross overflow. This increases the loss probability at each of the pools, which causes the overflow to increase too. However, the increase in the size of the overflow at each pool is lower than the increase in the size of the incoming flow, else we would have a situation where the occupation of the operators decreases with an increase in the incoming flow. Therefore, the marginal increase of the incoming load at each pool becomes lower and lower with each iteration of the procedure. This guarantees that the procedure converges to an equilibrium.

We keep the notation of Section 5.2 for the procedure to determine the loss probability at each pool. For each pool j, we need to

1. Merge the incoming flows to obtain λ_{in}^j and z_{in}^j, using (5.4) and (5.5), i.e. taking the covariances into account. At the first level, all covariances are equal to zero, as all subflows are independent. The correlation between two flows that have a common pool in the upstream part of their respective overflow paths will be positive.

2. Compute the loss probability at the pool using (4.2). Determine λ_{out}^j and compute z_{out}^j, the peakedness of the overflow, using (4.5).

3. Compute the overflow rates of each subflow k using (5.8).

4. For each subflow k, compute $\alpha_{k,out} = \lambda_k \mathrm{LP}_k / \lambda_{out}$, where $\alpha_{k,out}$ is the proportion of k in the overflow, and compute $z_{k,out}$ from (5.6).

5. For each overflow k, compute the covariance with all other subflows l:

   - if $l, k \in \mathcal{K}^j$, $\mathrm{Cov}(S_k, S_l)$ is given by (5.7).
   - if $k \in \mathcal{K}^j$, $l \in \mathcal{K} \setminus \mathcal{K}^j$ and if $\mathrm{Cov}(S_k, S_l) > 0$ before pool j, the covariance is updated using

$$\mathrm{Cov}(S_k, S_l) = \frac{\alpha_k \alpha_l (\lambda_k z_k + \lambda_l z_l)}{1 - 2\alpha_l \alpha_k}, \tag{5.14}$$

     which is obtained by isolating z_{out} in (5.7), by then replacing z_{in} in (5.5)[1] with this new expression and finally by isolating $\mathrm{Cov}(S_k, S_l)$. This proce-

1 Although presented with λ_{in} and z_{in}, (5.5) holds for any combination of flows

dure is an approximation as we assume that the covariance between any two dependent flows can be computed with (5.7).

This procedure is repeated with all pools of the system until all overflows are determined.

As mentioned earlier, this procedure is an approximation. The simulation experiments described in the next section show us that the procedure gives very good estimates of the loss probabilities.

5.4 COMPARISON OF THE METHODS

We now compare the methods described in Sections 5.2 and 5.3 using the datasets of Tables A.3 and A.4 given in Appendix A. The first dataset presents a series of configurations with three types of calls and the second dataset presents configurations with five types of calls. In practice we compare four different methods: the method proposed in [Tabordon, 2002], as described in Section 5.2 and noted "Uncorr", a similar method where the approximation proposed in Section 5.1.4 is included, noted "Uncorr(b)", the method presented in Section 5.3 with the approximation of Section 5.1.4 dropped, noted "Corr", and the method presented in Section 5.3, noted "Corr(b)". In all four methods, we used the iterative procedure described in Section 5.3 to deal with cross-overflows. We compare all four methods to simulation results. The average error and the average error in absolute value for each type of calls are presented in Tables 5.1 and 5.2.

	Call Type	x-calls	y-calls	z-calls
Mean AErr	Uncorr	0.0013	0.0013	0.0051
	Uncorr(b)	0.0047	0.0026	-0.0045
	Corr	0.0075	0.0067	0.0101
	Corr(b)	0.0078	0.0079	0.0083
Mean AAbsErr	Uncorr (AV)	0.0036	0.0045	0.0051
	Uncorr(b) (AV)	0.0062	0.0065	0.0073
	Corr (AV)	0.0075	0.0074	0.0101
	Corr(b) (AV)	0.0078	0.008	0.0083

Table 5.1: Average errors of the three methods as compared to simulation results for the configurations of dataset of Table A.3.

We observe that for the configurations with three types of demand the method proposed in [Tabordon, 2002] is the best performing. Differentiating the loss probability and including correlation effects both decrease the performance of Hayward's approximation. This confirms the results presented in Chapter 3 of [Tabordon, 2002]. The results for the configurations with five types of demand are not as clear as for

	Call Type	v-calls	w-calls	x-calls	y-calls	z-calls
Mean AErr	Uncorr	-0.0068	0.0008	0.0037	0.0043	-0.0115
	Uncorr(b)	0.0022	0.0038	-0.0014	-0.0062	0.017
	Corr	-0.0063	0.0015	0.0047	0.0049	-0.0109
	Corr(b)	-0.0002	0.0069	0.0031	0.0116	-0.0098
Mean AAbsErr	Uncorr	0.0101	0.0056	0.0068	0.0058	0.0119
	Uncorr(b) (AV)	0.0126	0.0166	0.0156	0.0167	0.0201
	Corr	0.0099	0.0061	0.0066	0.0064	0.0112
	Corr(b) (AV)	0.01	0.0103	0.0091	0.0131	0.0102

Table 5.2: Average errors of the three methods as compared to simulation results for the configurations of dataset of Table A.4.

three types of calls. Tabordon's method still performs the best but the differences are so small that it is not significant. The fact that the results for Uncorr, Corr and Corr(b) are so close to each other suggests that correlations among flows has a bigger impact in more complex settings (with increased number of call types and pools). A comparison of Corr and Corr(b) shows that differentiating the loss probabilities seems not to improve the performance for multi-skill loss systems.

5.5 CONCLUDING REMARKS

Hayward's approximation is a useful tool in the context of multi-skill loss models. Such models are much more complex than their single-skill counterparts. Effects such as correlations among flows or different variabilities of flows play a much bigger role and the modeller must decide which of many elements he should include in his model, depending on the application. In this chapter we presented different methods that each include a different set of assumptions. In our numerical application we confirmed the conclusion of [Tabordon, 2002] that taking into account correlations between flows in three-skill systems decreases the quality of the approximation for systems with three types of calls. For the loss systems with five types of calls that we analyzed, the results are slightly different: including correlations and assuming that calls of different types have equal loss probabilities at each pool, no matter the variability of their flow, does not improve the approximation but does not deteriorate it either. This leaves the modeler with different alternatives, with different sets of assumptions, to estimate the loss probability. One can choose the one that best fits its application.

Note also that even if the methods described here are designed to be used with Hayward's approximation, they could be adapted to the methods presented in Section 4.2, to some extent at least. In addition to the presentation of the hyper-exponential ap-

proximation, [Franx et al., 2006] proposes a method to estimate the performance of multi-skill loss systems.

Altogether, these methods pave the way to deeper analysis and manipulation of multi-skill models. In the remainder of this thesis we use them in two main applications: the development of approximations for the performance in multi-skill queueing models and the optimisation of the staffing of small-size call center. The latter is described in the next chapter.

6

OPTIMISATION OF MULTI-SKILL SYSTEMS IN THE ABSENCE OF QUEUES: A BRANCH AND BOUND APPROACH

With the Hayward-based methods developed in the previous chapter, we can estimate the performance of multi-skill loss systems. A first application of these methods is the possibility to compare different configurations and select the best performing one. In this chapter, we propose a method to find the best configurations for small-size loss systems, using the approximations of the previous chapter. The content of this chapter is actually a reproduction of the ideas and results published in [Chevalier and Van den Schrieck, 2008].

6.1 INTRODUCTION

We showed in Section 2.1 that a lot of research in call center analysis focuses on large call centers and on the economies of scale that can be achieved. Small size call centers are however frequent in B2B environments. Typical situations where one would find such type of call centers are:

- helpdesks for very specialized services, where agents might need special tools or software programs to solve the client's problems remotely (e.g. mobile phone operators);
- management of large accounts. In order to better serve the particular needs of its clients, a company might want to have operators specially trained to handle the specific problems that might arise with its large accounts (e.g. banks);
- hotline for retailers. In some situations the retailers provide part of a service (e.g. activation of mobile phone subscription): when a customer has an exceptional demand, the retailer must be able to call on very specialized agents to quickly find a solution (e.g. mobile phone operator).

These types of services are very specialized and will not generate a large demand justifying many agents. Often, companies try to group services that require skills relatively close to each other and where it is then possible to increase pooling through cross-training. Given the specialized nature of the services, cross-training can be expensive. Therefore it is necessary to find the best trade-off between the cost of cross-training and the economies of pooling. In the context of small call centers, this requires combining the stochastic aspects of arrival and service processes with the combinatorial aspects of the staffing decisions.

There is to our knowledge little attention paid to the specific problems of such small size call centers in the literature. In this context, the contribution of this chapter is two-fold: we contribute to filling a gap in the research by providing answers to important questions that managers of such call centers have and we present an original method that combines the stochastic and combinatorial aspects in an optimization model.

In the following section we describe the problem we address and show that solutions proposed in the literature do not treat adequately the specific features of small size call centers. We then formulate the problem in a more formal way. This is followed by a section that presents a branch and bound type algorithm to find an optimal configuration. Then, some numerical experiments are presented to illustrate the usefulness of our method and the results of a simulation experiment are given to validate the main hypothesis of our model. We end with a concluding section.

6.2 RELATION TO EARLIER WORK

To operate a multiple skill call center in an efficient way, the manager has to decide on a routing policy for incoming calls, and on a staffing level for each group of operators.

Assigning an incoming call to a group of operators that can handle the corresponding call type is referred to as skill-based routing. Optimizing the routing policy is a difficult problem, and it is only recently that researchers have started investigating this domain. [Garnett and Mandelbaum, 2001] present an introduction to the different types of routing schemes and their complexity. [Örmeci, 2004] shows the form of the optimal routing policy when there are two skills. In the general case with an arbitrary number of skills, [Bhulai, 2009] derives a heuristic rule to route calls based on the state of the system, and [Borst and Seri, 2000] propose a heuristic rule to route calls based on a credit scheme that depends on allocations made previously. [Sisselman and Whitt, 2007b] extend the classical skill-based routing by introducing value-based routing. The idea is to add some quality measurement in the model: instead of simply maximizing the proportion of calls that are answered, value-based routing tries to optimize an expected revenue that is an indirect measure of qualitative elements

such as first-call resolution or the preference of the operators. The authors show that it is possible to use value-based routing by adapting the skill-based routing algorithm presented in [Wallace and Whitt, 2005]. There is a large literature on the staffing of call centers, but most of the articles are about the staffing of systems with operators that all have the same skill. The general trend in those articles is to show that different variants of what has become known as the "square root staffing rule" enable one to reach the best trade-off between service quality and staffing cost.

In multiple skill environments, there are some articles about call centers with two skills. For example, [Stanford and Grassman, 2000] study a call center with a group of monolingual agents handling calls of the majority of callers and a group of bilingual agents able to answer calls of the minority language too (see also [Shumsky, 2004] for another approach to a similar setting).

For the staffing in an environment with an arbitrary number of skills, to our knowledge, all articles focus on large call centers and neglect as a result the integrality constraints on the staffing levels (at any time you can only have an integer number of agents in each group). [Wallace and Whitt, 2005] study the case where all agents master the same number of skills. They show that there is a staffing rule for the one in which all operators have two skills that already enables one to capture most of the efficiency gains that can be obtained from cross-training. Using a fluid model approach [Harrison and Zeevi, 2005] develop a way to compute the optimal staffing level when most of the traffic randomness comes from the non-stationarity over time of the arrival process. [Whitt, 2006d] and [Green et al., 2007] are other recent references of models that are dealing with variability on a long time scale. [Chevalier et al., 2005] show that in a similar (but somewhat simpler) setting to the one studied here, a very simple heuristic gives results close to optimality. This heuristic essentially says that when only specialists and fully polyvalent operators are available then 20% of the budget of the call center should be allocated to cross-trained operators. This confirms a general finding of articles in this area that a little flexibility goes a long way. The same type of conclusion can be drawn from several articles that analyze call centers using heavy traffic models (see for example [Gans and van Ryzin, 1997] or [Harrison and López, 1999]). [Pot et al., 2008] propose a 2-step method that first find the required staffing capacity of a multi-skill call center and then solves the problem of scheduling the agents according to the staffing requirement of the first step. They use the method of [Franx et al., 2006] to estimate the level of performance.

The methods developed in these articles are not applicable to small size call centers where the combinatorial side of the problem is predominant.

Our objective is to find the optimal number of servers in each group and the corresponding routing rule in order to meet a service quality target at the minimum

operating cost. Our service quality indicator is the proportion of lost calls if the system were a pure loss system (that is, when all operators are busy, callers do not have the opportunity to stay on hold). Although in practice almost all call centers provide the opportunity for callers to wait on hold, a loss model will provide a good approximation of the *relative* performance of different configurations in many situations. In particular, when callers are impatient and renege soon or when the call center wants to provide small waiting times to their callers, it has been shown that the loss probability is a very good indicator of the call center performance (see for example [Tabordon, 2002], [Chevalier et al., 2005]). [Whitt, 2006c] also finds that in a queueing model with abandonments, the rate of abandonments has a much smaller influence on performance than the arrival and/or processing rate. Note that these statements do not context the importance of including abandonments in queueing models.

Even for a small size center (less than 10 agents), with few different skills (3 or 4) the number of possible configurations is very large (several hundred millions with 3 skills and several hundred billions with 4 skills). In order to be able to find an optimal configuration it is necessary to compute bounds on a set of configurations. To estimate the performance of a configuration, we use an overflow model where arriving calls overflow from one agent group to another. The model we use is described in Section 2.2.2. The method for computing the performance in such a model is the Hayward approximation that is presented in Chapters 4 and 5. In the present chapter we go one step further and develop a method to optimize the staffing of a call center. In particular we show that it is possible to combine the hierarchical structure of the routing policy with the characterization of the call overflows to derive useful bounds that significantly speed up the search for an optimal configuration. Since we use a combinatorial optimization approach based on partial enumeration of the solution space, the size of the problems we can handle is limited.

6.3 PROBLEM FORMULATION

We call $\mathcal{I}$ the set of all different call types. We assume that the arrival process for calls of each type is a Poisson process with parameter λ_i ($i \in \mathcal{I}$). To each call type corresponds one skill for the agents: to answer a call of type i an agent needs to have the skill i. As agents can have multiple skills, we call a group of agents all the agents that have the same skill set. Formally, group j has the skills $\mathcal{S}^j \subseteq \mathcal{I}$ which means that all agents of group j can answer any call of type $i : \forall i \in \mathcal{S}^j$.

We suppose that the processing time for calls handled by all groups is exponentially distributed with rate μ. The assumption that service times are exponential is not so important for the loss probability as it is well known that the Erlang-B formula, on

which our approximation scheme is based, is valid for any service time distribution. Our simulation results suggest the insensitivity of the service time distribution holds even for the non-Poisson overflow processes; details of the simulation tests are not reported because they are beyond the scope of this chapter. Note that waiting times, on the other hand, are strongly influenced by the service time distribution. The other limitation of this hypothesis is that the service time distribution is identical for all groups and all call types. This is in fact not such a bad approximation, as there is little benefit to pool calls that would have very different processing time distributions. [van Dijk, 2004] and [van Dijk and van der Sluis, 2008] show that efficiency could actually decrease in such circumstances. In practice, we observe that call center operators try to pool call types that are relatively similar.

Since the optimal staffing depends on the routing policy, we analyze the two decisions together. For the routing policies, we restrict ourselves to a type of policy that is often used in practice because it is both simple and intuitive: the "hierarchical routing". It has been described in Section 2.2.2. Here again we choose this routing because it aims at keeping more flexible operators available for future uncertain demand.

In this context, we call $G(m)$ the set of all possible groups of level m[1].

$$G(m) = \{g : |\mathcal{S}^j| = m\}.$$

For analytical tractability we do not allow "horizontal" routing between two groups at the same level. Although a system with such "horizontal" routing would be efficient, when the routing probabilities are optimized it appears from our simulation section (see Section 6.6) that the gain is not very important.

The hourly cost of an agent is a function of the number of skills she has. We normalize the hourly cost of an agent with a unique skill to 1. An agent with m skills has a cost of $1 + (m-1)p$. This corresponds to a "premium" of p for each additional skill. This cost structure has been observed in several call centers we studied. Inpractice, the value of the premium depends on many factor such as the kind of skills required or the social context.

Our objective is to find the configuration that minimizes the total operating cost subject to a given service level constraint. We express the service level constraint in terms of the loss probability that the system would incur if calls could not wait at all. We choose this performance measure because it is much easier to compute than the expected waiting time or than a given percentile for the waiting times, i.e. a service level. As already mentioned, earlier work has shown that there is a very good correlation between these performance measures (especially for small loss probabili-

1 Recall that level m contains all the groups of agents that have m skills. See Figure 2.2 and Section 2.2.2.

ties/small waiting times). Another reason to handle such a system is for loss systems where waiting is not possible at all (for example paying services). Given the very high combinatorial complexity of the problem, it is important to have a performance measure that is not too hard to compute, since otherwise there is no hope to be able to find optimal solutions.

The maximum loss probability for any incoming call is β. This corresponds to the probability that an entering call will find no group with an available agent and will overflow from successive levels until it is finally overflown from the last level and thus rejected. We impose this loss probability constraint for each type of call to ensure an identical quality of service. We could just as easily impose a distinct loss probability constraint for each call type, but we must remember that the loss probability is only a proxy for the service performance of the call center. If it is clear that imposing the same loss probability should lead to a similar service quality for each type of call, so far we have no clue as to how a measure of service quality (e.g. average waiting time) would vary between the different call types as a function of the distinct loss probabilities for each call type.

We now introduce the MP formulation:

$$\min \sum_j c^j s^j \tag{6.1}$$

$$\mathrm{LP}_i(\mathbf{s}, \mathbf{q}) \leq \beta \quad \forall i \tag{6.2}$$

$$\sum_{j \in G(m): i \in \mathcal{S}^j} q_{imj} = 1 \quad \forall i, m \tag{6.3}$$

$$s^j \in \mathbb{N} \quad \forall j \tag{6.4}$$

Where:

- c^j is the server cost for pool j,
- s^j is the number of servers in pool j,
- $\mathrm{LP}_i(\mathbf{s}, \mathbf{q})$ is the proportion of type i calls that are lost,
- β is the maximum proportion of lost calls,
- q_{imj} is the proportion of type i calls routed to group j at level m,
- $\mathbf{s}, \mathbf{q}$ stand respectively for the vector of decision variables s^j and the matrix of decision variables q_{imj}.

The loss probability $\mathrm{LP}_i(\mathbf{s}, \mathbf{q})$ is estimated using Hayward's approximation, which is presented in Chapter 4 and adapted to multi-skill loss systems in Chapter 5.

In the model used in this chapter we consider that after each level all calls of the same type are first merged together and then, if need be, split again among the different pools handling this particular type of calls. When such a split is carried out, the different fractions are the decision variables q_{imj}. For the sake of simplicity we do not consider correlations between flows but the model can easily be adapted to account for correlations in a similar way to that proposed in Section 5.3.

In addition, we make the simplifying assumption that when a group handles different call types, the overflow probability and the peakedness of the resulting overflow is identical for each call type. Our simulation experiments presented in Sections 5.4 and 6.6 as well as those performed earlier in [Tabordon, 2002] suggest that this assumption does not create a large distortion in the approximated loss for the system.

6.4 FINDING AN OPTIMAL CONFIGURATION

Because we want to find the optimal staffing for call centers with small loads, it is likely that some of the groups of agents (especially the more flexible ones) will be of very small size. Therefore, it is important to take the integrality constraints into account. In all but the smallest instances the number of possible configurations is tremendously large. In order to be able to find an optimal configuration, we developed a Branch and Bound type search algorithm. In the following subsections we first show how it is possible to limit the search space in the branching process and then how the Hayward approximation scheme can be used to derive bounds that are useful to restrict the search even further. Finally we present two different search strategies in the Branch and Bound tree. The ideas implemented in this algorithm are quite similar to the ideas used in integer linear programming (see [Wolsey, 1998] for a good presentation of branch and bound techniques in combinatorial optimization).

6.4.1 *Branching*

In order to enumerate the different configurations, we gradually fix the sizes of the groups and the routing probabilities starting from the first level and then proceeding level by level until we reach the last level. The routing probabilities are enumerated with a stepsize of 0.1.

At level m, each group has m skills so there are $\binom{I}{m}$ possible groups at that level (I is the total number of call types). This can be a very large number and having operators in each of these groups will completely ruin the pooling gains. Because of the concave nature of the Erlang-B formula, it is always optimal to have the smallest number of

groups possible at any level (given that, for a given level, the cost is identical for all groups). To determine the number of groups that might be needed at level m, we note that we must have enough groups to keep I degrees of freedom to determine an adequate service level for each type of calls at that level. With k different groups at level m, we have km flows of call entering the different groups (remember that at level m all groups can handle m different call types). This gives $km - I$ independent decision variables because, for each call type, we know that the sum of the routing probabilities at level m should be 1 (see (6.3)). Practically, this removes I degrees of freedom. Additionally, we have k variables for the sizes of the groups. In total, we have $km - I + k = k(m+1) - I$ independent decision variables. This number should be larger than I in order to be able to decide independently on the service level for each call type. Note that we neglect the fact that two configurations yielding the same service levels for each call type could have overflows with a different peakedness. In practice with small configurations the difference in peakedness that could be observed for configurations with equivalent loss probabilities will only have a second order effect on the performance of the call center.

This implies that $k \geq 2I/(m+1)$. For any configuration with more than $M(m) = \lceil 2I/(m+1) \rceil$ groups it is possible to find a combination yielding the same service levels with a smaller number of groups. In our branching procedure we use this fact whenever at level m we have $M(m)$ groups with a positive number of agents; we immediately fix the number of agents to 0 for all other possible groups of that level (and for all the corresponding routing variables).

It can easily be checked that $M(m) \leq \binom{I}{m}$ for all values of m except the last level where only one group is present (fully cross-trained agents). Note that $M(I) = 2$ which is the number of groups that would be needed in order to be able to assign independent service levels for each call type at the last level: one would only need to route a portion of the flow to one of the two pools and the remainder to the second one to reach the desired service level. Such a configuration is beyond the scope of our model. It seems anyway that the diseconomies of scale linked to splitting a group in two would almost never make such a configuration economically interesting.

Remember that, in our approximation method for computing the loss probabilities, all different types of calls handled by a group have the same overflow probability. Since at the last level we can have only one group of (fully cross-trained) operators, the search space is significantly reduced. Indeed, the loss probability for a type of call is the product of its overflow probabilities across all levels. Consequently at the next to last levels we need only consider configurations with identical loss probabilities.

Example 6.1 *Figure 6.1 depicts the typical configuration one can expect for a system with three types of calls. As a result of $M(m) \leq \binom{I}{m}$, there are only two pools at the second level.*

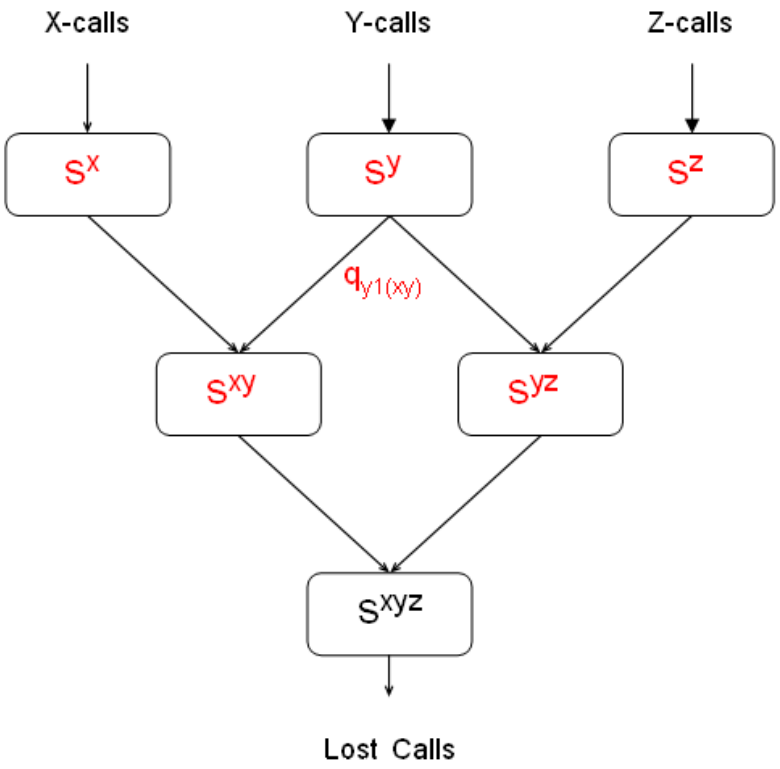

Figure 6.1: Structure of a configuration with three types of calls.

The number of decision variables is therefore reduced. They are displayed in red. s^{xyz} *is not a decision variable as it is the number of operators that would be necessary to meet the service requirement once the number of operators in the other pools is fixed. The only decision about the routing of flows is the proportion of y-calls routed to pool xy.*

6.4.2 *Bounding*

A node in our branch-and-bound search tree corresponds to fixing one of the decision variables. Fixing the routing of the y-calls is another node but we do not look for bounds at this node.

Given the decision variables that were fixed, we compute a lower bound on the cost of the possible configurations that would satisfy the service constraint. For this, we solve a relaxation of the problem where we suppose that :

- all remaining flows (overflows of agent groups and arrival processes) can be treated together without having to incur the cost of fully cross-trained agents. In other words, the remaining system is treated as a single pool;
- the service constraint has only to be satisfied globally for all call types and not individually for each one.

This relaxation gives a lower bound on the cost of a configuration at this node.

Example 6.2 *Continuing on Figure 6.1, the lower bound on a solution after fixing s^x would be the cost of fully cross-trained operators required to handle the overflow from pool x and the incoming flows of y- and z-calls according to the service level constraint, paid at the wages of dedicated operators. The lower bound after fixing s^x, s^y and s^z would be the cost of cross-trained operators required to serve the overflows of these three pools, paid as 2-cross-trained operators.*

Since we do not know how the different flows of calls will be combined, we must be careful in the way we treat the peakedness of the different flows. Remember that in the Hayward approximation the loss probability is computed by dividing the arrival rate and the number of servers in the Erlang-B formula by the peakedness. The concavity of the Erlang-B formula implies that for a given load and a given number of agents, a lower peakedness will always result in a smaller loss probability (or equivalently for a given loss probability and a given load, a lower peakedness will require a smaller number of agents). For this reason, we compute the lower bound in such a way that the peakedness is lower than it would be in reality. In practice here,

1. we start by "normalizing" the flows of calls, i.e. by dividing their rate by their peakedness.
2. We can then merge the different flows as if their peakedness was one.
3. Using the Erlang-B formula, we compute the number of servers needed to reach the desired loss probability. This is the global loss probability for all call types determined by dividing (i) the maximum rate of calls overflowing from the system when the service constraint is reached for all call-classes ($\beta \sum \lambda_i$) by (ii) the sum of the flow rates of all overflows at the current partial configuration.
4. The number of servers found is then divided between the different flows in proportion to their normalized rates.
5. All parts are multiplied by the peakedness of their corresponding flows. We finally obtain the lower bound by summing them.

When searching for the minimal number of agents to satisfy the global loss probability constraint, we use values for the number of servers that (after multiplying back by the different peakedness) give an integer number of agents.

This procedure determines a lower bound because we compute the number of servers needed with the full benefit of pooling (all types of calls are handled by a single group) but the peakedness of each stream of calls is kept separate. Indeed, when joining two flows of calls the peakedness is the weighted average of their peakedness. In the Hayward approximation the total flow is divided by this average peakedness. Dividing each flow by its own peakedness ("normalizing" as we call it in our procedure) is equivalent to joining the flows and computing the peakedness as the weighted harmonic average of the flows. Since the weighted harmonic average is always lower than the weighted (arithmetic) average and since a lower peakedness is always favorable, this will result in a smaller number of agents needed to reach a target loss rate.

Example 6.3 *To illustrate the bounding procedure, we present a small numerical example. Consider two flows, x ($\lambda_x = 3$, $z_x = 1.5$) and y ($\lambda_y = 5$, $z_y = 2$), that would be the overflows resulting from the groups that were already fixed. The required global loss probability (GLP) is* 0.2.

The first step to find the bound is to normalize the flows, i.e. to divide both flows by their peakedness. We obtain, for x, $\overline{\lambda}_x = \frac{\lambda_x}{z_x} = \frac{3}{1.5} = 2$ and, for y, $\overline{\lambda}_y = \frac{\lambda_y}{z_y} = \frac{5}{2} = 2.5$. The total normalized flow, $\overline{\lambda}_{tot}$, is thus equal to $\overline{\lambda}_x + \overline{\lambda}_y = 4.5$.

The next step is to determine the number of agents that are required. In order to do that we have to compute $\overline{s}$, the number of agents required to reach the required global loss probability with the normalized loads. After that, $\overline{s}$ is modified to get s, the equivalent number of operators that takes into account the different peakednesses. This is done by multiplying a proportion of $\overline{s}$ by z_x and the remaining part by z_y. This proportion corresponds to the weight of $\overline{\lambda}_x$ in the total normalized flow. Therefore, we have $s = (\frac{\overline{\lambda}_x}{\overline{\lambda}_{tot}}\,\overline{s})\,z_x + (\frac{\overline{\lambda}_y}{\overline{\lambda}_{tot}}\,\overline{s})\,z_y = (\frac{\overline{\lambda}_x}{\overline{\lambda}_{tot}}\,z_x + \frac{\overline{\lambda}_y}{\overline{\lambda}_{tot}}\,z_y)\,\overline{s} = (\frac{2}{4.5}\,1.5 + \frac{2.5}{4.5}\,2)\,\overline{s} = 1.78\,\overline{s}$. Note that 1.78 corresponds to the weighted harmonic average of the peakedness of the flows ($1.78 = \frac{\lambda_x+\lambda_y}{\lambda_x/z_x+\lambda_y/z_y}$).

However we wish to have s as an integer. This implies that we cannot take whatever value of $\overline{s}$. We should choose it accordingly. From $s = 1.78\,\overline{s}$, we immediately have $\overline{s} = \frac{1}{1.78}\,s = 0.5625\,s$. Consequently, in order to have s integer, we should choose for $\overline{s}$ to be a multiple of 0.5625.

Table 6.1 gives the loss probabilities for integer values of s. We see that with one operator we would have a global loss probability of 0.8965. By gradually increasing $\overline{s}$ by multiples of 0.5625 we find that $s = 10$ is the smallest value for which the global loss probability is smaller than 0.2. *$s = 10$ is therefore the value of the bound.*

s	1	2	3	4	5	6	7	8	9	10
$\bar{s}$	0.563	1.125	1.688	2.25	2.813	3.375	3.938	4.5	5.063	5.625
GLP	0.897	0.796	0.67	0.608	0.521	0.439	0.365	0.297	0.237	0.185

Table 6.1: Global loss probabilities as a function of s.

6.4.3 *An improved bound*

Until now we have not taken into account the cost of the different groups of agents. We can improve our lower bound by noting that for each flow there is a lower bound on the cost of agents that will handle the overflow. For example, if one of the merged flows is an overflow from a group at the second level, then, with the hierarchical routing policy, these calls can be handled in groups belonging to level 3 or higher. Since all agents at the m-th level have m different skills, they also all have the same cost. If we suppose that all calls from the overflow of level 2 are handled by agents with a level 3 cost this is a lower bound (as some calls might actually overflow again and be handled by agents in even higher levels).

In the preceding bounding procedure, after having multiplied the parts of operators by their peakedness, we also multiply these parts by the cost of agents at the next level. We then add up the costs of the different flows.

Example 6.4 *To illustrate this, let us continue Example 6.3.*
We calculated that the bound must be equal to 10 with $\lambda_x = 3$, $z_x = 1.5$, $\lambda_y = 5$, $z_y = 2$ *and with a required global loss probability of* 0.2*. We suppose now that x is a second level overflow and y a first level overflow and that the premium p is equal to 0.1. We wish to compute* $c(s)$*, the bound that takes the cost of the different groups into account.*
Remember that, in order to have $s = 10$ *as a bound, we needed* $\bar{s} = 5.625$ *(see table 6.1).* $c(s)$ *is also computed from* $\bar{s}$ *but, instead of computing* $s = (\frac{\overline{\lambda}_x}{\overline{\lambda}_{tot}}\,\bar{s})\,z_x + (\frac{\overline{\lambda}_y}{\overline{\lambda}_{tot}}\,\bar{s})\,z_y$*, we have to weigh each term with the corresponding cost. Therefore, we have* $c(s) = (\frac{\overline{\lambda}_x}{\overline{\lambda}_{tot}}\,\bar{s})\,z_x\,(1 + 2p) + (\frac{\overline{\lambda}_y}{\overline{\lambda}_{tot}}\,\bar{s})\,z_y\,(1 + p) = ([\frac{2}{4.5}\,1.5]\,1.2 + [\frac{2.5}{4.5}\,2]\,1.1)\,\bar{s}$*. With* $\bar{s} = 5.625$ *(this corresponds to* $s = 10$*), we have* $c(s) = 11.5125$*. It is the new bound.*
Note that this is a simplified example. If there really were only two flows of calls, one could assume that they can only be merged at the third level and use the cost of 1.2 for both flows. When there are more than two flows we obtain a lower bound by assuming that each call type will be handled by the first possible level. This is what is done in this little example.

6.4.4 *Branch and Bound tree search*

One of the questions that need to be solved when implementing a branch and bound approach is to choose a search strategy in the enumeration tree. The most common search strategies are depth first search (dfs) and a best bound search (bbs). The former proposes to explore the tree by selecting the node that has the most variables already fixed (in order to obtain as soon as possible a feasible solution that constitutes an upper bound). The latter suggests to select the node that has the best lower bound. The dfs strategy tends to minimize the number of active nodes at any time by improving the upper bound. The bbs strategy, on the other hand, tends to minimize the total number of nodes to explore by improving the lower bound. In order to try to combine the advantages of both approaches, we implemented a mixed strategy where we use bbs when fixing the number of operators for the groups of the first level. When exploring a node where all decision variables of the first level are fixed, we switch to dfs in order to find the best feasible solution corresponding to the given first level configuration.

6.5 NUMERICAL EXPERIMENTS

We have implemented our optimization algorithm in Java and tested it on a variety of small 3- and 4-skill examples. We implemented several programs that gradually include the different ideas of Section 6.4.

In the following subsections, we analyze the computational performance of our optimization procedure for 3- and 4-skill call centers. We then analyze the characteristics of the optimal configurations found on a sample of 100 randomly generated 3-skill examples.

6.5.1 *3-skill call centers*

For the 3-call type systems, there are five parameters to be chosen. The three first are the arrival rates for each call type (we normalize the time scale by fixing $\mu = 1$). For these three we considered five combinations. They are presented in Table 6.2. The fourth parameter is the value of the cost premium p and the fifth is the value of β, the maximum loss probability. We took three values for the premium, namely 0.1, 0.15 and 0.25 and three values for the loss probability, 0.05, 0.1 and 0.2. This gave us forty-five cases to analyze. We tried to make a balanced and representative choice. That is why we tested some cases in which all three flows are the same, other cases in which

two flows are similar while the third is quite different or cases where all flows are different.

Example	Flow X	Flow Y	Flow Z
1	10	2	1
2	4	4	4
3	8	7	3
4	3	2	1
5	5	4	2

Table 6.2: Description of the five combinations of flows considered for the 3-call type systems.

We solved these examples using various programs. The first one (R.Met) consists of an algorithm that makes an exhaustive search using the branching rule of Section 6.4.1 and only uses upper bounds to limit its search of possible configurations. The starting upper bound is the minimum between the cost of a configuration with only dedicated agents and a configuration with only fully cross-trained agents.

In the second one (L.Bou.), we include the bounding procedure of Section 6.4.2 with a dfs search strategy.

In the next program (N.Bou.), we use the same branching with the improved bound according to the method described in Section 6.4.3 with the same search strategy.

Finally, we tested all examples using a program (BBS) that uses the same branching and bounding strategy as (N.Bou) with our mixed search strategy presented in Section 6.4.4.

In order to check how effective the programs were, we measured the time it took to solve each instance using each program. The results are summarized in Figure 6.2. This figure shows, for each improvement of the algorithm, the number of examples for which the improvement in solution time lies in each percentage range compared to the exhaustive search method.

The results confirm that each successive improvement in our resolution method improves the speed to find the optimal configuration. The average time reductions are respectively 32%, 46% and 50% for the L.Bou, N.Bou and BBS programs (compared to the time to perform an exhaustive search).

We also evaluated the performance of the programs using another measure: we counted the number of configurations that had to be completely evaluated. The results were very similar to the time measures. That is why they are not presented here.

To evaluate our method, we also investigated the cost savings of our proposed solution compared to "naïve" configurations such as using either only fully cross-trained agents or only dedicated agents. To check this, we compared the optimal configuration obtained with the method proposed here to both extreme configurations, i.e. a

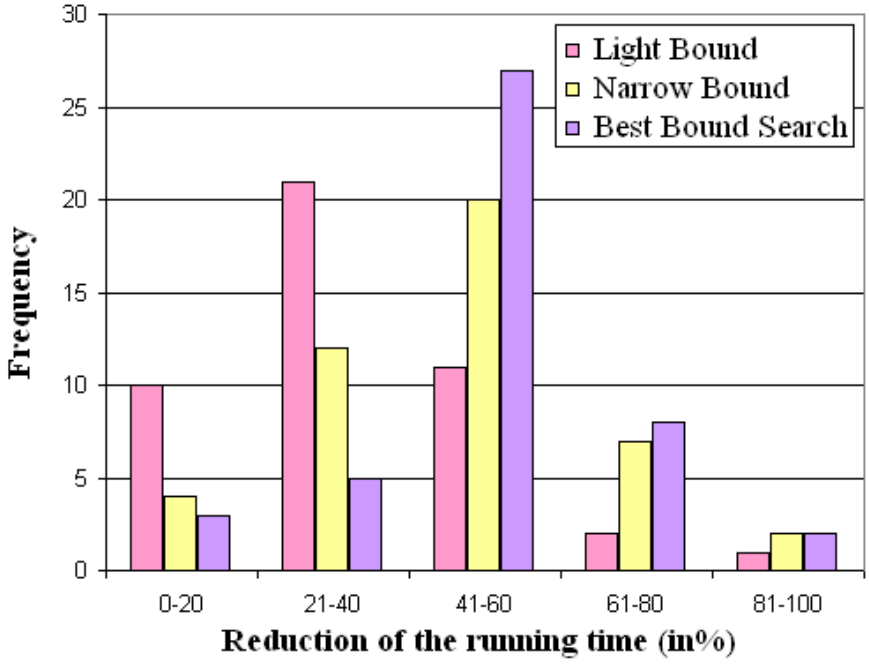

Figure 6.2: Time improvements brought by each algorithm improvement for the 3-call type systems.

configuration that satisfies the loss probability constraint with only dedicated operators (DED) and with only cross-trained operators (POL). Figure 6.3 shows the number of examples for each range of percentage cost improvement of the optimal solution compared to the two "naïve" policies. Over all 45 examples we observed an average cost reduction of 14% compared to the DED staffing and 11% compared to the POL staffing.

6.5.2 *4-skills call centers*

We now investigate the performance of our procedure for 4-call type call centers. As the benefit of the improved bound over the basic bound appeared almost constant in our 3-call type instance, we did not include the (L.Bou) program in our comparison. Because the number of possible configurations increases tremendously, we had to restrict ourselves to smaller loads than the loads chosen in the 3-call-type systems in order to keep the resolution times manageable (especially of the exhaustive search

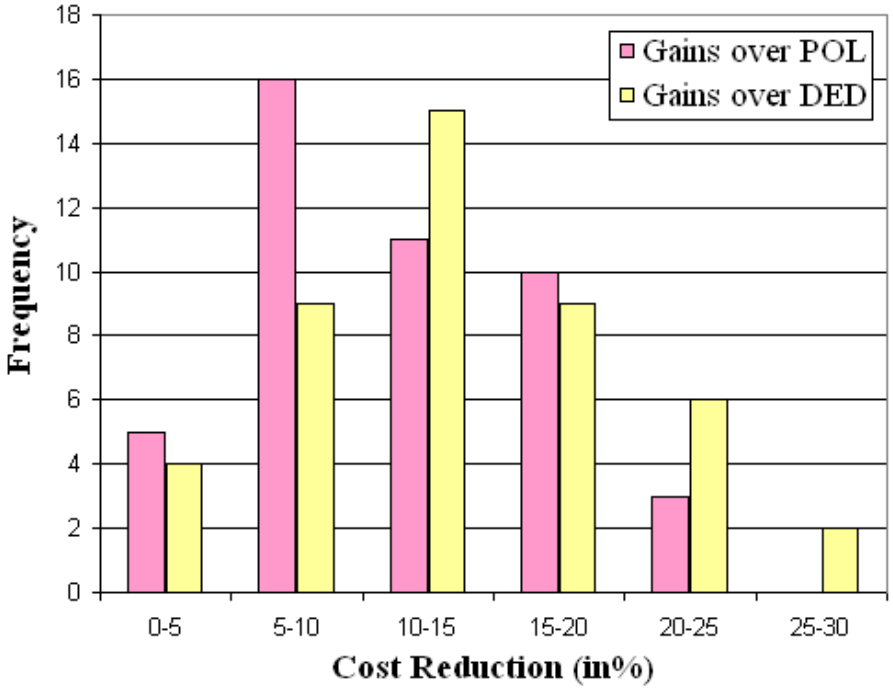

Figure 6.3: Percentage reduction of the total cost compared to configurations with only polyvalent operators (POL) or with only only dedicated operators (DED).

program). For the same reason, we only work with five examples. The parameters of each of them are presented in Table 6.3.

Example	Flow A	Flow B	Flow C	Flow D	Premium	β
1	1	1	1	1	0.05	0.05
2	2	1	1	1	0.05	0.05
3	3	2	1	1	0.05	0.05
4	0.5	4	1	0.5	0.15	0.05
5	2	1	1	1	0.2	0.05

Table 6.3: Description of the test set for the 4-call type systems.

As expected, the computation times were much longer than for the 3-call type systems. The results are presented in Table 6.4: the first column gives the time duration for the exhaustive search algorithm, the other ones present the time duration for each improvement in a normalized way.

Here, we see that our procedure really makes a huge difference for the computation time. We now have a reduction in computation time of at least 88% in all instances,

Example	Time	R.Met	N.Bou	BBS
1	6921172	100	2.97	3.06
2	8734740	100	3.16	3.15
3	29316375	100	3.71	3.78
4	21111376	100	12.39	11.48
5	33257332	100	7.08	4.91

Table 6.4: Time improvements brought by each algorithm improvement for the 4-call type systems.

and a reduction of more than 95% in all but one instance. However, it may look like the BBS program is not improving as much as, for the first three examples, the processing time is similar or even longer with BBS than with N.Bou. All the three first examples consider a relatively low premium (see Table 6.3). The consequence is that the optimal configuration in two of the three cases is a configuration with only fully cross-trained operators (see Table 6.5 where we may observe no cost decrease for these two examples). As the initial bound (the one used for starting the algorithm) is the minimum between the two extreme configurations (POL and DED), the algorithm begins with a bound that is the best configuration. The consequence is that, in this case, the mixed search strategy cannot find a better upper bound faster. This is supported by the fact that we did not observe any difference in the number of configurations evaluated using both methods. When we look at example 5, which is exactly the same as example 2 except for the higher premium, we see that the mixed search strategy still provides a non-negligible improvement.

Example	DED	POL	OPT	% gains
1	16.0	9.2	8.8	4.35
2	17.0	10.35	10.35	0
3	20.0	12.65	12.65	0
4	18.0	14.5	13.1	9.66
5	17.0	14.4	12.4	13.89

Table 6.5: Comparison of the staffing costs of the optimal and the DED and POL policies for the 4-call type systems.

Looking at Table 6.5, we see that, again, the optimal configuration brings important gains relative to both extreme configurations. These gains tend to be higher if the premium is large, as the comparison between examples 2 and 5 suggests.

6.5.3 *Analysis of optimal configurations for 3-skills call centers*

We used our optimization procedure to further investigate the structure of the optimal configurations. In order to do that, we randomly generated one hundred instances. In each of these instances, the incoming flows were randomly chosen in the interval between 0 and 10, the premium for cross-training laid between 0.05 and 0.3 and the loss probability could go from 0.01 to 0.25. For each case, the optimal configuration was found. When analyzing the results we could observe some interesting facts, mainly concerning the second level. As shown in Figure 6.4, the large majority of optimal configurations has only one pool of 2-cross-trained operators. In our sample, only four optimal solutions out of one hundred have a second pool in level 2. Moreover, that second pool consists, in all four cases, of only one operator, suggesting that the flow passing through it is minor. Actually, this is not surprising: it tends to show that dividing flows in multiple parts goes against optimality. This is coherent with the well-know phenomenon of economies of scale.

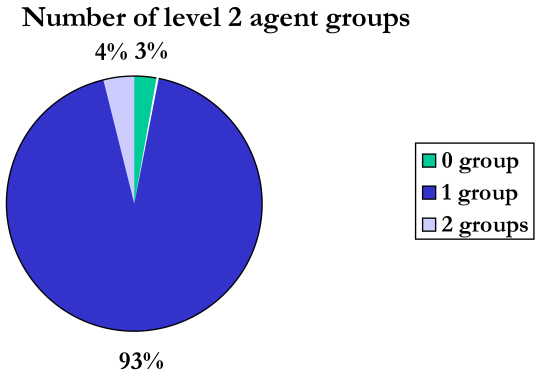

Figure 6.4: Percentage of cases with 0, 1 or 2 groups in the optimal solution at level 2.

A second important point to notice is that the 80%-20% rule stated in [Chevalier et al., 2005] does not appear to work in our case. As Figure 6.5 shows, the proportion of budget allocated to the dedicated operators is lower than one could expect. Still the general finding that "a little flexibility goes a long way" can be observed here again with an average of only about 20% of the staffing cost used for second level and third level operators. This could be compared for example with the observations of [Wallace and Whitt, 2005] where there is no cost for additional skills and operators all have the same number of skills. In that setting the authors find that most benefits from cross-training are obtained by having agents with 2 skills. Here we see that, when there is a cost for additional skills but it is possible to have agents with different levels of skills,

the number of additional skills in the best solutions is again very limited. [Jordan and Graves, 1995] and [Bassamboo et al., 2009] are other works that illustrate the benefits of a little flexibility in other types of models.

A closer analysis shows that there are huge differences between optimal configurations (see Figure 6.6) across our sample of instances. The proportion of dedicated operators, operators who can deal only one type of calls, may go from 0 to 100 percent. It provides actually a justification to the specific study of call centers of small size: the integrality condition may change the optimal solution a lot.

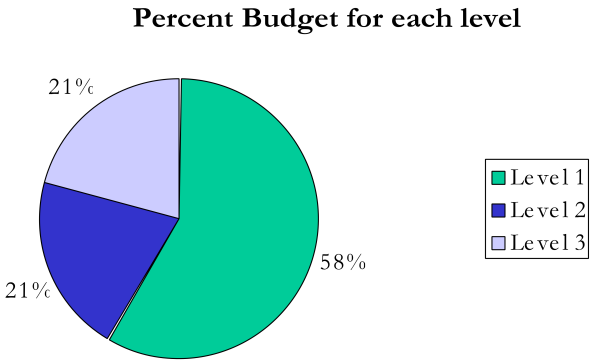

Figure 6.5: Optimal allocation of operators across the different levels.

6.6 SIMULATION STUDY

For analytical tractability many simplifying assumptions have to be made. We would now like to investigate their impact on the results we obtain. In order to do that, we ran a simulation experiment. First, we compare the loss probability computed using our model with the loss probabilities observed in the simulations. Second, we evaluate the relevance of our model for situations in which waiting is allowed. Finally, we check whether allowing horizontal routing makes a large difference.

6.6.1 *Validation of the results*

We start this simulation study by examining whether the loss probabilities estimated with our model match the loss probabilities observed in a simulation experiment. Because the objective of our model is to find the best configuration, we focus our investigation on how well the relative performance of different configurations is esti-

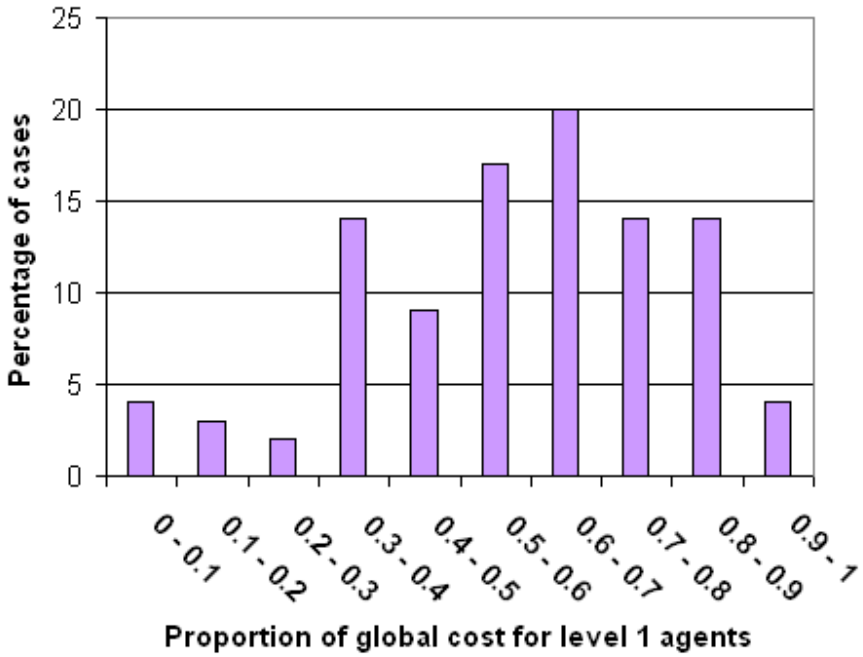

Figure 6.6: The optimal proportion of the total cost that is spent for level 1 agents.

mated. The test consists in taking a series of relatively similar configurations with the same combination of incoming flows. We took examples with 3 call types and arrival rates of 8, 7 and 3. We started with the optimal configuration with a cost premium of 0.25 and maximum loss probability of 0.1 found in Section 6.5.1. We performed an exhaustive search and selected 29 other configurations that, according to our model, give loss probabilities very close to 0.1 for each call type. In Figure 6.7 we compare the computed and simulated loss probabilities for those 30 examples.

This figure confirms that there is a very good agreement between the simulated and computed loss probabilities.

6.6.2 *Evaluation of the queueing approximation*

For analytical tractability we cannot explicitly consider queues. We argued in Section 6.3 that the results obtained with a loss model can be a good proxy for models with waiting. In other words, our idea is that the best configurations in the blocking model are also among the best configurations in a queueing model.

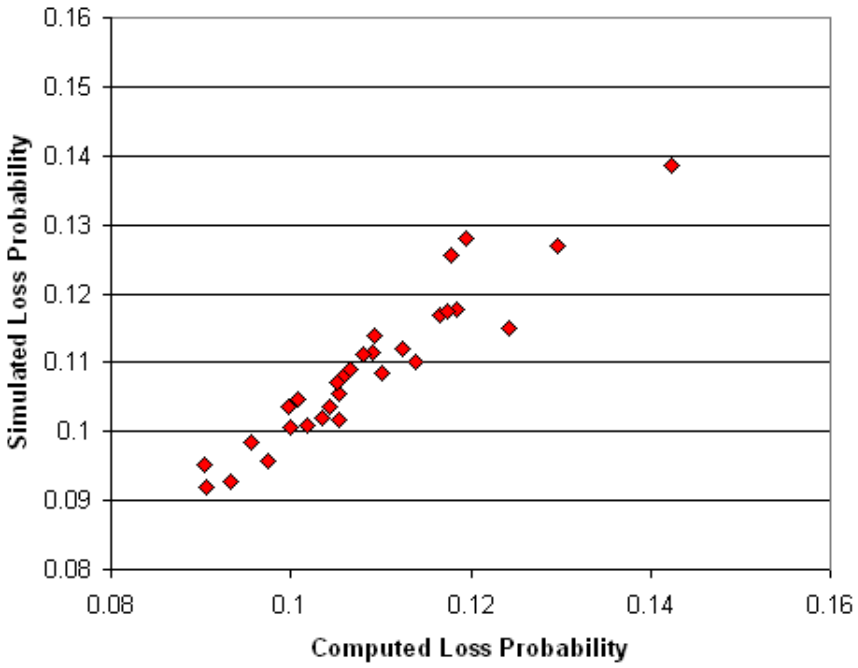

Figure 6.7: Correlation between computed loss probability and simulated loss probability.

In order to verify that, we took the set of thirty configurations we used in Section 6.6.1. We now compare the results we get when simulating the configurations with waiting and the results we compute using our model. For each configuration, we consider two cases.

Firstly, we introduce queues of limited sizes. More precisely we allow a maximum of two calls to be put on hold for each of the two first types of calls and a maximum of one call for the third flow. This gives queues of relatively comparable sizes proportional to the arrival rates. All callers are considered to have infinite patience. In the present case, it is possible to compare the computed loss probability to two performance measures. Because some calls are still lost we can compare it to the simulated loss probability as in Section 6.6.1. The second performance measure is the average waiting time. Figure 6.8 summarizes the results when the size of the queues is restricted.

From Figure 6.8 it appears that there is a remarkably strong correlation between the performance in both systems. Consequently, using the loss probability as a performance when waiting is possible is not bad at all.

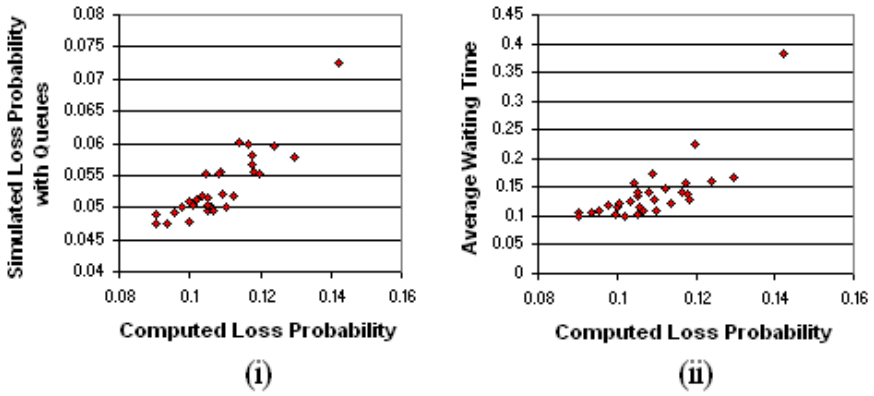

Figure 6.8: Correlation between computed loss probability and (i) simulated loss probability and (ii) average waiting time.

Secondly, we consider queues of infinite size. Again, callers never become impatient. The consequence is that a call is never lost and the blocking probability is thus equal to zero. That is why we only compare the computed loss probability to the average waiting time. Figure 6.9 summarizes the results when the size of the queue is unlimited.

The correlation is not as strong in this case. Still there clearly is a correlation. Moreover, one must keep in mind that the configurations we try to compare are configurations that have quite similar performances.

To conclude, it is clear that call centers where waiting is possible are much more prevalent than loss systems, yet a loss model can give good performance indications.

6.6.3 *Horizontal routing*

For analytical tractability in our model we do not allow horizontal routing. When flows have the possibility to be routed from one pool to another pool of the same level, the computation of the incoming flows at each pool becomes much more complicated. Deriving a bound for the Branch and Bound search would also be much harder.

From the results of Section 6.5.3 we could wonder whether the fact that very often there is only one group present at the second level could be a consequence of the strictly hierarchical routing. Maybe, with the possibility of horizontal routing between groups at the second level, it would be interesting to have more groups at that level?

The queueing simulation gives an indirect answer to the question already. Indeed, when waiting is allowed, a call will not be put on hold if there is a pool that can

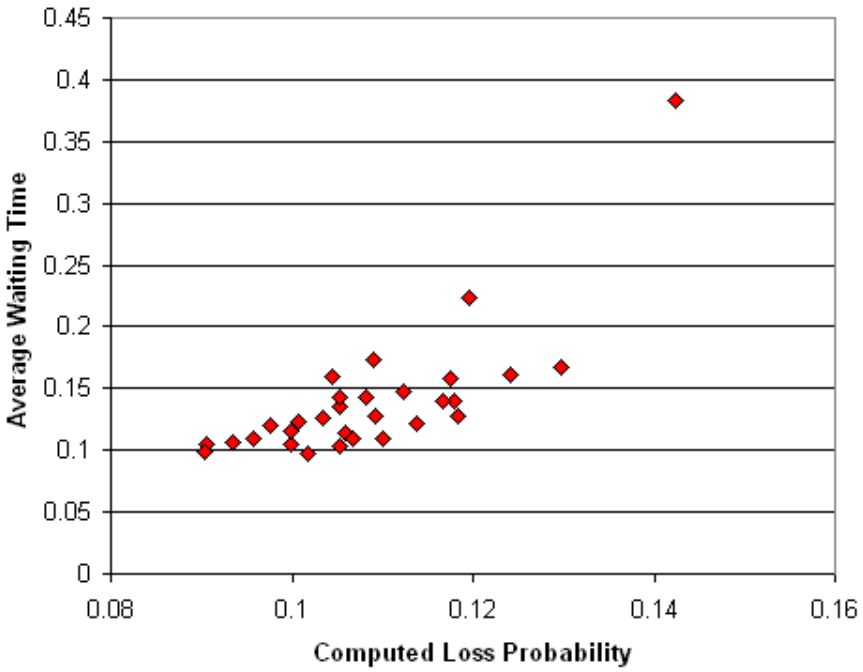

Figure 6.9: Correlation between computed loss probability and simulated loss probability with queues.

handle it so horizontal routing must be allowed. We saw that this did not prevent a very good agreement between the simulated and computed performance.

The following simulation experiment is meant to give a more direct answer to this question. We study 3 different cases of call centers with 3 call types, 2 cases with identical arrival rates for each call type and one case with one call type having a much smaller arrival rate than the two others. These seem to be the cases in which horizontal routing would be most beneficial. The justification would be for the equal arrival rates case that horizontal transfers could make a symmetrical configuration more attractive and for the case with a much smaller demand for one type that a second level server that can handle calls of the infrequent type would have spare capacity that is easier to use with horizontal transfers.

For each case, we simulated more or less all configurations that had a global loss probability close to 0.1 with not too large an imbalance between the 3 classes. The configurations, along with the simulated loss probabilities and the staffing costs are presented in Tables 6.6, 6.7 and 6.8.

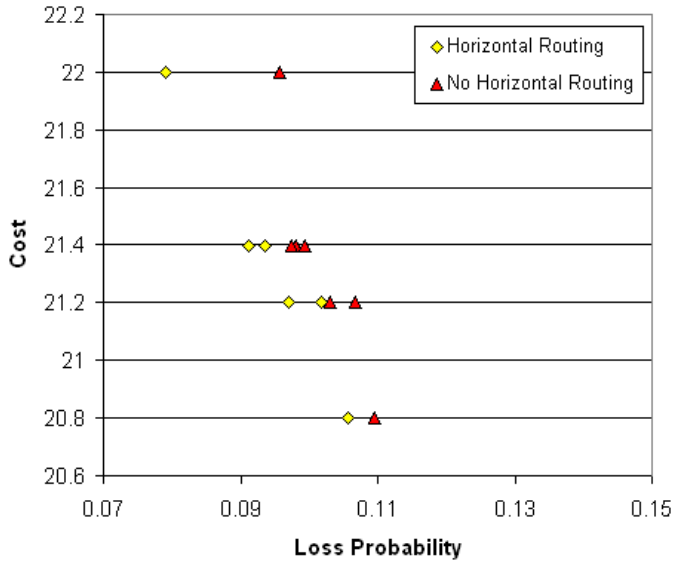

Figure 6.10: Cost vs average loss trade-off for policies with and without horizontal routing (example with arrival rates = 5 for each type).

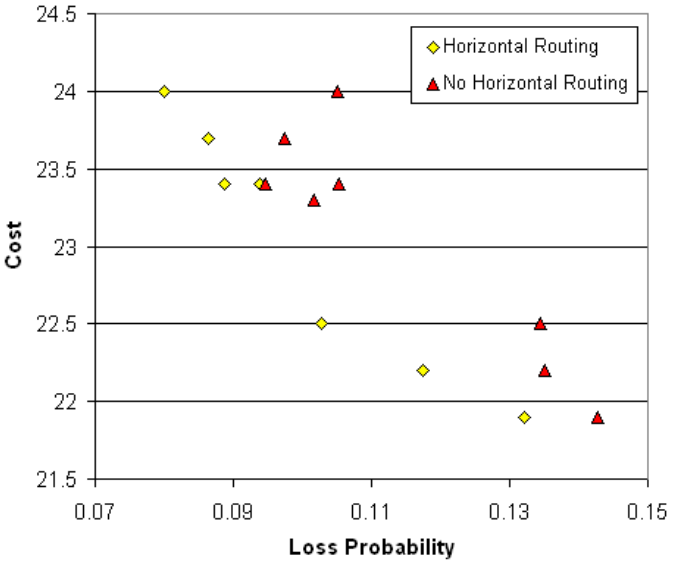

Figure 6.11: Cost vs average loss trade-off for policies with and without horizontal routing (example with arrival rates = 6 for each type).

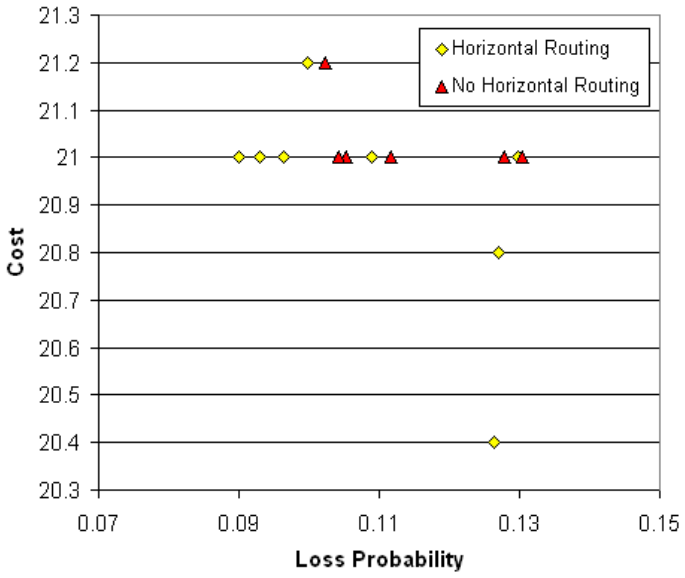

Figure 6.12: Cost vs average loss trade-off for policies with and without horizontal routing (example with arrival rates = {7,7,1} respectively).

In Figures 6.10, 6.11 and 6.12 we observe that horizontal routing reduces the loss probability indeed, especially when there are numerous operators in the second level. Generally speaking, the gains do not appear to be large: in order to get an important gain from horizontal routing we have to change the optimal configuration a lot to get a large number of second level operators and this results in a degradation of the performance that is seldom compensated by the horizontal routing. The fact that we have a discrete problem implies nevertheless that there are large jumps between configurations and that there are points where the difference can be significant.

6.7 CONCLUDING REMARKS

In this chapter, we presented a method to optimize small multi-skill call centers typical of B2B environments. We showed that in such environments, unless both the cross-training cost and the loads are very low, optimizing the configuration of the call center yields significant gains over simpler all dedicated or all cross-trained configurations. Moreover, we could observe that in settings with small loads, the combinatorics of the problem really influence the solution a lot.

To obtain an efficient optimization algorithm, we translated the main ideas of Branch and Bound techniques for a stochastic model. This seems a promising way to handle stochastic models where there are some discrete decision variables.

Our optimization procedure led us to the findings of Section 6.5.3. They tend to show an interesting property: it is in general optimal not to divide overflows in multiple parts. This property has, so far, only been empirically observed but it is quite logical: division of flow goes against the benefit of pooling. This property should be further investigated as it could lead to the development of faster heuristics to optimize the configuration of larger call centers.

Other ideas for improving the optimization procedure could be to study whether it would be possible to develop a tighter bound, or develop a more sophisticated search strategy. This would in turn make it possible to use this approach for larger call centers.

Other questions that remain to be studied are "how could non-exponential service times be handled?", "how to extend the routing policies?", "how to have a better estimation of the more interesting performance measures which are related to waiting times?". We provide some answers to the latter question in the next chapters.

Example	Sx	Sy	Sz	Sxy	Sxz	Syz	Sxyz	Horiz. Rout.	LPx	LPy	LPz	LPtot	Cost
1	4	4	4	2	2	2	2	1	0.0829	0.0744	0.0799	0.079	22
2	4	4	4	2	2	2	2	0	0.0942	0.092	0.1008	0.0957	22
3	5	5	5	1	1	1	2	1	0.0968	0.0917	0.0922	0.0936	21.4
4	5	5	5	1	1	1	2	0	0.1002	0.0926	0.1012	0.098	21.4
5	4	4	4	1	1	0	5	1	0.0967	0.0795	0.097	0.0911	21.4
6	4	4	4	1	1	0	5	0	0.1018	0.0988	0.0979	0.0995	21.4
7	5	5	6	1	0	0	3	0	0.1025	0.1061	0.0839	0.0975	21.4
8	6	4	6	1	1	0	2	1	0.0818	0.1278	0.0818	0.0971	21.2
9	6	4	6	1	1	0	2	0	0.0816	0.1465	0.0811	0.1031	21.2
10	5	6	5	1	1	0	2	1	0.1212	0.063	0.1208	0.1017	21.2
11	5	6	5	1	1	0	2	0	0.1256	0.0746	0.1199	0.1067	21.2
12	6	5	6	1	1	0	1	1	0.1038	0.1135	0.0997	0.1057	20.8
13	6	5	6	1	1	0	1	0	0.1045	0.1271	0.097	0.1095	20.8

Table 6.6: Comparison of configurations with and without horizontal routing when the incoming arrival rates are all equal to 5 and the cost premium is equal to 0.2.

Example	Sx	Sy	Sz	Sxy	Sxz	Syz	Sxyz	Horiz. Rout.	LPx	LPy	LPz	LPtot	Cost
1	2	2	2	4	4	4	4	1	0.0773	0.0811	0.0812	0.0799	24
2	2	2	2	4	4	4	4	0	0.1084	0.1064	0.1004	0.1051	24
3	4	4	4	1	1	1	7	1	0.0875	0.0848	0.087	0.0864	23.7
4	4	4	4	1	1	1	7	0	0.096	0.0982	0.0983	0.0975	23.7
5	4	4	4	3	3	0	4	1	0.0939	0.0694	0.103	0.0888	23.4
6	4	4	4	3	3	0	4	0	0.0989	0.0878	0.0973	0.0947	23.4
7	4	4	4	2	2	2	4	1	0.0948	0.0924	0.0944	0.0939	23.4
8	4	4	4	2	2	2	4	0	0.1054	0.1045	0.1061	0.1053	23.4
9	4	4	6	3	0	0	5	0	0.1043	0.1119	0.0885	0.1016	23.3
10	3	3	3	3	3	3	3	1	0.1049	0.1029	0.1004	0.1027	22.5
11	3	3	3	3	3	3	3	0	0.1316	0.1379	0.1341	0.1345	22.5
12	4	4	4	2	2	2	3	1	0.1161	0.1193	0.1171	0.1175	22.2
13	4	4	4	2	2	2	3	0	0.1329	0.1335	0.1392	0.1352	22.2
14	5	5	5	1	1	1	3	1	0.1304	0.1322	0.1341	0.1322	21.9
15	5	5	5	1	1	1	3	0	0.1388	0.1412	0.148	0.1427	21.9

Table 6.7: Comparison of configurations with and without horizontal routing when the incoming arrival rates are all equal to 6 and the cost premium is equal to 0.1.

Example	Sx	Sy	Sz	Sxy	Syz	Sxz	Sxyz	Horiz. Rout.	LPx	LPy	LPz	LPtot	Cost
1	7	7	2	0	1	1	2	1	0.1015	0.1071	0.0375	0.0998	21.2
2	7	7	2	0	1	1	2	0	0.108	0.1027	0.0602	0.1023	21.2
3	6	6	0	1	1	2	3	1	0.0978	0.0768	0.1267	0.0899	21
4	6	6	0	1	1	2	3	0	0.1104	0.0882	0.172	0.1042	21
5	6	6	0	1	3	0	3	1	0.1204	0.061	0.1246	0.093	21
6	6	6	0	1	3	0	3	0	0.1274	0.0839	0.0999	0.1053	21
7	5	5	0	2	3	3	1	1	0.0967	0.084	0.0976	0.0963	21
8	5	5	0	2	3	3	1	0	0.1306	0.1278	0.1485	0.1305	21
9	5	5	2	4	0	0	3	0	0.1068	0.1073	0.0665	0.1043	21
10	5	5	0	0	4	4	1	1	0.1136	0.0508	0.1125	0.1089	21
11	5	5	0	0	4	4	1	0	0.1087	0.1679	0.1065	0.1116	21
12	2	3	1	0	4	5	3	1	0.1377	0.1341	0.043	0.1297	21
13	2	3	1	0	4	5	3	0	0.1308	0.1312	0.0826	0.1278	21
14	3	3	1	0	4	4	3	1	0.1325	0.1339	0.0416	0.1271	20.8
15	3	3	1	0	4	4	3	0	0.1312	0.1349	0.0899	0.1302	20.8
16	4	4	1	0	3	3	3	1	0.1312	0.1323	0.0503	0.1263	20.4
17	4	4	1	0	3	3	3	0	0.1305	0.1294	0.0996	0.1279	20.4

Table 6.8: Comparison of configurations with and without horizontal routing when the arrival rates are equal to {7,7,1} respectively, and the cost premium is equal to 0.2.

7

A SAMPLE PATH APPROACH TO COOPER'S FORMULA

Our second application of Hayward's approximation is to use it for approximating the performance in multi-skill queueing systems. This is carried out in Chapters 8 and 9 in which we use the existing relationship between a single-skill loss system and the similar system with a queue to build similar relationships between a multi-skill loss system and the equivalent system with the addition of queues. Before that, in the present chapter, we analyze and propose an interpretation of the existing relationship in single-skill models.

The relationship which establishes an algebraic link between the Erlang-B and Erlang-C formulae has been mentioned in Section 4.3.6. For the sake of clarity we reproduce the relationship in (7.1), alongside the Erlang-B and Erlang-C formulae in (7.2) and (7.3), respectively.

$$C_E(s,a) = \frac{B_E(s,a)}{1 - a/s(1 - B_E(s,a))}, \tag{7.1}$$

$$B_E(s,a) = \frac{\dfrac{a^s}{s!}}{\displaystyle\sum_{i=0}^{s} \frac{a^i}{i!}}, \tag{7.2}$$

$$C_E(s,a) = \frac{\dfrac{a^s}{\overline{[(s-1)!(s-a)]}}}{\displaystyle\sum_{i=0}^{s-1} \frac{a^i}{i!} + \frac{a^s}{(s-1)!(s-a)}}, \tag{7.3}$$

where a is the offered load of the system, obtained by dividing the arrival rate λ by the service rate of an operator μ, $a = \lambda/\mu$. Let us remind that (7.2) is used to compute the probability for a call of being rejected upon its arrival to a M/M/s/s system and that gives the probability that a call has to wait before being serviced in a M/M/s model.

Because, as we show in Section 7.4, $(\frac{a}{s}(1 - B_E(s,a))$ in (7.1) must be smaller than one, an alternative expression for this relationship is

$$C_E(s,a) = B_E(s,a) \sum_{i=0}^{\infty} \left(\frac{a}{s}(1 - B_E(s,a))\right)^i. \tag{7.4}$$

This last expression is a geometric series with common ratio equal to $\frac{a}{s}(1 - B_E(s,a))$. It suggests that the introduction of a queue induces a series effect on the blocking probability. Intuitively, it is as if the service of delayed calls prevents the immediate service of other calls that would be serviced in the absence of a queue. The latter must therefore wait and this increases the proportion of calls that have to wait in the queue.

Although Formula (7.1) is widely referenced in the literature (see [Cooper, 1972], [Harel, 1988], [Koole et al., 2003] among others), to the best of our knowledge no research provides an interpretation of this algebraic connection. In this chapter we take an innovative "sample path" approach to the M/M/s and M/M/s/s models and use it to derive (7.1) in a way that corroborates our intuition and explains the mechanics that relate $B_E(s,a)$ and $C_E(s,a)$. Practically, we compare the behavior of the two models when they receive the same arrivals and when each operator processes a job in the same time in both models and we analyze the effect that the queue has on the service of the calls.

7.1 MODEL DESCRIPTION AND NOTATION

In this section we describe the main elements and the behaviour of the models we use and we present the related notation. The two models we use throughout this chapter are the M/M/s and M/M/s/s models. They are described in such a way that the situation in each model is kept as close as possible to the other. This facilitates the sample path analysis used in the next sections.

Let us define the incoming flow to a system as $\mathcal{C} = \{C_n, n \in \mathbb{N}_0\}$, where C_n is the arrival time of call n. We suppose that these calls arrive according to a Poisson process with rate λ and $\mathcal{C}$ is defined on the sample path $\{\Omega, \mathcal{F}, \mathcal{P}_\lambda\}$. Put differently:

$$\begin{cases} C_1 & \sim \quad \text{expo}(\lambda) \\ C_n - C_{n-1} & \sim \quad \text{expo}(\lambda) \quad \forall n \in \mathbb{N}_0 \setminus \{1\}. \end{cases}$$

Let us also define s different flows on the probability space $\{\Omega, \mathcal{F}, \mathcal{P}_\mu\}$. Each flow represents ends of processing, or service, times for one of the s operators in the system. Flow $i \in \{1, .., s\}$ is noted $\mathcal{D}_i = \{D_{i,m}, m \in \mathbb{N}_0\}$ where $D_{i,m}$ represents the time of the m-th end of service of operator i. We suppose that each flow $\mathcal{D}_i$ is a Poisson process

with rate μ. The arrivals are numbered from one to infinity and they form the set χ. Put another way:

$$\begin{cases} D_{i,1} & \sim \quad \text{expo}(\mu) \\ D_{i,m} - D_{i,m-1} & \sim \quad \text{expo}(\mu) \quad \forall m \in \mathbb{N}_0 \setminus \{1\}. \end{cases}$$

Let us now consider the set $\{c_n\}_{n=1}^{\infty} \in \mathcal{F}$, a realisation of $\mathcal{C}$ and the sets $\{d_{i,m}\}_{m=1}^{\infty}$ $\in \mathcal{F}$, $i \in \{1,..,s\}$, where $\{d_{i,m}\}_{m=1}^{\infty}$ is a realisation of $\mathcal{D}_i$.

Based on these sets, $\{c_n\}_{n=1}^{\infty}$ and $\{d_{i,m}\}_{m=1}^{\infty}$, $i \in \{1,..,s\}$, we construct two models. In both models $\{c_n\}_{n=1}^{\infty}$ represents the arrival process for a queueing system. These arrivals, also named calls, will be accepted or blocked by the system, depending on a condition that we define later in this section. If arrival $n \in \mathbb{N}_0$ is accepted, it is associated to, i.e. served by, an operator $i \in \{1,..,s\}$ until the next end of service occurs in the i-th end-of-service process. Note that we use i indifferently to designate the end-of-service process and the operator. Let us define $\sigma_i(t) = d_{i,m} - t, \quad d_{i,m-1} < t \leq d_{i,m}$ as the residual time in t until the next end-of-service time in process i. Because the exponential distribution has the memoryless property, we can prove that $\sigma_i(t)$ is exponentially distributed with parameter μ. The service time of arrival n is thus equal to $\sigma_i(c_n)$.

We define the binary function $b_i^j(t)$ that is equal to one if an arrival is associated to operator i at time t in model j and equal to zero otherwise.

From now on we are going to use t^- to denote the moment immediately before t. In short $\tau \leq t^- < t \, \forall \tau < t$. Depending on whether $b_i^j(d_{i,m}^-)$ is equal to 1 or 0 the end of service that will occur at that time will correspond to the exit of a serviced call or not. In general we say that there is an end of service at time $d_{i,m}^-$ but we will call this event a departure when $b_i^j(d_{i,m}^-) = 1$.

We now define $s_i^j(t)$ as the function defined on $\mathbb{N}$ that gives the reference of the call which is associated to operator i in model j at time t. This function takes the value 0 when no call is associated to operator i. $s_i^j(t)$ and $b_i^j(t)$ are related as

$$[s_i^j(t) = 0] \Leftrightarrow [b_i^j(t) = 0]$$

and

$$[s_i^j(t) > 0] \Leftrightarrow [b_i^j(t) = 1].$$

Let $S^j(t)$ be the function defined on $\{0,..,s\}$ that gives the number of operators that are busy in model j at time t. It is obvious that

$$S^j(t) = \sum_{k=1}^{s} b_k^j(t) \quad \forall j,t. \tag{7.5}$$

In both models the condition for call n to be accepted by the system is $\exists i \in \{1,..,s\}$: $s_i(c_n^-) = 0$. When a call n arrives and there is more than one operator i for which $s_i(c_n^-) = 0$, the call is associated to the operator with the most recent end of service. In other words n is associated to the operator for which $d_{i,m} = \max_{i\in\{1,..,s\},m\in\mathbb{N}_0}(d_{i,m} : d_{i,m} < c_n \leq d_{i,m+1}, s_i(c_n) = 0)$.

The function $y^j(t)$ defined on $\{0,..,s\}$ takes the value i at time t if at t operator i is the available operator with the most recent end of service. Note that $y^j(t) = 0$ means that no operator is available at time t in model j but we usually use $S^j(t) = s$ to describe this type of event. Note also that for any call n that is serviced immediately, $y^j(c_n^-)$ gives the reference of the operator that is going to serve n in model j. Observe finally that

$$[y^j(t) = i] \Rightarrow [b_i^j(t) = 0].$$

The function $l_i(t)$ defined on $\mathbb{R}^+$ gives the last end-of-service time observed in $\{d_{i,m}\}_{m=1}^{\infty}$ before time t. Formally,

$$l_i(t) = \max_{m\in\mathbb{N}_0}(d_{i,m} : d_{i,m} < t).$$

We call the first of the two models LM[1]. In LM, each call n for which $S^{\mathrm{LM}}(c_n^-) = s$ is rejected, i.e. it is not answered and considered lost to the system. LM is in fact a M/M/s/s model. In the second model, noted QM[2], any call n for which $S^{\mathrm{QM}}(c_n^-) = s$ is put on hold and waits until time t when $S^{\mathrm{QM}}(t) < s$. In other words n waits until an operator becomes available. This model is a M/M/s model. In LM, blocked calls are said to be rejected whereas in QM blocked calls are said to be delayed. We now define the following subsets of calls:

- $\mathcal{A}^j$ is the subset of arrivals n that are serviced immediately in model j.
- $\mathcal{R}$ is the subset of arrivals n that are rejected in LM, because $S^{\mathrm{LM}}(c_n^-) = s$.
- $\mathcal{W}$ is the subset of arrivals n that are put on hold in QM, because $S^{\mathrm{QM}}(c_n^-) = s$.

Because the number of arrivals and services is not bounded, we cannot really compute the size of any of those sets. We solve this problem by defining that the size of χ; the

1 LM is the abbreviation of Loss Model.
2 QM for Queueing Model.

set of all arrivals, is normalized to 1. By doing this we ensure that the size of a set corresponds to the proportion of calls that belong to this set.

When we need to restrict our attention to the calls in the system at time t, we use $\mathcal{A}^j(t)$, $\mathcal{R}(t)$ and $\mathcal{W}(t)$, respectively. As any call is either handled immediately or rejected in LM, $\mathcal{A}^{\text{LM}}$ and $\mathcal{R}$ are disjoint, i.e.

$$\mathcal{A}^{\text{LM}} \cap \mathcal{R} = \emptyset$$

and

$$\mathcal{A}^{\text{LM}} \cup \mathcal{R} = \chi.$$

For similar reasons, we have the same type of relationship between the subsets of calls defined on QM:

$$\mathcal{A}^{\text{QM}} \cap \mathcal{W} = \emptyset$$

and

$$\mathcal{A}^{\text{QM}} \cup \mathcal{W} = \chi.$$

In QM the discipline in the queue when there is more than one call waiting can be of any kind but it is usually assumed to be FCFS[3]. Independently of the priority rule in the queue, that does not matter in our present analysis, we suppose that all operators are busy when there are waiting calls. We shall thus not meet situations where a call n waits at time t whereas an operator i is available at the same time. Formally, we have

$$[|\mathcal{W}(t)| > 0] \Rightarrow [S^{\text{QM}}(t) = s]. \tag{7.6}$$

Calls in $\mathcal{W}$ are serviced when an operator becomes available. This type of event occurs when there is an end of service, more precisely a departure, in one of the s end-of-service processes. This event is described as:

$$\forall n \in \mathcal{W}, \quad \exists i \in \{1, .., s\}, \exists m \in \mathbb{N}_0 : s_i^{\text{QM}}(d_{i,m}^-) \neq n,\ s_i^{\text{QM}}(d_{i,m}^-) \neq 0, s_i^{\text{QM}}(d_{i,m}) = n. \tag{7.7}$$

Finally, it is important to note that in the coming sections we assume that the stability condition of the M/M/s model, i.e. $\lambda < s\mu$ holds. We explain in Section 7.5 how this condition is included in our approach.

3 Discipline in the queue does not affect on the performance of the operators and the system in general. Only the distribution of the waiting time depends on the chosen priority rule. In case of FCFS, the latter distribution is exponential with parameter $s\mu - \lambda$, λ being the arrival rate and μ the service rate.

7.2 A FEW PROPERTIES AND THE NOTION OF DISRUPTION

Formula (7.4) shows that the proportion of calls that are required to wait in a M/M/s model is higher than the proportion of calls that are rejected in the equivalent M/M/s/s model. In other words, $|\mathcal{R}| < |\mathcal{W}|$.

In this section we show that if both models behave according to the rules described in Section 7.1 then all calls that are rejected in LM have to wait in QM. In other words $\mathcal{R} \subseteq \mathcal{W}$. After that, we are going to focus on the remaining delayed calls. They all must belong to $(\mathcal{W} \cap \mathcal{A}^{\text{LM}})$ as they are calls that would have been serviced immediately in the absence of a queue.

Let us start by showing that $\mathcal{R} \subseteq \mathcal{W}$. This requires to first present a few interesting properties. They are all consequences of the rules described in the previous section.

Proposition 7.1

$$b_i^{QM}(t) \geq b_i^{LM}(t), \quad \forall i, t. \tag{7.8}$$

This means that if at a given time t an operator is busy serving a call in LM, then this operator is also busy at the same time t in QM.

Proof. We prove this result by assuming that there exist values of i and t such that $b_i^{\text{QM}}(t) < b_i^{\text{LM}}(t)$. If so, amongst the values of t there is a value τ such that $\tau = \min_{i\in\{1,..,s\},t\in\mathbb{R}^+}(t : b_i^{\text{QM}}(t) < b_i^{\text{LM}}(t))$. In other words τ is the first time when there is an operator i^* for which $b_{i^*}^{\text{QM}}(\tau) < b_{i^*}^{\text{LM}}(\tau)$. Formally, the following must be true

$$\exists i^* \in \{1,..,s\} : b_{i^*}^{\text{QM}}(\tau) < b_{i^*}^{\text{LM}}(\tau),\, b_i^{\text{QM}}(\tau^-) \geq b_i^{\text{LM}}(\tau^-) \quad \forall i \in \{1,..,s\}. \tag{7.9}$$

There are only two ways for (7.9) to hold:

- in τ^-, $b_{i^*}^{\text{QM}}(\tau^-) = b_{i^*}^{\text{LM}}(\tau^-) = 1$ and i^* becomes available in QM and not in LM. Because ends of service occur at the same epochs $d_{i,m}$ in both models, operator i^* must immediately start serving a call in LM and not do so in QM. Obviously, this is not possible as there is no queue in LM.

- $b_{i^*}^{\text{QM}}(\tau^-) = b_{i^*}^{\text{LM}}(\tau^-) = 0$ and there is an arrival in τ in both models. This call is serviced by i^* in LM, but not in QM. According to the allocation rule for the calls that are serviced immediately, the former means that $y^{\text{LM}}(\tau^-) = i^*$. Because arrivals occur at common epochs $\{c_n\}_{n=1}^{\infty}$ in both models, the only way for the call not to be associated to i^* in QM is to have $y^{\text{QM}}(\tau^-) \neq i^*$. There must therefore exist in QM an operator $i' \neq i^*$ with $b_{i'}^{\text{QM}}(\tau^-) = 0$, for whom

$l_{i^*}(\tau) < l_{i'}(\tau)$ and who is not available in LM, i.e. $b_{i'}^{\text{LM}}(\tau^-) = 1$. Therefore in $\tau^- < \tau$ we have $b_{i'}^{\text{QM}}(\tau^-) < b_{i'}^{\text{LM}}(\tau^-)$, which cannot be, according to the assumptions.

This proves Proposition 7.1. ■

This result also implies that any operator available at time t in QM is available at the same time in LM. Another implication of Proposition 7.1 is:

Corollary 7.1

$$S^{LM}(t) \leq S^{QM}(t), \quad \forall t.$$

Proof. This is a direct consequence of Proposition 7.1 and of Formula (7.5). ■

We can now show that all rejected calls in LM have positive waiting times in QM.

Proposition 7.2

$$\mathcal{A}^{QM} \subseteq \mathcal{A}^{LM} \tag{7.10}$$

and

$$\mathcal{R} \subseteq \mathcal{W}. \tag{7.11}$$

Proof. Let us first prove (7.10). We suppose that $\exists n \in \mathcal{A}^{\text{QM}} : n \notin \mathcal{A}^{\text{LM}}$. From (7.6), we must have $S^{\text{QM}}(c_n) < s$. We also know from the description of LM in Section 7.1 that $[n \notin \mathcal{A}^{\text{LM}}] \Leftrightarrow [S^{\text{LM}}(c_n) = s]$. We immediately deduce $S^{\text{QM}}(c_n) < S^{\text{LM}}(c_n)$, which contradicts Corollary 7.1. (7.10) is verified. (7.11) follows immediately as $\mathcal{A}^{\text{QM}} \cup \mathcal{W} = \mathcal{A}^{\text{LM}} \cup \mathcal{R} = \chi$ and as $\mathcal{A}^{\text{QM}} \cap \mathcal{W} = \mathcal{A}^{\text{LM}} \cap \mathcal{R} = \emptyset$. ■

The calls in $\mathcal{R}$ form a portion of the calls with positive waiting times. The other waiting calls are calls that are serviced immediately in LM. They form the set $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$. We now show that $\mathcal{R}$ and $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ are mutually exclusive and that there is no call of $\mathcal{W}$ outside these two subsets.

Proposition 7.3

$$\mathcal{W} = \mathcal{R} \cup (\mathcal{A}^{LM} \cap \mathcal{W})$$

and

$$\mathcal{R} \cap (\mathcal{A}^{LM} \cap \mathcal{W}) = \emptyset.$$

Proof. The first statement is easily proven with basic manipulations on the sets of calls.

$$\begin{aligned} \mathcal{W} &= \chi \cap \mathcal{W} \\ &= (\mathcal{R} \cup \mathcal{A}^{\text{LM}}) \cap (\mathcal{R} \cup \mathcal{W}) \\ &= \mathcal{R} \cup (\mathcal{A}^{\text{LM}} \cap \mathcal{W}), \end{aligned}$$

where $\mathcal{W} = \mathcal{R} \cup \mathcal{W}$ is a consequence of Formula (7.11).

The second statement directly follows from $\mathcal{A}^{\text{LM}} \cap \mathcal{R} = \emptyset$. ■

The calls of $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ are blocked as a result of the introduction of a queue in the model. They see all operators busy upon their arrival in QM. Formally,

$$[o \in (\mathcal{A}^{\text{LM}} \cap \mathcal{W})] \Leftrightarrow [S^{\text{LM}}(c_o) < s, S^{\text{QM}}(c_o) = s].$$

This means that other calls occupy the operators that should answer them, according to the situation in LM.

Our next objective is to identify which call is responsible for the delay of each call in $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$. This requires us to first present an interesting property that follows from our description of the model.

Proposition 7.4

$$\forall n \in \mathcal{A}^{QM},\ y^{QM}(c_n) = y^{LM}(c_n).$$

In other words, any call that is serviced immediately in QM is answered in LM and it is associated to the same operator in both models.

Proof. Because we know from Proposition 7.2 that $\mathcal{A}^{\text{QM}} \subseteq \mathcal{A}^{\text{LM}}$, we have

$$\forall n \in \mathcal{A}^{\text{QM}}, \exists i \in \{1, .., s\} : s_i^{\text{LM}}(c_n) = n.$$

Consequently, for this same operator i, we have $y^{\text{LM}}(c_n^-) = i$. Suppose now that n is not associated to i in QM. Put another way, suppose that $y^{\text{QM}}(c_n^-) \neq i$. This situation is possible only if one of the following two conditions holds: either $b_i^{\text{QM}}(c_n^-) = 1$ or $b_i^{\text{QM}}(c_n^-) = 0$ and $\exists i' \neq i : l_i(c_n) < l_{i'}(c_n)$, $b_{i'}^{\text{QM}}(c_n^-) = 0$. The latter means that there exists in QM another available operator for which an end of service occurs in $[l_i(c_n); c_n)$ in the associated end-of-service process.

- Let us investigate the first condition. Because $n \in \mathcal{A}^{\text{QM}}$, we have $S^{\text{QM}}(c_n^-) < s$. As $b_i^{\text{QM}}(c_n^-) = 1$, $\exists i' \neq i :\ b_{i'}^{\text{QM}}(c_n^-) = 0$. From Proposition 7.1, we have $b_{i'}^{\text{LM}}(c_n^-) = 0$ and because operator i is the one chosen in LM to serve n, we also have $l_i(c_n) > l_{i'}(c_n)$. Let us assume $l_i(c_n) = \tau$. If $b_i^{\text{QM}}(c_n^-) = 1$, then one of the two following events had to occur: either an arriving call n' has been associated to i in the interval $[\tau; c_n^-)$, or i started serving in τ a call of $\mathcal{W}(\tau)$. We can assume without loss of generality that no arriving call was attributed to i in $[\tau; c_n^-)$: if this happens, the situation for call n' would be identical to the current situation of n and one only needs to remplace n by n' and apply our whole argument to n'. In the second case, we would have $\mathcal{W}(\tau^-) > 0$. As we also have $b_{i'}^{\text{QM}}(\tau^-) = 0$, we are contradicting (7.6). The first condition cannot be fulfilled.

- We now show that the second condition, where we suppose that $\exists i' \neq i : l_i(c_n) < l_{i'}(c_n)$, $b_{i'}^{\text{QM}}(c_n^-) = 0$, cannot be true. Because ends of service are simultaneous is both models, the fact that $y^{\text{LM}}(c_n^-) = i$ whereas $l_i(c_n) < l_{i'}(c_n)$ implies that $b_{i'}^{\text{QM}}(c_n^-) = 1$. This contradicts Proposition 7.1.

As none of the two conditions can be fulfilled, we conclude that $y^{\text{QM}}(c_n^-) = i$ and this proves the proposition. ■

Let us now focus again on the calls in $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ and continue the identification of the calls responsible for their blocking in QM. We can deduce from Proposition 7.4 that

$$[o \in (\mathcal{A}^{\text{LM}} \cap \mathcal{W})] \Leftrightarrow [y^{\text{LM}}(c_o^-) = i, b_i^{\text{QM}}(c_o^-) = 1].$$

This equivalence basically says that a call of $\mathcal{A}^{\text{LM}}$ has to wait in QM if and only if the operator that serves him in LM is not available in QM. Consequently we can also write that

$$\forall o \in (\mathcal{A}^{\text{LM}} \cap \mathcal{W}), \exists n \in \chi : y^{\text{LM}}(c_o^-) = i, s_i^{\text{QM}}(c_o) = n.$$

In this case we say that "n disrupts o", that we note $o \in \mathcal{G}_n$, where $\mathcal{G}_n$ is defined as the set of calls that are disrupted by n.

Definition 7.2 *Let $o \in \mathcal{A}^{LM}$, $n \in \chi$ and let i be the operator such that $s_i^{LM}(c_o) = o$ then*

$$[s_i^{LM}(c_o) = o, s_i^{QM}(c_o) = n] \Leftrightarrow [o \in \mathcal{G}_n].$$

This definition permits to associate to each call in $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ one and only one other call responsible for its delay. Before commenting it, let us present a few important implications of this definition. The first proposition states that a call disrupts other calls only if it had to wait before being serviced.

Proposition 7.5 *If $o \in \mathcal{G}_n$, then $n \in \mathcal{W}$.*

Proof. Let i be the operator that answers call o in LM and call n in QM, i.e. $s_i^{\text{LM}}(c_o) = i$ and $s_i^{\text{QM}}(c_o) = n$. Let us assume that $n \in \mathcal{A}^{\text{QM}}$. Then, $y^{\text{QM}}(c_n^-) = i$. From Proposition 7.2, we must also have $n \in \mathcal{A}^{\text{LM}}$. Because of Proposition 7.4, $y^{\text{QM}}(c_n^-) = y^{\text{LM}}(c_n^-) = i$ and $s_i^{\text{LM}}(c_o) = n$ must hold too. This is obviously not possible. Therefore $n \in \mathcal{W}$, which is precisely our proposition. ■

Note that a direct consequence of this proposition is that

$$\forall n \in \mathcal{A}^{\text{QM}}, \mathcal{G}_n = \emptyset.$$

With Proposition 7.5, we know the exact set of calls that will cause disruptions with a strictly positive probability. It reduces the field of analysis in the next propositions.

Proposition 7.6

$$[o \in (\mathcal{R} \cup \mathcal{A}^{QM})] \Leftrightarrow [\nexists n \in \mathcal{W} : o \in \mathcal{G}_n]. \tag{7.12}$$

Proof. We first prove $\Rightarrow$. If $o \in \mathcal{A}^{QM}$, we know from Proposition 7.4 that o is serviced by the same operator in LM and in QM. Suppose operator i serves o. Then $s_i^{LM}(c_o) = s_i^{QM}(c_o) = o$. Consequently, $\nexists n \in \mathcal{W} : s_i^{QM}(c_o) = n$. This proves that $\nexists n \in \mathcal{W} : o \in \mathcal{G}_n$ when o in $o \in \mathcal{A}^{QM}$. If $o \in \mathcal{R}$ then $\nexists i \in \{1, .., s\} : s_i^{LM}(c_o) = o$. We immediately deduce from Definition 7.2 that $\nexists n \in \mathcal{W} : o \in \mathcal{G}_n$.

Implication $\Leftarrow$ is proven by contradiction. Suppose $o \in (\mathcal{W} \cap \mathcal{A}^{LM})$. Because $o \in \mathcal{A}^{LM}$, There must exist an operator i such that $y_i^{LM}(c_o^-) = i$. We also know from Definition 7.2 that $[\nexists n \in \mathcal{W} : o \in \mathcal{G}_n] \Rightarrow [\nexists n \in \mathcal{W} : s_i^{QM}(c_o) = n]$. In the mean time, because Proposition 7.5 tells us that if a call disrupts o, the former must be in $\mathcal{W}$, we deduce that no call from $\mathcal{A}^{QM}$ disrupts o and therefore no call from $\mathcal{A}^{QM}$ is associated to operator i at time c_o in QM. As $\mathcal{A}^{QM} \cup \mathcal{W} = \chi$, we immediately deduce that no call is associated to i at time c_o^- in QM. Operator i is thus available in c_o^- so o can be serviced in c_o, implying $o \notin \mathcal{W}$. This is clearly a contradiction. (7.12) is verified. ■

Corollary 7.3

$$[o \in (\mathcal{A}^{LM} \cap \mathcal{W})] \Leftrightarrow [\exists n \in \mathcal{W} : o \in \mathcal{G}_n]. \tag{7.13}$$

Proof. Because $\mathcal{R} \cap \mathcal{A}^{LM} = \mathcal{W} \cap \mathcal{A}^{QM} = \emptyset$ and $\mathcal{R} \cup \mathcal{A}^{LM} = \mathcal{W} \cup \mathcal{A}^{QM} = \chi$, (7.13) is the contraposition of (7.12) and it is thus true. ■

This result indicates that the calls that are additionaly blocked in comparison to the situation in LM are blocked because of a disruption. Practically, any call that incurs a delay is likely to take the place of future arriving calls when it is finally serviced. This would further increase the number of calls with positive waiting times.

We can consequently present an intuitive interpretation of "disruption" in the single-skill models used here: a disrupted call is a call that is serviced in LM but is blocked and must wait in QM. We will see however in Chapter 8 that this interpretation does not apply in models where there is more than one type of call and several pools of operators. In these cases a call can be serviced without delay despite being disrupted in the sense of Definition 7.2.

As presented previously, our intuition is that allowing the calls rejected in LM to be serviced after some delay blocks the immediate service of calls that would be serviced without delay otherwise and this amplifies the proportion of delayed calls. This intuition remains coherent with our results so far. In order to verify that the amplification factor is correct we are now going to characterize $E[|\mathcal{G}_n|]$, the amount

of calls that are disrupted because of call n. After that, we are going to focus on the amplification itself.

The computation of $E[|\mathcal{G}_n|]$ is much simplified by the next proposition that shows that $|\mathcal{G}_n| \leq 1$.

Proposition 7.7

$$|\mathcal{G}_n| \leq 1, \forall n \in \mathcal{W}$$

In other words, any waiting call disrupts at most one other call when being served.

Proof. We know from (7.7) that the service of a waiting call begins immediately after the departure of a serviced call. Without loss of generality suppose that n is serviced at time $d_{i,m}$. This implies that $s_i^{\text{QM}}(t) = n, \forall t \in [d_{i,m}; d_{i,m+1})$. Suppose also that o is the first call to be disturbed by n. From Definition 7.2, we immediately see that $[o \in \mathcal{G}_n] \Rightarrow [c_o \in [d_{i,m}; d_{i,m+1})]$. It follows that $s_i^{\text{LM}}(t) = o, \forall t \in [c_o; d_{i,m+1})$. Therefore, no call k such that $c_o \leq c_k < d_{i,m+1}$, i.e. no call that arrives in the time interval $[c_o; d_{i,m+1})$, will be associated in LM to operator i as $b_i^{\text{LM}}(t) = 1, \forall t \in [c_o; d_{i,m+1})$. This shows that o is the only call disrupted by n. ■

Proposition 7.7 implies that $E[|\mathcal{G}_n|] = \sum_{i=0}^{\infty} iP[|\mathcal{G}_n| = i]$ simplifies to

$$\begin{aligned} E[|\mathcal{G}_n|] &= \sum_{i=0}^{1} iP[|\mathcal{G}_n| = i] \\ &= P[|\mathcal{G}_n| > 0]. \end{aligned}$$

Moreover, $E[|\mathcal{G}_n|] < 1$. This means that on average a call serviced from the queue will disrupt less than one call. Interestingly, this illustrates a well-known feature of the comparison of the M/M/s and M/M/s/s models: including a queue increases the utilization of the operators as for each call serviced from the queue less than one call is blocked on average. More importantly, this also means that although the disruption is further propagated, there will be a convergence to a final and stable quantity of disrupted calls. We now explain how the disruption is propagated.

7.3 THE NOTION OF INDIRECT DISRUPTION

Suppose a blocked call n is serviced when an operator becomes available. So far we have shown that the service of n disrupts on average $E[|\mathcal{G}_n|]$ other calls[4]. The latter calls will be put on hold and serviced too, after a delay. They will in turn possibly disrupt other calls from $\mathcal{A}^{\text{LM}}$ during their own service. We consider that these calls are indirectly disrupted by call n. The amount of such indirectly disrupted calls is

4 Although we know from Proposition 7.7 that this value is lower than one, we consider the quantity as plural, because it is an average value.

expected to be equal to $E[|\mathcal{G}_n|]^2$. The effect is further propagated by the latter calls, and so on.

Let us define $\mathcal{H}_n$ as the set of calls that are disrupted either directly or indirectly by call n. We also call $\mathcal{H}_n$ the chain of disruptions caused by call n.

Definition 7.4 *Let $o, n \in \mathcal{W}$, we say that $o \in \mathcal{H}_n$ if and only if:*

$$\exists o_1, .., o_k \in \mathcal{W} : o_1 \in \mathcal{G}_n,\ o \in \mathcal{G}_{o_k},\ o_m \in \mathcal{G}_{o_{m-1}}\ \forall m \in \{2, .., k\}.$$

Note that we say that if $o \in \mathcal{H}_n$, we say that n is in the upstream part of the chain of disruptions of o. Obviously the following holds:

$$\mathcal{G}_n \subseteq \mathcal{H}_n.$$

The next step is to determine the amount of calls that are indirectly disrupted by a call n, $E[|\mathcal{H}_n|]$. By repeating the argument of the previous paragraph, we can easily identify $E[|\mathcal{H}_n|]$, the average number of calls that are directly or indirectly disrupted by n:

$$\begin{aligned} E[|\mathcal{H}_n|] &= \sum_{k=1}^{\infty} E[|\mathcal{G}_n|]^k \\ &= \frac{E[|\mathcal{G}_n|]}{1 - E[|\mathcal{G}_n|]}. \end{aligned} \tag{7.14}$$

We observe here a geometric series with common ratio equal to $E[|\mathcal{G}_n|]$, in the disruptions due to the service of each call in the queue. In short, (7.14) gives the amplitude of the consequences of allowing one blocked call to be serviced when an operator becomes available.

Figure 7.1 depicts a realisation of QM with the calls arranged according to our definitions. $\mathcal{R}$ includes four calls, n_1, n_2, n_3 and n_4. Call n_3 does not disrupt any call but the three remaining ones each cause one disruption. This is the beginning of three chains of disruptions that end with the disruption of p_1, q_2 and o_4. Basically, our intuition is that the amplitude of the consequences of allowing all calls of $\mathcal{R}$, i.e. all rejected calls, to be serviced is equal to $B_E(s, a) \cdot E[|\mathcal{H}_n|]$ and that this value is equal to $C_E(s, a)$. We now prove this intuition by showing that $\mathcal{W} \cap \mathcal{A}^{\text{LM}} = \cup_{n \in \mathcal{R}} \mathcal{H}_n$ on the one hand, and that $\mathcal{H}_n \cap \mathcal{H}_{n'} = \emptyset\, \forall n \in \mathcal{R}, \forall n' \in \mathcal{R} \setminus \{n\}$ on the other hand. The former statement means that there is no call of $\mathcal{W} \cap \mathcal{A}^{\text{LM}}$ that does not belong to some $\mathcal{H}_n$, $n \in \mathcal{R}$ and the latter statement means that any call of $\mathcal{W} \cap \mathcal{A}^{\text{LM}}$ belongs to only one chain of disruptions caused by a call of $\mathcal{R}$. Proving the latter permits to show that $E[|\mathcal{H}_n \cup \mathcal{H}_{n'}|] = E[|\mathcal{H}_n|] + E[|\mathcal{H}_{n'}|] = 2E[|\mathcal{H}_n|], \forall n \in \mathcal{R}\, \forall n' \in \mathcal{R} \setminus \{n\}$.

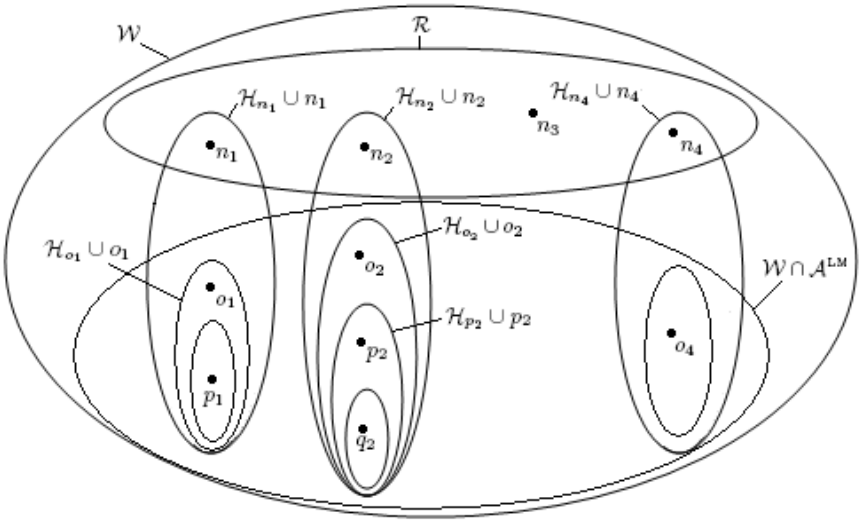

Figure 7.1: Representation of a realisation of QM with chains of disruptions.

Before presenting these two statements in Propositions 7.8 and 7.9, we need to introduce three lemmas that are required for the proofs of the two propositions.

Lemma 7.5

$$\forall o \in (\mathcal{A}^{LM} \cap \mathcal{W}),\ \exists n \in \mathcal{W} :\ \mathcal{H}_o \subset \mathcal{H}_n.$$

Proof. We know from Corollary 7.3 that $\exists n \in \mathcal{W} :\ o \in \mathcal{G}_n$. Besides, we know from Definition 7.4 that $o \notin \mathcal{H}_o$. For call n, we have $\mathcal{H}_n = \{o\} \cup \mathcal{H}_o$, and the lemma is thus proven. ■

This result formally presents something that has been previously stated, namely that the calls of $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ all belong to sets of calls indirectly disrupted by other calls. Consequently they should not be accounted as calls that cause a new chain of disruptions. Otherwise, some calls would be counted multiple times.

Lemma 7.6 $\forall o_i, n \in \mathcal{W}$,

$$[o_i \in \mathcal{H}_n] \Leftrightarrow [\mathcal{H}_{o_i} \subset \mathcal{H}_n], \tag{7.15}$$

and

$$[o_i \notin \mathcal{H}_n] \Leftrightarrow [\mathcal{H}_{o_i} \not\subset \mathcal{H}_n]. \tag{7.16}$$

Proof. (7.16) is the contraposition of (7.15) and therefore we only need to prove one of the two affirmations to prove the other one as well. We take (7.15).

Let us first prove $\Rightarrow$. From Definition 7.4, we know that

$$[o_i \in \mathcal{H}_n] \Leftrightarrow [\exists o_1, .., o_{i-1} \in \mathcal{W} : o_1 \in \mathcal{G}_n, o_m \in \mathcal{G}_{o_{m-1}} \quad \forall m \in \{2, .., i\}].$$

If, for the sake of simplicity we assume that $i, k \in \mathbb{N}$ and that $i < k$, we can write

$$\forall o_k \in \mathcal{H}_{o_i}, \exists o_{i+1}, .., o_{k-1} \in \mathcal{W} : o_m \in \mathcal{G}_{o_{m-1}} \quad \forall m \in \{i+1, .., k\}.$$

As a result, by combining those two results we find that:

$$\forall o_k \in \mathcal{H}_{o_i}, \exists o_1, .., o_{i-1}, o_i, o_{i+1}, .., o_{k-1} \in \mathcal{W} : o_1 \in \mathcal{G}_n, \, o_m \in \mathcal{G}_{o_{m-1}} \quad \forall m \in \{2, .., k\}.$$

This implies that $o_k \in \mathcal{H}_n$ and also $\mathcal{H}_{o_i} \subset \mathcal{H}_n$.

We now prove $\Leftarrow$. Suppose $\exists o_i : \mathcal{H}_{o_i} \subset \mathcal{H}_n, o_i \notin \mathcal{H}_n$. For all element in $\mathcal{H}_{o_i}$, we have

$$\forall o_k \in \mathcal{H}_{o_i}, \exists o_{i+1}, .., o_{k-1} \in \mathcal{W} : o_m \in \mathcal{G}_{o_{m-1}}, \forall m \in \{i+1, .., k\}$$

and, since $\mathcal{H}_{o_i} \subset \mathcal{H}_n$,

$$\forall o_k \in \mathcal{H}_{o_i}, \exists o'_1, .., o'_{k-1} \in \mathcal{W} : o'_1 \in \mathcal{G}_n, \, o'_m \in \mathcal{G}_{o'_{m-1}} \forall m \in \{2, .., k\}.$$

Definition 7.2 says that for each disruption there exists only one call responsible. We must therefore have $o'_m = o_m$, $\forall m \in \{i+1, .., k-1\}$. We then also have $o_i = o'_i$. As a result the sequence $o'_1, .., o'_{i-1}$ is such that $o_i \in \mathcal{G}_{o'_{i-1}}$, $o'_m \in \mathcal{G}_{o'_{m-1}} \forall m \in \{2, .., i-1\}$, $o'_1 \in \mathcal{G}_n$. This means that $o_i \in \mathcal{H}_n$. $\Leftarrow$ is thus proven by contradiction. ■

The following lemma indicates that the set of calls that are indirectly disrupted by another call is only made up of calls of $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$.

Lemma 7.7

$$\mathcal{H}_o \subseteq (\mathcal{A}^{LM} \cap \mathcal{W}), \forall o \in \mathcal{W}.$$

This result is another way of stating that all disrupted calls belong to $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$.

Proof. Suppose that $\exists o' \in (\mathcal{H}_o \cap \mathcal{R})$. From Proposition 7.6, we must have $\nexists n \in \mathcal{W} : o' \in \mathcal{G}_n$. This clearly contradicts Definition 7.4. ■

We now can prove that the chains of disruptions generated by the calls of $\mathcal{R}$ include all the calls of $\mathcal{W} \cap \mathcal{A}^{\text{LM}}$ and only them.

Proposition 7.8

$$\mathcal{W} \cap \mathcal{A}^{LM} = \cup_{n \in \mathcal{R}} \mathcal{H}_n.$$

Proof. Because from Proposition 7.2 $\mathcal{R} \subseteq \mathcal{W}$, we deduce from Lemma 7.7 that $\cup_{n \in \mathcal{R}} \mathcal{H}_n \subseteq (\mathcal{W} \cap \mathcal{A}^{\text{LM}})$.

Let us now show by contradiction that $(\mathcal{W} \cap \mathcal{A}^{\text{LM}}) \subseteq \cup_{n \in \mathcal{R}} \mathcal{H}_n$. Suppose $\exists o \in (\mathcal{W} \cap \mathcal{A}^{\text{LM}}) : \forall n \in \mathcal{R}, o \notin \mathcal{H}_n$. From Lemma 7.5 we know that $\mathcal{H}_o \not\subset \mathcal{H}_n$.

Because $o \in (\mathcal{W} \cap \mathcal{A}^{\text{LM}})$, $\exists o' \in \mathcal{W} : o \in \mathcal{H}_{o'}$.

- If $o' \in (\mathcal{W} \cap \mathcal{A}^{\text{LM}})$, then we deduce from (7.16) in Lemma 7.6 that $\forall n \in \mathcal{R}$, $o' \notin \mathcal{H}_n$. In this case, o' is in the same situation as o and we can repeat our analysis and replace o by o' until we find a call that does not belong to $(\mathcal{W} \cap \mathcal{A}^{\text{LM}})$.
- If $o' \notin (\mathcal{W} \cap \mathcal{A}^{\text{LM}})$, then from Proposition 7.3 we have $o \in \mathcal{R}$. This is a contradiction.

As we know from Proposition 7.3 that $\mathcal{W} = (\mathcal{W} \cap \mathcal{A}^{\text{LM}}) \cup \mathcal{R}$, we cannot find a call o such a described hereabove without contradicting (7.15) in Lemma 7.6.

This proves Proposition 7.8. ■

We now prove that the sets of indirectly disrupted calls of two different calls of $\mathcal{R}$ are mutually exclusive. Put another way we show that a call of $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$ is indirectly disrupted by at most one call of $\mathcal{R}$.

Proposition 7.9

$$\mathcal{H}_n \cap \mathcal{H}_{n'} = \emptyset, \forall n \in \mathcal{R}, n' \in \mathcal{R} \setminus \{n\}.$$

Proof. Suppose there exists an element $o \in \mathcal{W}$ such that $o \in \mathcal{H}_n$, $o \in \mathcal{H}_{n'}$. By definition,

$$\exists o_1, .., o_k : o_1 \in \mathcal{G}_n, o_m \in \mathcal{G}_{o_{m-1}}, \forall m \in \{2, .., k\}$$

and

$$\exists o'_1, .., o'_l : o'_1 \in \mathcal{G}_{n'}, o \in \mathcal{G}_{o'_l}, o'_m \in \mathcal{G}_{o'_{m-1}}, \forall m \in \{2, .., l-1\}.$$

Now assume $l > k$. From Definition 7.2, we know that there is only one call responsible for the disruption of o. We thus have $o_k = o'_l$. Also, $o_m = o'_{l-k+m}$, $\forall m \in \{1, .., k-1\}$, and $n = o'_{l-k-1}$. We deduce that $n \in \mathcal{H}_{n'}$. Because $n \in \mathcal{R}$, this result contradicts Lemma 7.7. We must have $l \leq k$. Repeating the argument for $l < k$, we conclude that $l = k$ and that in order to simultaneously have $o \in \mathcal{H}_n$ and $o \in \mathcal{H}_{n'}$, we must have $n = n'$. This contradicts the assumptions of Proposition 7.9. ■

Using Propositions 7.8 and 7.9, we can write that:

$$E[|\mathcal{W}|] = E[|\mathcal{R}|] + \sum_{n \in \mathcal{R}} E[|\mathcal{H}_n|].$$

We thus have:

$$\begin{aligned} E[|\mathcal{W}|] &= E[|\mathcal{R}|] + E[|\mathcal{R}|] \cdot E[|\mathcal{H}_n|] \\ &= B_E(s,a) + B_E(s,a)\frac{E[|\mathcal{G}_n|]}{1 - E[|\mathcal{G}_n|]} \\ &= \frac{B_E(s,a)}{1 - E[|\mathcal{G}_n|]}, \end{aligned} \tag{7.17}$$

Recalling that we scaled $E[|\mathcal{W}|]$ in function of the benchmark $|\chi| = 1$, $E[|\mathcal{W}|]$ is equal to the proportion of delayed calls. We thus have $E[|\mathcal{W}|] = C_E(s,a)$, meaning that $C_E(s,a) = \frac{B_E(s,a)}{1-E[|\mathcal{G}_n|]}$.

It only remains to determine $E[|\mathcal{G}_n|]$. This is the object of the following section.

7.4 COMPUTATION OF $E[|\mathcal{G}_n|]$

In the previous sections we described the consequences of allowing rejected calls in LM to be serviced in QM after a delay. We showed that other calls are disrupted. They form the set $(\mathcal{A}^{\text{LM}} \cap \mathcal{W})$. We showed in Proposition 7.7 that each call in $\mathcal{W}$ disrupts at most one call and deduced that $E[|\mathcal{G}_n|] = P[|\mathcal{G}_n| = 1] = P[|\mathcal{G}_n| > 0]$. We now compute $E[|\mathcal{G}_n|]$. To do that we identify a necessary and sufficient condition in LM for the occurence of a disruption in QM. We then compute the probability that this condition occurs in LM by solving a Markov chain.

Take any call $n \in \mathcal{W}$. By definition,

$$[|\mathcal{G}_n| = 1] \Leftrightarrow [\exists o \in \chi, \exists i \in \{1,..,s\} : s_i^{\text{LM}}(c_o) = o, s_i^{\text{QM}}(c_o) = n] \tag{7.18}$$

If we assume that in QM operator i serves n, then according to relationship (7.7) we can assume, without loss of generality, that $d_{i,m}$ is the time when n starts being serviced. The equivalence in (7.18) can then be written as

$$[|\mathcal{G}_n|=1] \Leftrightarrow [\exists o \in \chi : c_o \in [d_{i,m}; d_{i,m+1}),\, s_i^{\text{LM}}(c_o) = o,\, s_i^{\text{QM}}(t) = n \ \forall t \in [d_{i,m}, d_{i,m+1})] \tag{7.19}$$

and this implies

$$P[|\mathcal{G}_n| = 1] \; P[\exists o \in \chi : c_o \in [d_{i,m}; d_{i,m+1}), s_i^{\text{LM}}(c_o) = o,\, s_i^{\text{QM}}(t) = n \ \forall t \in [d_{i,m}, d_{i,m+1})].$$

The equivalence in (7.19) basically says that n disrupts a call if and only if in the interval $[d_{i,m}; d_{i,m+1})$, i.e. during the period of service of n in QM, a call, o in our

notation, arrives and is served in LM by the operator i that serves n in QM. This is our condition in LM for a call to be blocked in QM.

The fact that a call o is associated to operator i in LM implies that upon the arrival of o at time $c_o \in [d_{i,m}, d_{i,m+1})$, operator i is the available operator with the most recent end of service. Before that, at time $d_{i,m}$, i is also the process with the most recent end of service. If it remains the case all along $[d_{i,m}, c_o)$, it means that the first event after $d_{i,m}$ is the arrival of call o. o is then serviced by operator i in LM and o is disrupted by n in QM. Other events, ends of service actually, could occur before c_o, resulting in i not being the operator with the latest end of service anymore. In this case, a sequence of successive events can occur during the interval $[d_{i,m}; d_{i,m+1})$ with different outcomes. First, the sequence of events can be such that at a given epoch in $[d_{i,m}; d_{i,m+1})$ the system might return to the situation where i is the available operator with the latest end of service and such that the next following event is an arrival. Obviously this arrival, o in fact, will be associated to i. Second, some sequences might result in operator i not serving any call in $[d_{i,m}, d_{i,m+1})$. In this case, i finishes serving n in QM at time $d_{i,m+1}$ without any disruption by n. We present hereafter an illustration of the two types of sequences.

Example 7.1 *Figure 7.2 is a Gantt-like representation of the occupation of the three operators in a system, each layer representing one of them. Red areas represent periods of time when the corresponding operator is busy. Arrivals are represented at the top of the figure. There are five of them, named from c_{o-4} to c_o. Ends of services are represented below the layers. With some ends of service, such as the one at time $d_{2,k}$, there is a departure. With others, like the one at time $d_{2,k+2}$, there is no corresponding departure. In this sequence of events, call o is associated to operator $i = 3$.*

Now, if c_o and $d_{3,m+1}$ are inverted in the sequence, i.e. the m-th end of service in process 3 occurs before call o arrives, then the operator does not serve any call in $[d_{3,m}, d_{3,m+1})$.

What we are basically looking for is the probability that the sequence of events in $[d_{i,m}; d_{i,m+1})$ results in a call o being served by i. Note that as end-of-service and arrival processes are all Poisson processes, the time between two successive events is exponentially distributed. This is also the case for the time $d_{i,m+1} - d_{i,m}$. Consequently the event $[\exists o \in \chi : c_o \in [d_{i,m}; d_{i,m+1}), s_i^{\mathrm{LM}}(c_o) = o, s_i^{\mathrm{QM}}(t) = n \ \forall t \in [d_{i,m}, d_{i,m+1})]$ can be represented in the Markov chain that is described in Figure 7.3.

There are two absorbing states in this Markov chain, namely G and E. G is the state that is reached if an incoming call is served by operator i in LM before the next end of service in the corresponding process. E is reached if the next end of service in the i^{th} process, in $d_{i,m+1}$, occurs before a call is allocated to operator i. There are also s transient states numbered from 1 to s. Transient state j represents the situation where

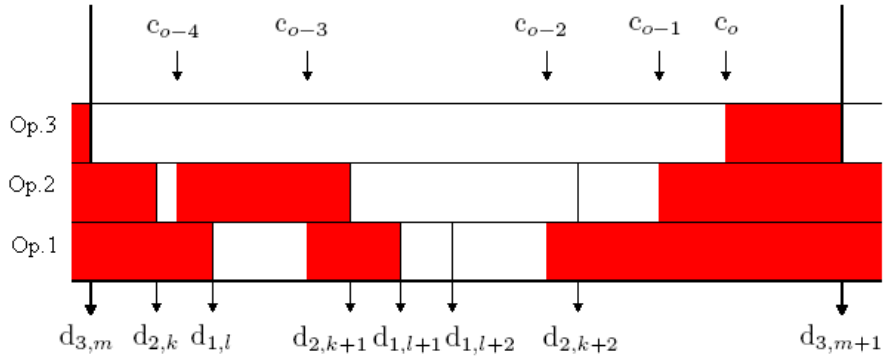

Figure 7.2: A sequence of events that leads to call o being answered by operator $i = 3$ in the interval $[d_{3,m}; d_{3,m+1})$ when $s = 3$.

the last end of service to date in i, that occured at time $d_{i,m}$, is the j^{th} most recent of the last ends of service in the system. As the time horizon considered here starts in $d_{i,m}$, the Markov chain begins in state 1.

We now identify the possible transitions.

- There is one transition from each of the s transient states toward state E. The transition rate is equal to μ and such a transition corresponds to the $(m+1)^{th}$ end of service in i.
- The only transition toward state G originates from transient state 1. Its rate is equal to λ and the transition corresponds to an arriving call that is associated to i in LM, as in state 1 i is the available operator with the most recent end of service to date.
- There is a transition from state j to state $j-1$ with rate λ. This corresponds to an arrival in LM that will occupy the operator with the most recent end of service. In this case, operator i gains a rank in the list.
- Moving from transient state j to transient state $j+1$ occurs at rate $(s-j)\mu$. This corresponds to an end of service for one of the $s-j$ processes for which the most recent end of service up to this time was older than the m^{th} end of service of i.
- Finally, there is a transition from each transient state to itself. In transient state j, the transition corresponds to an end of service for one of the $(j-1)^{th}$ available operators that have a more recent end of service than operator i. This event does not change the rank of operator i in the priority list.The rate of this transition is equal to $(j-1)\mu$.

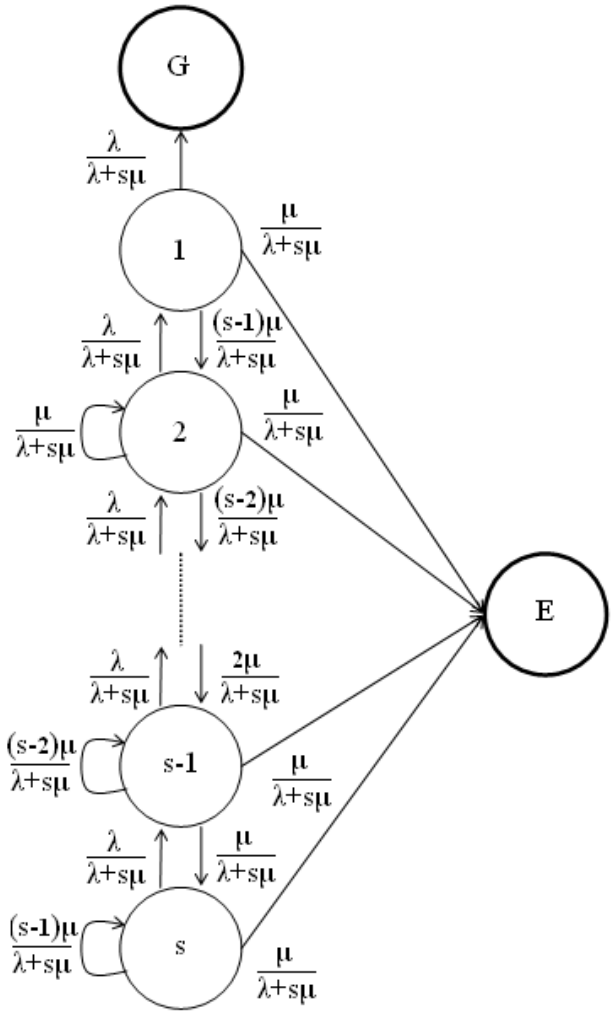

Figure 7.3: Markov chain representation of the sequence of events leading to a disruption or not.

Observe that the total rate of the transitions from one transient state is equal to $\lambda + s\mu$. Finding the associated transition probabilities is trivial. They are displayed in Figure 7.3.

Example 7.2 *The sequence of events described in Figure 7.2 would induce the following sequence of transitions in the Markov chain. Starting from state 1, the system would switch to state 2 in $d_{2,k}$. With the arrival in c_{o-4}, the system moves back to state 1. A departure in $d_{1,l}$ moves the system to state 2. It switches again to state 1 in $c_o - 3$ and returns to state 2 with*

the departure at time $d_{2,k+1}$. At time $d_{1,l+1}$ the system moves to state 3. It remains in state 3 with the end of service in $d_{1,l+2}$. An arrival in c_{o-2} moves the system to state 2, where it remains until c_{o-1} despite the end of service in $d_{2,k+2}$, that corresponds to a transition within the same state. The arrival in c_{o-1} moves the system back to state 1 and the system reaches state G with the arrival at time c_o.

Let ν_j be the probability of joining state G when the system is in transient state j. Because the chain begins in state 1, we are looking for ν_1. The values of ν_j are found by solving the following set of equations:

$$\begin{aligned}
\nu_1 &= \frac{\lambda}{\lambda+s\mu} + \frac{(s-1)\mu}{\lambda+s\mu}\nu_2 && (7.20)\\
\nu_2 &= \frac{\lambda}{\lambda+s\mu}\nu_1 + \frac{\mu}{\lambda+s\mu}\nu_2 + \frac{(s-2)\mu}{\lambda+s\mu}\nu_3 && (7.21)\\
&\vdots \\
\nu_j &= \frac{\lambda}{\lambda+s\mu}\nu_{j-1} + \frac{(j-1)\mu}{\lambda+s\mu}\nu_j + \frac{(s-j)\mu}{\lambda+s\mu}\nu_{j+1} && (7.22)\\
&\vdots \\
\nu_{s-1} &= \frac{\lambda}{\lambda+s\mu}\nu_{s-2} + \frac{(s-2)\mu}{\lambda+s\mu}\nu_{s-1} + \frac{\mu}{\lambda+s\mu}\nu_s && (7.23)\\
\nu_s &= \frac{\lambda}{\lambda+s\mu}\nu_{s-1} + \frac{(s-1)\mu}{\lambda+s\mu}\nu_s. && (7.24)
\end{aligned}$$

This system of equations is solved by first isolating ν_s in (7.24) and by replacing ν_s with the latter result in (7.23). The procedure is repeated up to (7.20), where ν_2 is replaced by a function of ν_1.

For ν_s we find:

$$\nu_s = \frac{\lambda}{\lambda+\mu}\nu_{s-1}.$$

In the mean time, basic manipulations on (7.2) when $s = 1$ lead to

$$\frac{\lambda}{\mu}(1 - B_E(1,a)) = \frac{\lambda}{\lambda+\mu}.$$

As a result, we find that

$$\nu_s = \frac{\lambda}{\mu}(1 - B_E(1,a))\nu_{s-1}.$$

Based on this last result, we now determine ν_j assuming ν_{j+1} is equal to

$$\nu_{j+1} = \frac{\lambda}{(s-j)\mu}(1 - B_E(s-j,a))\nu_j. \qquad (7.25)$$

This yields:

$$\begin{aligned}\nu_j &= \frac{\lambda}{\lambda+s\mu}\nu_{j-1} + \frac{(j-1)\mu}{\lambda+s\mu}\nu_j + \frac{(s-j)\mu}{\lambda+s\mu}\frac{\lambda}{(s-j)\mu}(1-B_E(s-j,a))\nu_j \\ &= \frac{\lambda}{\lambda+s\mu}\nu_{j-1} - \frac{(s-j+1)\mu-(\lambda+s\mu)+\lambda B_E(s-j,a)}{\lambda+s\mu}\nu_j \\ &= \frac{\lambda\nu_{j-1}}{(s-j+1)\mu+\lambda B_E(s-j,a)}.\end{aligned}$$

As we find from (7.2) that

$$\begin{aligned}\frac{\lambda}{(s-j+1)\mu}(1-B_E(s-j+1,a)) &= \frac{\lambda}{(s-j+1)\mu}\frac{\sum_{k=0}^{s-j}\frac{(\lambda/\mu)^k}{k!}}{\sum_{k=0}^{s-j+1}\frac{(\lambda/\mu)^k}{k!}} \\ &= \frac{\lambda}{(s-j+1)\mu}\frac{1}{1+\frac{\frac{(\lambda/\mu)^{s-j+1}}{(s-j+1)!}}{\sum_{k=0}^{s-j}\frac{(\lambda/\mu)^k}{k!}}} \\ &= \frac{\lambda}{(s-j+1)\mu+\lambda\frac{\frac{(\lambda/\mu)^{s-j}}{(s-j)!}}{\sum_{k=0}^{s-j}\frac{(\lambda/\mu)^k}{k!}}} \\ &= \frac{\lambda}{(s-j+1)\mu+\lambda B_E(s-j,a)},\end{aligned}$$

we prove that if (7.25) holds the following is true:

$$\nu_j = \frac{\lambda}{(s-j+1)\mu}(1-B_E(s-j+1,a))\nu_{j-1}. \tag{7.26}$$

Because we know that this relationship is true for $j=s$, we can prove recursively that (7.26) holds for any $j \in \{2,..,s\}$. In particular, for $j=2$ we find an expression for ν_2 that only depends on ν_1. This expression can be coupled with (7.20) to form a system of equations that can be solved for ν_1. We obtain:

$$\begin{aligned}\nu_1 &= \frac{\lambda}{\lambda+s\mu} + \frac{(s-1)\mu}{\lambda+s\mu}\frac{\lambda}{(s-1)\mu}(1-B_E(s-1,a))\nu_1 \\ &= \frac{\lambda}{s\mu+\lambda(1-B_E(s-1,a))} \\ &= \frac{\lambda}{s\mu}(1-B_E(s,a)).\end{aligned}$$

Finally we have

$$P[|\mathcal{G}_n|>0] = \frac{\lambda}{s\mu}(1-B_E(s,a)). \tag{7.27}$$

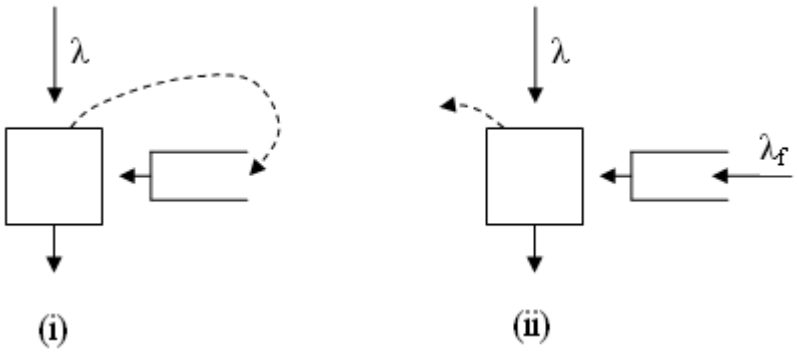

Figure 7.4: Representations of a queueing system when blocked calls in the arrival process feed the queue (i) and when the calls in the queue are not rejected calls of the main arrival process (ii).

We immediately deduce that

$$E[|\mathcal{G}_n|] = \frac{\lambda}{s\mu}(1 - B_E(s,a)). \tag{7.28}$$

Including the latter result in (7.17) yields Formula (7.1). We conclude that our sample path approach with the disruption correctly explains the link between $B_E(s,a)$ and $C_E(s,a)$.

7.5 COMMENTS ON THE STABILITY OF THE MODEL

Let us remind that the stability condition for a M/M/s model is $\lambda < s\mu$.

When we considered in Section 7.2 a call n and investigated the effect of its injection in the system from the queue, we assumed n is a call of the arriving process $\mathcal{C}$. This need not be the case. One could instead assume that all calls of $\mathcal{C}$ that arrive when the system is full are rejected and that call n is the unique arrival from a secondary source that feeds the queue. The two situations are represented in Figure 7.4. In scheme (i) rejected calls of $\mathcal{C}$ go to the queue and in scheme (ii), these calls are rejected to be lost and calls from a secondary external flow with rate λ_f supply the queue. We have that each service of the calls in the second flow can only start when an end of service occurs in one of the s end-of-service processes. The difference between both situations is slight but helps determining where the stability condition plays a role.

In particular we see that the stability condition does not concern the main arrival process $\mathcal{C}$: the arrival rate does not matter as from the point of view of these calls, the model is a M/M/s/s model. If no operator is available upon its arrival, a call is just

lost. On the other side, calls from the secondary flow enter the system to be served only when there is an end of service for one of the s service processes. They therefore exit the system at a maximum rate equal to $s\mu$. The system will be unstable if on the long run the number of arrivals in the secondary process exceeds the number of possible departures. The condition for stability is thus:

$$\lambda_f < s\mu. \tag{7.29}$$

There is therefore a maximum quantity of calls that can use the queue.

In Section 7.3 we basically established a link between the flow of rejected calls of $\mathcal{C}$ and the total flow of calls to the queue. We established that :

$$\lambda_f = \frac{\lambda B_E(s,a)}{1 - \frac{\lambda}{s\mu}(1 - B_E(s,a))}.$$

The stability condition becomes thus:

$$\begin{aligned} \frac{\lambda B_E(s,a)}{1 - \frac{\lambda}{s\mu}(1 - B_E(s,a))} &< s\mu \\ \lambda B_E(s,a) &< s\mu - \lambda(1 - B_E(s,a)) \\ \lambda &< s\mu. \end{aligned}$$

This is the required condition for steady state in M/M/s models.

7.6 CONCLUDING REMARKS

In this chapter we described a sample path approach to construct a M/M/s system from the equivalent M/M/s/s system. The approach allowed us to understand how more calls end up waiting in the queueing model than only the calls rejected in LM. In particular we showed that the service of each waiting call induces a series effect resulting in strictly positive waiting times for more calls. In other words we found out that the waiting probability is obtained by dividing the proportion of lost calls in the equivalent M/M/s/ system by $1 - E[|\mathcal{G}_n|]$, where $E[|\mathcal{G}_n|]$ is the expected number of calls disrupted by any call $n \in \mathcal{W}$. It is the common ratio of the geometric series caused by the disruptions.

This is the main contribution of the chapter. We are going to use this approach and the resulting interpretation of (7.1) in the next chapter to develop approximations for the blocking probability in multi-skill models with queues.

8

ESTIMATING THE PERFORMANCE USING THE SAMPLE PAST ANALYSIS

In Chapter 7 we used a sample path approach to explain the link between a queueing system and its equivalent loss system in models with only one type of call. Here we apply the approach to models where some or all call types can wait in queue before being serviced and we develop an approximation for the blocking, i.e. waiting or loss depending on the situation, probability.

The chapter is structured as follows. In Section 8.1 we briefly discuss the importance of an approximation for multi-skill queueing systems. In Section 8.2 we apply the principle of disruption described in the previous chapter to two systems. Both systems are known to have the same performance as the single-skill system with identical parameters and we exploit this property to deduce two rules useful for our multi-skill approximation. We present in Section 8.3 the notation we use for developing this approximation. We also describe the related assumptions. This is followed by a short section that presents the general three-step procedure that we are going to follow to build approximation formulae. Based on this procedure, we find in Section 8.5 formulae to compute the probability of waiting or being rejected in models where two types of calls are served by three pools of operators. In Section 8.6, we generalize the formulae of Section 8.5 to systems with more than two types of calls and more pools of operators. In Section 8.7 we perform a series of numerical experiments and evaluate the quality of the proposed approximations. Section 8.8 concludes the chapter.

8.1 MOTIVATION FOR A MULTI-SKILL QUEUEING APPROXIMATION

We previously commented that flexibility is of increasing importance in service industries in general and for call center companies in particular. Nowadays they often handle many different types of calls each requiring specific skills. Flexibility is critical as demand is stochastic and requires quick response from the service provider. This

has resulted in the development of multi-class queueing models in the literature. For a review of the main models used, we already mentioned [Gans et al., 2003] and [Akşin et al., 2007a]. On the general problem of routing and staffing in multi-skill call centers we recommend [Koole and Pot, 2006].

Evaluating the performance of a multi-skill queueing system is a challenging issue per se. [Shumsky, 2004] proposes an interesting approximation to evaluate the performance of a queueing system with two types of demands and two types of operators, one being single-skilled and the other type being polyvalent. The author divides the state-space into different areas. This procedure permits to diminish the computation burden significantly, even for a large number of operators, while keeping accurate results. [Whitt, 1992] develops simple approximations for common performance measures, in the case of G/G/s systems. The approximations are based on the infinite server model. We use that approach to some extent when using the Hayward approximation. Nevertheless we take a different direction by considering several types of calls and different types of operators, some dedicated, others more polyvalent.

[Maglaras, 1999], [Armony and Bambos, 2003], [Bambos and Walrand, 1993] and [Gans and van Ryzin, 1997] all consider multi-skill models.

The model presented in [Maglaras, 1999] and [Armony and Bambos, 2003] is a generalization of the queueing models we use here.

Among the queueing systems, the loss systems have the advantage of being much simpler and, consequently, much easier to analyze than other queueing systems. In Chapter 5 we presented a general method to estimate the loss probability in these systems, along with some variants. In Chapter 6, we presented a method to optimize the configuration of multi-skill loss systems. In Section 6.6, we noticed that the relative performance of loss systems is often a good proxy for the relative performance of similar queueing systems. This gives us the idea of a more thorough investigation of the potential to use loss systems and their qualities to build approximation techniques for the performance measurement of queueing systems. This would pave the way for the optimization of multi-skill queueing systems. In the current chapter and in Chapter 9 we show that very good results can be obtained with such techniques.

8.2 PROPERTIES

As said in the introduction, we present here two interesting properties of M/M/s models that are easily proven with the approach described in Chapter 7. The first property gives the number of disrupted calls when we know that a delayed call is serviced in one of several pools. This is shown by dividing a pool into two or more subpools and by computing the proportion of calls serviced immediately in each sub-

pool in QM. The second property tells us that if calls of different types can be serviced in a same pool then the service of a delayed call can disrupt calls of all types and explains how to determine the amount of disrupted calls in this context. We derive the second property by dividing the arrival process in two Poisson subprocesses. In both cases we use the two models, LM and QM, of Chapter 7 to derive the property.

8.2.1 *Two successive pools*

Let us modify the structure in LM and QM by dividing the pool of s operators into two subpools, refered as 1 and 2, with s^1 and s^2 operators respectively, such that $s^1 + s^2 = s$. Pools 1 and 2 are put in series as shown in Figure 4.2 (i) on page 50. This means that the latter receives calls only if all operators of the former are busy. We define $\mathcal{A}^{\text{LM},1}$ and $\mathcal{A}^{\text{LM},2}$ as the sets of calls that are answered in LM by an operator of pool 1 and pool 2, respectively. In QM, calls that cannot find an available operator either in pool 1 or in pool 2 are put on hold and are answered when an operator in any of the two pools becomes available.

We argued in Section 4.2 that the performance of LM and QM remains the same as the one obtained with a unique pool. The total loss probability in LM is still equal to $B_E(s, a)$ and we have $\mathcal{A}^{\text{LM}} = \mathcal{A}^{\text{LM},1} \cup \mathcal{A}^{\text{LM},2}$ as well as $\mathcal{A}^{\text{LM},1} \cap \mathcal{A}^{\text{LM},2} = \emptyset$. This division is thus purely artificial. Its purpose is to be able to focus on one specific part of the system and to compute the proportion of disrupted calls in this part.

Because pool 1 faces all the arrivals of the Poisson process, the loss probability at the first pool is equal to $B_E(s^1, a)$, and the proportion of calls that are answered by an operator of the first pool in LM is equal to $(1 - B_E(s^1, a))$. As the time between two successive arrivals at the second pool is not always exponentially distributed, the exact loss probability cannot be obtained from Formula (4.1). It can however be easily computed using

$$\text{LP}^2 = \frac{B_E(s, a)}{B_E(s^1, a)}, \tag{8.1}$$

where, generalizing to more than two successive pools, LP^k is the loss probability at pool k.

Let n be a call of $\mathcal{W}$. We are now going to compute the average number of calls in $\mathcal{A}^{\text{LM},1}$ that are put on hold when call n is answered by an operator of pool 1, $E[|\mathcal{G}_n| \, | s^{\text{QM}}(n) \in Q^1]$, where $s^j(n)$ is defined as the function which associates to call n the operator that serves n in model j and Q^j is the set of operators of pool j.

Because calls arrive to pool 1 according to a Poisson process, this pool forms a $M/M/s^1$ system and, using (7.28), we find that

$$E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^1] = \frac{\lambda}{s^1 \mu}(1 - B(s^1, a)). \tag{8.2}$$

We cannot reproduce the same argument for pool 2 and use (7.28) to compute $E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^2]$ because the arrivals to pool 2 do not form a Poisson process. However, recalling that the performance of the system is not affected by the division into two subpools, we can deduce $E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^2]$ in the following way. We know that $E[|\mathcal{G}_n|] = \frac{\lambda}{s\mu}(1 - B_E(s, a))$. It is easy to see that

$$E[|\mathcal{G}_n|] = \sum_{k=1}^{2} E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^k] \cdot P[s^{QM}(n) \in \mathcal{Q}^k]. \tag{8.3}$$

This formula gives the expected amount of disrupted calls, independently of the pool that serves n after the delay. We can therefore easily find $E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^2]$ once we know $P[s^{QM}(n) \in \mathcal{Q}^k]$, $\forall k \in \{1, 2\}$, the probability that call n is answered by an operator of pool k.

Let m be the index of the end of service that precedes n in the service process $s^{QM}(n)$. Because at time $d^-_{s^{QM}(n),m}$ all operators are busy in QM, $P[s^{QM}(n) \in \mathcal{Q}^k]$ is equal to the probability that an operator of pool k is first to complete a service. As all service times are i.i.d according to an exponential distribution with parameter μ, we have:

$$P[s^{QM}(n) \in \mathcal{Q}^k] = \frac{s^k}{s^1 + s^2}, \quad k \in \{1, 2\}. \tag{8.4}$$

By including this last result in (8.3), by replacing $E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^1]$ with (8.2) and $E[|\mathcal{G}_n|]$ with (7.28) and by introducing LP^2 from (8.1), we find:

$$E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^2] = \frac{\lambda}{s^2 \mu} B_E(s^1, a)(1 - LP^2).$$

If we divide $\mathcal{Q}^1$ several times in two successive pools and if we repeat after each division the procedure for finding $E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^2]$, we can generalize the latest result and prove that for K successive pools,

$$E[|\mathcal{G}_n| \, | s^{QM}(n) \in \mathcal{Q}^k] = \frac{\lambda}{s^k \mu}\Big(\prod_{l=1}^{k-1} LP^l\Big)(1 - LP^k) \quad \forall k \in \{1, .., K\}. \tag{8.5}$$

It is interesting to note that the structure of this formula is the same as the one of (7.28), for any pool k. We summarize our results in the following property.

Property 2 *If in a M/M/s model the operators are divided into subgroups, the average number of calls disrupted by the service of a call $n \in \mathcal{W}$ at pool k is equal to the rate of the calls of $\mathcal{A}^{LM}$ which are answered at pool k, $\lambda\big(\prod_{l=1}^{k-1} LP^l\big)(1 - LP^k)$, divided by the total service rate of pool k, $s^k\mu$.*

If the stability condition in the system, $\lambda < s\mu$, is fulfilled, then we can reproduce the argument to deduce $E[|\mathcal{H}_n|]$ with (7.14) and then verify that the waiting probability, is effectively equal to $C_E(s,a)$. We illustrate this for $k = 2$.

Example 8.1 *With $k = 2$, $E[|\mathcal{G}_n|] = \sum_{k=1}^{2} E[|\mathcal{G}_n| \ \ |s^{QM}(n) \in \mathcal{Q}^k] \cdot P[s^{QM}(n) \in \mathcal{Q}^k]$. By using this result as the common ratio in Formula (7.14), we find $E[|\mathcal{H}_n|]$. We deduce therefore, with (7.17), that*

$$\begin{aligned} WP(s^1+s^2,a) &= B_E(s^1,a)\cdot LP^2 \\ &\quad + \frac{P[s^{QM}(n)\in\mathcal{Q}^1]\frac{\lambda}{s^1\mu}(1-B_E(s^1,a))\,B_E(s^1,a)\,LP^2}{1-\sum_{k=1}^{2}E[|\mathcal{G}_n|\,|s^{QM}(n)\in\mathcal{Q}^k]\cdot P[s^{QM}(n)\in\mathcal{Q}^k]} \\ &\quad + \frac{P[s^{QM}(n)\in\mathcal{Q}^2]\frac{\lambda}{s^1\mu}B_E(s^1,a)(1-LP^2)\,B_E(s^1,a)\,LP^2}{1-\sum_{k=1}^{2}E[|\mathcal{G}_n|\,|s^{QM}(n)\in\mathcal{Q}^k]\cdot P[s^{QM}(n)\in\mathcal{Q}^k]}. \end{aligned}$$

If we replace $P[s^{QM}(n) \in \mathcal{Q}^k]$ with (8.4) and simplify we have

$$\begin{aligned} WP(s^1+s^2,a) &= B_E(s^1+s^2,a) \\ &\quad + \frac{\frac{\lambda}{(s^1+s^2)\mu}(1-B_E(s^1,a))\,B_E(s^1+s^2,a)}{1-\frac{\lambda}{(s^1+s^2)\mu}(1-B_E(s^1,a))-\frac{\lambda}{(s^1+s^2)\mu}B_E(s^1,a)(1-LP^2)} \\ &\quad + \frac{\frac{\lambda}{(s^1+s^2)\mu}B_E(s^1,a)(1-LP^2)\,B_E(s^1+s^2,a)}{1-\frac{\lambda}{(s^1+s^2)\mu}(1-B_E(s^1,a))-\frac{\lambda}{(s^1+s^2)\mu}B_E(s^1,a)(1-LP^2)}. \\ &= \frac{B_E(s^1+s^2,a)}{1-\frac{\lambda}{(s^1+s^2)\mu}B_E(s^1+s^2,a)}. \end{aligned}$$

Because $s^1 + s^2 = s$, we have $WP(s_1 + s_2, a) = C_E(s,a)$.

8.2.2 *Two different types of calls*

Suppose now that $\lambda = \lambda_x + \lambda_y$ where λ_x and λ_y are the arrival rates of two Poisson processes. Suppose also that the service time of any call of the two types is exponentially distributed with parameter μ. The notation used so far is adjusted with the addition of a subscript to designate the call type. In the absence of subscripts we refer to any type of call. As in Section 8.2.1, the perfomance of LM and QM is basically not changed by the subdivision of the arrival process.

Let $n_x \in \mathcal{W}_x$be a type-x call that is delayed. The objective is to determine the proportion of type-x calls and type-y calls that are put on hold because of the service of n_x. From (7.28), we deduce:

$$\begin{aligned} E(|\mathcal{G}_{n_x}|) &= \frac{\lambda_x + \lambda_y}{s\mu}(1 - B_E(s, a_x + a_y)) \\ &= \frac{\lambda_x}{s\mu}(1 - B_E(s, a_x + a_y)) + \frac{\lambda_y}{s\mu}(1 - B_E(s, a_x + a_y)), \end{aligned} \tag{8.6}$$

where $a_i = \frac{\lambda_i}{\mu}$. We identify here two groups of disrupted calls. The first term in (8.6) gives the average number of type-x calls disrupted by n_x, the second one gives the average number of disrupted y-calls. We note here that Property 2 holds for both terms: the average number of calls of one type is obtained by dividing the arrival rate of the calls of this type that are answered in LM by the total service rate of the pool. Any call in $\mathcal{W}$ can disrupt calls of any type. As the disrupted calls will possibly disrupt other calls of any type, the series effect is more complex to compute recursively. For example n_x can indirectly disrupt a call $o_x \in \mathcal{A}_x^{\text{LM}}$ by first disrupting a call o_y such that $o_x \in \mathcal{G}_{o_y}$. However, as $\lambda_x + \lambda_y = \lambda$, we know from (7.14) that:

$$\begin{aligned} E(|\mathcal{H}_{n_x}|) &= \sum_{i=1}^{\infty} \left(\frac{\lambda_x + \lambda_y}{s\mu}(1 - B_E(s, a_x + a_y)) \right)^i \\ &= \frac{\frac{\lambda_x}{s\mu}(1 - B_E(s, a_x + a_y))}{1 - \frac{\lambda_x}{s\mu}(1 - B_E(s, a_x + a_y)) - \frac{\lambda_y}{s\mu}(1 - B_E(s, a_x + a_y))} \\ &\quad + \frac{\frac{\lambda_y}{s\mu}(1 - B_E(s, a_x + a_y))}{1 - \frac{\lambda_x}{s\mu}(1 - B_E(s, a_x + a_y)) - \frac{\lambda_y}{s\mu}(1 - B_E(s, a_x + a_y))}. \end{aligned} \tag{8.7}$$

This result is obtained by first computing the geometric series and by then factorizing the expression in factors of λ_x and λ_y, in the numerator and the denominator. Each of the two terms gives the average number of calls of one type that are disrupted by n_x. The series factor, as presented in the denominator, consists in two terms that each corresponds to the propagation of the disruption for one of the two types of calls. We deduce from this result the following property:

Property 3 *If in a M/M/s model there is more than one type of arrival, a delayed call can disrupt a call of any type upon its service. Independently of the type of the call responsible for the disruption, the expected number of disrupted type-i calls is found by dividing the rate of the answered type-i calls in the loss model by the total service rate of the pool.*

Note that Property 2 and Property 3 complement each other: the structure of the expression that gives the expected number of disrupted calls is similar as it consists in dividing a rate of answered calls in the loss model by the service rate in one pool.

8.2.3 *Conclusion*

Each of these two properties gives a partial insight on what happens when several types of call share different groups of operators. The properties are valid when the system can be represented by an M/M/s model. This requires in particular that all operators should be able to answer all types of calls with equal efficiency (i.e. same exponential service time distribution). In the next section, we assume that both properties apply when operators can answer only a subset of the call types. Based on this assumption, we build approximation formulae to compute the waiting probability or the loss probability in systems with more than one type of call.

8.3 NOTATION AND ASSUMPTIONS

We wish to estimate the probability for a call to wait in models where all calls can wait but also in models where only a subset of the call types can access the queue, blocked calls of other types being rejected.

As a reminder, the general idea of our approach is to draw on the analogy with a loss system where arrivals, operator pools and the routing of arriving calls are identical but all blocked calls are rejected rather than put on hold in a queue if no available operator can be found to handle the call immediately. This system is the equivalent loss system. Figure 8.1 shows a simple example of the type of systems we are studying.

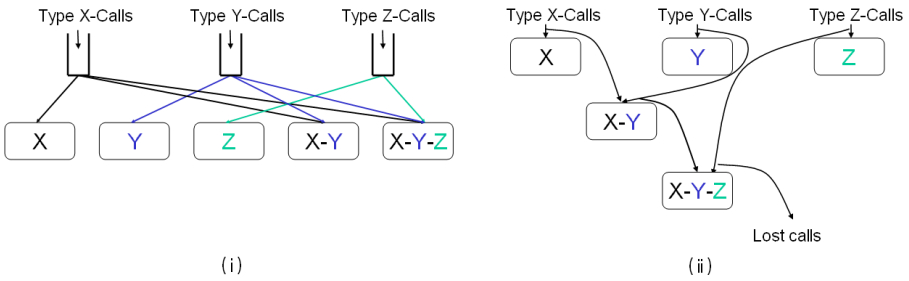

Figure 8.1: A simple example of a multi-class, multi-pool system. (i) represents the system with queues for all types of calls and (ii) is the equivalent loss system.

Note that we consider all the formulae proposed in the next sections to be approximations. Some results are exact in some cases but identifying the exact conditions under which the formulae are true is cumbersome and would not bring much added

value as compared to the space and time required to prove it. Basically, our ultimate goal is to present handy formulae that accurately evaluate the waiting probability.

8.3.1 *Notation*

We extend the notation of the previous chapter to include the call types and the different types of operators. Calls are classified according to their type $i \in \mathcal{I}$ and their arrival rate is noted λ_i[1].

Let $j \in \{1, .., J\}$ denote the pool of operators with skill set $\mathcal{S}^j$ [2]. The set of operators in pool j is denoted $\mathcal{Q}^j$ and $|\mathcal{Q}^j| = s^j$. $\mathcal{P}_i$ is defined as the set of pools that have the skill to answer calls of type i. We use as in the previous chapter $\mathcal{G}_n$ to refer to the set of calls in $\mathcal{A}^{\text{LM}}$ that are disrupted by the service of n, and $\mathcal{H}_n$ as previously defined as well. Note that the calls in $\mathcal{G}_n$ or $\mathcal{H}_n$ can be of any type.

χ is the set of all arrivals in the model. We use $\chi_i \subseteq \chi$ to denote the set of calls of type i. The set $\mathcal{A}_i^{\text{LM},j}$ regroups all type-i calls that are answered in LM by an operator from $\mathcal{Q}^j$. $\mathcal{W}_i^j$ is the set of type-i calls in $\mathcal{W}$ that are answered after waiting by an operator from $\mathcal{Q}^j$. As in the preceding chapter, we apply the normalization process so that the size of each subset of χ_i corresponds to the proportion of i calls that belong to this subset. Note however that this normalization poses problems when we analyze the interactions between calls of different types: because for each call type i the corresponding sets would all be rescaled based on the benchmark $|\chi_i| = 1$, information is lost about the relative size of the different flows. For this reason on the one hand we set $|\chi| = 1$ and $|\chi_i| = \frac{\lambda_i}{\sum_{k\in\mathcal{I}} \lambda_k}$ and we define $|.|_i$ as the operator that measures the size of a set by using the size of χ_i as the normalized benchmark, i.e. $|\chi_i|_i = 1$. Therefore when we compare calls of different types, we need to rescale these amounts to account for the relative sizes of calls of each type. We use scaling factors that are ratios of the respective arrival rates. The example hereafter clarifies this.

Example 8.2 *Suppose a queueing system with calls of type x and calls of type y such that $\lambda_y = 2\lambda_x$. The amount of delayed y-calls is equal to WP_y when we use the operator $|.|_y$ to scale the size of $\mathcal{W}_y$. Let α be the proportion of delayed y-calls that are serviced at pool xy, the unique pool in $\mathcal{P}_x \cap \mathcal{P}_y$. If we estimate based on Property 3 that each delayed y-call answered at pool xy disrupts on average $\frac{\lambda_x M_x^{xy}}{s^{xy}\mu}$ calls of type x, then the total amount of x-calls directly disrupted by delayed y-calls is equal to $\frac{\lambda_x M_x^{xy}}{s^{xy}\mu} \cdot WP_y$ if we use $|.|_y$, i.e. if χ_y is used as the normalized benchmark, to measure the size of this set, and to $\frac{\lambda_x M_x^{xy}}{s^{xy}\mu} \cdot 2WP_y$ when measured with operator $|.|_x$, as there are on average λ_y/λ_x times, i.e. twice, as many y-calls as x-calls.*

1 We usually use subscripts to refer to anything related to call types.
2 We use superscripts to refer to anything related to the groups of operators.

We define L_i as the size of the queue for type-i calls. L_i is equal to either zero or infinity. Because of the increasing number of call types, more models than the pure loss and pure queueing models are going to be analyzed. There is a different model for each combination of queue sizes. We define $\mathcal{U} = \{i : i \in \mathcal{I},\, L_i = \infty\}$ as the set of call types with a queue of infinite capacity and $\mathcal{V} = \{i : i \in \mathcal{I},\, L_i = 0\}$ as the set of call types that do not have access to a queue. Obviously $\mathcal{U} \cup \mathcal{V} = \mathcal{I}$. We use QM($\mathcal{U}$) to refer to the model where the system includes a queue for the call types in $\mathcal{U}$ and not for the others. We use LM for QM($\emptyset$) and drop ($\emptyset$) for all elements and results related to LM.

The waiting probability for a call of type $i \in \mathcal{U}$ in QM($\mathcal{U}$) is noted $\mathrm{WP}_i(\mathcal{U})$ and the loss probability for a call of type $i \in \mathcal{V}$ is noted $\mathrm{LP}_i(\mathcal{U})$. When we need to refer to the proportion of calls that cannot find an available operator upon their arrival, we use $\mathrm{BP}_i(\mathcal{U})$ [3]. $\mathrm{BP}_i(\mathcal{U}) = \mathrm{LP}_i(\mathcal{U})$ when $L_i = 0$, and $\mathrm{BP}_i(\mathcal{U}) = \mathrm{WP}_i(\mathcal{U})$ when $L_i = \infty$.

$s^{\mathrm{QM}(\mathcal{U})}(n)$ gives the reference of the operator that serves call n in QM($\mathcal{U}$).

For the sake of brevity, we use M_i^j to designate the proportion of type-i calls that are answered by an operator from $\mathcal{Q}^j$ in LM. This avoids using combinations of sums (when flows are subdivided) and products (when there are successive pools of operators) of loss probabilities, as in equation (8.5).

To illustrate the concepts and procedures we present in the coming section, we are going to refer to graphs similar to the one presented in Figure 8.2. It is a tree representation of the possible chains of disruptions with two types of calls. As for a node to exist in practice there must be a disruption, we associate a call to each node in the three. In reality there would be at most one branch that leaves a node but here, as we are going to work with average disruptions, we consider all possible branches. Here, we named the call at the top layer n_i, it can either be a call of type x or a call of type y.

8.3.2 *Assumptions*

Before describing the methodology in the next sections, we review three issues that arise when using the procedure of Chapter 7 to multi-skill systems. We solve the first two issues by making assumptions when developing the approximation methods. For the numerical applications of Section 8.7, approximations are proposed to drop these assumptions. For the third issue, an assumption is made and kept throughout this chapter, both for the development of the approximations and for the numerical applications. There is a brief discussion about how to relax this assumption in Section 8.8.

3 BP stands for blocking probability, i.e. loss or waiting probability, depending on the presence of a queue or not.

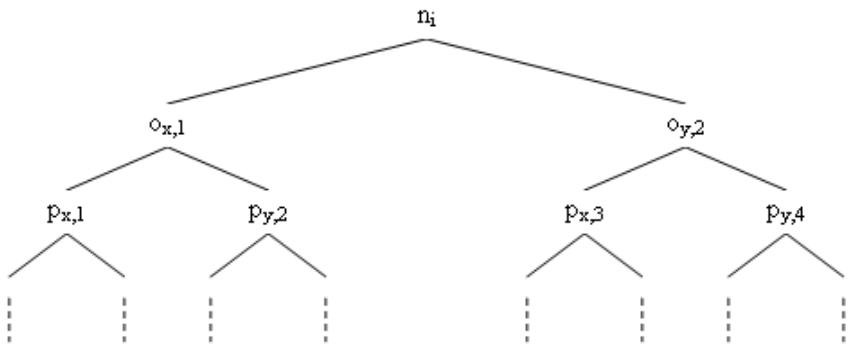

Figure 8.2: A tree representation of the chains of disruptions.

1. We saw in Chapter 5 that computing the loss probability in multi-skill systems is not straight-forward and requires one to make several approximations. We discuss the choice of the approximation method in Section 8.7 and until then we assume that the performance of LM is known.

2. In the coming sections we will need the probability for a call of $\mathcal{W}_i$ to be answered by an operator of pool j. This value first depends on the number of operators in $\mathcal{Q}^j$. It also depends on the presence of waiting calls of other types that can be answered by operators of pool j. In other words, this probability also depends on $\mathcal{U}$. Let $\pi_i^j(\mathcal{U})$ denote this probability and let $\Pi(\mathcal{U})$ be the set of all $\pi_i^j(\mathcal{U})$ in QM($\mathcal{U}$). As for the performance of LM, we suppose up to Section 8.6 that $\pi_i^j(\mathcal{U})$ is known.

3. The third issue is related to the definition of disruption. In Section 7.2, we defined a disruption of call o by call n in QM($\mathcal{U}$) as the fact that n occupies at time c_o in QM($\mathcal{U}$) the operator that serves o in LM. We commented after Corollary 7.3 that in the single-skill model such a situation is equivalent to call n causing a delay in the service of o. We also hinted in Section 7.2 that this might not be true in multi-skill models. The following example based on Figure 8.3 illustrates this.

   Example 8.3 *Suppose a call o_x is disrupted by a call $n_y \in \mathcal{W}_y$ at pool xy. All operators in $\mathcal{Q}^{xy}$ are thus busy. If, as in Figure 8.3, the routing of calls is hierarchical, all operators in $\mathcal{Q}^x$ will be unavailable as the only calls that are routed to pool xy are the ones that could not find an available operator in pool x. If the routing is reverted and pool xy is visited first, the opposite holds: type-x calls are routed to pool x only if there is no available operator at pool xy. In this case, it is possible for o_x to be rejected from pool xy*

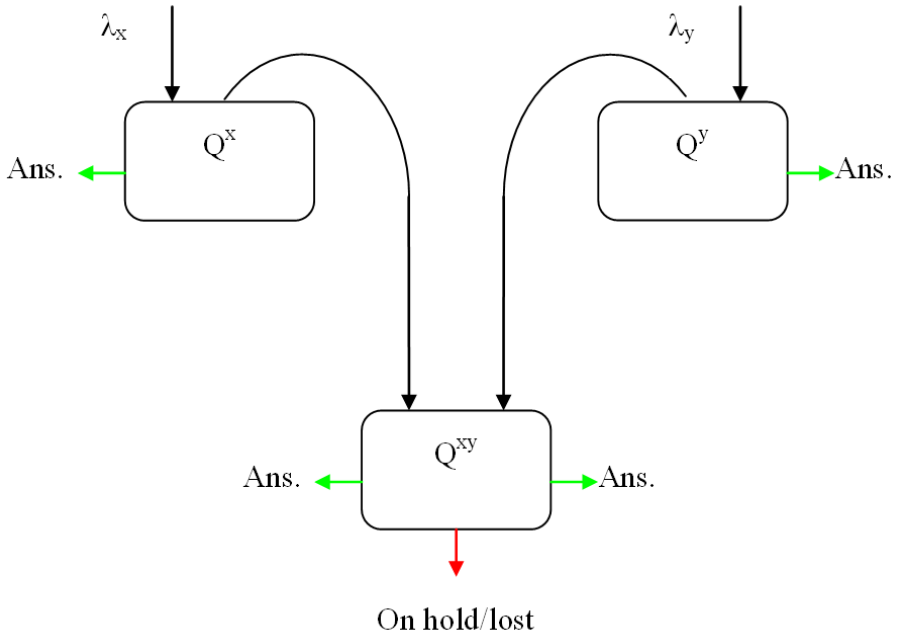

Figure 8.3: Representation of a multi-skill system with two types of calls and three pools.

because of the service of calls from $\mathcal{W}_y$ and to immediately find an operator from pool x, as operators from $\mathcal{Q}^x$ do not serve y-calls.

Example 8.3 shows that under some circumstances a disrupted call might immediately find an available and adequately skilled operator: "his" operator might be busy serving another call whereas skilled operators in other pools are available to answer it. Despite this immediate answer, this call is disrupted, according to Definition 7.2. The earlier interpretation of disruption does not hold and a better description of this event is for a call not to be serviced by the same operator or at the same time in QM($\mathcal{U}$) as compared to LM. Calls in the situation illustrated with Example 8.3 are said to have a zero-waiting time and the event is described as the "zero-waiting-time" phenomenon. Our third assumption is basically that the "zero-waiting-time" phenomenon occurs with a zero probability.

With hierarchical routing, the zero-waiting-time phenomenon is very often negligible. Only operators in downstream pools in the overflow path can immediately serve disrupted calls. With hierarchical routing, dedicated operators, that are more likely to be available when there is a disruption at another pool, are

visited first. In systems with a different routing of calls the assumption on the absence of zero-waiting-time events may cause larger errors.

8.4 A THREE-STEP PROCEDURE

As we said in the previous sections, our intention in this chapter is to exploit the results of Chapter 7 to build an approximation for computing the waiting probability in multi-skill models. Basically we are going to use a procedure similar to the one used in the single-skill model to determine this probability. As a reminder the procedure consisted in first identifying the calls in $\mathcal{R}$, i.e. the lost calls, and compute $P[n \in \mathcal{R}] = B_E(s,a)$, the proportion of all calls they represent, then in determining for each delayed call n the amount of calls that are expected to be disrupted, $E[|\mathcal{G}_n|]$, i.e. in identifying the common ratio of the geometric series induced by the service of n in QM, and in finally computing $C_E(s,a)$ by dividing $P[n \in \mathcal{R}]$ by $1 - E[|\mathcal{G}_n|]$.

In multi-skill models adaptations of this procedure are required. We present them here, along with the definition of a few new sets. The three-step procedure as adapted to multi-skill situations is presented afterward.

First, if we want to compute the proportion of type-i calls expected to wait when they are allowed to queue we need to identify all type-i calls that are lost in the absence of that queue. These include the lost calls in LM but also all i-calls that are disrupted by delayed calls of other types and lost if they cannot wait in queue. In other words, assuming $i \in \mathcal{U}$, we need to find the proportion of lost i-calls in model QM($\mathcal{U} \setminus \{i\}$) to determine the set of calls that generate a new chain of disruption. Let us define $\mathcal{Z}_i(\mathcal{U})$ in the following way.

Definition 8.1 *We define $\mathcal{Z}_i(\mathcal{U})$ as a subset of type-i calls such that:*

$$\mathcal{Z}_i(\mathcal{U}) = \{n_i | n_i \in \mathcal{W}_i(\mathcal{U}),\ (\nexists n_i' \in \mathcal{W}_i(\mathcal{U}) : n_i \in \mathcal{H}_{n_i'}(\mathcal{U}))\}.$$

In other words, $\mathcal{Z}_i(\mathcal{U})$ comprises all the type-i calls for which we cannot find another type-i call in the upstream part of their chain of disruptions.

Example 8.4 *Figure 8.4 represents different sets of calls in a similar way as Figure 7.1 but considers two types of calls and represents different chains of disruptions one might encounter in multi-skill systems. In this realisation of QM, $n_{y,3}$, $n_{y,4}$, $o_{y,2}$ and $p_{y,1}$ are all the elements of $\mathcal{Z}_y(\{x,y\})$. $o_{y,4}$ and $p_{y,2}$ do not belong to this set as $o_{y,4} \in \mathcal{H}_{n_{y,4}}$ and as $p_{y,2} \in \mathcal{H}_{o_{y,2}}$. $n_{x,1}$ and $n_{x,2}$ are the only elements of $\mathcal{Z}_y(\{x,y\})$, so that here, $\mathcal{Z}_y(\{x,y\}) = \mathcal{R}_x$. $o_{x,1}$ and $q_{x,2}$ do not belong to this set as $o_{x,1} \in \mathcal{H}_{n_{x,1}}$ and as $q_{x,2} \in \mathcal{H}_{n_{x,2}}$.*

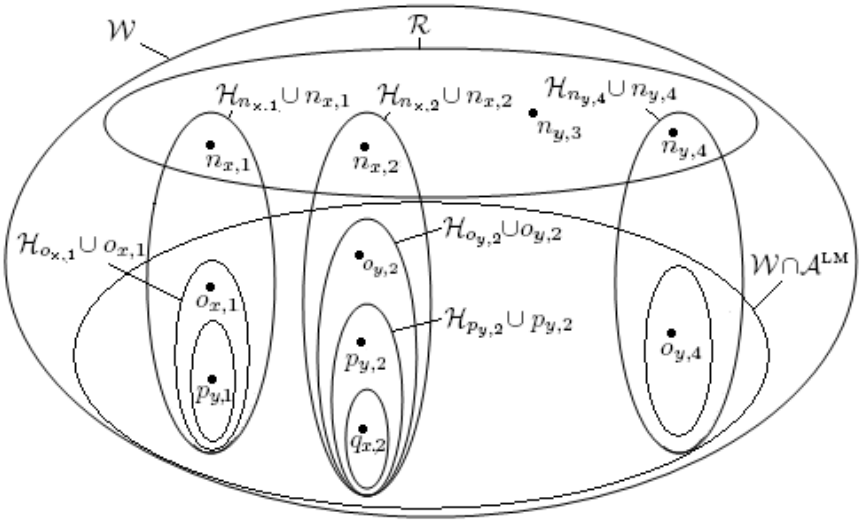

Figure 8.4: Representation of a realisation of QM with two types of calls and the chains of disruptions.

$\mathcal{Z}_i(\mathcal{U})$ comprises the calls in $\mathcal{R}_i$ and all i-calls that are disrupted by calls of other types that do not have any type-i calls in the upstream part of their chain of disruptions. The latter i-calls are divided in sets $\mathcal{O}_{i,i'}(\mathcal{U})$ in the following way.

Definition 8.2 *We define $\mathcal{O}_{i,i'}(\mathcal{U})$ as a subset of $\mathcal{W}_i(\mathcal{U})$ such that:*

$$\mathcal{O}_{i,i'}(\mathcal{U}) = \{o_i | \exists n_{i'} \in \mathcal{W}_{i'}(\mathcal{U}) : o_i \in \mathcal{G}_{n_{i'}}, (\nexists n_i \in \chi_i : n_{i'} \in \mathcal{H}_{n_i}(\mathcal{U}))\}$$

This set contains all calls of type i that are disrupted by calls of type i' that do not have any type-i call in the upstream part of their chain of disruptions. Note that $\mathcal{O}_{i,i'}(\mathcal{U})$ is a subset of $\mathcal{Z}_i(\mathcal{U})$.

Example 8.5 *In Figure 8.4, $o_{y,2}$, $p_{y,1}$ are the two elements of $\mathcal{O}_{y,x}(\{x,y\})$. $\mathcal{O}_{x,y}(\{x,y\})$ is here an empty set: $q_{x,2} \in \mathcal{H}_{p_{y,2}}$ but $p_{y,2} \in \mathcal{H}_{n_{x,2}}$.*

Secondly, the common ratio of the geometric series is more complex than in single-skill models, because of the presence of calls of more than one type. Remember that the common ratio in the single-skill model fives the amount of calls that join the queue as a result of the service of a delayed call. The service of these calls after some waiting will have similar consequences, further extending the geometric series. When looking for the common ratio for type-i calls in a multi-skill model we need to identify the amount of calls of type i that have to join the queue as a consequence of the service

of another type-i call, n_i. The service of these calls will cause similar disruption. This amount differs from $E[|\mathcal{G}_{n_i}|]$ in two ways.

- First $\mathcal{G}_{n_i}$ includes calls of all types. Although they might be delayed because of n_i they should not be accounted in the common ratio as we look at calls of type-i exclusively and, more importantly, they will not cause the kinds of disruptions when served as they might be served by operators outside $\mathcal{P}_i$.
- Secondly, even though calls of other types should not be counted as disrupted calls, they might disrupt, directly or indirectly, calls of type i. As the service of these calls of type i will result in similar disruptions as the disruptions due to n_i and as n_i is the type-i call most direclty responsible for their disruption, the amount of such calls should be included in the common ratio.

Put another way, as the common ratio gives the average number of calls of type i that will be disrupted, join the queue and cause similar disruption, we need to identify all fragments of the chains of disruptions from n_i that result in the disruption of an x-call. The bottom line here is that although we should not include them in the common ratio of the type-i geometric series, we need to identify the number of calls of other types for which n_i is the type-i call most directly responsible for their disruption: it will permit us to identify the number of calls of type-i for which n_i is the closest i-call in the upstream part of their chain of disruption. This is why we introduce a new set of calls, $\mathcal{F}_{n_i,i'}(\mathcal{U})$.

Definition 8.3 *Let $n_i \in \mathcal{W}_i(\mathcal{U})$. We define $\mathcal{F}_{n_i,i'}(\mathcal{U}) \subseteq \mathcal{H}_{n_i}(\mathcal{U})$ as the set of arrivals of type i' such that:*

$$\mathcal{F}_{n_i,i'}(\mathcal{U}) = \{o_{i'} | o_{i'} \in \mathcal{H}_{n_i}(\mathcal{U}), \nexists o_i \in \mathcal{H}_{n_i}(\mathcal{U}) : o_{i'} \in \mathcal{H}_{o_i}(\mathcal{U})\}.$$

$\mathcal{F}_{n_i,i'}(\mathcal{U})$ thus comprises the type-i' calls in $\mathcal{H}_{n_i}(\mathcal{U})$ for which call n_i is the type-i call which is most directly responsible for their disruptions. In what follows we use $\mathcal{F}_{n_i}(\mathcal{U})$ for $\mathcal{F}_{n_i,i}(\mathcal{U})$.

Example 8.6 *Continuing on Figure 8.4, we see that calls $o_{y,2}$ and $p_{y,2}$ both belong to the set $\mathcal{F}_{n_{x,2},y}(\{x,y\})$. $q_{x,2}$ does not belong to $\mathcal{F}_{n_{x,2},y}(\{x,y\})$ but is an element of $\mathcal{F}_{n_{x,2}}(\{x,y\})$. $p_{y,1} \in \mathcal{F}_{o_{x,1},y}(\{x,y\})$ and therefore it does not belong to $\mathcal{F}_{n_{x,1},y}(\{x,y\})$.*

With these two modifications, the three-step methodology used to estimate the proportion of waiting type-i calls in QM($\mathcal{U}$) becomes the following.

STEP 1. Identify $\mathcal{Z}_i(\mathcal{U})$, the set of calls that are first responsible for a chain of disruptions and estimate $P[n_i \in \mathcal{Z}_i(\mathcal{U})]$, the proportion[4] of type-i calls that are included in $\mathcal{Z}_i(\mathcal{U})$. In the previous chapter, when there is only one type of call, this proportion is equal to $B_E(s,a)$. Note that in this step, if $i \in \mathcal{U}$, we compare QM($\mathcal{U}$) to QM($\mathcal{U} \setminus \{i\}$), i.e. the equivalent model without a queue for the i-calls.

STEP 2. For any call $n_i \in \mathcal{Z}_i(\mathcal{U})$ approximate the common ratio of the geometric series. This is equivalent to estimating $E[|\mathcal{F}_{n_i}(\mathcal{U})|]$. In practice, this is done by applying the results in Properties 2 and 3.

STEP 3. Estimate the probability for a type-i call to find all operators from the pools in $\mathcal{P}_i$ busy. This probability is obtained by combining the results of steps 1 and 2 in the following way:

$$\widehat{\mathrm{BP}}_i(\mathcal{U}) = \frac{\widehat{P}[n_i \in \mathcal{Z}_i(\mathcal{U})]}{1 - \widehat{E}[|\mathcal{F}_{n_i}(\mathcal{U})|]}. \tag{8.8}$$

This formula is similar to the formula used in the single-skill model: $\widehat{P}[n_i \in \mathcal{Z}_i(\mathcal{U})]$ is the equivalent of $B_E(s,a)$ and $\widehat{E}[|\mathcal{F}_{n_i}(\mathcal{U})|]$ is the equivalent of $E[|\mathcal{G}_n|]$.

Once again, this procedure provides only an estimation of $\mathrm{BP}_i(\mathcal{U})$. Note also that this procedure applies for calls of type $i \in \mathcal{U}$ as well as for calls of type $i \in \mathcal{V}$. When $i \in \mathcal{V}$, we find at step 2 that $E[|\mathcal{F}_{n_i}(\mathcal{U})|] = 0$ and the procedure provides an estimate for $\mathrm{LP}_i(\mathcal{U})$. When $i \in \mathcal{U}$, we obtain in step 3 an estimate for $\mathrm{WP}_i(\mathcal{U})$.

8.5 TWO TYPES OF CALLS

In this section we use the procedure described in Section 8.4 on models that include two types of calls and three pools of operators, like the one depicted in Figure 8.3. As they are the simplest multi-skill models, we can apply the procedure and illustrate the main concepts while avoiding as much as possible the complications in the notation brought by an increased number of call types and operators. Two types of models are considered. We first analyze the situation where only one of the two types of calls is allowed to queue. After that, in Section 8.5.2, we compute the waiting probability for each type of calls when calls of both types can wait.

4 Despite the fact that the expected amount of calls in one set is scaled such that its value corresponds to a proportion, here, because we are going to compute the blocking probability, we explicitely use the proportion rather the expected number of calls.

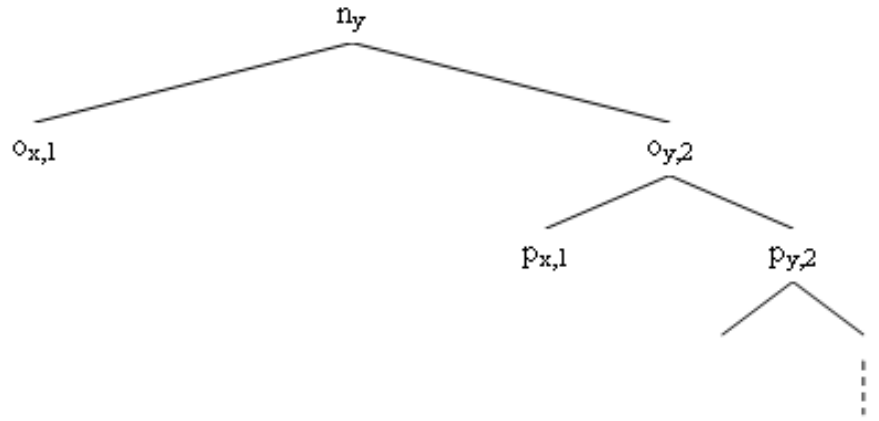

Figure 8.5: A tree representation of the chains of disruptions when only the calls of type y are allowed to wait in queue.

8.5.1 *Intermediate situation: $WP_i(\{y\})$ and $LP_i(\{y\})$*

Let us consider QM($\{y\}$), i.e. we analyze the hybrid situation where $L_x = 0$ and $L_y = \infty$. QM($\{x\}$) can be analyzed in the same way. If only calls of one type can wait, the proportion of blocked calls increases for both types of calls. For type y, whose calls have access to the queue, this need not be further explained. For the x-calls, the amount of rejected calls increases too because delayed y-calls disrupt x-calls that are rejected too.

8.5.1.1 *Estimating $WP_y(\{y\})$*

We start by applying the procedure presented in Section 8.4 to type-y calls. Therefore we will compare the situation in $QM(\{y\})$ with the situation in LM.

STEP 1. As type-y calls are the only ones that can wait, $\mathcal{Z}_y(\{y\}) = \mathcal{R}_y$. As a result:

$$P[n_y \in \mathcal{Z}_y(\{y\})] = \mathrm{LP}_y. \tag{8.9}$$

Figure 8.5 gives the tree representation of the possible disruptions caused by n_y. The dashed vertical lines at the bottom indicate which branches continue to expand downward. All branches with a call of type x terminate. In a similar graph for n_x, there would not be any branch so that the only element of $\mathcal{Z}_y(\{y\})$ in these graphs is n_y.

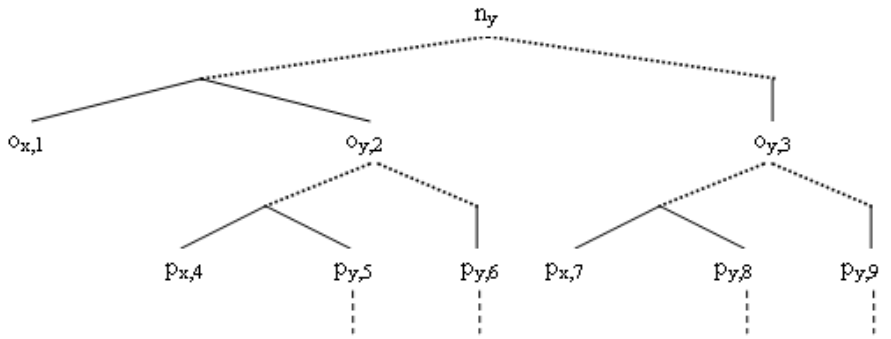

Figure 8.6: A tree representation of the chains of disruptions when only the calls of type y are allowed to wait in queue and which includes the visited pools.

STEP 2. Let $n_y \in \mathcal{W}_y$. n_y will be served by an operator in $\mathcal{Q}^y$ with probability $\pi_y^y(\{y\})$ and by an operator in $\mathcal{Q}^{xy}$ with probability $\pi_y^{xy}(\{y\})$. We modify Figure 8.5 to account for the pool that serves the blocked y-calls by duplicating the branches. In Figure 8.6 the dotted lines represent the pool where a call is served. A dotted line to the left indicates a service at pool xy and a dotted line to the right a service at pool y. As at pool xy a call can disrupt calls of both types, the branch is further divided, with plain lines. At pool y the dotted line is continued with a single plain vertical line as only calls of type y can be disrupted. $\pi_y^{xy}(\{y\})$ gives the probability to go to the left in the tree and $\pi_y^y(\{y\})$ the probability to choose the right path.

Let us first analyze the situation when n_y is serviced at pool xy, i.e. when going to the left branch. There, as calls of both types are serviced, we use Property 3 to estimate that

$$\widehat{E}[|\mathcal{G}_{n_y}(\{y\})| \, | s^{\mathrm{QM}(\{y\})}(n_y) \in \mathcal{Q}^{xy}] = \frac{\lambda_y}{s^{xy}\mu} M_y^{xy} + \frac{\lambda_x}{s^{xy}\mu} M_x^{xy}. \tag{8.10}$$

This is shown in the graph as the dotted branched is divided in two parts. However, because type-x calls are not allowed to queue, they never disrupt any call and can be put aside. Therefore, at pool xy, the total amount of calls disrupted by n_y, that can disrupt other y-calls is estimated as equal to

$$\widehat{E}[|\mathcal{F}_{n_y}(\{y\})| \, | s^{\mathrm{QM}(\{y\})}(n_y) \in \mathcal{Q}^{xy}] = \frac{\lambda_y}{s^{xy}\mu} M_y^{xy}. \tag{8.11}$$

This reflects the fact that the chain is further propagated only through $o_{y,2}$ in the graph.

The analysis when n_y is serviced at pool y is similar: no x-calls can be serviced there and, consequently,

$$\widehat{E}[|\mathcal{F}_{n_y}(\{y\})|\,|s^{\mathrm{QM}(\{y\})}(n_y) \in \mathcal{Q}^y] = \widehat{E}[|\mathcal{G}_{n_y}(\{y\})|\,|s^{\mathrm{QM}(\{y\})}(n_y) \in \mathcal{Q}^y].$$

We deduce from Property 2 that

$$\widehat{E}[|\mathcal{F}_{n_y}(\{y\})|\,|s^{\mathrm{QM}(\{y\})}(n_y) \in \mathcal{Q}^y] = \frac{\lambda_y}{s^y\mu} M_y^y. \tag{8.12}$$

In other words, this permits to take into account that if n_y is serviced at pool y, he might disrupt $o_{y,3}$ which might in turn disrupt other calls at both pools.

We can combine (8.11) and (8.12) along with $\pi_y^{xy}(\{y\})$ and $\pi_y^y(\{y\})$ to find that the common ratio of the geometric series can be estimated as:

$$\widehat{E}[|\mathcal{F}_{n_y}(\{y\})|] = \pi_y^y(\{y\})\frac{\lambda_y}{s^y\mu} M_y^y + \pi_y^{xy}(\{y\})\frac{\lambda_y}{s^{xy}\mu} M_y^{xy}.$$

This is indeed the common ratio of the geometric series as the service of each delayed y-call will have on average the same consequences, as can be seen in Figure 8.6 by comparing the third layer, with the p-labeled calls to the second layer with the o-labeled calls.

STEP 3. By inserting the results of steps 1 and 2 in (8.8) we find that the probability for a type-y call to wait can be approximated as:

$$\widehat{\mathrm{WP}}_y(\{y\}) = \frac{\mathrm{LP}_y}{1 - \pi_y^y(\{y\})\frac{\lambda_y}{s^y\mu} M_y^y - \pi_y^{xy}(\{y\})\frac{\lambda_y}{s^{xy}\mu} M_y^{xy}}. \tag{8.13}$$

8.5.1.2 *Estimating $LP_x(\{y\})$*

As we said, the proportion of lost type-x calls in QM($\{y\}$) is higher than in LM. We can estimate the proportion of lost x-calls with the three-step procedure of Section 8.4.

STEP 1. $\mathcal{Z}_x(\{y\})$ includes $\mathcal{R}_x$ as well as all elements of $\mathcal{A}_x^{\mathrm{LM}}$ that have been rejected because of the service of a call from $\mathcal{W}_y(\{y\})$ by an operator in $\mathcal{Q}^{xy}$. Formally these elements form the set $\{o_x | o_x \in \mathcal{A}_x^{\mathrm{LM}}, (\exists n_y \in \mathcal{W}_y(\{y\}) : o_x \in \mathcal{G}_{n_y})\}$. As pool xy is the only pool of $\mathcal{P}_x \cap \mathcal{P}_y$, where calls of both types are answered, all these disruptions occur at pool xy and only the calls of $\mathcal{W}_y^{xy}(\{y\})$ cause such disruptions. Examples of such elements are calls $o_{x,1}$, $p_{x,1}$ and $p_{x,4}$ in Figure 8.6. They are all disrupted at pool xy.

Based on Property 2 we estimate that each call $n_y \in \mathcal{W}_y^{xy}(\{y\})$ disrupts on average $\frac{\lambda_x}{s^{xy}\mu} M_x^{xy}$ calls of $\mathcal{A}_x^{\text{LM}}$. As $|\mathcal{W}_y^{xy}(\{y\})|_y = \pi_y^{xy}(\{y\})\text{WP}_y(\{y\})$, by including the scale factor λ_y/λ_x we find that the proportion of disrupted type-x calls is evaluated as being equal to $\frac{\pi_y^{xy}(\{y\})\lambda_y}{s^{xy}\mu} M_x^{xy}\text{WP}_y(\{y\})$. Therefore, using the estimate of $\text{WP}_y(\{y\})$ given by (8.13) and adding this proportion to the proportion of lost x-calls in LM, we obtain the following approximation for the proportion of type-x calls in $\mathcal{Z}_x(\{y\})$:

$$\widehat{P}[n_x \in \mathcal{Z}_x(\{y\})] = \text{LP}_x + \frac{\pi_y^{xy}(\{y\})\lambda_y}{s^{xy}\mu} M_x^{xy} \cdot \widehat{\text{WP}}_y(\{y\}).$$

STEP 2. Because all calls of type i that do not find an available skilled operator upon their arrival are rejected, obviously we have $E[|\mathcal{F}_{n_x}(\{y\})|] = 0$.

STEP 3. We therefore obtain the following approximation for the proportion of rejected type-x calls in QM($\{y\}$):

$$\widehat{\text{LP}}_x(\{y\}) = \text{LP}_x + \frac{\pi_y^{xy}(\{y\})\lambda_y}{s^{xy}\mu} M_x^{xy} \cdot \widehat{\text{WP}}_y(\{y\}). \tag{8.14}$$

8.5.2 *Two queues*

We now focus on QM($\{x, y\}$) and find an approximation for $\text{WP}_x(\{x, y\})$. As we focus on the x calls, we are going to compare the situation in QM($\{x, y\}$) with the model that differs from QM($\{x, y\}$) only by the absence of a queue for the type-x calls. This is QM($\{y\}$). We illustrate the different steps of the procedure with Figure 8.7 which represents the disruptions that a call $n_y \in \mathcal{R}_y$ and a call $n_x \in \mathcal{R}_x$ can cause.

STEP 1. $\mathcal{Z}_x(\{x, y\})$ is composed of all calls in $\mathcal{R}_x$ on the one hand and of the type-x calls in the set $\mathcal{O}_{x,y}(\{x, y\})$ on the other hand. As a reminder, the latter set contains all x-calls that are disrupted by calls of $\mathcal{W}_y(\{x, y\})$ that have not been directly or indirectly disrupted by a call of type x, i.e. these y-calls do not have any x-call in the upstream part of their chain of disruptions. In Figure 8.7, n_x is a call of $\mathcal{R}_x$ and $o_{x,1}$, $p_{x,4}$ and $p_{x,7}$ are calls of $\mathcal{O}_{x,y}(\{x, y\})$. They are all the x-calls of their respective chains of disruptions that are the most directly disrupted by n_y.

It is easy to generalize from Figure 8.7 and say that the disruptions of the calls in $\mathcal{O}_{x,y}(\{x, y\})$ all originate at some point from the rejected y-calls in LM. In the tree, $o_{x,1}$ is directly disrupted by n_y and $p_{x,4}$ and $p_{x,7}$ are indirectly disrupted by n_y, through $o_{y,2}$ and $o_{y,3}$, respectively. We now need to identify the set of

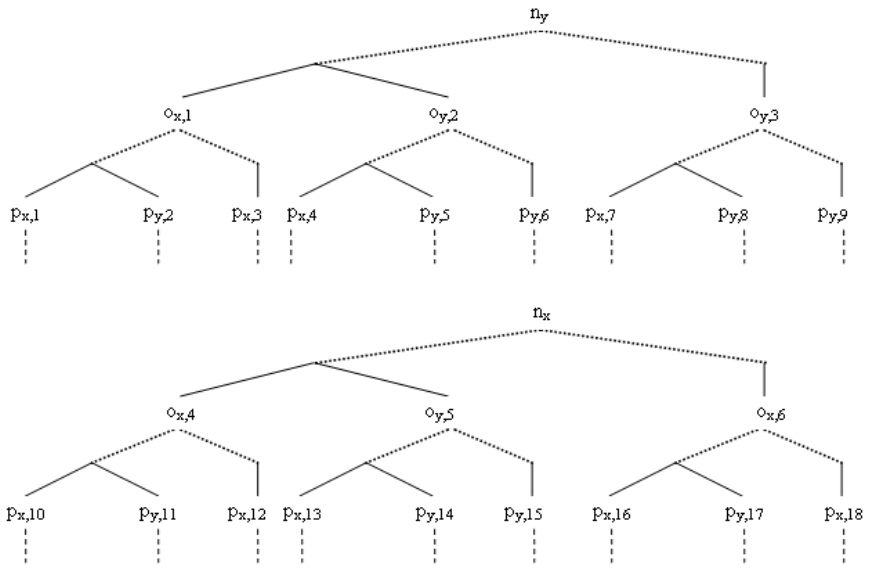

Figure 8.7: Two tree representations of the chains of disruptions when calls of both types are allowed to wait in queue and which include the visited pools. The first tree represents the possible disruptions resulting from serving a call of type y and the second when serving a call of type x.

y-calls that are indirectly disrupted by rejected y-calls without any participation of type-x calls in the chain of disruptions.

If we put aside the disruptions caused by the calls of $\mathcal{O}_{x,y}(\{x,y\})$ and only consider the red part in Figure 8.8, we observe that the situation is close to what is displayed in Figure 8.6. Therefore, the proportion of the type-y calls that do not have any y-calls in the upstream part of their chain of disruptions is close to the proportion of delayed y-calls in QM($\{y\}$), where x-calls cause no disruptions. The only difference is due to the differing allocations of the delayed y-calls to the pools. In QM($\{y\}$), when there are waiting y-calls, there is no competition with delayed x-calls for the capacity at pool xy whereas in QM($\{x,y\}$) delayed calls of both types are answered there. Therefore, $\pi_i^j(\{y\}) \neq \pi_i^j(\{x,y\})\ \forall i,j$. We estimate that the y-calls that do not have any x-call in the upstream part of their chain of disruptions represent a proportion of the y-calls equal to $\frac{\mathrm{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}$.

To estimate the size of $\mathcal{O}_{x,y}(\{x,y\})$, we repeat the arguments used to determine $\widehat{P}[n_x \in \mathcal{Z}_x(\{y\})]$ in the preceding section.

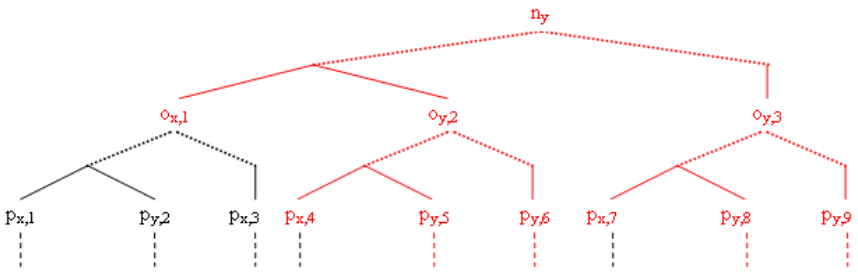

Figure 8.8: Tree representation of the chains of disruptions caused by a type-y call when calls of both types are allowed to wait in queue and which include the visited pools. The red part depicts all fragments of chains that reach a call of type x.

- First, we estimate the proportion of y-calls in $\{n_y | n_y \in \mathcal{W}_y(\{x,y\}), (\nexists n_x \in \chi_x : n_y \in \mathcal{H}_{n_x}(\{x,y\}))\}$ and that are serviced at pool xy. This proportion is equal to $\frac{\pi_y^{xy}(\{x,y\})\mathrm{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}$.
- Second, we rescale this proportion to the x-call referent by multiplying it by λ_y/λ_y.
- Thirdly, as according to Property 3 each of those y-calls will disrupt on average $\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}$ calls of $\mathcal{A}_x^{\mathrm{LM}}$, we estimate that the proportion of x-calls disrupted by those y-calls amounts to $\frac{\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_x^{xy}\cdot\mathrm{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}$.

When we add this quantity to the x-calls lost in LM, we evaluate the proportion of x-calls in $\mathcal{Z}_x(\{x,y\})$ as equal to:

$$\widehat{P}[n_x \in \mathcal{Z}_x(\{x,y\})] = \mathrm{LP}_x + \frac{\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_x^{xy}\cdot\mathrm{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}. \tag{8.15}$$

STEP 2. Let us take a call $n_x \in \mathcal{W}_x(\{x,y\})$ and represented in Figure 8.7. In order to identify the elements in $\mathcal{F}_{n_x}(\{x,y\})$, we first determine $\mathcal{G}_{n_x}(\{x,y\})$.

n_x has a probability $\pi_x^x(\{x,y\})$ of being served by an operator from $\mathcal{Q}^x$, i.e. choosing the right dotted branch in the graph, and, if so, by extrapolating from Property 2, we estimate that n_x will disrupt on average $\frac{\lambda_x}{s^x\mu}M_x^x$ calls of $\mathcal{A}_x^{\mathrm{LM}}$ and,

$$\widehat{E}[|\mathcal{G}_{n_x}(\{x,y\})|\,|s^{\mathrm{QM}(\{x,y\})}(n_x) \in \mathcal{Q}^x] = \frac{\lambda_x}{s^x\mu}M_x^x.$$

n_x also has a probability $\pi_x^{xy}(\{x,y\})$ to be serviced by an operator of $\mathcal{Q}^{xy}$ and thus go to the left branch of the second tree in Figure 8.7. In this case, drawing upon Property 3, we estimate $E[|\mathcal{G}_{n_x}(\{x,y\})|\,|s^{\mathrm{QM}(\{x,y\})}(n_x) \in \mathcal{Q}^{xy}]$ in the following way:

$$\widehat{E}[|\mathcal{G}_{n_x}(\{x,y\})|\,|s^{\mathrm{QM}(\{x,y\})}(n_x) \in \mathcal{Q}^{xy}] = \frac{\lambda_x}{s^{xy}\mu} M_x^{xy} + \frac{\lambda_y}{s^{xy}\mu} M_y^{xy}. \tag{8.16}$$

The second term in (8.16) gives the average quantity of type-y calls that are disrupted. That would be call $o_{y,5}$ in Figure 8.7. These y-calls do not belong to $\mathcal{F}_{n_x}(\{x,y\})$ but as disrupted type-y calls are put on hold before being serviced, they will possibly cause other disruptions, perpetuating a chain of disruptions that could cause the disruption of other calls of $\mathcal{A}_x^{\mathrm{LM}}$. In the tree, call $p_{x,13}$ is such a call. These x-calls belong to $\mathcal{F}_{n_x}(\{x,y\})$. To evaluate the amplitude of this propagating effect, we first need to estimate $E[|\mathcal{F}_{n_x,y}(\{x,y\})|]$, the amount of type-$y$ calls for which n_x is the call of type x most directly responsible for their disruption.

Suppose $\exists o_y \in \mathcal{G}_{n_x}(\{x,y\})$ (obviously, $o_{y,5}$ in our tree). Using once again Properties 2 and 3, we estimate that the amount of calls disrupted by o_y must be equal to

$$\widehat{E}[|\mathcal{G}_{o_y}(\{\cancel{x},y\})\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y + \pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy} + \pi_y^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}.$$

The first two terms give the expected amount of y-calls disrupted in pools y and xy, respectively. In Figure 8.7, with $o_y = o_{y,5}$, they are represented by calls $p_{y,15}$ and $p_{y,14}$, respectively. The third term gives the expected amount of type-x calls disrupted by o_y in pool xy, represented by $p_{x,13}$ in the tree. These x-calls are obviously not elements of $\mathcal{F}_{n_x,y}(\{x,y\})$ and all the y-calls that they disrupt neither: these x-calls are the x-calls most directly responsible for the disruption of such y-calls. In short, only the y-elements that are counted in the first two terms of the expression are elements of $\mathcal{F}_{n_x,y}(\{x,y\})$. Other elements of $\mathcal{F}_{n_x,y}(\{x,y\})$ are all the type-y calls that are indirectly disrupted by the y-calls of $\mathcal{G}_{o_y}(\{x,y\})$ without any interaction with x-calls. These elements are not represented in Figure 8.7 but are in the branches that start from $p_{y,15}$ and $p_{y,14}$. The size of $\mathcal{F}_{n_x,y}(\{x,y\})$ is estimated by taking one y-element of $\mathcal{G}_{o_y}(\{x,y\})$, $p_{y,15}$ for example, and by computing the amount of calls disrupted by this call in the same way as for computing $\widehat{E}[|\mathcal{G}_{o_y}(\{x,y\})|]$, which is easy as the structure

of the tree remains the same in lower branches. By repeating this process an infinite number of times, we eventually find:

$$\widehat{E}[|\mathcal{F}_{n_x,y}(\{x,y\})|\,|s^{\mathrm{QM}(\{x,y\})}(n_x)\in\mathcal{Q}^{xy}] = \frac{\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}.$$

The numerator represents the expected amount of y-calls directly disrupted by n_x as determined with (8.16) and the denominator contains the common ratio of the geometric series that results from the propagation of the disruption to the y-calls.

Because call n_x can disrupt y-calls only if it is served at pool xy, we have $\widehat{E}[|\mathcal{F}_{n_x,y}(\{x,y\})|] = \pi_x^{xy}\widehat{E}[|\mathcal{F}_{n_x,y}(\{x,y\})|\,|s^{\mathrm{QM}(\{x,y\})}(n_x)\in\mathcal{Q}^{xy}]$. As the service of any call $n^y\in\mathcal{W}_y(\{x,y\})$ by an operator of $\mathcal{Q}^{xy}$ occurs with probability $\pi_y^{xy}(\{x,y\})$ and disrupts on average $\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}$ calls of $\mathcal{A}_x^{\mathrm{LM}}$, the amount of calls in $\widehat{E}[|\mathcal{F}_{n_x}(\{x,y\})|]$ is increased and we obtain the following approximation:

$$\begin{aligned}\widehat{E}[|\mathcal{F}_{n_x}(\{x,y\})|] &= \pi_x^x(\{x,y\})\frac{\lambda_x}{s^x\mu}M_x^x+\pi_x^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}\\ &\quad+\pi_x^{xy}(\{x,y\})\,\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}\frac{\pi_y^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}.\end{aligned} \tag{8.17}$$

Looking once more at Figure 8.7, each of the three terms corresponds to one of the three branches that start from n_x. The third term is a collection of branches. All branches terminat when they reach a node with a x-call. $o_{x,4}$ and $o_{x,6}$ are reached directly. In the third term, the branches are longer: one ends at $p_{x,13}$, others continue to lower layers, through $p_{y,14}$ or $p_{y,15}$.

STEP 3. As a result of the first two steps, the waiting probability for a type-x call is estimated as

$$\widehat{\mathrm{WP}}_x(\{x,y\}) = \frac{\mathrm{LP}_x+\frac{\frac{\pi_y^{xy}(\{x,y\})\lambda_y}{s^{xy}\mu}M_x^{xy}\cdot\mathrm{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}}{1-\left(\pi_x^x(\{x,y\})\frac{\lambda_x}{s^x\mu}M_x^x+\pi_x^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}+\frac{\pi_x^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}\pi_y^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y-\pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}\right)}. \tag{8.18}$$

The analysis of (8.18) provides interesting observations. In particular we see how the calls of type y interfere with the type-x calls and influence their waiting probability. First they increase the number of chains of disruptions that cause the delay of x-calls. This increase is accounted for in the second term of the numerator. Roughly speaking these calls would not have been disrupted without the presence of a queue for type-y calls and would be lost in QM($\{y\}$) [5]. Second, disrupted type-y calls also interfere in the chains of disruptions created by the delayed type-x calls. The effect is represented by the last term of the common ratio in the denominator. Note that the two effects are two geometric series with a same common ratio, namely $\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y - \pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}$.

Formula (8.18) is ugly but shows in the best possible way all the interactions in QM($\{x,y\}$) that influence the proportion of delayed x-calls. We propose to simplify this expression by including a new parameter, $\text{SWP}_i(\mathcal{U},\Pi(\mathcal{U}'))$.

We have noted the important similarity between the proportion of disrupted y-calls that have no x-calls in the upstream part of their chain of disruption and the proportion of delayed y-calls in QM($\{y\}$): they have the same structure and only differ by the sets of $\Pi(\mathcal{U})$ that they use. $\pi_i^j(\{y\}) \neq \pi_i^j(\{x,y\})\,\forall i,j$. We define $\text{SWP}_i(\mathcal{U},\Pi(\mathcal{U}'))$ as the waiting probability for type-i calls if the routing probabilities were equal to $\Pi(\mathcal{U}')$ rather than $\Pi(\mathcal{U})$. Obviously,

$$\text{SWP}_y(\{y\},\Pi(\{y\})) = \text{WP}_y(\{y\}).$$

But, for example,

$$\text{SWP}_y(\{y\},\Pi(\{x,y\})) = \frac{\text{LP}_y}{1-\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y - \pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}}.$$

We commented that $\pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y - \pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}$ is the common ratio of the geometric series in $\text{SWP}_y(\{y\},\Pi(\{x,y\}))$ and of the geometric series in the last term of $\widehat{E}[|\mathcal{F}_{n_x}(\{x,y\})|]$. Exploiting this observation, we propose to introduce the following parameter to further simplify (8.18). We define $E[|\mathcal{F}_{n_i}(\mathcal{U}_i,\Pi(\mathcal{U}))|]$ as the common ratio of the geometric series of $\text{SWP}_i(\mathcal{U}_i,\Pi(\mathcal{U}))$. Here,

$$E[|\mathcal{F}_{n_y}(\{y\},\Pi(\{x,y\}))|] = \pi_y^y(\{x,y\})\frac{\lambda_y}{s^y\mu}M_y^y - \pi_y^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy},$$

5 This is not completely true, as we discussed in Step 1.

and

$$E[|\mathcal{F}_{n_y}(\{y\},\Pi(\{y\}))|] = \widehat{E}[|\mathcal{F}_{n_y}(\{y\})|] = \pi_y^y(\{y\})\frac{\lambda_y}{s^y\mu}M_y^y - \pi_y^{xy}(\{y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}.$$

Including these two simplifications in (8.18) yields

$$\widehat{\mathrm{WP}}_x(\{x,y\}) = \frac{\mathrm{LP}_x + \frac{\pi_y^{xy}(\{x,y\})\lambda_y}{s^{xy}\mu}M_x^{xy}\cdot \mathrm{SWP}_y(\{y\},\Pi(\{x,y\}))}{1-\left(\pi_x^x(\{x,y\})\frac{\lambda_x}{s^x\mu}M_x^x+\pi_x^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}+\frac{\pi_x^{xy}(\{x,y\})\frac{\lambda_y}{s^{xy}\mu}M_y^{xy}\pi_y^{xy}(\{x,y\})\frac{\lambda_x}{s^{xy}\mu}M_x^{xy}}{1-E[|\mathcal{F}_{n_y}(\{y\},\Pi(\{x,y\}))|]}\right)}. \tag{8.19}$$

As from the routing point of view the system is symmetric for x and for y, we can obtain $\mathrm{WP}_y(\{x,y\})$ with the same procedure, just by inverting x and y.

8.6 GENERALIZATION TO I TYPES OF CALLS

We now extend the approximation to systems where J pools of operators answer calls of I different types. Keeping the assumptions of Section 8.3, this extension raises no major difficulty compared to the approximation for two types of calls and three different pools presented in Section 8.5. Only the increased number of call types and pools makes the notation a little bit more complex. We therefore introduce additional elements of notation.

We define $\mathcal{U}_i \subseteq \mathcal{U}$ in the following way:

$$\begin{cases} \mathcal{U}_i = \mathcal{U} & \text{if} \quad i \notin \mathcal{U} \\ \mathcal{U}_i = \mathcal{U}\setminus\{i\} & \text{otherwise.} \end{cases}$$

Similarly, $\mathcal{V}_i \supseteq \mathcal{V}$ is defined as:

$$\begin{cases} \mathcal{V}_i = \mathcal{V} & \text{if} \quad i \notin \mathcal{U} \\ \mathcal{V}_i = \mathcal{V}\cup\{i\} & \text{otherwise.} \end{cases}$$

We do not provide as many details as for the description of the three-steps procedure in the models with two types of calls. Very often, as the procedure is similar, we refer to the procedures of the preceding section for a more detailed explanation and only give the results as adapted for the increased number of call types and pools.

8.6.1 $|\mathcal{U}| = 1$

Suppose now first that only type-i calls can wait before being serviced when there is no operator available with the required skill. We estimate $\mathrm{WP}_i(\{i\})$ by generalyzing (8.13). All steps proceed similarly. In Step 2, the analysis of the amount of disrupted calls at one pool is repeated for each pool where operators can answer type-i calls. The analysis itself does not change as even though calls of many types can be disrupted by the service of delayed i-calls, only disrupted calls of type i could propagate the disruptions. In the end we find:

$$\widehat{\mathrm{WP}}_i(\{i\}) \quad = \quad \frac{\mathrm{LP}_i}{1 - \sum_{j \in \mathcal{P}_i} \pi_i^j(\{i\}) \frac{\lambda_i}{s^j \mu} M_i^j}.$$

The procedure used to estimate $\mathrm{LP}_{k \neq i}(\{i\})$ is exactly the same as the one that leads to (8.14). The only change is caused by the possibly higher number of pools where k-calls can be disrupted by i-calls that increases the number of terms in $\widehat{\mathrm{LP}}_{k \neq i}(\{i\})$:

$$\widehat{\mathrm{LP}}_{k \neq i}(\{i\}) \quad = \quad \mathrm{LP}_k + \sum_{j \in \mathcal{P}_k} \pi_i^j(\{i\}) \frac{\lambda_i}{s^j \mu} M_k^j \cdot \widehat{\mathrm{WP}}_i(\{i\}).$$

8.6.2 $|\mathcal{U}| \geq 1$

Let us now consider any set $\mathcal{U}$ and analyze what happens to calls of type i.

STEP 1. In order to estimate $\mathcal{Z}_i(\mathcal{U})$, we make the same observation as the one for $\mathcal{Z}_x(\{x, y\})$ in Section 8.5: $\mathcal{Z}_i(\mathcal{U})$ consists in all elements of $\mathcal{R}_i$ but also in all calls of $\mathcal{A}_i^{\mathrm{LM}}$ which are disrupted by calls of other types for which no type-i call can be held responsible, directly or indirectly, for their own disruption. Here, we need to identify the proportion of calls in $\mathcal{O}_{i,k}(\mathcal{U})\ \forall k \in \mathcal{U}_i$, i.e. the set of i-calls that are disrupted by type-k calls and that do not have any i-call in the upstream part of their chain of disruptions. This is done by looking at one call at a time and apply the same procedure as for identifying $\mathcal{O}_{x,y}(\{x, y\})$ in Section 8.5.2. The only complication comes from the higher number of pools where calls interact.

Let us consider call type $k \in \mathcal{U}_i$. We use the same arguments as the ones used to find (8.15) to deduce that a good estimate for the proportion of type-k calls that

are delayed and that do not have any type-i call in the upstream part of their chain of disruptions, $P[n_k \in (\mathcal{W}_k(\mathcal{U}) \setminus (\cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U})))]$, would be:

$$\widehat{P}[n_k \in (\mathcal{W}_k(\mathcal{U}) \setminus \cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U}))] = \mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U})) \quad \forall k \in \mathcal{U}_i.$$

Note here that $\mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U}))$ is similar in structure to $\mathrm{WP}_k(\mathcal{U}_i)$ in the same way as $\mathrm{SWP}_y(\{y\}, \Pi(\{x, y\}))$ is similar to $\mathrm{WP}_y(\{y\})$.

The calls in $\mathcal{W}_k(\mathcal{U}) \setminus \cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U})$ are dispatched to the different pools of the system according to $\pi_k^j(\mathcal{U})$, $\forall j \in \mathcal{J}$. Therefore, the proportion of k-calls in $\mathcal{W}_k(\mathcal{U}) \setminus \cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U})$ is equal to

$$\begin{aligned}\widehat{P}[n_k \in (\mathcal{W}_k(\mathcal{U}) \setminus \cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U})), s^{\mathrm{QM}(\mathcal{U})}(n_k) \in \mathcal{Q}^j] \\ = \pi_k^j(\mathcal{U}) \cdot \mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U})) \qquad \forall k \in \mathcal{U}_i,\ \forall j \in \mathcal{J}. \qquad (8.20)\end{aligned}$$

Note that $\pi_k^j(\mathcal{U}) = 0$, $\forall j \notin \mathcal{P}_k$.

Type-i calls can be disrupted by type-k calls at each pool $j \in (\mathcal{P}_i \cap \mathcal{P}_k)$, that can answer both type-i and type-k calls. For each pool $j \in (\mathcal{P}_i \cap \mathcal{P}_k)$, the service of a delayed type-k call will disrupt on average $\frac{\lambda_i}{s^j \mu} M_i^j$ type-i calls. By combining this result with (8.20) and by using the rescaling factor λ_k / λ_i, we can finally estimate that the proportion of i-calls disrupted by type-k calls of $\mathcal{W}_k(\mathcal{U}) \setminus \cup_{n_i \in \mathcal{W}_i(\mathcal{U})} \mathcal{H}_{n_i}(\mathcal{U})$ will be of the form $\pi_k^j(\mathcal{U}) \frac{\lambda_k}{s^j \mu} M_i^j \cdot \mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U}))$. By summing on all pools in $\mathcal{P}_i$, we obtain an estimate for $P[n_i \in \mathcal{O}_{i,k}(\mathcal{U})]$, namely:

$$\widehat{P}[n_i \in \mathcal{O}_{i,k}(\mathcal{U})] = \sum_{j \in \mathcal{P}_i} \pi_k^j(\mathcal{U}) \frac{\lambda_k}{s^j \mu} M_i^j \cdot \mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U})).$$

Note that each term in this sum corresponds to one pool where calls of type i can be serviced. For the pools that cannot serve calls of type k, the corresponding term is equal to 0 as $\pi_k^j(\mathcal{U}) = 0$ in this case.

If we repeat this procedure for each $k \in \mathcal{U}_i$ and if we add the resulting proportions to the proportion of rejected calls in $\mathcal{A}_i^{\mathrm{LM}}$, we obtain the following generalization of (8.15):

$$\widehat{P}[n_i \in \mathcal{Z}_i(\mathcal{U})] = \mathrm{LP}_i + \sum_{j \in \mathcal{P}_i} \sum_{k \in (\mathcal{S}^j \cap \mathcal{U}_i)} \pi_k^j(\mathcal{U}) \frac{\lambda_k}{s^j \mu} M_i^j \cdot \mathrm{SWP}_k(\mathcal{U}_i, \Pi(\mathcal{U})).$$

STEP 2. As we could obtain $\widehat{P}[n_i \in \mathcal{Z}_i(\mathcal{U})]$ from a generalization of the method used to get (8.15), obtaining an approximation for $E[|\mathcal{F}_{n_i}(\mathcal{U})|]$ is done by generalizing

the approach leading to (8.17) to I call types and J pools. It suffices to repeat the procedure presented in Step 2 of Section 8.5.2 to each pool of $\mathcal{P}_i$ and each call type of $\mathcal{U}_i$. This gives

$$E[|\mathcal{F}_{n_i}(\mathcal{U})|] = \sum_{j\in\mathcal{P}_i}\Big(\frac{\pi_i^j(\mathcal{U})\lambda_i}{s^j\mu}M_i^j + \sum_{k\in\mathcal{U}\cap\mathcal{S}^j}\frac{\pi_i^j(\mathcal{U})\lambda_k}{s^j\mu}M_k^j\sum_{l\in\mathcal{S}_i}\frac{\pi_k^l(\mathcal{U})\frac{\lambda_i}{s^l\mu}M_i^l}{1-E[|\mathcal{F}_{n_i}(\mathcal{U}_i,\Pi(\mathcal{U}))|]}\Big).$$

The first terms in the sum represent the amount of i-calls that are directly disrupted by call n_i at each pool of $\mathcal{P}_i$. The terms in the second sum give the amount of i-calls for which n_i is the call of type i most directly responsible for their disruptions. Each of these term is obtained by repeating the analysis to find the third and last term in (8.17). Note that this expression is equal to zero when $i \in \mathcal{V}$ as in this case $\pi_i^j(\mathcal{U}) = 0$, $\forall j \in \mathcal{J}$.

STEP 3. Combining the results of the two first steps in (8.8) yields our approximation for the probability for a call of type i to be put on hold or rejected. Formally we have:

$$\widehat{\mathrm{BP}}_i(\mathcal{U}) = \frac{\mathrm{LP}_i + \sum_{j\in\mathcal{P}_i}\sum_{k\in(\mathcal{S}^j\cap\mathcal{U}_i)}\pi_k^j(\mathcal{U})\frac{\lambda_k}{s^j\mu}M_i^j\cdot\mathrm{SWP}_k(\mathcal{U}_i,\Pi(\mathcal{U}))}{1-\sum_{j\in\mathcal{P}_i}\left(\frac{\pi_i^j(\mathcal{U})\lambda_i}{s^j\mu}M_i^j + \sum_{k\in\mathcal{U}\cap\mathcal{S}^j}\frac{\pi_i^j(\mathcal{U})\lambda_k}{s^j\mu}M_k^j\sum_{l\in\mathcal{S}_i}\frac{\pi_k^l(\mathcal{U})\frac{\lambda_i}{s^l\mu}M_i^l}{1-E(|\mathcal{F}_{n_i}(\mathcal{U}_i,\Pi(\mathcal{U}))|)}\right)} \quad \forall i\in\mathcal{U}. \tag{8.21}$$

8.7 NUMERICAL APPLICATION

We now test the formulae of Sections 8.5 and 8.6 by comparing them to what is obtained with simulations. We first detail the approximations and methods used to evaluate the parameters required in the formulae. We then describe and comment the results. Finally we end with a brief conclusion.

8.7.1 *Approximations*

In section 8.3 we presented three assumptions we used to build the approximation formulae for computing the blocking probability. We assumed

1. that the performance in LM is known,

2. that $\pi_i^j(\mathcal{U})$ is known too and

3. that a disrupted call has a non-zero waiting time.

In this section we detail how to effectively compute approximated values for the parameters in the first two assumptions. By narrowing the field of application of our approximations to systems where hierarchical routing holds, we believe that the third assumption has a limited impact on the quality of the results: this impact is even nonexistent when there are only two types of calls.

8.7.1.1 *Evaluating the performance in LM*

To estimate the performance of LM we use the Hayward-based multi-skill methodology detailed in Section 5.3, i.e. we choose the method that includes the correlation between subflows in the system and differentiates the loss probability at each pool in function of the peakedness. From our own experience this is the method that best fits to this queueing application. Using another version of the method proposed in Chapter 5 is possible too.

8.7.1.2 *Estimating $\pi_i^j(\mathcal{U})$*

$\pi_i^j(\mathcal{U})$ is the unique parameter in the formulae of Sections 8.5 and 8.6 that is not a result observed in LM. Therefore it needs to be estimated.

If there is only one queue, say for type-i calls, then the probability for a call of this type to be answered by an operator from pool $j \in \mathcal{P}_i$ is equal to the probability that an operator of pool j is first to become available. In this case, as all service times are exponentially distributed, we have

$$\pi_i^j(\{i\}) = \frac{s^j \mu}{\sum_{k \in \mathcal{P}_i} s^k \mu} \quad \forall j \in \mathcal{P}_i.$$

This result is nothing else than a generalization of (8.4), where $I = 2$.

When more than one type of calls can wait before service, operators from pool j can possibly select for service a waiting call of another type than i, if there is one. Although the likelihood of choosing a type-i call in this situation depends on the number of calls of other types in queue at the time, we assume there exists a constant, i.e. independent of the situation in the queues, value for $p_i^j(\mathcal{U})$, the probability that an available operator from $\mathcal{Q}^j$ chooses a call from $\mathcal{W}_i$ to serve it in QM($\mathcal{U}$). Then,

$$\begin{aligned}\pi_i^j(\mathcal{U}) &= p_i^j(\mathcal{U})\pi^j \sum_{a=0}^{\infty}\left[\sum_{k\in\mathcal{J}} \pi^k(1-p_i^k(\mathcal{U}))\right]^a \\ &= \frac{\pi^j p_i^j(\mathcal{U})}{\sum_{k\in\mathcal{J}} \pi^k p_i^k(\mathcal{U})}, \end{aligned} \tag{8.22}$$

where $\pi^j = \frac{s^j\mu}{\sum_{k\in\mathcal{J}} s^k\mu}$ is the probability that the next available operator comes from pool j. There is here a geometric series, with common ratio $\sum_{k\in\mathcal{J}} \pi^k(1-p_i^k(\mathcal{U}))$ that gives the probability that an operator in another pool is first to finish a service and that he selects a call of another type.

Finding an approximate value for $p_i^j(\mathcal{U})$ is difficult. Intuitively, $p_i^j(\mathcal{U})$ should be proportional to the average number of calls of each type in the queue. However, as we can only find very loose proxies for these values, we rather make the assumption that $p_i^j(\mathcal{U})$ is proportional to the average utilisation of the operators in $\mathcal{Q}^j$ by calls of type-i. Although it is something that is not known, we can approximate it if we know the capacity dedicated to each call type.

The capacity dedicated to each call type, that we note as $s_i\mu$, can be estimated with the following method. It relies on the observation that in a multi-skill queueing model all calls will be answered. Consequently, the dispatch of calls in the queue must be such that $s_i\mu > \lambda_i$ for all $i \in \mathcal{I}$ ([Armony, 1999]). If this inequality does not hold for one of the call types then the system cannot reach a steady-state.

The method consists in considering a minimum capacity for each call type, equal to λ_i, on the one hand and an exceeding capacity of $s_i\mu - \lambda_i$ on the other hand. We call the exceeding capacity idle capacity or idleness, as on average this capacity is not used.

What really needs to be divided among the call types is the idle capacity. We will use a fluid approach where we first estimate the total idleness for each pool and then "allocate" this idleness among the call types. We define IC^j as the average idle capacity at pool j.

To evaluate IC^j, we suppose that the routing is hierarchical. If s_i^j is the number of operators allocated on average to type-i calls, then:

- if $\sum_{k\in\mathcal{S}^j} \lambda_k^j \geq s^j\mu$, then $\widehat{\mathrm{IC}}^j = 0$ and $s_i^j = \frac{\lambda_i^j}{\sum_{k\in\mathcal{S}^j}\lambda_k^j} s^j$. The overflow λ_{out}^j, is equal to $\lambda_{\text{out}}^j = \sum_{k\in\mathcal{S}^j}\lambda_k^j - s^j\mu$. For type-$i$ calls, the outgoing flow, $\lambda_{i,\text{out}}^j$, is equal to $\lambda_{i,\text{out}}^j = \lambda_i^j - s_i^j\mu$.
- If $\sum_{k\in\mathcal{S}^j} \lambda_k^j < s^j\mu$, then $\widehat{\mathrm{IC}}^j = s^j\mu - \sum_{k\in\mathcal{S}^j}\lambda_k^j$, $s_i^j = \lambda_i^j + \widehat{\mathrm{IC}}^j \frac{\lambda_i \mathrm{LP}_i}{\sum_{k\in\mathcal{S}^j}\lambda_k \mathrm{LP}_k}$, $\quad \forall i \in \mathcal{S}^j$ and $\lambda_{i,\text{out}}^j = 0$.

The total capacity allocated to one type of call is equal to the sum of the capacities allocated at each pool.

At each pool, we estimate that the utilization of the operators by each type of call is proportional to the estimated capacity allocated to each call type. We thus estimate $p_i^j(\mathcal{U})$ as

$$\widehat{p}_i^j(\mathcal{U}) = \frac{s_i^j}{s^j}.$$

If we introduce this last result in (8.22) and simplify, we end up with

$$\widehat{\pi}_i^j(\mathcal{U}) = \frac{s_i^j}{\sum_{k\in\mathcal{J}} s_i^k}.$$

8.7.2 *Numerical results*

The numerical study consists in comparing the method presented in Sections 8.5 and 8.6, that we call GSM[6], to simulation results. We use the datasets presented in Tables A.1, A.2, A.4 and A.5 in Appendix A. The first and second datasets contains configurations of systems with two types of calls and the two other datasets consists in configurations with with five types of calls and different numbers of pools. They are all described in Section 2.5.

8.7.2.1 *Two types of calls*

For two types of calls we analyze three different models, the ones with $\mathcal{U} = \{x, y\}$, $\mathcal{U} = \{x\}$ and $\mathcal{U} = \{y\}$, respectively. The results presented in the tables are the mean values of the differences in absolute value, AAbsErr, between the results obtained by simulation and the results computed with GSM. The mean difference, Mean AErr is also displayed along with the minimum and the maximum differences. For each model, we present the errors per call type. For the pure queueing model, i.e. when $\mathcal{U} = \{x, y\}$, we also present the errors on the overall waiting probability in the systems. These errors are presented in column "all calls".

For illustrative purposes, we also present the results with the first dataset in graphs. They are displayed in Figures 8.9, 8.10 and 8.11.

The results show that the approximation gives good estimates of the waiting probability with on average less than 0.02 difference to the simulated probabilities. The results are remarkably good when there is only one queue. This is due to the fact that $\pi_i^j(\{i\})$ is exactly computed as $p_i^{xy}(\{i\}) = 1$, $i \in \{x, y\}$. This is further confirmed by the results obtained when, instead of using Hayward's approximation, we compute

6 GSM stands for Geometric Series Method.

	$\mathcal{U} = \{x, y\}$			$\mathcal{U} = \{x\}$		$\mathcal{U} = \{y\}$	
	x-calls	y-calls	all calls	x-calls	y-calls	x-calls	y-calls
Mean AAbsErr	0.0154	0.0157	0.0068	0.0088	0.0054	0.006	0.01
Mean AErr	-0.0038	0.0085	0.0043	0.0085	0.0046	0.006	0.0053
Min AErr	-0.0788	-0.0522	-0.01	-0.0031	-0.0048	0	-0.0278
Max AErr	0.0599	0.0591	0.0159	0.055	0.0165	0.0253	0.0274

Table 8.1: Mean difference in absolute value and mean, minimum and maximum differences observed on the 90 configurations of the first dataset between the simulated and the computed results for the three possible models that include a queue.

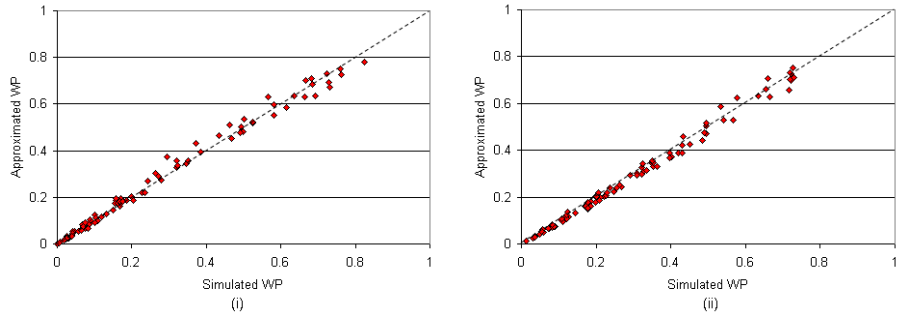

Figure 8.9: Comparison of the blocking probabilities observed in simulation with the GSM approximation when $\mathcal{U} = \{x, y\}$ for the dataset of Table A.1. (i) gives WP_x and (ii) gives WP_y.

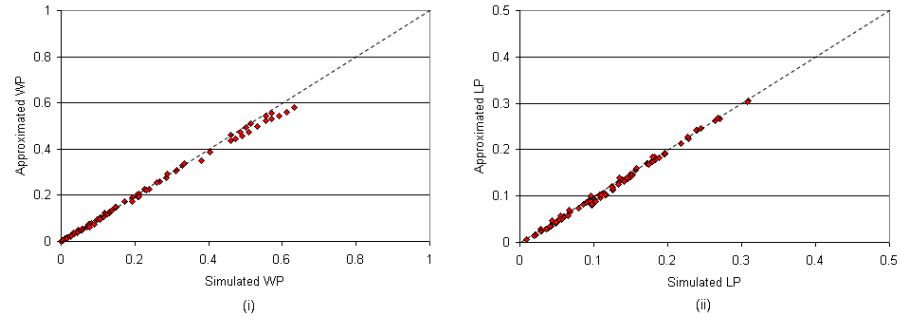

Figure 8.10: Comparison of the blocking probabilities observed in simulation with the GSM approximation when $\mathcal{U} = \{x\}$ for the dataset of Table A.1. (i) gives WP_x and (ii) gives LP_y.

$\mathrm{WP}_i(\{i\})$ with the values of M_i^j observed when simulating LM: the average error is more or less equal to 0.0005 for both types of calls. This error is well within the expected accuracy of our simulations and this suggest that the results provided by GSM are correct in the conditions described above.

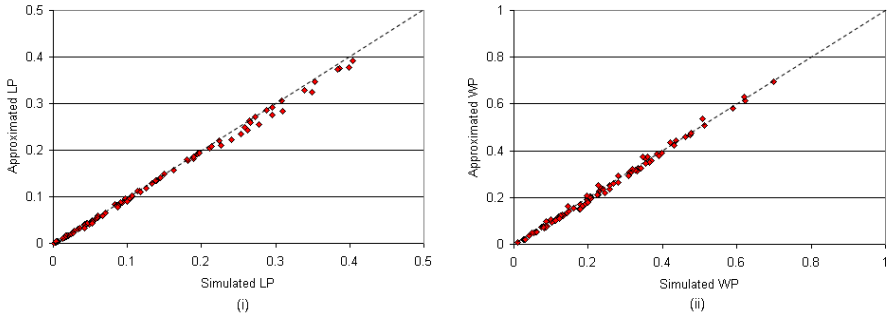

Figure 8.11: Comparison of the blocking probabilities observed in simulation with the GSM approximation when $\mathcal{U} = \{y\}$ for the dataset of Table A.1. (i) gives LP_x and (ii) gives WP_y.

When there is a queue for each of the two call types, the error may become large, as indicated by the value of the maximum error. We isolated the five configurations where the difference is higher than 0.05 for at least one of the two call types. These are configurations 6, 36, 42, 48 and 61. All five are systems where the incoming loads or the number of dedicated operators are unbalanced. For all five cases, the error is large for both call types but the error on the overall waiting probability, i.e. when we consider all the calls together is smaller and similar to what is observed in other configurations. Both observations suggest that the GSM approach performs well as it permits to accurately estimate the overall waiting probability and that the secondary approximations described in Section 8.7.1 that determine the loss probability at each pool and for each call type and the allocation of waiting calls to the different operators are responsible for an important part of the error in the waiting probability per call type. We believe that a better approximation of these two parameters would decrease the error.

In all three models the average difference, Mean AErr, is smaller than the average difference in absolute value, Mean AAbsErr. This suggests that the approximation is not significantly biased in comparison to the simulation results.

Figures 8.9 to 8.11, visually confirm our conclusions from the tables. They also tend to show that the difference tend to be higher when the blocking probability is higher. The results are excellent in all cases anyway.

There are no significant differences in quality between the results of Table 8.1 and the results presented in Table 8.2. All the conclusions from Table 8.1 apply for the larger systems presented here. This basically means that the quality of the approximation is not affected by the size of the system. This is very encouraging as systems with higher loads are more frequent in call center applications.

	$\mathcal{U} = \{x, y\}$			$\mathcal{U} = \{x\}$		$\mathcal{U} = \{y\}$	
	x-calls	y-calls	all calls	x-calls	y-calls	x-calls	y-calls
Mean AAbsErr	0.0181	0.0148	0.0103	0.0051	0.0008	0.0012	0.0056
Mean AErr	-0.0099	-0.0123	-0.0103	0.0013	-0.0007	0.0001	-0.0053
Min AErr	-0.0686	-0.1011	-0.0269	-0.0128	-0.0025	-0.0028	-0.0268
Max AErr	0.0643	0.0089	0	0.0287	0.0005	0.0058	0.0013

Table 8.2: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset presented in Table A.2 between the simulated and the computed results when there is a queue for only one of the two call types.

The only noticeable difference is that a comparison of the mean difference in absolute value and the mean difference when $\mathcal{U} = \{x, y\}$ reveals the presence of a bias in the approximation method, at least in the whole system (all calls). The approximation overestimates the waiting probability. Because this is the only case in Table 8.2 where it occurs and because the difference, 0.0103, is small, we do not believe it causes much concern to the approximation.

8.7.2.2 *Five types of calls.*

When there are five types of calls the number of different models increases. We therefore only simulate some of all the possible models. We arbitrarily chose to compare the cases where $\mathcal{U} = \{v, w, x, y, z\}$, (Table 8.3), $\mathcal{U} = \{y\}$ (Table 8.4), $\mathcal{U} = \{v, w\}$ (Table 8.5), $\mathcal{U} = \{w, x, y\}$ (Table 8.6) and $\mathcal{U} = \{v, x, y, z\}$ (Table 8.7). Note that in Table 8.3 we also present the statistics for the overall waiting probability.

	v-calls	w-calls	x-calls	y-calls	z-calls	all calls
Mean AAbsErr	0.0243	0.0272	0.0356	0.0239	0.0242	0.0077
Mean AErr	0.0074	0.0225	0.0041	-0.003	-0.0145	0.0075
Min AErr	-0.0778	-0.0362	-0.1234	-0.0735	-0.067	-0.0012
Max AErr	0.1253	0.1147	0.1179	0.0454	0.0393	0.0425

Table 8.3: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset with systems with five type of calls between the simulated and the computed results when there is a queue for each of the call types.

The results presented in the above tables are excellent. They confirm the qualities of the GSM approach that were observed in Section 8.7.2.1: the mean error is usually inferior to 0.02. It tends to be slightly higher when the size of $\mathcal{U}$ increases. Note however that, as suggested by the maximum error in all tables, the difference between the

	v-calls	w-calls	x-calls	y-calls	z-calls
Mean AAbsErr	0.0101	0.0107	0.009	0.0168	0.009
Mean AErr	0.0004	0.008	0.0055	0.0143	-0.0081
Min AErr	-0.0189	-0.0092	-0.0171	-0.0189	-0.042
Max AErr	0.0282	0.0437	0.021	0.0684	0.0023

Table 8.4: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset with systems with five type of calls between the simulated and the computed results when there is a queue for only one of the call types.

	v-calls	w-calls	x-calls	y-calls	z-calls
Mean AAbsErr	0.0215	0.0166	0.0083	0.0137	0.007
Mean AErr	-0.0144	0.0036	0.0047	0.0126	-0.005
Min AErr	-0.0681	-0.0711	-0.0154	-0.0083	-0.0252
Max AErr	0.0285	0.0402	0.0167	0.0592	0.0055

Table 8.5: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset with systems with five type of calls between the simulated and the computed results when there is a queue for two of the call types.

	v-calls	w-calls	x-calls	y-calls	z-calls
Mean AAbsErr	0.0088	0.0139	0.0167	0.0119	0.0059
Mean AErr	0.0086	0.0056	-0.0078	-0.006	-0.0012
Min AErr	-0.0068	-0.0367	-0.0848	-0.0581	-0.0151
Max AErr	0.0345	0.0449	0.0362	0.0142	0.0123

Table 8.6: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset with systems with five type of calls between the simulated and the computed results when there is a queue for three of the call types.

	v-calls	w-calls	x-calls	y-calls	z-calls
Mean AAbsErr	0.0192	0.0181	0.0252	0.0222	0.0405
Mean AErr	0.0022	0.0194	-0.0172	-0.0235	-0.0321
Min AErr	-0.1192	0.0034	-0.1317	-0.1356	-0.1186
Max AErr	0.0517	0.049	0.0597	0.0127	0.023

Table 8.7: Mean difference in absolute value and mean, minimum and maximum differences observed on the dataset with systems with five type of calls between the simulated and the computed results when there is a queue for four of the call types.

simulated and estimated blocking probabilities can be quite large. In particular, focusing on Table 8.3, when we analyze the configurations with higher errors we observe that the maximum error on all calls is much smaller, very much as in Section 8.7.2.1, suggesting again that the approximations described in Section 8.7.1 are responsible for an important part of the errors. The maximum error on all calls is however more

important (0.0425) than the one observed with only two call types (0.0159 or 0.0269, depending on the dataset). This is probably due to the third assumption which assumes that disrupted calls have a waiting time higher than zero. As expected, the effect of this assumption is limited when the routing is hierarchical. The difference between the simulated and approximated performance does not reveal a significant bias. Unsurprisingly the error is more important when we compute a waiting probability than when we compute a loss probability.

Finally, we present in Table 8.8 the results when we use GSM on the dataset of Table A.5, i.e. when there are fourteen different pools. Because of the small size of the dataset we only present the results for the pure queueing multi-skill system: we can then aggregate the waiting probability for each type of call together and have a sufficiently large sample to draw some conclusions.

		WP
Individual calls	Mean AAbsErr	0.0304
	Mean AErr	0.0245
	Min AErr	-0.0316
	Max AErr	0.0709
All calls	Mean AAbsErr	0.0301
	Mean AErr	0.0301
	Min AErr	0.0162
	Max AErr	0.0574

Table 8.8: Errors observed on the waiting probability when estimating it on the dataset of Table A.5.

The key observation here is that despite the increased complication of the structure, GSM continues to perform well, with an average difference of 0.03. Note also that the average difference, Mean AErr, suggests that the approximation tends to underestimate the waiting probability.

8.8 CONCLUDING REMARKS

In this chapter, we used the interpretation of Formula (7.1) presented in Chapter 7 as a basis for developing an approximation of the performance of systems with more than one type of calls and for which blocked calls of one type have access to a waiting queue and can wait instead of being rejected. Numerical experiments show that the method provides excellent results for systems with hierarchical routing.

The problem of the disrupted calls that are immediately serviced by an operator in a downstream pool is the big weakness of the GSM approach when there is a large number of pools. This can probably be solved to some extent by adapting the approxi-

mation procedure, making it much more reliable for configurations with other routing policies than hierarchical. For example, we could develop an iterative procedure in the following way. It would start from the situation where it is assumed that no disrupted call has a zero waiting time. One would then focus on the first level pools and deduce from the WP_i the rate of calls that are not serviced immediately at each of the first-level pools. The resulting rates would become the incoming flows to the pools at the next level and new loss probabilities, or equivalently the proportion of calls answered immediately[7], would be computed at each pool of this and the following levels. One would use the new loss values to estimate new waiting probabilities in the system. The procedure could be repeated, handling at some point the second level pools, then the following ones, in a similar manner as the ones of the first level, until it converges to a solution. Developing such a procedure and analyzing its convergence is left for future research.

Another possible extension for our method would be to use it in models where waiting calls only have access to a subset of the pools of operators.

7 The difference between the initial proportion of answered calls found in the loss system and the new proportion would be an approximation for the disrupted calls that have a zero waiting time and are eventually serviced at that pool.

9

PERFORMANCE ESTIMATION BASED ON SINGLE-SKILL EQUIVALENT SYSTEMS

The GSM approach presented in the previous chapter permits to accurately estimate the blocking, i.e. loss or waiting, probability in a multi-skill queueing system. It is accurate but it has two weaknesses in our opinion. Firstly the required assumption about the absence of "zero-waiting-time" events limits its field of application. Secondly Formula (8.21) becomes cumbersome when the number of call types and the number of operators increase. Although such an increase does not really matter in practice as implementing it in a computing tool is not particularly difficult, it is not very amenable to further analysis or utilisation in other applications.

In this chapter we present an alternative method, that we call MAO[1], to compute the waiting probability in multi-skill models where all calls can wait before being serviced. The advantage of the method is that it is simpler to use than the GSM approach and can be easily exploited to compute other performance measures of queueing systems: the average waiting time and the service level.

In the first section of this chapter, we present a simple approximation for the waiting probability. This is followed by a section where we propose two bounds and an approximation for the average waiting time. In Section 9.3 we present a formula to estimate the service level in a multi-skill pure queueing system. In all three sections we provide visual illustrations of the results that can be obtained with these approximations, using the dataset of Table A.1. After that, in Section 9.4 we analyze the quality of our approximations based on the datasets described in Section 2.5 and displayed in Appendix A, in a similar way as we did in Section 8.7.2 for GSM. We end with a few comments on the strengths and limitations of the approximations presented in this chapter. We also list some of the possible extensions.

1 For Method with Allocation of Operators.

9.1 APPROXIMATING THE WAITING PROBABILITY

As in the GSM approach we use the connection between loss and queueing systems in the single-skill model to estimate the waiting probability in multi-skill systems. We propose here to start from the multi-skill loss system and build a new system in which a single pool of dedicated operators is associated to each call type. Basically we build I single-skill loss systems that do not interact with each other. We suppose that the loss probability in each of the single-skill systems is equal to the loss probability of the associated call type in the multi-skill system. The I single-skill loss systems are thus each equally efficient as the multi-skill loss system. We then use (7.1) with each of the I single-skill loss systems to compute the waiting probability in the I equivalent single skill queueing systems that we can build from the loss systems. The resulting waiting probabilities are our approximations for the waiting probabilities in the original multi-skill queueing system. Figure 9.1 illustrates the approach.

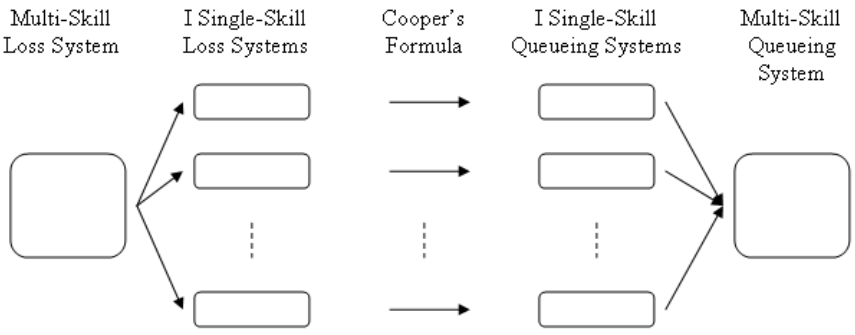

Figure 9.1: Comparison of the waiting probability observed in simulation with the approximation based on the equivalent loss systems. (i) gives WP_x and (ii) gives WP_y.

Based on this and drawing upon (7.1) we estimate the waiting probability for calls of type i as:

$$\widehat{\mathrm{WP}}_i = \frac{\mathrm{LP}_i}{1 - a_i/s_i(1 - \mathrm{LP}_i)}, \tag{9.1}$$

where, like in the previous chapters,

LP_i is the loss probability for type-i calls in the equivalent loss system,

a_i is the load for type i arrivals ($a_i = \lambda_i/\mu_i$),

s_i is the equivalent number of operators that handle type-i calls in the single-skill system.

LP_i can be estimated with the methods presented in Chapter 5. The load a_i is given. Finding s_i, the number of operators handling type i calls in the single-skill system is more difficult as we do not precisely know to what it actually corresponds. It could be the number of operators required to have a loss probability equal to LP_i in the single-skill system or it could be related to the capacity available for call type i in the multi-skill model. Or it could be something else.

We conducted two experiments to find what would be a good evaluation for s_i in the equivalent single-skill loss model. Both experiments used the dataset of Table A.3 that contains configurations with three types of calls and seven different pools. We simulated the systems as queueing and equivalent loss models and recorded the resulting waiting and loss probabilities for each call type.

In the first experiment we determined the value of s_i such that using s_i and a_i as input parameter for Formula (4.1) results in a loss probability equal to LP_i. We then used the resulting s_i, along with the corresponding a_i and LP_i, in Formula (9.1) to estimate WP_i. The obtained results were quite different from the simulated waiting probabilities with differences (Mean AAbsErr) as high as 0.13. We therefore concluded that s_i is not the number of operators required in the equivalent single-skill system to have a loss probability equal to the one observed in the multi-skill system.

In the second experiment we took the opposite approach and first determined what should be the value of s_i for (9.1) to work effectively. After simulating the multi-skill queueing and equivalent loss systems to determine the waiting and loss probabilities, respectively, we introduced the resulting values in (9.1) and isolated s_i to obtain its desired value. When summing the values of s_x, s_y and s_z in each configuration, with $\sum s_i$ as result, we observed that it is higher but usually close to the sum of all operators, $\sum s^j$, in the multi-skill system. The resulting sums are presented in Table 9.1.

The difference is high for configurations 2 and 8 but both are systems with very low loss and waiting probabilities. Putting aside these two outliers, we see that values in both columns are similar. We conclude that there is a connection between the number of operators in the multi-skill system and the number of operators in the single-skill "equivalent" systems. This implies that there is a division of the operators in the system that would permit to obtain a close estimate of the waiting probability from (9.1). Therefore, despite the presence of outliers, we propose to consider that s_i is the average number of operators to handle type-i calls in the multi-skill queueing system. This needs to be estimated as well.

In Section 8.7.1.2, we presented a fluid-based approach to estimate the proportion of delayed calls of one type that are serviced at each pool. In the procedure we build

Conf.	$\sum s_i$	$\sum s^j$
1	28.72	28
2	43.84	38
3	33.73	32
4	28.98	27
5	23.63	23
6	24.63	24
7	30.79	28
8	38.00	33
9	26.60	25
10	17.82	17
11	20.15	19
12	15.10	15

Table 9.1: Comparison of the sum of the desired s_x, s_y and s_z, i.e. $\sum s_i$ with the sum of all operators in the system, $\sum s^j$, for each configuration of the dataset in Table A.3.

an estimate of the capacity dedicated to each type of calls in each pool. We propose to use this approximation to compute s_i. In other words, if $\widehat{s}_i^j$ is the capacity dedicated to call type i at pool j as estimated with the method presented in Section 8.7.1.2, $s_i = \sum_{j \in \mathcal{P}_i} \widehat{s}_i^j$.

For the sake of clarity we present here a close but alternative description of this approximation:

- First we determine an estimate of the global idleness for each pool, IC^j, with a fluid approximation and then "allocate" this idleness among the call types. Suppose a flow of rate λ is sent to pool j with capacity $s^j\mu$.
  - if $\lambda > s^j\mu$, then there is an overflow of rate $\lambda - s^j\mu$ and $\text{IC}^j = 0$. The overflow is then sent to other pools, according to the routing paths.
  - if $\lambda < s^j\mu$, then there is sufficient capacity and all calls are handled. There is an idle capacity: $\text{IC}^j = s^j\mu - \lambda$.

  Note that if the flow rate is a combination of several different flows with $\lambda = \sum_{i=1}^{I} \lambda_i$, then the utilized capacity is shared proportionally to the flow rates. Because of the routing policy, the idle capacity is found at the pools with the more polyvalent operators of the system.

- After that, we divide the idle capacity between all call types that can be answered by an operator in pool j. The purpose of the idle capacity is to absorb the delays in service due to the burstiness of the traffic (the higher the idle capacity, the quicker a queue is emptied). The allocation of the idleness depends on the amount of delayed calls of each type: if there are many calls of one type in

the queues, they will use many operators. We should therefore allocate a greater part of the idle capacity to them.

Obviously, a good estimate for the amount of delayed workload is the flow of waiting calls, $\lambda_i \mathrm{WP}_i$. Because it is our objective to compute WP_i we here use another metric to estimate the latter: the loss probability, LP_i. In doing this we assume that $\mathrm{LP}_i/\mathrm{WP}_i$ is constant. The allocation of the idle capacity at pool j is thus:

$$\widehat{\mathrm{IC}}_i^j = \frac{\lambda_i \mathrm{LP}_i}{\sum_{k \in \mathcal{S}^j} \lambda_k \mathrm{LP}_k} \widehat{\mathrm{IC}}^j.$$

- Eventually, we obtain s_i in the following way:

$$s_i \mu = \lambda_i + \sum_{j \in \mathcal{S}_i} \widehat{\mathrm{IC}}_i^j.$$

Note that if in the fluid approximation, some calls are overflown from the system, i.e. after visiting the last pool there is a call type i for which the overflow is higher than zero, the system is declared to be not stable.

Like we said earlier on, the waiting probability computed with (9.1) serves as our estimate for the waiting probabilities per call type in the original multi-skill queueing model. Figure 9.2 presents a comparison between the results obtained with (9.1) and the simulation results for the waiting probability for the dataset of Table A.1.

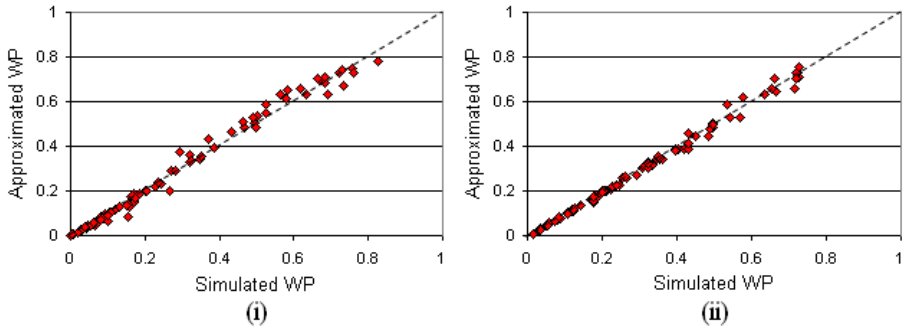

Figure 9.2: Comparison of the waiting probability observed in simulation with the approximation based on the equivalent loss systems. (i) gives WP_x and (ii) gives WP_y.

These results show that the approximation is accurate. It tends to be more accurate when the waiting probability is lower and it is a bit less precise for systems with higher waiting probabilities. Section 9.4 analyzes the results in more details.

It is interesting to note that, provided both methods use the same approximations for the loss probability and for the allocation of the idle capacity, Formula (9.1) gives the same results as Formula (8.19) when the system consists in two call types and when $\lambda_x \geq s^x\mu$ and $\lambda_y \geq s^y\mu$. In this case, $s_i\mu = \lambda_i + (s^{xy}\mu + s^x\mu + s^y\mu - \lambda_x - \lambda_y)\frac{\lambda_i \mathrm{LP}_i}{\lambda_x \mathrm{LP}_x + \lambda_y \mathrm{LP}_y}$ and algebraic manipulations show that both formulae are equivalent. This is also the case when the system is symetrical in load and number of operators. This proximity provides a light theoretical background to the purely intuitive approach presented here.

9.2 THE AVERAGE WAITING TIME

Another important performance measure is the average waiting time before being served. In a $M/M/s$ system, it is equal to:

$$\mathrm{WT} = \frac{C_E(s,a)}{s\mu - \lambda}. \tag{9.2}$$

We observe that the average waiting time is the ratio of the waiting probability (i.e. the proportion of time when there is a queue) over the idle capacity in the queueing system. Because the average waiting time is equal to the average waiting time of the calls with a strictly positive waiting time multiplied by the probability of waiting, we find that $\frac{1}{s\mu-\lambda}$ is the expected waiting time for the calls that have to wait. The denominator, $s\mu - \lambda$, is the difference between the service rate when there are waiting calls, and the rate of arrivals.

9.2.1 *Bounding the average waiting time*

Intuitively, the waiting times for one type of calls depend on the waiting times for other types of calls, because of the FCFS rule and the shared operators. The effect is important even when the shared capacity is small: for a waiting call of a particular type, the fact that calls of other types are handled quickly (thanks to their dedicated operators for instance) increases the probability for the former to be handled quickly at the cross-trained pool.

From this observation we can derive bounds on the estimations of the waiting time. Indeed, a lower bound is obtained if we suppose that all calls of all types equally benefit from the service of all operators. This means that the system would be equivalent to a single pool system with $s = \sum_{j\in\mathcal{J}} s^j$ and $\lambda = \sum_{i\in\mathcal{I}} \lambda_i$. Therefore, an estimate of the lower bound for the average waiting time, $\mathrm{WT}_{i,low}$, would be:

$$\widehat{\mathrm{WT}}_{i,low} = \frac{\widehat{\mathrm{WP}}_i}{\left(\sum_{j\in\mathcal{P}_i} s^j\mu - \sum_{k\in\mathcal{I}} \lambda_k\right)}. \tag{9.3}$$

On the other hand we can derive an upper bound on the waiting time if we suppose that the system behaves as if there was no interaction between the different call types. To compute this we use the equivalent number of operators dedicated to type-i calls as derived in section 9.1. An estimate for this upper bound, $\mathrm{WT}_{i,up}$ would thus be:

$$\widehat{\mathrm{WT}}_{i,up} = \frac{\widehat{\mathrm{WP}}_i}{(s_i\mu - \lambda_i)}. \tag{9.4}$$

We use the same dataset as in the previous section to illustrate these bounds. In Figure 9.3 we compare these bounds (vertical axis) with the simulated average waiting times (horizontal axis). We observe that the values obtained by computation are very

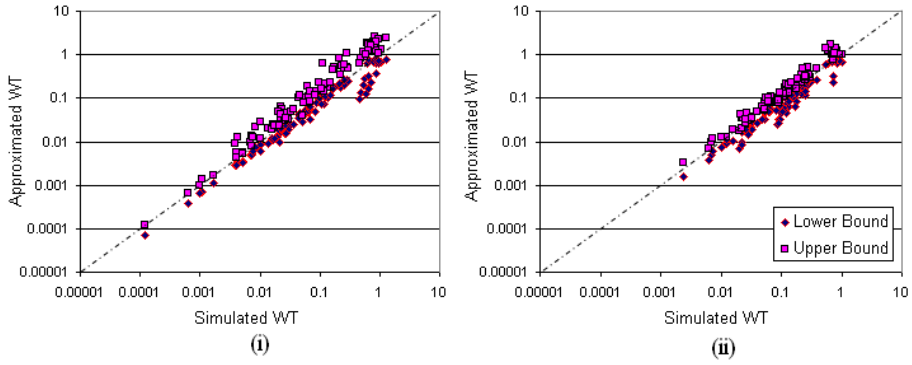

Figure 9.3: The bounds on the average waiting time, as functions of the simulated waiting time. (i) gives the bounds on WT_x and (ii) on WT_y. The axes have a logarithmic scale.

good bounds on the waiting time: the upper bounds lie above the 45 degrees line while the lower bounds are below.

9.2.2 *An approximation for the waiting time*

The difference between the total service rate and the arrival rate is actually the rate at which the queue is emptied. Drawing upon Formula (9.2), we build an approximation for the average waiting time in multi-skill queueing systems by estimating the average speed at which a queue is emptied. Practically, to estimate the speed at which the queue for calls of type i is emptied, we are going to compare the total service rate of

the operators in pools of $\mathcal{P}_i$ to the arrival rate of all calls that are serviced by operators in pools of $\mathcal{P}_i$, provided that calls of type-i are served from the queue. For the sake of clarity, we describe the approximation for a system with two types of demands. Generalizing it to systems with more types of calls does not raise any difficulty but would require a cumbersome notation.

Let $L_i(t)$ be the number of type-i calls in the queue at time t and let us focus on call type x. When $L_x(t)$ is strictly positive, i.e. arriving x-calls have to wait, all operators in pools of $\mathcal{P}_x$ are busy. Therefore, the total service rate when $L_x(t) > 0$ is equal to $\sum_{j \in \mathcal{P}_i} s^j \mu = s^x \mu + s^{xy} \mu$. These operators serve calls of type x but could also serve calls of type y, depending on whether there are y-calls waiting. The arrival rate of the calls serviced by operators of $\mathcal{P}_i$ varies in function of $L_y(t)$.

Let AR_i be the arrival rate of the calls serviced by operators in the pools of $\mathcal{P}_i$ when $L_i(t)$ is strictly positive and let $\mathrm{AR}_i(t)$ denote the rate of the arrivals to the pools of $\mathcal{P}_i$ at time t. When $L_x(t)$ is strictly positive, two situations are possible:

1. $L_y(t) = 0$. Because x-calls are waiting, no arriving y-call will be serviced at pool xy until $L_y(t)$ becomes strictly positive or until the x-queue is emptied. In this case, $\mathrm{AR}_x = \lambda_x$.

2. $L_y(t) > 0$. In this case, the rate of the arrivals serviced by the operators of pool xy is equal to the portion of x-calls and the portion of y-calls that are being serviced by an operator in $\mathcal{Q}^{xy}$ when calls of both types are waiting. Let π_y^{xy} [2] be the proportion of waiting y-calls that are answered at pool xy. We assume that this proportion is independent of the situation in the other queue. This parameter is estimated using $\widehat{\pi}_i^j = \frac{s_i^j}{\sum_{k \in \mathcal{J}} s_i^k} = \frac{s_i^j}{s_i}$. In doing this we assume that the proportion of calls answered by pool j is equal to the portion of pool j capacity used by i-calls in the total capacity available for them. In short, when $L_y(t) > 0$ we estimate that $\widehat{\mathrm{AR}}_x = \lambda_x + \widehat{\pi}_y^{xy} \lambda_y$.

Putting these results together, along with their associated probability we find that the arrival rate for the pools of $\mathcal{P}_x$ when there is a queue can be estimated as

$$\widehat{\mathrm{AR}}_x = \widehat{P}[L_y(t) > 0 | L_x(t) > 0] \cdot \widehat{\mathrm{AR}}_{x,1} + \widehat{P}[L_y(t) = 0 | L_x(t) > 0] \cdot \widehat{\mathrm{AR}}_{x,2}, \tag{9.5}$$

where $\widehat{\mathrm{AR}}_{i,k}$ is the estimate of the arrival rate to the pools of $\mathcal{P}_i$ when there are waiting calls of type i and when $L_y(t)$ is in situation k, as estimated with our method. Index k

2 In section 8.3 we defined $\pi_i^j(\mathcal{U})$ as the probability that a waiting type- i is serviced by an operator of pool j in model $\mathcal{U}$. We here drop $(\mathcal{U})$ as we only work with pure queueing models.

is equal to 1 when $L_y(t) > 0$ and to 2 when $L_y(t) = 0$. Let $P[L_y(t) > 0|L_x(t) > 0]$ be the probability that the first situation occurs. We propose to evaluate it with:

$$\widehat{P}[L_y(t) > 0|L_x(t) > 0] = \widehat{\mathrm{WP}}_y + (1 - \widehat{\mathrm{WP}}_y)\frac{s^{xy}}{s^y + s^{xy}}. \tag{9.6}$$

This approximation is based on a linear interpolation between two extreme cases: $\mathcal{Q}^{xy} = \emptyset$ on the one hand and $\mathcal{Q}^y = \emptyset$ on the other hand. In the first case the situation in the type-y queue is independent of the queue of x calls and we have $P[L_y(t) > 0|L_x(t) > 0, \mathcal{Q}^{xy} = \emptyset] = P[L_y(t) > 0|\mathcal{Q}^{xy} = \emptyset] = \mathrm{WP}_y$. In the second case, when there are no dedicated operators for type-y calls, they all have to be served by an operator of pool xy. When there are x-calls waiting, all operators of pool xy are busy. Consequently, $P[L_y(t) > 0|\mathcal{Q}^y = \emptyset] = P[L_x(t) > 0|\mathcal{Q}^y = \emptyset]$ and $P[L_y(t) > 0|L_x(t) > 0, \mathcal{Q}^y = \emptyset] = 1$.

Introducing (9.6) in (9.5) yields:

$$\widehat{\mathrm{AR}}_x = \lambda_x + \widehat{\pi}_y^{xy}\lambda_y \left(\widehat{\mathrm{WP}}_y + (1 - \widehat{\mathrm{WP}}_y)\frac{s^{xy}}{s^y + s^{xy}}\right). \tag{9.7}$$

The generalization to any number of call types and pools is given below:

$$\widehat{\mathrm{AR}}_i = \sum_{j \in \mathcal{P}_i} \sum_{k \in \mathcal{S}^j} \widehat{\pi}_k^j \lambda_k \left(\widehat{\mathrm{WP}}_k + (1 - \widehat{\mathrm{WP}}_k)\frac{\sum_{l \in (\mathcal{P}_k \cap \mathcal{P}_i)} s^l}{\sum_{l \in \mathcal{P}_k} s^l}\right). \tag{9.8}$$

This gives us the following estimate for the expected waiting time:

$$\widehat{\mathrm{WT}}_i \quad = \quad \frac{\widehat{\mathrm{WP}}_i}{\sum_{j \in \mathcal{P}_i} s^j \mu - \widehat{\mathrm{AR}}_i}. \tag{9.9}$$

We tested Formula (9.9) on the dataset of Table A.1. The comparison with simulation experiments is presented in Figure 9.4.

The results of Figure 9.4 show again that the quality of the approximation is quite good, with some deterioration for the heavily loaded systems. Note that we switched to a logarithmic scale in order to have more evenly spread values and to have relative errors rather than absolute errors. This presentation does not reflect the relative accuracy of our approximation, which is not as good as the one for the waiting probability.

9.3 THE SERVICE LEVEL

A third measure of performance is the service level. Recall that it gives the proportion of calls that wait less than a given time limit before being serviced.

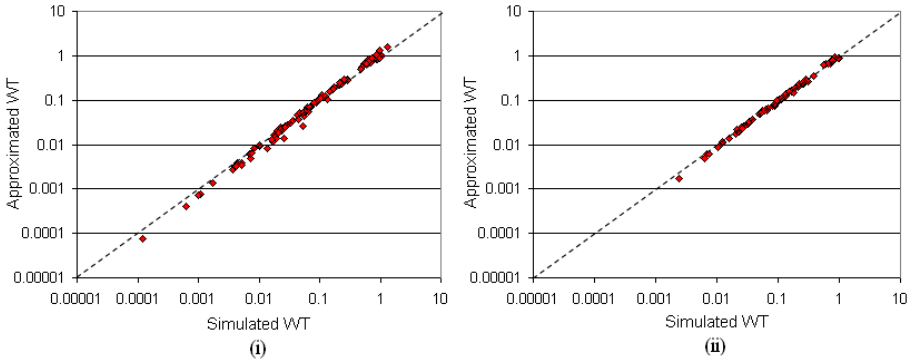

Figure 9.4: The approximation of the average waiting time for the x-calls (i) and the y-calls (ii) based on computations. The axes have a logarithmic scale.

As we already mentioned in Section 4.3.6, in a single-skill $M/M/s$ setting, it is easy to compute a service level because the distribution of the waiting time is known. Conditionally on the fact that an arrival has to wait, the waiting time is exponentially distributed. Formally,

$$P(\mathrm{WT} < t | \mathrm{WT} > 0) = 1 - e^{-(s\mu-\lambda)t}. \tag{9.10}$$

The service level is thus computed as:

$$P(\mathrm{WT} < t) = \mathrm{WP} \cdot (1 - e^{-(s\mu-\lambda)t}) + 1 - \mathrm{WP} = 1 - \mathrm{WP} \cdot e^{-(s\mu-\lambda)t}. \tag{9.11}$$

The parameter of the exponential distribution is $s\mu - \lambda$, which, as we discussed in the previous section, can be interpreted as the difference between the service rate when the queue is not empty and the arrival rate. Based on this fact we propose the following approximation for the service level:

$$\widehat{P}(\mathrm{WT}_i < t | \mathrm{WT}_i > 0) = 1 - e^{-(\sum_{j\in\mathcal{P}_i} s^j\mu - \widehat{\mathrm{AR}}_i)t} \tag{9.12}$$

$$\widehat{P}(\mathrm{WT}_i < t) = 1 - \widehat{\mathrm{WP}}_i \cdot e^{-(\sum_{j\in\mathcal{P}_i} s^j\mu - \widehat{\mathrm{AR}}_i)t}. \tag{9.13}$$

In other words, we suppose that the waiting time distribution in a multi-skill system is exponential with a rate equal to $\sum_{j\in\mathcal{P}_i} s^j\mu - \widehat{\mathrm{AR}}_i$.

We tested this approximation on the same dataset as in the preceding sections. We first check our approximation for the conditional waiting time distribution by comparing computation results obtained with (9.12) to simulations. Figures 9.5 (i) to

(v) present the results for maximum waiting time of 5, 10, 25, 50 and 100 percent of the average service time.

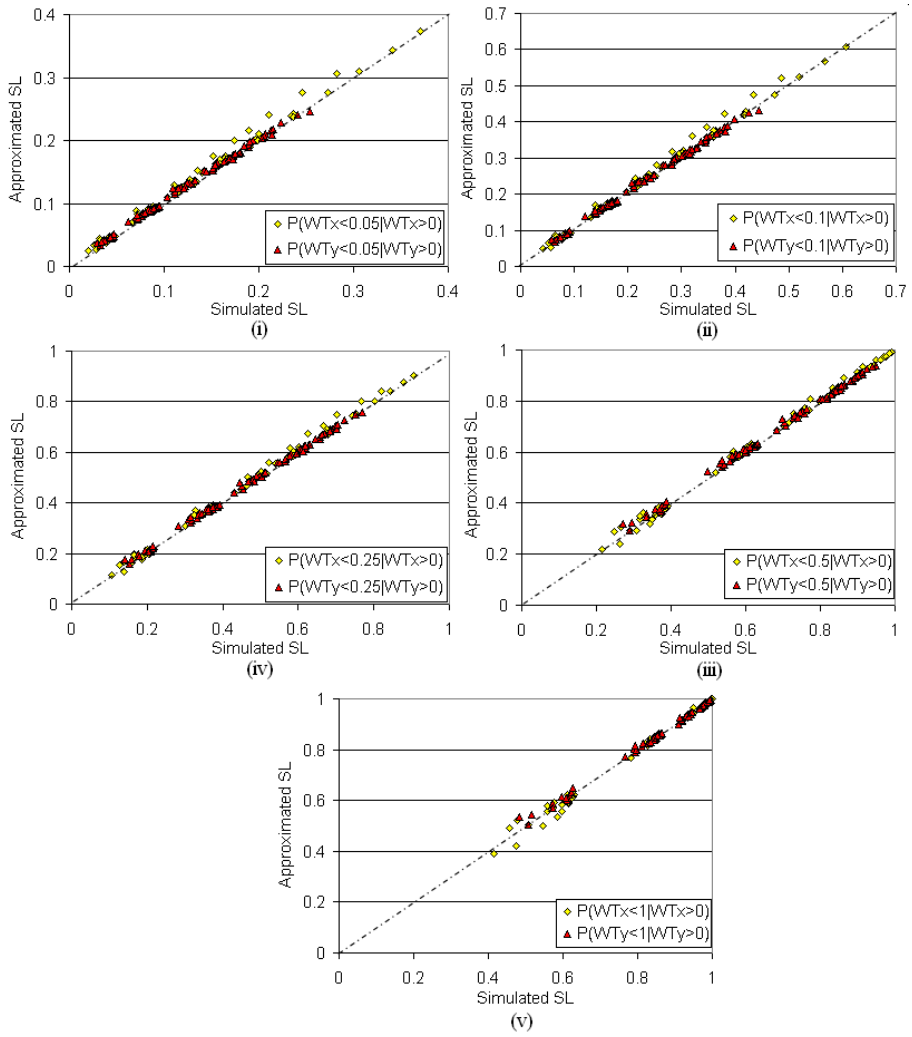

Figure 9.5: Approximation of the conditional service level for five different maximum waiting times.

As it may be observed, the results are particularily good for smaller maximum waiting times. There are some deviations at the higher ones (Cases (iv) and (v)) for the smallest probabilities. Once again this corresponds to the systems with large waiting probabilities: the waiting time is usually very high in these cases, resulting in a small proportion of calls answered within the proposed bounds.

In Figures 9.6 (i) to (iii), we present the unconditional service level as it is approximated with Formula (9.13).

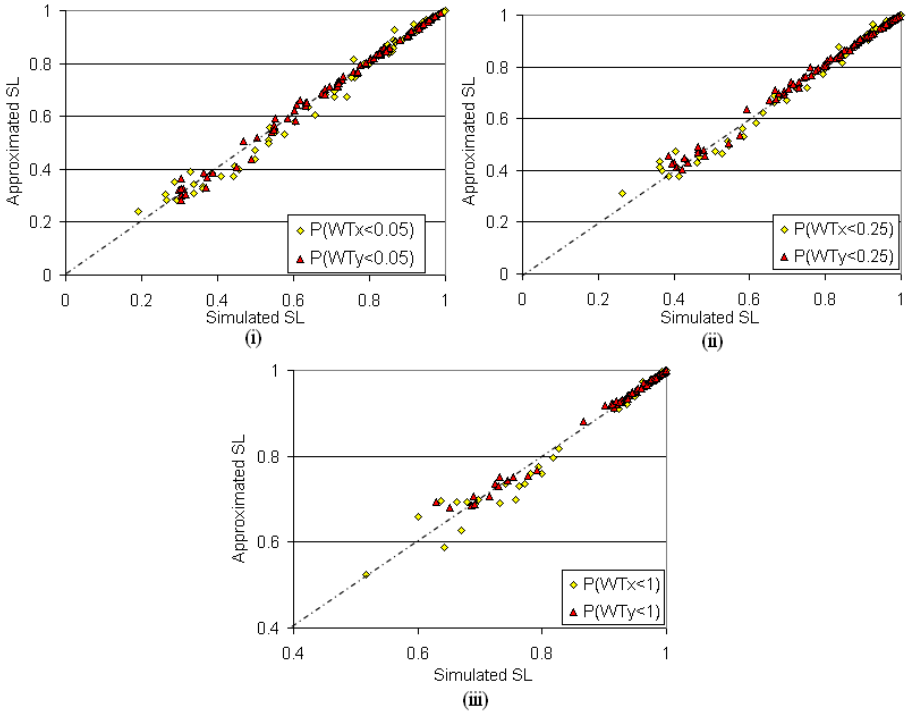

Figure 9.6: Approximation of the service level for three different maximum waiting times.

We observe that although the approximation is accurate in most cases, there are several cases for which the approximation is of lesser quality. This is again the case with large waiting probabilities. A comparison of Figures (9.5) and (9.6) reveals that most of the difference comes from the earlier approximation on the waiting probability. We should note also that the approximation tends to underestimate the service level compared to what is observed in the simulations.

9.4 QUALITY OF THE APPROXIMATIONS

In the previous sections we presented approximations to compute the waiting probability, the average waiting time and the service level in multi-skill queueing systems. Based on the simulation datasets presented in Section 2.5, we now present some qualititative results about the behavior of the approximations in different types of config-

urations. The purpose here is to identify the conditions in which the approximations work best and when they are of lesser quality.

In Sections 9.4.1 and9.4.2, we present the results for systems with hierarchical routing policies, where arriving calls will be handled preferentially by operators that have a smaller number of skills, as explained earlier. In Section 9.4.3, we test other routing policies. The study is not as detailled as the study in Sections 9.4.1 and 9.4.2 as the results are not significantly different and as hierarchical routings are prevalent in practice.

9.4.1 *Two types of calls.*

In this section we test the quality of the approximation on systems with two types of calls. We use the same two datasets as in Section 8.7.2.1, where we test the GSM method. In Table 9.2 we present the observed errors on the waiting probability and on the waiting time with the dataset of Table A.1, that we used for illustrating the approximations. In Table 9.3, we present the same errors obtained using the the dataset of Table A.2, i.e. the dataset with larger arrival rates and numbers of operators. Tables 9.4 and 9.5 display errors on the service level for 5 and 25 percent of the average service time, for each of the two datasets.

	Errors on the Waiting Probability			Errors on the Average Waiting Time		
	X-calls	*Y*-calls	All calls	*X*-calls	*Y*-calls	All calls
Mean AAbsErr/RAbsErr	0.0183	0.0134	0.0077	0.1355	0.0684	0.058
Mean AErr/RErr	-0.0015	0.0079	0.0046	0.0471	0.052	0.0269
Min AErr/RErr	-0.0788	-0.0521	-0.0111	-0.3484	-0.1456	-0.0134
Max AErr/RErr	0.0702	0.0591	0.0325	0.5095	0.2895	0.1655

Table 9.2: The mean, min and max differences between the simulation results for the Dataset Table A.1 for the waiting probability and the waiting time. The former are absolute errors and the latter are relative errors.

	Errors on the Waiting Probability			Errors on the Average Waiting Time		
	X-calls	*Y*-calls	All calls	*X*-calls	*Y*-calls	All calls
Mean AAbsErr/RAbsErr	0.0181	0.0148	0.0103	0.2479	0.2885	0.125
Mean AErr/RErr	-0.0099	-0.0123	-0.0103	0.0979	-0.1199	0.0727
Min AErr/RErr	-0.0686	-0.1011	-0.0269	-0.4046	-3.3950	-0.3175
Max AErr/RErr	0.0643	0.0089	0	0.791	0.2659	0.2956

Table 9.3: The mean, min and max differences between the simulation results for the Dataset Table A.2 for the waiting probability and the waiting time. The former are absolute errors and the latter are relative errors.

We observe in Table 9.2 that the approximation for the waiting probability works fine with two types of calls. We do not observe significant differences between the errors in Table 9.2 and the errors provided in Table 9.3, and we therefore conclude that the quality of the approximation is not affected by the size of the system. The performance remains very good in all cases, with average errors less than 0.02. Lower errors on the overall waiting probability suggest that our method gives a good estimate of the waiting probability in the whole system but that the allocation of operators used to compute s_i tends to decrease the accuracy of the approximation. In general we observed that asymmetry in the configuration and/or in the arrival rates negatively affects the accuracy of the approximation. Interestingly, the approximation performs well with low waiting probability. It seems to contradict our previous experiment of Section 9.1 that indicates that s_i does not correspond to the capacity allocated to calls of type i. We explain this apparent contradiction by the fact that, here, we evaluate the approximations with the absolute errors. Figure 9.7 displays the relative errors on the waiting probability as a function of the simulated waiting probability for the first dataset. We observe higher relative errors with smaller waiting probabilities. In fact, since the waiting probability is low, high relative errors result in small absolute errors in comparison to the errors observed with large waiting probabilities. In conclusion, the approximation on s_i has a weak impact on the quality of the approximation when it is measured in absolute error.

The results observed for the waiting time are good even if it is not as impressive as the results for the waiting probability. Although not shown here, we observed that the asymmetric cases show the highest errors, as for the waiting probability. Note here that the errors are bigger for the large systems.

	Errors on the service level (5%)			Errors on the service level (25%)		
	X-calls	Y-calls	All calls	X-calls	Y-calls	All calls
Mean AAbsErr	0.0167	0.0128	0.0069	0.0128	0.0104	0.0061
Mean AErr	0.0004	-0.0082	-0.0052	-0.0006	-0.0073	-0.0051
Min AErr	0.0634	-0.0626	-0.0327	-0.0713	-0.0714	-0.0309
Max AErr	0.0662	0.0497	0.0105	0.0622	0.0419	0.0098

Table 9.4: The mean, min and max differences between the simulation results for the Dataset of Table A.1 for two service levels. In both cases, this is the absolute errors.

Concerning the service level, the quality of the approximation is comparable to the one for the waiting probability. Note that it is partially due to the presence of the waiting probability approximation in the computation of the service level (see (9.13)).

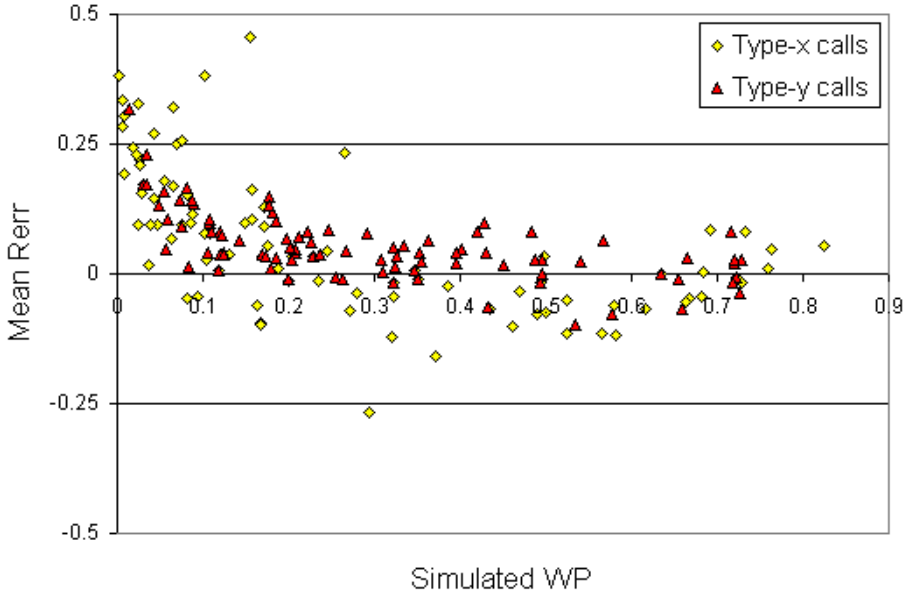

Figure 9.7: Relative errors on the waiting probability in function of the simulated waiting probability for the dataset of Table A.1.

	Errors on the service level (5%)			Errors on the service level (25%)		
	X-calls	*Y*-calls	All calls	*X*-calls	*Y*-calls	All calls
Mean AAbsErr	0.0124	0.0096	0.0076	0.0158	0.0110	0.0127
Mean AErr	-0.0119	-0.0028	-0.0066	-0.0158	-0.0110	-0.0127
Min AErr	-0.0669	-0.0298	-0.0335	-0.0713	-0.0716	-0.0714
Max AErr	0.0018	0.0358	0.0077	0	0	0

Table 9.5: The mean, min and max differences between the simulation results for the Dataset of Table A.2 for two service levels. In both cases, this is the absolute errors.

9.4.2 *Five types of calls*

We now investigate how well the approximations perform when there is a larger number of different calls and see if a more complex structure affects the performance. We analyze the results for the systems with five types of demands, presented in Tables A.4 and A.5. The types of errors displayed here are the same as in the preceding section but we aggregate the errors into a single statistic for each performance measure: we put the performance measures per call type in a single common set, that thus contains measurements for each of the five call types, and compute the average, maximum and

minimum errors on this set of measurements. The results are displayed in Table 9.6. In the second part of the table we present the average, maximum and minimum errors for the global performance measure.

		WP (A)	WT (R)	SL(0.05) (A)	SL(0.25) (A)
Individual calls	Mean AAbsErr/RAbsErr	0.0303	0.1468	0.0294	0.0495
	Mean AErr/RErr	0.0061	0.0506	-0.0133	-0.0461
	Min AErr/RErr	-0.1409	-0.4089	-0.1307	-0.2678
	Max AErr/RErr	0.1222	0.5823	0.1131	0.0487
All calls	mean AAbsErr/RAbsErr	0.0196	0.0628	0.015	0.0481
	Mean AErr/RErr	0.0023	0.0443	-0.015	-0.0481
	Min AErr/RErr	-0.0747	-0.0942	-0.0529	-0.1649
	Max AErr/RErr	0.0448	0.2631	-0.0003	-0.0006

Table 9.6: Errors observed on the waiting probability (absolute), waiting time (relative) and service level (absolute) at 5 and 25% when estimating these measurements on the dataset of Table A.4.

In all cases, the overall waiting probability is accurately estimated. The approximation for each call type is quite good as well, although the error is bigger than for the overall probability. This confirms our previous observation that a better allocation of operators could improve the quality of the results. We observe higher errors for the average waiting time than for the waiting probability but the error is on average not bigger than it is in a 2-call type system (see Table 9.3). This shows that complexification of the system does not deteriorate the quality of the approximation by much. The results for the service level support our conclusion.

We provide in Table 9.7 the same statistics for the results obtained from the dataset in Table A.5.

		WP	WT	SL(0.05)	SL(0.25)
Individual calls	Mean AAbsErr/RAbsErr	0.0239	0.2744	0.0288	0.0286
	Mean AErr/RErr	-0.0055	0.2551	-0.002	-0.009
	Min AErr/RErr	-0.0853	-0.1271	-0.066	-0.0631
	Max AErr/RErr	0.0441	0.8248	0.077	0.1044
All calls	mean AAbsErr/RAbsErr	0.0148	0.268	0.0073	0.013
	Mean AErr/RErr	-0.0008	0.268	-0.0055	-0.013
	Min AErr/RErr	-0.0144	0.1144	-0.0096	-0.0294
	Max AErr/RErr	0.0189	0.7407	0.0045	-0.0018

Table 9.7: Errors observed on the waiting probability (absolute), waiting time (relative) and service level (absolute) at 5 and 25% when estimating these measurements on the dataset of Table A.5.

The results further confirm our previous observations and we can conclude that the complexification does not affect the waiting probability and the service level but the error on the average waiting time is higher in this last dataset. Note that for the two other performance measures, the error is in general smaller than with the dataset of Table A.4. We do not think this has any significance, as the number of examples tested here, with 14 different pools, is small.

9.4.3 *Alternative routings*

Hierarchical routing is a common routing policy. Others are possible and in this section we evaluate how well MAO performs in this context. Obviously, this requires changing the routing accordingly in the equivalent loss model. Note however that it is necessary to keep the hierarchical routing policy in the fluid approximation in order to have an adequate allocation of operators. This might look a bit counter-intuitive but one should keep in mind that the allocation of calls to the different operators when there is a queue is independent of the routing policy: the first skilled operator to become available answers the waiting call, no matter the chosen routing policy.

We use the dataset of Table A.1, with two types of demands, to test two alternative routing policies. They are depicted in Figure 9.8 (i) and (ii). In the former case arriving x-calls first visit pool xy and overflown calls are routed to pool x. The routing of y-calls remains hierarchical. We call this case the semi-inverted routing model (SIRM). In the latter case, calls of both types first visit pool xy and the overflows are routed to their dedicated pool. This is the inverted routing model (IRM).

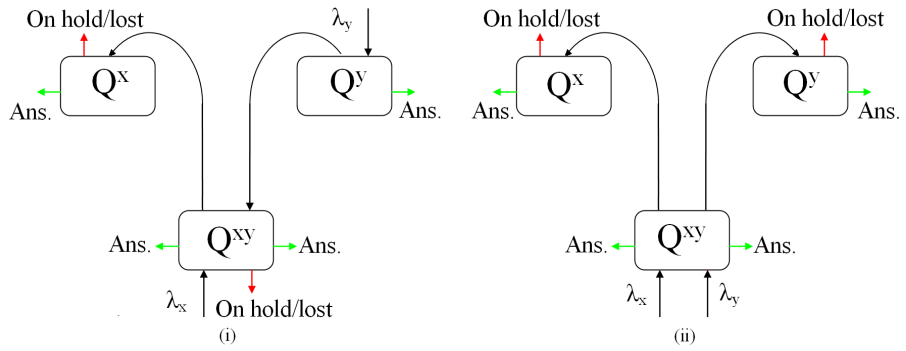

Figure 9.8: Two alternative routing policies. In (i) x-calls first visits the pool with polyvalent pools, the routing of y-calls being hierarchical. In (ii) both types of calls first visits pool xy.

Table 9.8 presents the usual statistics used to evaluate the quality of the approximation. We only present the approximation of the waiting probability. The experiments in the previous section showed that presenting the average waiting time and the service levels do not bring additional information as both are heavily correlated to the waiting probability approximation.

	WP SIRM			WP IRM		
	X-calls	Y-calls	All calls	X-calls	Y-calls	All calls
Mean AAbsErr	0.0337	0.0186	0.0114	0.0284	0.0159	0.0096
Mean AErr	0.0072	0.004	0.008	-0.0105	0.0105	0.0042
Min AErr	-0.1249	-0.053	-0.0138	-0.1573	-0.0228	-0.0183
Max AErr	0.1214	0.0485	0.0471	0.1062	0.0668	0.0345

Table 9.8: The mean, min and max differences between the simulation results for the Dataset Table A.2 for the waiting probability when the routing is not hierarchical.

According to the results in Table 9.8, using MAO when the routing is not hierachical gives good estimates of the waiting probability. The error is higher than when the routing is hierarchical but it remains quite low, especially when computing the waiting probability in the whole system. The highest errors are observed in configurations where the loads are unbalanced.

9.5 CONCLUSION

In this chapter a method was presented to approximate the most important performance measures of multi-class queueing systems based on equivalent loss systems. We successively developed approximations for the waiting probability, the average waiting time and the service level that are easy to compute. Our approximations were validated using a series of simulations.

The accuracy of our approximations is usually quite good. One should nevertheless be aware that the quality of the approximations could deteriorate a little bit for systems with asymmetry in the configuration or in the arrival rates and for longer waiting times. Although many call centers work close to saturation, which are cases where we observed some deviation, the methods provide fairly good approximations even in these cases. This and their easy computation make them appropriate to be used in practice.

10

COMPARATIVE EVALUATION OF THE METHODS FOR APPROXIMATING THE PERFORMANCE OF MULTI-SKILL QUEUEING SYSTEMS

In the previous two chapters we presented two different methods to evaluate performance measures in multi-skill queueing systems. Both are useful as they each have their own field of application. GSM can give estimate of the performance of systems with only a subset of call types that are allowed to wait. MAO on the other hand only applies when calls of all types can wait. However it is applicable to situations where the routing is different from the classical hierarchical routing and it is more amenable to other applications, such as computing the waiting time or service levels, than GSM.

Nevertheless, they can both estimate the waiting probability in multi-skill queueing systems with hierarchical routing. Under the conditions described in Section 9.1 both methods yield identical results. It remains to compare the methods when they provide different results and this is the purpose of Section 10.1. In Section 10.2, we compare one of our two methods to an alternative method proposed in [Koole et al., 2003]. To our knowledge, this method is the only comparable existing method for approximating the performance of multi-skill queueing systems. We end with concluding remarks on the two approximations.

10.1 COMPARISON OF THE METHODS OF CHAPTER 8 AND CHAPTER 9

As said in the introduction we compare here the GSM approach presented in Chapter 8 to the MAO approach presented in Chapter 9. Simulation results are used as benchmarks. As it does not make sense to compare both methods when they are known to give the same result, we cleaned the first dataset from all instances where the arrival rates exceed the dedicated capacity or where the configuration is symetrical and we do not use the dataset of Table A.2.

To summarize, we perform the comparisons based on two datasets:

- the first dataset consists in 50 systems with two call types and three pools, such as the one represented in Figure 8.3. They are the remaining configurations after cleaning the dataset of Table A.1.
- The second dataset is described in Table A.4. The systems there consist in calls of five different types that are dispatched to 9 different pools. In some pools operators can answer only one type of calls, in others more.

The results are presented hereafter in Table 10.1 in a similar way as in the previous two chapters: the value presented are the mean value of the differences in absolute value between the simulation results and the results obtained with the approximation method, Mean AAbsErr, as well as the mean, minimum and maximum differences, Mean, Min and Max AErr.

	x-calls		y-calls		All calls	
	GSM	MAO	GSM	MAO	GSM	MAO
Mean AAbsErr	0.0117	0.0168	0.0156	0.0115	0.083	0.0101
Mean AErr	-0.0013	0.0028	0.0125	0.0113	0.0075	0.0081
Min AErr	-0.0374	-0.0679	-0.021	-0.0024	-0.008	-0.0111
Max AErr	0.0354	0.0702	0.0363	0.0263	0.0159	0.0325

Table 10.1: Comparison of GSM and MAO methods for the 50 configurations with two call types.

We observe that GSM and MAO give results of the same quality: in both cases the average absolute error lies between 0.01 and 0.02. The results appear to be slightly better for GSM but the difference is too small to be significant. Note that the maximum error observed with GSM is quite low. Such a lower error on the overall waiting probability in the systems for GSM, combined with the observation that the mean errors are similar in both methods, suggests that GSM inherently provides a better estimate but that the approximations of the parameters used in the methods, namely the approximation for the loss probability and the fluid approximation for the allocation of calls, have a larger impact in the GSM approach than in MAO.

Table 10.2 confirms our conclusions. The results indicate that GSM provides a better estimate for the waiting probability in a multi-skill system. Like, for systems with two types of calls, the difference is never very important. The difference is however slightly more important for systems with five types of calls: there might be a close to 0.01 difference in accuracy between the two methods. This is especially true when we look at the waiting probability for all calls together.

	v-calls		w-calls		x-calls	
	GSM	MAO	GSM	MAO	GSM	MAO
Mean AAbsErr	0.0243	0.0328	0.0272	0.0276	0.0356	0.0357
Mean AErr	0.0074	0.0057	0.0225	0.0201	0.0041	0.0096
Min AErr	-0.0778	-0.1022	0.-0.0362	-0.0453	-0.1234	-0.1409
Max AErr	0.1253	0.0999	0.1147	0.0959	0.1179	0.108
	y-calls		z-calls		All calls	
Mean AAbsErr	0.0239	0.0252	0.0242	0.0302	0.0077	0.0196
Mean AErr	-0.003	0.0044	-0.0145	-0.0093	0.0075	0.0023
Min AErr	-0.0735	-0.081	-0.067	-0.0735	-0.0012	-0.0747
Max AErr	0.0454	0.1222	0.0393	0.0902	0.0425	0.0448

Table 10.2: Comparison of GSM and MAO methods for the 21 configurations with five call types.

10.2 COMPARISON TO AN EXISTING METHOD

To our knowledge, the only reference presenting a method to compute the performance of systems similar to the ones studied in this article is [Koole et al., 2003]. Their method consists in summing all flow rates on the one hand and all operators on the other hand in order to get a single arrival rate and a single number of operators to use with Formula (7.1). They then apply the loss approximation proposed in [Koole and Talim, 2000], the Poisson approximation of overflows to estimate the loss probability. We compare this method to MAO on the set of examples presented in [Koole and Talim, 2000] and described hereafter.

In each example, all five types of calls arrive at a rate of 10 calls per time unit and the general structure of the system remains the same: there are five single-skilled pools with five operators each, four other pools, with skill sets {v,w}, {w,x}, {x,y} and {y,z}, respectively, have five operators each. There is one last pool with either 8, 10, 12 or 14 fully cross-trained operators, depending on the example. All operators are equally efficient and the service times are exponentially distributed with rate $\mu = 1$. The routing of overflows is similar to ours: dedicated pools are first checked, then the 2-skilled pools and finally the fully polyvalent pool. Note that as there are two 2-skilled pools able to answer calls of type w, x and y, there are two possible paths for those types of call. One for each is chosen such that each of the four 2-skilled pools receives at least one overflow from the dedicated pools. Figure 10.1 represents one of the systems.

Table 10.3 displays the results. We compare the overall waiting probability obtained when aggregating all flows and all operators and using their overflow Poisson Approximation (KPT) with MAO and using the Hayward-based approximation of Section 5.3.

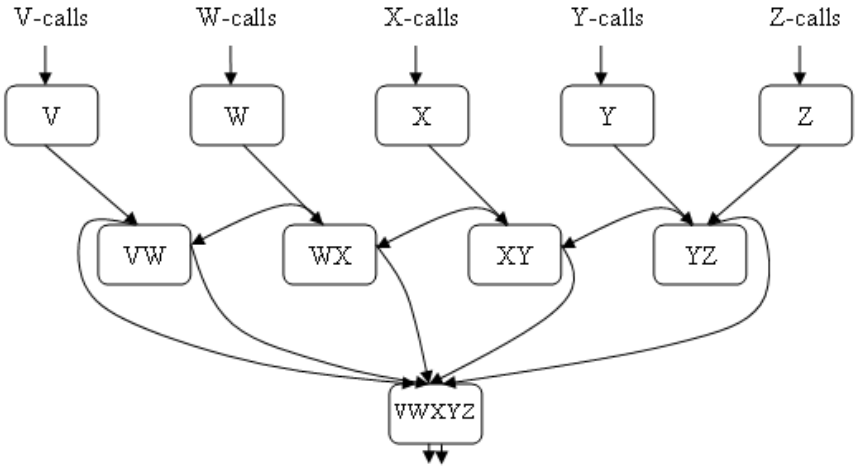

Figure 10.1: Structure of the systems presented in [Koole et al., 2003].

Ex.	Sim. WP	KPT	% error KPT	MAO	% error MAO
1	0.6622	0.6194	-6.46	0.6886	3.99
2	0.475	0.4004	-15.71	0.5074	6.82
3	0.324	0.2283	-29.54	0.3548	9.51
4	0.2108	0.1117	-47.01	0.2341	11.05

Table 10.3: Waiting probabilities obtained by simulation, with the method of [Koole et al., 2003] and with MAO. The relative errors are also displayed.

The results of Table 10.3 show that as a way for estimating the overall waiting probabilities, MAO provides better results. It is also interesting to note that our loss approximation method has a tendency to overestimate the waiting probability while theirs underestimates it. A comparison of KPT with GSM would yield very similar results.

10.3 RESPECTIVE STRENGTHS OF THE TWO APPROACHES AND CONCLUSIONS

GSM and MAO are two similar approaches for solving a same problem: to evaluate the performance of multi-skill queueing systems. In practice they have different fields of applications, which very much depend on their strengths and limitations. As a conclusion to this comparative chapter, we list the strong points of each of the two methods.

The strengths of GSM are:

1. it rests on some theoretical background. This brings confidence in the obtained results.

2. It applies to a broader range of queueing systems than the systems where queueing is allowed for all types of calls: the performance of systems where only a subset of the call types can be put on hold can be accurately estimated as well.

3. Within its range of application, it is slightly more accurate than MAO.

On the other side, the strengths of MAO are:

1. the formula remains simple and easy to read, no matter the number of calls types and of pools.

2. It can work for other routing policies than the hierarchical routing.

In conclusion we think that neither of the two methods really outperforms the other. The two approaches are actually complementary. They are both of good quality, with average errors that are within the precision of the estimates that can often be obtained for the arrival rate or the service rate in real-life situations.

There are many possible extensions to the work presented in the last three chapters. We believe in particular that the approach of using equivalent loss models, that are easy to solve, to estimate the performance of more complex models is promising.

In the context of call centers, we see the following possible extensions. First one should investigate whether the results presented here could be used in systems with limited queues and/or impatient customers. Because we are able to approximate queues of infinite length based on queues of zero length (loss systems), two situations at both ends of the spectrum, we believe that approximating queues of limited size, something in between the two extreme cases, would be simple. Secondly, one could investigate the method a step further for more complicated settings. In particular, we think about assuming general service time distributions. Adapting the method to systems with priority rules different from the FCFS rule is also something to be done. Finally we believe that loss models could be used to estimate the performance in models where abandonment is considered: there exist connections between the Erlang-B and Erlang-A formulae as well (see f.i. [Mandelbaum and Zeltyn, 2005]).

11

CONCLUSION

We can divide the contributions of this thesis into four main parts, each with a managerial and a scientific dimension.

The two first major contributions are related to multi-skill call centers.

- The first contribution is the development of a method for finding the best allocation of operators in multi-skill call centers in the absence of queues when integrality constraints matter. This is presented in Chapter 6. We showed that integrality constraints influence the solution a lot. From a managerial point of view this provides managers of small-size call centers with an optimization tool. We have shown that this method can reduce staffing costs by a substential amount as compared to "naïve" configurations, where all operators are either dedicated or fully polyvalent. Moreover, in the chapter, we have observed interesting and quite intuitive properties that can be exploited in multi-skill call center management. The most important one is that a good practice would be not to divide a flow into multiple subflows. From a scientific point of view, the originality of the approach lies in the utilization of combinatorial optimization to solve a stochastic model.

- The second contribution is the development of approximation formulae in Chapters 8 and 9 to evaluate the performance of multi-skill queueing systems. These methods exploit the similarities in performance between loss and queueing models to deduce the performance of queueing models from the performance of loss models. Simulation experiments showed that the resulting formulae are efficient: they accurately estimate the waiting probability, the average waiting time or the service level in multi-skill models and they are easy to use. Because of their connection with the algebraic relationship between the Erlang-B and the Erlang-C formulae and because we assumed in our model Poisson arrivals and exponential service times, the formulae can be viewed as multi-skill versions of the Erlang-C formula. Despite the limitations brought by these assumptions, we

believe they are valuable updates to the latter formula, which is still being used in practice by managers of many call centers.

From a scientific point of view, we believe that the sample path approach and the resulting interpretation of Formula (7.1) presented in Chapter 7 used to develop our approximations bring a new vision of the relationship between loss and queueing models.

We see two other major contributions to the study of call centers. These are not specifically related to multi-skill call centers and in that sense may have larger implications.

- In Chapter 3 we introduced the peakedness, which is a an indirect measure of variability, and we proposed methods to effectively determine the peakedness of real-life arrival streams. The method covers many types of situations that managers could face: it is possible to compute the peakedness from the collection of interarrival times as well as from the record of arrivals over time intervals. Moreover the methods are quite easy to use. The scientific contribution lies here in the adaptation of existing methods to the specificities of call center analysis.

- Fourthly, in Section 4.3 we presented an application of Hayward's approximation, which basically estimates the loss probability in loss systems where the arrival flow is not Poisson distributed. This application contributes to solving outsourcing decisions. The originality of our analysis is that we take the perspective of the outsourcer and propose methods to help him decide whether or not he should accept proposals from outsourcing companies and under which conditions. At the same time, this application illustrates how Hayward's principle of resizing a model by dividing its parameters by the peakedness to account for the variability of the incoming flow can be applied to other situations than loss models.

There are many possible extensions to this work. In particular, each restrictive assumption in our models paves the way to new research.

The extensions and possible improvements we see of the branch and bound optimization procedure described in Chapter 6 are listed in Section 6.7. These include looking for tighter bounds in the branch and bound procedure, or other search strategies.

The multi-skill models we work with are usually pure loss or pure queueing models. We have shown in Chapter 8 how it is possible to approximate the performance when some of the call types can wait and other cannot but we have not investigated models where the size of the queue is limited. Because we are able to approximate

queues of infinite length based on queues of zero length (loss systems), two opposite situations, we believe that approximating queues of limited size, something in between two extreme cases, would be possible.

Abandonment are not included either. It is one of the possible refinements to this work and, here again, we believe that possibilities of approximations exist: the Erlang-C and the Erlang-B formulae are special versions of the more general Erlang-A formula used to determine the waiting probability when callers may become impatient.

One could also investigate the approximation methods a step further for more complicated settings. In particular, we think about assuming general service time distributions. Adapting the method to systems with different priority rules between queues is also something to be done.

The two first contributions, the optimization of the configuration in loss models and the derivation of the performance in a queueing model from the performance observed in the equivalent loss model, are complementary in a sense: we actually believe that they can be combined to optimize the configuration of small-size queueing models. We have shown in Section 6.6 that the best of two configurations in a multi-skill loss model is likely to be the best of the two configurations in the presence of queues. We believe that it should be possible to develop an adequate methodology that exploits the formulae of Chapters 8 and 9 to derive the best configurations in the presence of queues from the best configurations in the absence of queue with more confidence. We think this is the most important immediate extension to this thesis.

A

DATASETS USED FOR VALIDATING THE APPROXIMATIONS

Arrival rates			series 1			series 2			series 3		
Ex.	λ_x	λ_y	n^x	n^x	n^{xy}	n^x	n^y	n^{xy}	n^x	n^y	n^{xy}
1	2	2	2	2	$x \in \{1,...,5\}$	$x \in \{1,...,5\}$	3	1	2	$x \in \{1,3,4,5,6\}$	2
2	2	3	2	2	$x \in \{2,...,6\}$	$x \in \{1,...,5\}$	3	2	1	$x \in \{1,...,5\}$	5
3	2	5	3	3	$x \in \{3,...,7\}$	$x \in \{1,...,5\}$	4	3	3	$x \in \{1,...,5\}$	2
4	3	10	5	5	$x \in \{6,...,10\}$	$x \in \{1,...,5\}$	8	5	2	$x \in \{10,...,14\}$	2
5	4	5	3	3	$x \in \{4,...,8\}$	$x \in \{2,4,6,8,10\}$	4	5	5	$x \in \{4,...,8\}$	3
6	5	5	4	4	$x \in \{3,...,7\}$	$x \in \{2,4,6,8,10\}$	4	5	5	$x \in \{4,...,8\}$	3

Table A.1: The main simulation dataset.

Ex.	λ_x	λ_y	n^x	n^x	n^{xy}
1	20	100	20	80	50
2	20	50	20	40	30
3	40	50	40	40	30
4	50	50	40	40	22
5	60	70	30	40	80
6	30	100	20	90	25
7	30	100	30	50	70
8	30	100	50	50	60
9	50	50	10	50	50
10	50	50	10	50	90
11	20	50	10	10	55
12	30	100	25	100	15
13	20	50	15	48	10
14	60	70	50	20	70
15	60	70	0	50	100
16	40	50	20	30	70
17	40	50	20	30	60
18	40	50	20	30	45
19	40	50	15	25	60
20	50	50	30	70	50

Table A.2: The simulation dataset with 2-demand systems and larger demand flows.

Ex.	λ_x	λ_y	λ_z	Pool x	Pool y	Pool z	Pool xz	Pool xy	Pool yz	Pool xyz
1	9	6	8	5	3	5	4	4	4	3
2	9	6	8	3	3	3	10	8	9	2
3	9	6	8	2	3	3	8	6	6	4
4	5	5	8	5	5	8	2	2	2	3
5	5	5	8	2	4	4	4	3	4	2
6	5	5	8	3	3	3	3	3	3	6
7	2	5	9	1	3	5	5	3	7	4
8	2	5	9	3	3	6	6	4	6	5
9	2	5	9	2	5	9	2	2	2	3
10	4	4	4	3	3	3	2	2	2	2
11	4	4	4	1	3	5	2	4	1	3
12	4	4	4	1	1	3	3	1	2	4

Table A.3: The simulation dataset with 3 types of flows and 7 different pools.

Ex.	λ_v	λ_w	λ_x	λ_y	λ_z	Conf.	Pool 1	Pool 2	Pool 3	Pool 4	Pool 5	Pool 6	Pool 7	Pool 8	Pool 9
1	7	8	5	6	3	(i)	5	5	4	3	2	5	4	3	8
2	4	4	8	4	3	(i)	4	3	6	4	3	2	2	3	6
3	5	5	3	8	2	(i)	2	2	3	2	3	1	4	4	11
4	4	4	6	6	4	(iii)	3	5	4	4	4	3	4	3	5
5	5	5	3	3	3	(iii)	7	6	3	3	4	1	2	2	2
6	5	7	9	2	5	(iii)	6	6	8	2	5	2	3	4	4
7	5	5	3	3	2	(ii)	6	5	3	3	3	2	2	1	2
8	6	7	5	5	5	(ii)	6	6	5	5	7	4	2	2	2
9	2	2	8	6	7	(ii)	1	1	6	6	9	1	4	4	3
10	7	8	5	6	3	(i)	4	4	3	2	1	5	4	3	5
11	4	4	8	4	3	(i)	3	2	5	3	2	2	2	3	3
12	5	5	3	8	2	(i)	1	1	3	2	3	1	4	4	8
13	4	4	6	6	4	(iii)	1	3	3	3	3	3	4	3	4
14	5	5	3	3	3	(iii)	5	4	2	2	3	1	2	2	1
15	5	7	9	2	5	(iii)	4	4	6	1	3	2	3	4	3
16	5	5	3	3	2	(ii)	4	4	3	2	1	2	2	1	2
17	6	7	5	5	5	(ii)	4	4	4	4	4	4	2	2	2
18	2	2	8	6	7	(ii)	1	1	5	5	5	1	4	4	3
19	7	8	5	6	3	(i)	4	4	3	2	1	5	4	3	8
20	4	4	8	4	3	(i)	3	2	5	3	2	2	2	3	6
21	5	5	3	8	2	(i)	1	1	3	2	3	1	4	4	11

Table A.4: The simulation dataset with 5 types of flows and 9 different pools. The *Conf.* column indicates which configuration among the three presented in Figure 2.5 is chosen for the example.

Ex.	λ_v	λ_w	λ_x	λ_y	λ_z	Pool 1	Pool 2	Pool 3	Pool 4	Pool 5	Pool 6	Pool 7	Pool 8	Pool 9	Pool 10	Pool 11	Pool 12	Pool 13	Pool 14
1	1	4	3	5	4	1	2	1	2	1	1	2	2	3	2	1	2	1	3
2	4	4	4	4	4	1	2	1	2	1	1	2	2	3	2	1	2	1	3
3	8	8	8	2	3	4	2	2	2	1	1	2	2	4	2	3	2	2	4
4	2	2	3	3	4	1	2	1	2	1	1	2	2	3	2	1	2	1	3
5	4	8	4	4	2	1	3	1	2	1	1	2	2	3	2	2	1	1	2

Table A.5: The simulation dataset with 5 types of flows and 14 different pools. The skills of the agents are given in Figure 2.6.

B

PROOF OF PROPOSITION 3.1

The proof is a reproduction of [Eckberg, 1983] adapted to our notation. We obtain $z_X(\mathsf{G}_\mu)$ by determining $E[S_X(t)]$ and $\mathrm{Var}[S_X(t)]$. Let us define here the function $s_X(t,\theta,g)$ as the number of busy operators in the infinite pool at epoch t due to an arrival occuring at time θ and with service time g:

$$s_X(t,\theta,g) = \begin{cases} 1, & \theta \leq t < \theta + g \\ 0, & \text{otherwise.} \end{cases}$$

If a batch of B_X arrivals occur at time θ and if their service times are denoted by g_i, $i \in \{1,..,B_X\}$ then the number of busy operators at time due to this batch arrival is the random variable

$$S_{\theta,B_X}(t) = \sum_{i=1}^{B_X} s_X(t,\theta,g_i).$$

Conditioned on B_X, $S_{\theta,B_X}(t)$ is Binomial with conditional mean and variance

$$\begin{aligned} E[S_{\theta,B_X}(t)|B_X] &= B_X \mathsf{G}_\mu^c(t-\theta) \\ \mathrm{Var}[S_{\theta,B_X}(t)|B_X] &= B_X \mathsf{G}_\mu^c(t-\theta)(1-\mathsf{G}_\mu^c(t-\theta)) \end{aligned}$$

Because arrivals in X occur continuously with batches of size $d\tilde{N}_X(\theta) = \tilde{N}_X(\theta, \theta + d\theta)$, it follows that $S_X(t)$ can be written as

$$S_X(t) = \int_0^t S_{d\tilde{N}_X(\theta)}(t) d\theta,$$

where $S_{d\tilde{N}_X(\theta)}(t) = S_{\theta,d\tilde{N}_X(\theta)}(t)$. We deduce that the mean and variance of $S_X(t)$ conditioned on $X(t)$ are equal to

$$\begin{aligned} E[S_X(t)|N_X(t)] &= \int_{-\infty}^{t} E[S_{d\tilde{N}_X(\theta)}(t)|N_X(t)]d\theta \\ &= \int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)d\tilde{N}_X(\theta) \end{aligned}$$

and

$$\mathrm{Var}[S_X(t)|N_X(t)] = \mathrm{Var}[\int_{-\infty}^{t} S_{d\tilde{N}_X(\theta)}(t)|N_X(t)]d\theta.$$

Because $S_{d\tilde{N}_X(\theta_1)}(t)$ and $S_{d\tilde{N}_X(\theta_2)}(t)$, $\forall\theta_1 \in \mathbb{R}^+$, $\theta_2 \in \mathbb{R}^+ \setminus \{\theta_1\}$, are independent random variables, we can write

$$\begin{aligned} \mathrm{Var}[S_X(t)|N_X(t)] &= \int_{-\infty}^{t} \mathrm{Var}[S_{d\tilde{N}_X(\theta)}(t)|N_X(t)]d\theta \\ &= \int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)(1-\mathsf{G}_\mu^c(t-\theta)d\tilde{N}_X(\theta). \end{aligned}$$

Using the law of total expectation, we find that

$$\begin{aligned} E[S_X(t)] &= E[E[S_X(t)|X]] \\ &= \int_{-\infty}^{t} E[\mathsf{G}_\mu^c(t-\theta)d\tilde{N}_X(\theta)] \\ &= \int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)\lambda_X d\theta \\ &= \lambda_X \int_0^{\infty} \mathsf{G}_\mu^c(\theta)d\theta \\ &= \lambda_X/\mu. \end{aligned}$$

With the law of total variance, we have

$$\begin{aligned} \mathrm{Var}[S_X(t)] &= E[\mathrm{Var}[S_X(t)|X]] + \mathrm{Var}[E[S_X(t)|X]] \\ &= E[\int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)(1-\mathsf{G}_\mu^c(t-\theta))d\tilde{N}_X(\theta)] + \mathrm{Var}[\int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)d\tilde{N}_X(\theta)] \\ &= \int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta)(1-\mathsf{G}_\mu^c(t-\theta))E[d\tilde{N}_X(\theta)] \\ &\qquad + \int_{-\infty}^{t}\int_{-\infty}^{t} \mathrm{Cov}[d\tilde{N}_X(\theta_1)\mathsf{G}_\mu^c(t-\theta_1); d\tilde{N}_X(\theta_1)\mathsf{G}_\mu^c(t-\theta_2)] \\ &= \lambda_X/\mu - \int_{-\infty}^{t} (\mathsf{G}_\mu^c(t-\theta))^2\lambda_X d\theta \\ &\qquad + \int_{-\infty}^{t}\int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta_1)\mathsf{G}_\mu^c(t-\theta_2)k_X(\theta_1-\theta_2)d\theta_1 d\theta_2 \\ &= \lambda_X/\mu - \int_{-\infty}^{t} \lambda_X \int_{\infty}^{\theta_1} \delta(\theta_1-\theta_2)\mathsf{G}_\mu^c(t-\theta_1)\mathsf{G}_\mu^c(t-\theta_2)d\theta_2 d\theta_1 \\ &\qquad + \int_{-\infty}^{t}\int_{-\infty}^{t} \mathsf{G}_\mu^c(t-\theta_1)\mathsf{G}_\mu^c(t-\theta_2)k_X(\theta_1-\theta_2)d\theta_1 d\theta_2, \end{aligned}$$

where in the second term of the last line we used the fact that $(G^c_\mu(t))^2 = \int_0^\infty \delta(\tau)$ $G^c_\mu(t)G^c_\mu(t-\tau)d\tau$ and substituted θ with θ_1 and τ with $\theta_1 - \theta_2$. Now, substituting $\theta_1 - \theta_2$ with τ and $t - \theta_1$ with τ_1 the expression becomes

$$\begin{aligned}
\mathrm{Var}[S_X(t)] &= \lambda_X/\mu + \int_{-\infty}^{\infty} \lambda_X \delta(t) \int_{\max(0,-\tau)}^{\infty} \mathsf{G}^c_\mu(\tau)\mathsf{G}^c_\mu(\tau-\tau_1)d\tau_1 d\tau \\
&\quad + \int_{-\infty}^{\infty} k_X(\tau) \int_{\max(0;-\tau_1)}^{\infty} \mathsf{G}^c_\mu(\tau)\mathsf{G}^c_\mu(\tau-\tau_1)d\tau_1 d\tau. \\
&= \lambda_X/\mu + \int_{-\infty}^{\infty} (k_X(t) - \lambda_X\delta(t))\rho_{G^c_\mu}(t)dt.
\end{aligned}$$

Dividing $\mathrm{Var}[S_X(t)]$ by $E[S_X(t)]$ eventually yields (3.3).

C

PROOF OF PROPOSITION 3.2

The proof is presented in [Macq, 2005]. With the fluid process, $G^c_\mu(t-\tau)$ gives the amount of operators still serving in t arrivals at time τ. The number of operators busy at time t is equal to

$$\bar{S}_X(t) = \int_{-\infty}^{t} G^c_\mu(t-\tau) d\tilde{N}_X(\tau).$$

Then,

$$\begin{aligned} E[\bar{S}_X(t)] &= \int_{-\infty}^{t} E[G^c_\mu(t-\tau) d\tilde{N}_X(\tau)] \\ &= \lambda_X \int_{-\infty}^{t} G^c_\mu(t-\tau) d\tau \\ &= \frac{\lambda_X}{\mu}. \end{aligned}$$

Before determining $\mathrm{Var}[\bar{S}_X(t)]$, we need to determine the autocovariance function of $\bar{S}_X$. Let us define $\bar{S}_{d\tilde{N}_X(\theta)}(t)$ as the number of busy operators in time t due to the arrivals in the process $X(t)$ at time θ. $\mathrm{Cov}[\bar{S}_{d\tilde{N}_X(\theta_1)}(t); \bar{S}_{d\tilde{N}_X(\theta_2)}(t)]$ is equal to

$$\begin{aligned} \mathrm{Cov}[\bar{S}_{d\tilde{N}_X(\theta_1)}(t); \bar{S}_{d\tilde{N}_X(\theta_2)}(t)] &= \int_{-\infty}^{t}\int_{-\infty}^{t} \mathrm{Cov}[G^c_\mu(t-t_1) d\tilde{N}_X(\theta_1); G^c_\mu(t-t_2) d\tilde{N}_X(\theta_2)] \\ &= \int_{-\infty}^{t}\int_{-\infty}^{t} \mathrm{Cov}[d\tilde{N}_X(\theta_1); d\tilde{N}_X(\theta_2)] G^c_\mu(t-t_1) G^c_\mu(t-t_2) \\ &= \int_{0}^{\infty} k_X(\tau) G^c_\mu(\tau_1) G^c_\mu(\tau_1+\tau) d\tau d\tau_1 \\ &= \int_{-\infty}^{\infty} k_X(\tau) \int_{max(0,-\tau)}^{\infty} G^c_\mu(\tau_1) G^c_\mu(\tau_1+\tau) d\tau d\tau_1 \\ &= \int_{-\infty}^{\infty} k_X(\tau) \rho_{G^c_\mu}(\tau) d\tau. \end{aligned}$$

Using this last result, we find that

$$\mathrm{Var}[\bar{S}_X(t)] = \int_{-\infty}^{\infty} k_X(\tau) \rho_{G^c_\mu}(\tau) d\tau.$$

Dividing $\mathrm{Var}[\bar{S}_X(t)]$ by $E[\bar{S}_X(t)]$ gives

$$\bar{z}_X(\mathsf{G}_\mu) = \frac{\mu}{\lambda_X} \int_{-\infty}^{\infty} k_X(\tau) \rho_{G_\mu^c}(\tau) d\tau.$$

Noting that $\rho_{G_\mu^c}(0) = \int_{-\infty}^{\infty} \delta\tau \rho_{G_\mu^c}(\tau) d\tau$ and using (3.3) we easily deduce the result in Proposition 3.2.

D

TABLES OF ARRIVAL TIMES, SERVICE TIMES AND MEASUREMENT TIMES USED IN SECTION 3.2

0.1574	4.5578	8.6309	14.4950	20.3141	25.2501	29.2269	35.0697
0.2750	4.5888	8.7309	14.5185	20.4512	25.4198	29.4422	35.1320
0.3587	4.6274	8.7637	14.7140	20.5882	25.5880	29.4901	35.1807
0.4207	4.7549	8.9381	14.8188	20.7536	25.5906	29.5842	35.2351
0.4285	4.8250	8.9908	14.9402	20.9248	25.8435	29.9743	35.3311
0.5885	4.9993	9.0382	14.9647	21.0401	25.8508	30.0158	35.3379
0.6000	5.0261	9.1333	14.9733	21.1407	25.8788	30.1745	35.4270
0.6599	5.0858	9.1444	14.9906	21.2343	25.9190	30.3488	35.4722
0.7246	5.1388	9.8160	15.2009	21.3500	26.0994	30.3863	35.6719
0.8576	5.1764	10.3124	15.3000	21.3642	26.3528	30.4315	35.7917
0.8582	5.1979	10.5366	15.3644	21.3868	26.3674	30.5611	36.1994
1.0507	5.2409	10.7432	15.6688	21.4341	26.3938	30.7024	36.4228
1.3369	5.2801	10.8241	15.8770	21.4641	26.4009	30.8238	36.5656
1.3769	5.3515	11.0285	15.9266	21.7823	26.6006	31.0316	36.6438
1.3873	5.6367	11.0657	16.0155	21.8438	26.8660	31.2814	36.6921
1.4422	5.7749	11.2731	16.1458	22.0639	27.0049	31.4123	36.8318
1.6333	5.8224	11.3418	16.2899	22.0970	27.0668	31.5042	36.8499
1.6501	5.8467	11.4427	16.6229	22.1069	27.0958	31.5895	36.9219
1.7894	5.8881	11.6511	16.6886	22.1150	27.1705	31.9654	37.0212
1.9289	6.0341	11.7110	16.7953	22.3769	27.2530	32.0442	37.1467
2.0178	6.1512	11.7359	16.8104	22.4889	27.2791	32.1361	37.4496
2.0988	6.2462	11.7928	17.0903	22.5244	27.2801	32.3390	37.5531
2.3432	6.3227	11.8096	17.1096	22.5593	27.3962	32.3680	37.5637
2.3651	6.3243	11.8770	17.1318	22.5613	27.4849	32.5971	37.8305
2.4027	6.3490	11.9110	17.2223	22.5675	27.6439	32.6354	37.9119
2.4535	6.3892	11.9362	17.3454	22.7333	27.7660	32.6931	38.0334
2.5150	6.5491	11.9517	17.3586	22.8352	27.8511	32.7406	38.0634
2.6320	6.5985	12.2008	17.4712	23.0093	27.9772	32.8626	38.1468
2.7538	6.6754	12.2563	17.5475	23.1630	28.1486	32.9572	38.3943
2.8011	6.7485	12.3221	17.6608	23.3303	28.1680	32.9915	38.4677
2.9373	6.7669	12.3265	17.6938	23.3452	28.2053	33.0808	38.5895
2.9473	6.7735	12.5571	17.8699	23.6793	28.2438	33.0851	38.6182
2.9724	6.7862	12.9381	18.3092	23.8721	28.3308	33.1017	38.7564
3.0811	6.8197	13.0536	18.3112	23.9624	28.3389	33.2992	39.5334
3.2195	6.8653	13.0882	18.3779	24.0301	28.3705	33.3270	39.5514
3.2527	6.9886	13.1469	18.4379	24.0370	28.4407	33.3957	39.6261
3.2677	7.0241	13.3357	18.6989	24.1307	28.4973	33.5079	40.1965
3.5476	7.1311	13.3475	18.7792	24.1822	28.5184	33.5955	40.3122
3.5828	7.2917	13.3636	18.8187	24.2895	28.5984	33.6374	40.3713
3.6549	7.4623	13.4003	18.8297	24.3434	28.6371	33.6486	40.4198
3.6963	7.6231	13.4642	19.0971	24.3907	28.6511	33.9078	40.5301
3.8420	7.7129	13.6398	19.1702	24.4091	28.7556	34.1390	40.5689
3.8474	7.8130	13.6794	19.3322	24.5045	28.7646	34.1785	40.6883
4.1552	8.0790	13.7326	19.4460	24.7358	28.9345	34.2260	40.9937
4.1995	8.1167	13.7683	19.4626	24.9090	29.0361	34.5206	41.0377
4.2835	8.1534	13.8978	19.4748	24.9398	29.0498	34.5840	41.1254
4.3678	8.2170	13.9320	19.8083	24.9638	29.0854	34.7196	41.2111
4.4180	8.2482	14.1278	19.8644	24.9723	29.1229	34.7774	41.2167
4.4338	8.2749	14.1339	19.8659	25.1153	29.1282	34.8338	41.3906
4.4746	8.4260	14.2194	20.2900	25.1768	29.1917	35.0178	41.3970

Table D.1: List of 400 arrivals of a Poisson process of rate 10.

0.2750	0.0735	0.0061	0.0040	0.0836	0.0558	0.1402	0.5596	0.9375
0.4737	0.0307	0.0044	0.3427	0.1803	0.4076	1.0783	0.2197	1.0077
0.6147	2.2521	0.4911	0.1319	0.3842	0.2466	2.1402	0.0921	1.6581
0.4962	2.0606	0.1623	0.0113	0.3454	0.2452	0.0672	1.2138	0.0275
0.8685	0.2895	0.1129	0.4183	0.3101	0.0909	0.4156	1.0533	0.4945
	0.3023	0.0238	0.1676	0.1130	0.0090	0.2292	1.4557	0.2742
	0.0465	0.3188	0.3472	0.3225	0.1808	0.1441	0.1792	0.1920
	1.2372	0.1531	0.0336	0.0637	0.0454	0.0470	0.0502	0.5944
	0.0893	0.2012	0.0088	0.9115	2.2115	0.1515	0.1181	0.0342
	2.0162	0.0993	0.3396	0.5938	0.0451	0.0105	0.5408	0.0950
	0.1972	0.5239	0.6823	0.6935	1.2905	0.7136	1.0686	0.0847
	0.3130	0.0610	0.3566	0.0284	0.1025	0.2026	0.0783	0.4413
	0.0678	0.6430	0.4436	0.3258	1.6942	1.4437	0.7915	0.7891
	0.2474	0.0500	0.3506	1.5866	0.0222	1.0083	0.0217	1.4288
	0.1786	0.6817	0.4817	1.4415	1.7809	0.3627	0.1291	0.2175
	1.3016	0.2725	0.3345	0.3399	0.1619	0.2750	0.3112	0.9304
	0.8128	0.0532	0.2286	0.2771	0.4258	0.0745	0.3338	0.4303
	0.9273	0.0122	0.3702	0.0021	0.0379	1.0523	0.1295	0.0821
	0.0950	0.3411	0.2428	0.9715	0.3329	0.4779	0.1823	1.3085
	1.2456	0.1383	0.9211	0.2181	0.0596	1.3308	0.2762	0.3466
	0.1763	0.4684	0.9756	0.7181	0.0578	0.7779	0.1426	1.1536
	0.5798	0.4657	0.3996	0.7366	1.4370	0.2914	0.6224	0.8100
	0.3017	0.3701	0.4446	0.3745	0.1039	1.1990	0.8208	0.3611
	0.1547	0.0129	0.1474	0.2925	0.4383	0.3144	0.0516	0.0217
	0.8071	1.9291	1.2664	1.0741	0.1358	0.1159	0.3260	0.3971
	1.2750	0.7836	0.5698	0.0126	0.0199	0.2836	0.2181	1.0057
	0.0690	0.0586	0.2510	0.4023	0.3487	0.0085	1.0167	0.0428
	0.8603	0.9357	0.4419	0.6275	2.2196	0.8666	0.3616	0.1778
	0.7404	0.4365	2.3478	1.1876	0.4428	2.5290	0.0409	0.7374
	0.3533	1.3877	0.3187	0.1038	1.0719	0.8183	0.1410	0.3353
	0.0351	0.0157	0.3170	0.3935	0.9509	0.1833	0.0249	0.5775
	0.0303	0.0233	1.0117	0.4092	2.5741	0.4433	0.7271	0.0596
	0.4103	0.6884	0.2258	0.2551	1.1799	0.3048	0.5559	0.0362
	0.2304	1.0605	1.0798	0.4760	0.0125	0.7506	1.0976	0.1252
	0.6000	0.3088	1.3182	0.1221	0.1209	0.2965	0.0079	0.2963
	0.6669	0.0134	0.3535	0.2929	0.1964	1.7118	0.4400	0.2476
	0.1817	0.7643	0.3245	0.3598	0.2346	0.9730	0.2745	0.0501
	0.2082	0.0066	0.0172	0.3008	0.2852	0.3318	0.0506	0.7760
	0.5373	0.5339	0.3257	0.8716	0.0103	0.2780	0.4739	0.0210
	0.7653	1.4050	0.1934	0.3912	0.1262	0.8895	0.7682	0.6255
	0.2560	0.5099	0.6032	0.0826	0.0727	0.1567	0.2752	0.5248
	0.2721	0.5560	0.1442	0.3186	0.0025	0.2641	0.6862	0.9174
	0.3949	0.2735	0.0314	0.0639	0.0392	0.0317	0.2004	0.3181
	0.0754	0.6851	0.2168	1.1455	0.1814	0.2851	0.2331	1.0527
	1.1831	0.0063	0.2689	0.7292	0.0429	0.0121	1.2645	0.3760
	0.2505	0.2390	0.5061	0.1476	0.0396	0.7254	0.9941	0.2920
	0.8267	1.0515	1.1483	0.0364	2.1235	0.7277	0.2074	0.0288
	0.4831	0.2662	1.0418	0.6949	0.0887	0.2884	0.5565	1.0341
	0.4850	0.2314	1.4747	1.8666	0.4720	0.2767	0.2175	0.2246
	0.5365	0.0315	0.9623	1.2813	1.1480	1.1593	0.8941	0.1132

Table D.2: List of 405 services times from an exponential distribution of parameter $\mu = 2$. The five first arrivals in the first column are the service times of the 5 calls in service at t=0.

Interval	Nbr arrivals	Interval	Nbr arrivals	Interval	Nbr arrivals	Interval	Nbr arrivals
0 - 0.25	1	10.5 - 10.75	2	20.75 - 21	2	31 - 31.25	1
0.25 - 0.5	4	10.75 - 11	1	21 - 21.25	3	31.25 - 31.5	2
0.5 - 0.75	4	11 - 11.25	2	21.25 - 21.5	5	31.5 - 31.75	2
0.75 - 1	2	11.25 - 11.5	3	21.5 - 21.75	0	31.75 - 32	1
1 - 1.25	1	11.5 - 11.75	3	21.75 - 22	2	32 - 32.25	2
1.25 - 1.5	4	11.75 - 12	6	22 - 22.25	4	32.25 - 32.5	2
1.5 - 1.75	2	12 - 12.25	1	22.25 - 22.5	2	32.5 - 32.75	4
1.75 - 2	2	12.25 - 12.5	3	22.5 - 22.75	5	32.75 - 33	3
2 - 2.25	2	12.5 - 12.75	1	22.75 - 23	1	33 - 33.25	3
2.25 - 2.5	4	12.75 - 13	1	23 - 23.25	2	33.25 - 33.5	3
2.5 - 2.75	2	13 - 13.25	3	23.25 - 23.5	2	33.5 - 33.75	4
2.75 - 3	5	13.25 - 13.5	5	23.5 - 23.75	1	33.75 - 34	1
3 - 3.25	2	13.5 - 13.75	3	23.75 - 24	2	34 - 34.25	3
3.25 - 3.5	2	13.75 - 14	3	24 - 24.25	4	34.25 - 34.5	0
3.5 - 3.75	4	14 - 14.25	3	24.25 - 24.5	4	34.5 - 34.75	3
3.75 - 4	2	14.25 - 14.5	1	24.5 - 24.75	2	34.75 - 35	2
4 - 4.25	2	14.5 - 14.75	2	24.75 - 25	4	35 - 35.25	5
4.25 - 4.5	5	14.75 - 15	5	25 - 25.25	2	35.25 - 35.5	4
4.5 - 4.75	3	15 - 15.25	1	25.25 - 25.5	2	35.5 - 35.75	1
4.75 - 5	3	15.25 - 15.5	2	25.5 - 25.75	2	35.75 - 36	1
5 - 5.25	6	15.5 - 15.75	1	25.75 - 26	4	36 - 36.25	1
5.25 - 5.5	2	15.75 - 16	2	26 - 26.25	1	36.25 - 36.5	1
5.5 - 5.75	1	16 - 16.25	2	26.25 - 26.5	4	36.5 - 36.75	3
5.75 - 6	4	16.25 - 16.5	1	26.5 - 26.75	1	36.75 - 37	3
6 - 6.25	3	16.5 - 16.75	2	26.75 - 27	1	37 - 37.25	2
6.25 - 6.5	4	16.75 - 17	2	27 - 27.25	4	37.25 - 37.5	1
6.5 - 6.75	4	17 - 17.25	4	27.25 - 27.5	5	37.5 - 37.75	2
6.75 - 7	6	17.25 - 17.5	3	27.5 - 27.75	1	37.75 - 38	2
7 - 7.25	2	17.5 - 17.75	3	27.75 - 28	3	38 - 38.25	3
7.25 - 7.5	2	17.75 - 18	1	28 - 28.25	4	38.25 - 38.5	2
7.5 - 7.75	2	18 - 18.25	0	28.25 - 28.5	5	38.5 - 38.75	2
7.75 - 8	1	18.25 - 18.5	4	28.5 - 28.75	4	38.75 - 39	1
8 - 8.25	5	18.5 - 18.75	1	28.75 - 29	3	39 - 39.25	0
8.25 - 8.5	2	18.75 - 19	3	29 - 29.25	7	39.25 - 39.5	0
8.5 - 8.75	2	19 - 19.25	2	29.25 - 29.5	2	39.5 - 39.75	3
8.75 - 9	3	19.25 - 19.5	4	29.5 - 29.75	1	39.75 - 40	0
9 - 9.25	3	19.5 - 19.75	0	29.75 - 30	1	40 - 40.25	1
9.25 - 9.5	0	19.75 - 20	3	30 - 30.25	2	40.25 - 40.5	3
9.5 - 9.75	0	20 - 20.25	0	30.25 - 30.5	3	40.5 - 40.75	3
9.75 - 10	1	20.25 - 20.5	3	30.5 - 30.75	2	40.75 - 41	1
10 - 10.25	0	20.5 - 20.75	1	30.75 - 31	1	41 - 41.25	4
10.25 - 10.5	1						

Table D.3: Number of arrivals in each 15 minute interval.

E

NUMERICAL STUDY TO EVALUATE FORMULA (3.11)

We build a set of scenarios which aims to represent the kind of demand processes likely to appear in a supply chain. Each scenario was simulated several times using a software that is able to compute both $\bar{z}$ and $\bar{z}_b$. Before giving details about the processes used in the scenarios, we briefly present how the simulations are performed.

We compare two processes. The first process is a simple arrival process X. The second process, Y, is the batch arrival process that is constructed by assuming that arrivals occur every T time units and the size of the batch of the nT-th arrival is equal to $\tilde{N}_X((n-1)T, nT)$. $\bar{z}$ and $\bar{z}_b$ in (3.11) are here $\bar{z}_X$ and $\bar{z}_Y$, respectively.

We assume that X is stationary in the long run but that the rate of arrivals can change from one time interval to another. Denote $\lambda_X(n)$ the average arrival rate observed during $[(n-1)T, nT)$. In practice, each simulation run is performed by considering one time interval $[(n-1)T, nT)$ at a time. For each time interval we perform the following sequence of actions.

STEP 1. Based on $\lambda_X(n)$, generate a number of arrivals, noted $A(n) \in \mathbb{R}^+$.

STEP 2. Launch $A(n)$ random variables Uniformy distributed overf the length T of the interval. This determines the exact arrival time of each of the $A(n)$ arrivals. These times are then chronoligically ranked.

STEP 3. Determine $\bar{S}_X(t)$ for each arrival time t. This is done by applying the procedure illustrated in Example 3.2.

STEP 4. Compute $\bar{S}_Y(nT) = e^{-\mu T}\bar{S}_Y((n-1)T) + A(n)$.

This procedure is repeated for each interval $[(n-1)T, nT)$ of the time horizon of the simulation. After that, like in Example 3.2 we pick sample values of both processes at random times. We then compute $\bar{z}_X$ and $\bar{z}_Y$ from those sample values.

$\lambda(n)$ is determined in a different way in each scenario of the simulation experiment.

In a first group of scenarios, $\lambda(n)$ is obtained from a random walk equal to:

$$R(n) = A(n-1) + W(n),$$

where $W(n) \sim N(0,\sigma)$. To avoid negative arrival rates and ensure the stationarity of the process, we put "mirrors" so that

$$\lambda(n) = \begin{cases} R(n), & 0 \leq R(n) \leq 2\lambda(0) \\ -R(n), & R(n) < 0 \\ 2\lambda(0) - (R(n) - 2\lambda(0)), & R(n) > 2\lambda(0), \end{cases} \tag{E.1}$$

where $\lambda(0)$ is the initial arrival rate. If $\lambda(n)$ is outside $[0; 2\lambda(0)]$, we repeat the procedure until $\lambda(n)$ falls within the threshold values. In each scenario of this group, we either change $\lambda(0)$ (10 or 100 arrivals per interval) or σ in the Normal random variables ($0.05\lambda(0)$, $0.1\lambda(0)$ or $0.2\lambda(0)$).

In a second group of simulations, $\lambda(n)$ is based on a ARIMA-like random variable. The general formula is:

$$\lambda(n) = \alpha A(n-1) + (1-\alpha)Z(n),$$

where α is a correlation factor and $Z(n) \sim$ lognormal(μ,σ). μ is chosen such that $E[Z(n)] = \lambda(0)$ Depending on the scenario, α might be equal to 0, 0.2, 0.4 or 0.6. $\lambda(0)$ is equal to either 10 or 100 arrivals per interval. σ is fixed at $0.2\lambda(0)$, $0.5\lambda(0)$ or $\lambda(0)$.

Each scenario is duplicated. First, it is assumed that A(n) is obtained by rouding $\lambda(n)$ to the nearest integer value. In the second case, A(n) is found by launching a Poisson random variable of mean $\lambda(n)T$. The first class of scenarios is called the Uniform set and the second calls the Poisson set.

The value of $\bar{z}_Y$ obtained by simulation is compared to the value computed with Formula 3.11, using the value of $\bar{z}_X$ found in simulation. All results are summarized in Tables E.1 to E.5. Each table is divided into two parts, one for the Poisson and the other for the Uniform cases. Each line represents the set of simulations that shares a same λ and a same σ and only differ by the service time. We display the Mean and maximum relative difference in absolute value between the simulated and the computed value of $\bar{z}_Y$.

The results show that the approximation proposed with Formula (3.11) performs well. The mean error is limited to less than 2% in most cases.

		Poisson		Uniform	
λ	σ	Mean RAbsErr	Max RAbsErr	Mean RAbsErr	Max RAbsErr
10	0.5	0.0009	0.0066	0.0005	0.0025
10	1	0.0016	0.0094	0.0012	0.0074
10	2	0.0017	0.0112	0.0007	0.0029
100	5	0.0007	0.0047	0.0007	0.0041
100	10	0.0009	0.0051	0.0008	0.0033
100	20	0.0012	0.0038	0.0009	0.0028

Table E.1: Relative errors observed when approximating the batch peakedness using Formula (3.11) compared to simulations when the arrival stream is of the random-walk type.

		Poisson		Uniform	
λ	σ	rel. deviation	Max deviation	rel. deviation	Max deviation
10	2	0.0065	0.0149	0.0212	0.0648
10	5	0.0142	0.0436	0.0116	0.0358
10	10	0.0195	0.0751	0.0166	0.056
100	20	0.0089	0.0226	0.0081	0.02
100	50	0.0177	0.0602	0.016	0.0448
100	100	0.0172	0.0405	0.0203	0.0663

Table E.2: Relative errors observed when approximating the batch peakedness using Formula (3.11) compared to simulations when the arrival stream is an ARIMA process with $\rho = 0$. In other words, the number of arrivals in each interval is obtained from a lognormal distribution.

		Poisson		Uniform	
λ	σ	rel. deviation	Max deviation	rel. deviation	Max deviation
10	2	0.0055	0.0263	0.0282	0.077
10	5	0.0101	0.0289	0.0073	0.0203
10	10	0.0119	0.0375	0.01	0.0239
100	20	0.0096	0.0273	0.0079	0.0185
100	50	0.0117	0.039	0.0136	0.0447
100	100	0.0128	0.0446	0.0128	0.0473

Table E.3: Relative errors observed when approximating the batch peakedness using Formula (3.11) compared to simulations when the arrival stream is an ARIMA process with $\rho = 0.2$.

		Poisson		Uniform	
λ	σ	rel. deviation	Max deviation	rel. deviation	Max deviation
10	2	0.0034	0.0073	0.024	0.0632
10	5	0.0056	0.02	0.0027	0.0041
10	10	0.0065	0.0233	0.0073	0.0287
100	20	0.0042	0.0096	0.0052	0.0169
100	50	0.0077	0.0274	0.0076	0.0253
100	100	0.009	0.0341	0.0089	0.0363

Table E.4: Relative errors observed when approximating the batch peakedness using Formula (3.11) compared to simulations when the arrival stream is an ARIMA process with $\rho = 0.4$.

		Poisson		Uniform	
λ	σ	rel. deviation	Max deviation	rel. deviation	Max deviation
10	2	0.0051	0.0115	0.0259	0.0516
10	5	0.0039	0.0093	0.0053	0.0142
10	10	0.0053	0.0172	0.0028	0.005
100	20	0.0086	0.0156	0.0059	0.0131
100	50	0.0063	0.0221	0.0043	0.0201
100	100	0.0044	0.0119	0.0046	0.0128

Table E.5: Relative errors observed when approximating the batch peakedness using Formula (3.11) compared to simulations when the arrival stream is an ARIMA process with $\rho = 0.6$.

F

DATASET USED IN SECTION 4.3.6

Example	λ	Pool1	z	Pool2	Example	λ	Pool1	z	Pool2	Example	λ	Pool1	z	Pool2
1	4	1	1.13	4	49	16	4	1.24	13	97	60	15	1.30	46
2	4	1	1.13	5	50	16	4	1.24	14	98	60	15	1.30	50
3	4	1	1.13	6	51	16	4	1.24	15	99	60	15	1.30	54
4	4	1	1.13	7	52	16	4	1.24	16	100	60	15	1.30	58
5	4	2	1.28	3	53	16	8	1.56	9	101	60	30	1.81	31
6	4	2	1.28	4	54	16	8	1.56	10	102	60	30	1.81	35
7	4	2	1.28	5	55	16	8	1.56	11	103	60	30	1.81	39
8	4	2	1.28	6	56	16	8	1.56	12	104	60	30	1.81	43
9	4	3	1.42	2	57	16	12	1.98	6	105	60	45	2.73	18
10	4	3	1.42	3	58	16	12	1.98	7	106	60	45	2.73	20
11	4	3	1.42	4	59	16	12	1.98	8	107	60	45	2.73	22
12	4	3	1.42	5	60	16	12	1.98	9	108	60	45	2.73	24
13	4	4	1.54	2	61	16	16	2.40	3	109	60	60	4.08	6
14	4	4	1.54	3	62	16	16	2.40	4	110	60	60	4.08	8
15	4	4	1.54	4	63	16	16	2.40	5	111	60	60	4.08	10
16	4	4	1.54	5	64	16	16	2.40	6	112	60	60	4.08	12
17	8	2	1.19	7	65	20	5	1.25	16	113	80	20	1.31	62
18	8	2	1.19	8	66	20	5	1.25	17	114	80	20	1.31	66
19	8	2	1.19	9	67	20	5	1.25	18	115	80	20	1.31	70
20	8	2	1.19	10	68	20	5	1.25	19	116	80	20	1.31	74
21	8	4	1.41	5	69	20	10	1.61	11	117	80	40	1.85	41
22	8	4	1.41	6	70	20	10	1.61	12	118	80	40	1.85	45
23	8	4	1.41	7	71	20	10	1.61	13	119	80	40	1.85	49
24	8	4	1.41	8	72	20	10	1.61	14	120	80	40	1.85	53
25	8	6	1.66	4	73	20	15	2.09	7	121	80	60	2.89	23
26	8	6	1.66	5	74	20	15	2.09	8	122	80	60	2.89	26
27	8	6	1.66	6	75	20	15	2.09	9	123	80	60	2.89	29
28	8	6	1.66	7	76	20	15	2.09	10	124	80	60	2.89	32
29	8	8	1.89	2	77	20	20	2.61	4	125	80	80	4.62	7
30	8	8	1.89	3	78	20	20	2.61	5	126	80	80	4.62	8
31	8	8	1.89	4	79	20	20	2.61	6	127	80	80	4.62	9
32	8	8	1.89	5	80	20	20	2.61	7	128	80	80	4.62	10
33	12	3	1.22	10	81	40	10	1.29	31	129	100	25	1.31	85
34	12	3	1.22	11	82	40	10	1.29	34	130	100	25	1.31	90
35	12	3	1.22	12	83	40	10	1.29	37	131	100	25	1.31	95
36	12	3	1.22	13	84	40	10	1.29	40	132	100	25	1.31	100
37	12	6	1.50	7	85	40	20	1.74	21	133	100	50	1.87	51
38	12	6	1.50	8	86	40	20	1.74	24	134	100	50	1.87	55
39	12	6	1.50	9	87	40	20	1.74	27	135	100	50	1.87	60
40	12	6	1.50	10	88	40	20	1.74	30	136	100	50	1.87	65
41	12	9	1.84	5	89	40	30	2.49	12	137	100	75	3.02	28
42	12	9	1.84	6	90	40	30	2.49	14	138	100	75	3.02	30
43	12	9	1.84	7	91	40	30	2.49	16	139	100	75	3.02	32
44	12	9	1.84	8	92	40	30	2.49	18	140	100	75	3.02	34
45	12	12	2.16	3	93	40	40	3.44	5	141	100	100	5.10	8
46	12	12	2.16	4	94	40	40	3.44	6	142	100	100	5.10	9
47	12	12	2.16	5	95	40	40	3.44	7	143	100	100	5.10	10
48	12	12	2.16	6	96	40	40	3.44	8	144	100	100	5.10	11

Table F.1: 144 combinations of arrivals and two successive pools with the peakedness of the intermediate flow.

BIBLIOGRAPHY

[Adelman, 2008] Adelman, D. (2008). A simple algebraic approximation to the erlang loss system. *Operation Research Letters*, 36:484 – 491.

[Agnihothri et al., 2003] Agnihothri, S., Mishra, A., and Simmons, D. (2003). Workforce cross-training decisions in field service systems with two job types. *Journal of the Operations Reseach Society*, 54:410–418.

[Akan et al., 2007] Akan, M., Ata, B., and Lariviere, M. A. (2007). Asymmetric information and economies of scale in service contracting. Technical report, Kellogg School of Management, Evanston, Illinois.

[Akşin et al., 2007a] Akşin, Z., Armony, M., and Mehrotra, V. (2007a). The Modern Call Center: A Multi-Disciplinary Perspective on Operations Management Research. *Production and Operations Management*, 16(6):665–688.

[Akşin et al., 2008] Akşin, Z., de Véricourt, F., and Karaesmen, F. (2008). Call center outsourcing contract analysis and choice. *Management Science*, 54(2):354–368.

[Akşin et al., 2005] Akşin, Z., Karaesmen, F., and Örmeci, E. L. (2005). On the Interaction between Resource Flexibility and Flexibility Structures. In *Proceedins of the Fifth International Conference on Analysis of Manufacturing Systems-Production Management*.

[Akşin et al., 2007b] Akşin, Z., Karaesmen, F., and Örmeci, E. L. (2007b). A review of workforce cross-training in call centers from an operations management perspective. In Nembhard, D., editor, *Workforce Cross Training Handbook*. CRC Press.

[Allon and Federgruen, 2005] Allon, G. and Federgruen, A. (2005). Outsourcing service processes to a common service provider under price and time competition. Technical report, Northwestern University, Evanston, IL.

[Armony, 1999] Armony, M. (1999). *Queueing Networks with Interacting Service Resources*. PhD thesis, Stanford University.

[Armony and Bambos, 2003] Armony, M. and Bambos, N. (2003). Queueing Dynamics and Maximal Throughput Scheduling in Switched Processing Systems. *Queueing Systems*, 44:209 – 252.

[Avramidis et al., 2009] Avramidis, N., Chan, W., and L'Ecuyer, P. (2009). Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions*, 41(6):483–497.

[Bambos and Walrand, 1993] Bambos, N. and Walrand, J. (1993). Scheduling and stability aspects of a general class of parallel processing systems. *Advances in Applied Probability*, 25:176 – 202.

[Baron and Milner, 2009] Baron, O. and Milner, J. (2009). Staffing to maximize profit for call centers with alternate service level agreements. *Operations Research*, 57(3):685–700.

[Bassamboo et al., 2009] Bassamboo, A., Randhawa, R. S., and Mieghem, J. A. V. (2009). A Little Flexibility is All You Need: On the Value of Flexible Resources in Queueing Systems. `http://www.kellogg.northwestern.edu/faculty/vanmieghem/htm/pubs/Bassamboo_Randhawa_VanMieghe_A%20little%20flexibility%20is%20all%20you%20need_21June2009.pdf`.

[Berezner et al., 1998] Berezner, S. A., Krzesinski, A. E., and Taylor, P. G. (1998). On the Inverse of Erlang's Function. *Journal of Applied Probability*, 35(13):246 – 252.

[Bhulai, 2009] Bhulai, S. (2009). Dynamic Routing Policies for Multi-Skill Call Centers. *Probability in the Engineering and Informational Sciences,*, 23(1):101–119.

[Bolotin, 1994] Bolotin, V. (1994). Telephone circuit holding time distributions. In *Proceedings of the 14th International Teletraffic Conference*, pages 125 – 134.

[Borst and Seri, 2000] Borst, S. and Seri, P. (2000). Robust Algorithms for Sharing Agents with Multiple Skills. Technical report, Bell Laboratories.

[Breukelen, 1995] Breukelen, G. (1995). Theoretical note: Parallel information processing models compatible with lognormally distributed response times. *Journal of Mathematical Psychology*, 39:396–399.

[Brown et al., 2005] Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., and Zhao, L. (2005). Statistical analysis of a telephone call center: A queueing science perspective. *Journal of the American Statistical Association*, 100:36–50.

[Cezik and L'Ecuyer, 2008] Cezik, M. T. and L'Ecuyer, P. (2008). Staffing Multiskill Call Centers via Linear Programming and Simulation. *Management Science*, 54(2):310–323.

[Chevalier et al., 2005] Chevalier, P., Shumsky, R. A., and Tabordon, N. (2005). Routing and Staffing in Large Call Centers with Specialized and Fully Flexible Servers. Technical report, Université catholique de Louvain.

[Chevalier and Tabordon, 2003] Chevalier, P. and Tabordon, N. (2003). Overflow analysis and cross-trained servers. *International Journal of Production Economics*, 85:47–60.

[Chevalier and Van den Schrieck, 2008] Chevalier, P. and Van den Schrieck, J.-C. (2008). Optimizing the staffing and routing of small-size hierarchical call centers. *Production and Operations Management*, 17(3):306 – 319.

[Cooper, 1972] Cooper, R. B. (1972). *Introduction to Queueing Theory*. North Holland, 2nd edition.

[Dawson, 2007] Dawson, K. (2007). ICMI's Contact Center Outsourcing Report – Key Findings. *Call Center Magazine*. `http://www.callcentermagazine.com/shared/article/showArticle.jhtml?articleId=201805251`.

[Eckberg, 1976] Eckberg, A. E. (1976). A generalization of peakedness to arbitrary arrival processes and service time distributions. In *Bell Laboratories Technical Memorandum*.

[Eckberg, 1983] Eckberg, A. E. (1983). Generalized Peakedness of Teletraffic Processes. In *Proceedings of the 10-th International Teletraffic Congress*, Montréal, Canada.

[Erlang, 1917] Erlang, A. K. (1917). Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Elektrotkeknikeren*, 13.

[Erlang, 1948] Erlang, A. K. (1948). On the rational determination of the number of circuits. In E. Brockmeyer, H. L. Halstrom, A. J., editor, *The life and works of A. K. Erlang*. The Copenhagen Telephone Company.

[Franx et al., 2006] Franx, G. J., Koole, G., and Pot, A. (2006). Approximating multi-skill blocking systems by HyperExponential Decomposition. *Performance Evaluation*, 63(8):799–824.

[Fredericks, 1980] Fredericks, A. A. (1980). Congestion in Blocking Systems - A Simple Approximation Technique. *The Bell System Technical Journal*, 59(6):805–827.

[Gans et al., 2003] Gans, N., Koole, G., and Mandelbaum, A. (2003). Telephone Call Centers: Tutorial, Review and Research Prospects. *Manufacturing and Service Operations Management*, 5(2):79–141.

[Gans and van Ryzin, 1997] Gans, N. and van Ryzin, G. (1997). Optimal Control of a Multiclass, Flexible Queueing System. *Operations Research*, 45(5):677 – 693.

[Gans and Zhou, 2002] Gans, N. and Zhou, Y.-P. (2002). Managing Learning and Turnover in Employee Staffing. *Operations Research*, 50(6):991–1006.

[Gans and Zhou, 2007] Gans, N. and Zhou, Y.-P. (2007). Call-Routing Schemes for Call-Center Outsourcing. *Manufacturing and Service Operations Management*, 9(1):33–50.

[Garnett and Mandelbaum, 2001] Garnett, O. and Mandelbaum, A. (2001). An introduction to skills-based routing and its operational complexities. Technical report, Technion.

[Garnett et al., 2002] Garnett, O., Mandelbaum, A., and Reiman, M. (2002). Designing a call center with impatient customers. *Manufacturing & Service Operations Management*, 4(3):208–227.

[Green et al., 2007] Green, L. V., Kolesar, P. J., and Whitt, W. (2007). Coping with Time-Varying Demand when Setting Staffing Requirements for a Service System. *Production and Operations Management*, 16(1):13–39.

[Gross and Harris, 1998] Gross, D. and Harris, C. M. (1998). *Fundamentals of Queueing Theory*. John Wiley & Sons, INC., 3rd edition.

[Halfin and Whitt, 1981] Halfin, S. and Whitt, W. (1981). Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588.

[Harel, 1988] Harel, A. (1988). Sharp bounds and simple approximations for the erlang delay and loss formulas. *Management Science*, 34(8):959 – 972.

[Harel, 1990] Harel, A. (1990). Convexity properties of the erlang loss formula. *Operations Research*, 38(3):499 – 505.

[Harrison and López, 1999] Harrison, J. M. and López, M. J. (1999). Heavy Traffic Resource Pooling in Parallel-Server Systems. *Queueing Systems*, 33:339–368.

[Harrison and Zeevi, 2005] Harrison, J. M. and Zeevi, A. (2005). A method for staffing large call centers based on stochastic fluid models. *Manufacturing and Service Operations Management*, 7(1):20–36.

[Hasija et al., 2008] Hasija, S., Pinker, E. J., and Shumsky, R. A. (2008). Call center outsourcing contracts under information asymmetry. *Management Science*, 54(4):793–807.

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

[Jagerman, 1984] Jagerman, D. L. (1984). Methods in traffic calculations. *AT&T Bell Laboratories Technical Journal*, 63(7):1283 – 1310.

[Jagerman et al., 1996] Jagerman, D. L., Melamed, B., and Willinger, W. (1996). Stochastic modeling of traffic processes. In *Frontiers in Queueing: Models, Methods and Problems*, pages 271–370. CRC Press.

[Jiménez and Koole, 2004] Jiménez, T. and Koole, G. (2004). Scaling and comparison of fluid limits of queues applied to call centers with time-varying parameters. *OR Spectrum (Special issue on Call Center Management)*, 26:413–422.

[Jongbloed and Koole, 2000] Jongbloed, G. and Koole, G. (2000). Managing uncertainty in call centers using Poisson mixtures. Technical report, Department of Stochastics - Vrije Universiteit Amsterdam.

[Jordan and Graves, 1995] Jordan, W. C. and Graves, S. C. (1995). Principles on the Benefits of Manufacturing Process Flexibility. *Management Science*, 54(4):577–594.

[Jouini et al., 2009] Jouini, O., Dallery, Y., and Akşin, Z. (2009). Queueing Models for Multiclass Call Centers with Real-Time Anticipated Delays. *International Journal of Production Economics*, 120:389–399.

[Jouini et al., 2010] Jouini, O., Dallery, Y., and Akşin, Z. (2010). Call Centers with Delay Information: Models and Insights. Submitted.

[Keblis and Chen, 2006] Keblis, M. and Chen, M. (2006). Improving customer service operations at amazon.com. *Interfaces*, 36(5):433–445.

[Khintchine, 1960] Khintchine, A. Y. (1960). *Mathematical methods in the theory of queueing*. Griffin.

[Kleinrock, 1975] Kleinrock, L. (1975). *Queueing Systems ; Volume 1: Theory*. Wiley.

[Koole and Pot, 2006] Koole, G. and Pot, A. (2006). An overview of Routing and Staffing algorithms in Multi-Skill Contact Centers. Technical report, Departement of Stochastics, Vrije Universiteit Amsterdam.

[Koole et al., 2003] Koole, G., Pot, A., and Talim, J. (2003). Routing heuristics for multi-skill call centers. In *Proceedings of the 2003 Winter Simulation Conference*, pages 1813–1816.

[Koole and Talim, 2000] Koole, G. and Talim, J. (2000). Exponential approximation of multi-skill call centers architecture. In *Proceedings of QNETs 2000*, pages 23/1–10.

[Kosten, 1973] Kosten, L. (1973). *Stochastic theory of service systems*. Pergamon.

[Kuczura, 1973] Kuczura, A. (1973). The interrupted poisson process as an overflow process. *The Bell System Technical Journal*, 52(3):437—448.

[Macq, 2005] Macq, J.-F. (2005). *Optimization of Multimedia Flows over Data Networks: the Core Location Problem and the Peakedness Characterization*. PhD thesis, Université catholique de Louvain - Faculté des Sciences Appliquées.

[Maglaras, 1999] Maglaras, C. (1999). Dynamic scheduling in multiclass queueing networks: Stability under discrete review policies. *Queueing Systems*, 31:171 – 206.

[Mandelbaum, 2006] Mandelbaum, A. (2006). Call centers (centres): Research bibliography with abstracts. Technical report, Technion - Israel Institute of Technology, Haifa, Israel.

[Mandelbaum and Zeltyn, 2005] Mandelbaum, A. and Zeltyn, S. (2005). Service Engineering in Action: The Palm/Erlang-A Queue, with Applications to Call Centers. Technion Technical Report.

[Milner and Olsen, 2008] Milner, J. M. and Olsen, T. L. (2008). Service-level agreements in call centers: Perils and prescriptions. *Management Science*, 54(2):238–252.

[Neal, 1971] Neal, S. (1971). Combining correlated streams of nonrandom traffic. *The Bell System Technical Journal*, 50(6):2015 – 2037.

[Örmeci, 2004] Örmeci, E. L. (2004). Dynamic admission control in a call center with one shared and two dedicated service facilities. *IEEE Transactions on Automatic Control*, 49:1157–1161.

[Palm, 1957] Palm, C. (1957). esearch on telephone traffic carried by full availability groups. *Tele*, 1(107). English translation of results first published in 1946 in Swedish in the same journal, which was then entitled Tekniska Meddelanden fran Kungl. Telegrfstyrelsen.

[Pinker and Shumsky, 2000] Pinker, E. J. and Shumsky, R. A. (2000). The Efficiency-Quality Trade-Off of Cross-Trained Workers. *Manufacturing and Service Operations Management*, 2(1):32–48.

[Pot et al., 2008] Pot, A., Bhulai, S., and Koole, G. (2008). A simple staffing method for multi-skill call centers. *Manufacturing and Service Operations Management*, 10(3):421 – 428.

[Rapp, 1964] Rapp, L. Y. (1964). Planning of junction networks in a multi-exchange area, part 1. *Ericsson Technics*, 20:77 – 130.

[Ren and Zhang, 2009] Ren, Z. J. and Zhang, F. (2009). Service Outsourcing: Capacity, Quality and Correlated Costs. Technical report, Boston University.

[Ren and Zhou, 2008] Ren, Z. J. and Zhou, Y.-P. (2008). Call center outsourcing: Coordinating staffing level and service quality. *Management Science*, 54(2):369 – 383.

[Shumsky, 2004] Shumsky, R. A. (2004). Approximation and Analysis of a Queueing System with Flexible and Specialized Servers. *OR Spectrum*, 26(3):307–330.

[Sisselman and Whitt, 2007a] Sisselman, M. E. and Whitt, W. (2007a). Value-based routing and preference-based routing in customer contact centers. *Production and Operations Management*, 16(1).

[Sisselman and Whitt, 2007b] Sisselman, M. E. and Whitt, W. (2007b). Value-based routing and preference-based routing in customer contact centers. *Production and Operations Management*, 16(3):277–2919.

[Stanford and Grassman, 2000] Stanford, D. K. and Grassman, W. K. (2000). Blingual server call centers. In McDonald, D. R. and Turner, S. R. E., editors, *Analysis of Communication Networks: Call Centers, Traffic and Performance*, volume 28, pages 31–48. Fields Institute Communications.

[Subba Rao, 1965] Subba Rao, S. (1965). Queuing Models with Balking, Reneging, and Interruptions. *Operations Research*, 13(4):596–608.

[Tabordon, 2002] Tabordon, N. (2002). *Modeling and Optimizing the Management of Operator Training in a Call Center*. PhD thesis, Université catholique de Louvain - Institut d'Administration et de Gestion.

[Ulrich and Miller, 1993] Ulrich, R. and Miller, J. (1993). Information processing models generating lognormally distributed reaction times. *Journal of Mathematical Psychology*, 37:513–525.

[van Dijk, 2004] van Dijk, N. M. (2004). Should we pool or not for call centers. In *First conference of the POMS College of Service Operations*.

[van Dijk and van der Sluis, 2008] van Dijk, N. M. and van der Sluis, E. (2008). To Pool or not to Pool in Call Centers. *Production and Operations Management*, 17(3):296–305.

[Van Muylder, 2001] Van Muylder, N. (2001). Phénomènes de pertes et de temps d'attente dans un call-center. Master's thesis, Université catholique de louvain, Faculté des Sciences Appliquées.

[Wallace and Whitt, 2005] Wallace, R. B. and Whitt, W. (2005). A Staffing Algorithm for Call Centers with Skill-Based Routing. *Manufacturing and Service Operations Management*, 7(4):276–294.

[Whitt, 1992] Whitt, W. (1992). Understanding the Efficiency of Multi-Server Service Systems. *Management Science*, 38(5):708 – 723.

[Whitt, 1995] Whitt, W. (1995). Variability Functions for Parametric-Decomposition Approximations of Queueing Networks. *Management Science*, 41(10):1704–1715.

[Whitt, 1998] Whitt, W. (1998). Improving service by informing customers about anticipated delays. *Management Science*, 45:192–207.

[Whitt, 2005] Whitt, W. (2005). Two fluid approximations for multi-server with abandonments. *Operation Research Letters*, 33:363–372.

[Whitt, 2006a] Whitt, W. (2006a). Fluid Models for Multiserver Queues with Abandonments. *Operation Research*, 54(1):37–54.

[Whitt, 2006b] Whitt, W. (2006b). A multi-class fluid model for a contact center with skill-based routing. *AEU - International Journal of Electronics and Telecommunications*, 60(2):95–102.

[Whitt, 2006c] Whitt, W. (2006c). Sensitivity of Performance in the Erlang-A Queueing Model to Changes in the Model Parameters. *Operations Research*, 54(2):247–260.

[Whitt, 2006d] Whitt, W. (2006d). Staffing a Call Center with Uncertain Arrival Rate and Absenteeism. *Production and Operations Management*, 15(1):88–102.

[Wilkinson, 1956] Wilkinson, R. (1956). Theories for toll traffic engineering in the U.S.A. *Bell System Technical Journal*, 35(2):421–514.

[Wolff, 1989] Wolff, R. W. (1989). *Stochastic Modeling and the Theory of Queues*. Prentice Hall.

[Wolsey, 1998] Wolsey, L. A. (1998). *Integer Programming*. Wiley.

Université catholique de Louvain
Faculté des sciences économiques, sociales et politiques
Louvain School of Management
Center of Excellence for Supply Chain Management –
Center for Operations Research and Econometrics

Place des Doyens 1, 1348 Louvain-la-Neuve, Belgique
Tél. : + 32 (0) 10 47 83 69 - fax : + 32 (0) 10 47 83 24
E-mail : jc.vandenschrieck@uclouvain.be
http://www.uclouvain.be/lsm.html

The objective of this thesis is to support the management of call centers by developing tools and methods to decide on the required number of operators. We use existing queueing theory models to analyze call centers with several types of calls. In this context the goal is to find the most efficient configurations, i.e. configurations that achieve the best performance at a minimum cost by combining different sorts of operators: some only answer calls of one type; others are skilled to answer calls of different types, but at a greater cost.

We propose a Branch and Bound method to find the best combinations of operators while keeping the integrality constraints on the number of operators, which matter in small-size call centers.

Looking for efficient configurations requires assessing the performance in the models. We present new methods to estimate the waiting probability, the average waiting time or the service level. These methods exploit the similarities observed in single-skill models between systems with no queue and the same systems with a queue of infinite size. Using Hayward's approximation, that permits to approximate the probability for a call to not be handled in a multi-skill system with no queues, we estimate the performance of the same multi-skill system when calls can be put on hold.

We also present the peakedness functional. It is a measure of variability of the stochastic flows. We explain how to extract the peakedness from real-life data and how to use it in our models. Different applications are proposed.

Jean-Christophe Van den Schrieck was born in Ottignies in 1980. During his master in business engineering at UCL, he developed an interest in Operations Research. In 2004 he became teaching-assistant in the Production and Operations Management department of the Louvain School of Management (Belgium). In 2007 he stayed for three months at Koç University (Turkey) as an invited researcher. His main fields of interest are queueing theory, stochastic modelling and supply chain management.

632 | 2010

Jean-Christophe Van den Schrieck

Multi-skill Queueing Models for Call Centers

UCL