Università della Calabria

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA

DOTTORATO DI RICERCA IN RICERCA OPERATIVA
MAT/09-XX CICLO

Agent Scheduling in a Multiskill Call Center
Schedulazione di operatori in un call center multi-skill

Ornella Pisacane

Anno Accademico 2007–2008

University of Calabria

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS, INFORMATICS AND SYSTEMS

PhD in Operations Research
MAT/09-XX cycle

Agent Scheduling in a Multiskill Call Center

Coordinator:
Prof.Lucio Grandinetti

[illegible]

Candidate:
Ornella Pisacane

[illegible]

Supervisors:
Prof.Pierre L'Ecuyer

[illegible]

Prof.Roberto Musmanno

[illegible]

2007–2008

Universitá della Calabria

FACOLTÁ DI INGEGNERIA

DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA

Dottorato di Ricerca in Ricerca Operativa
MAT/09-XX ciclo

Schedulazione di agenti in un Call Center Multiskill

Relatori:
Prof.Pierre L'Ecuyer

[illegible]

Prof.Roberto Musmanno

[illegible]

Coordinatore:
Prof.Lucio Grandinetti

[illegible]

Candidata:
Ornella Pisacane

[illegible]

2007–2008

A mia madre

Contents

List of Figures

List of Tables

Acknowledgements

I thank my father and my mother for their continuous moral support, my principal source of strength. During my PhD, they have always encouraged me to do my best with their words and, in particular, with their facts. I shall never forget their great help during my studies at the Université dé Montréal, allowing me to overcome some critical moments and to complete this important experience. They have taught me to use passion and sacrifice for reaching my goals!

My thanks also go to Prof. Pierre L'Ecuyer and Prof. Roberto Musmanno who have allowed me to increase my knowledge, teaching me more than what I have summarized in this PhD thesis.

My thanks go to my sister Ilaria and my best friend Adamo who always support me in all what I do, sharing with me the happy and the sad moments.

Finally, my thanks go to a special Angel that whenever is, she is supporting me as usually: Grazie Nonna!

Abstract

Il principale scopo del presente lavoro di tesi e' risolvere il problema di scheduling del personale di un call center multi-skill.

Un call center a skill multiplo gestisce diversi tipi di chiamate distinte in base agli skill richiesti per il servizio. Ciascun gruppo di operatori (agenti) e' distinto in base al numero di tipi diversi di chiamate (alias numero di skill) in grado di gestire. L'insieme di regole per assegnare le chiamate agli agenti (o viceversa) e' indicato con il nome di routing. In un tipico call center, le chiamate in ingresso arrivano in maniera casuale secondo un qualche sofisticato processo stocastico. Sono, quindi, anche componenti casuali: la durata delle chiamate, i tempi di attesa prima di ricevere il servizio, gli abbandoni (utenti abbandonanti il sistema prima di ricevere il servizio e dopo aver aspettato un certo tempo, detto di impazienza, anche esso casuale) e, in alcuni contesti operativi, anche il numero di clienti che riprova ad ottenere il servizio dopo aver abbandonato il sistema.

In genere, la giornata lavorativa e' suddivisa in periodi (tipicamente di 15 minuti ciascuno) in maniera tale da definire e prevedere anche pause per gli agenti e dettagliare l'andamento delle chiamate durante il giorno (specificandone il tasso di arrivo).Un turno (shift) per un agente e' una sequenza di periodi lavorativi (comprese le pause) ed e' definito attraverso i periodi in cui l'agente e' in servizio e puo' rispondere alle chiamate (cioe' possiede tutti gli skill richiesti).

Il principale obiettivo e' quello di garantire una data qualita' di servizio a minimo costo. La piu' comune misura di qualita' del servizio e' il livello di servizio, definito come la frazione di chiamate il cui tempo di attesa in coda non e' piu' lungo di una data soglia.

Tipicamente,vengono prese in considerazione diverse misure di livello di servizio: per un dato periodo del giorno, per una data chiamata, per una coppia (periodo, chiamata), globali (aggregati su tutti i periodi e su tutti i tipi di chiamate).

Sulla base delle previsioni effettuate sui volumi di chiamate, i manager dei call center devono decidere (tra le altre cose) quanti agenti di ciascun tipo avere nel centro in ogni istante della giornata lavorativa; devono costruire turni di lavoro per gli agenti disponibili;devono, anche, decidere le regole di routing. Queste decisioni sono prese sotto un elevato livello di incertezza.

I classici problemi relativi alla gestione degli agenti di un call center possono essere suddivisi in due grandi categorie: problemi di staffing e di scheduling.

Nel primo, si deve stabilire quanti agenti di ciascun tipo assegnare a ciascun periodo della giornata lavorativa, con l'obiettivo di soddisfare i livelli di servizio minimizzando il costo della soluzione finale.

Nel secondo, invece, si procede all'individuazione dell'assegnamento a costo minimo degli agenti ai vari turni al fine di soddisfare sempre i livelli di servizio.

I due problemi non sono separati: dalla soluzione finale di scheduling si puo' agevolmente risalire alla relativa soluzione di staffing.

Si noti che, in entrambi i casi, l'ammissibilita' della soluzione deve essere verificata utilizzando la simulazione, a causa della non linearita' di alcuni vincoli, e, per questo motivo, i due suddetti problemi rientrano nella categoria di problemi di simulazione-ottimizzazione.

In letteratura, il classico approccio per risolvere il problema di scheduling e' quello a due stadi: nel primo stadio si risolve un problema di staffing per quanti sono i periodi individuati; nel secondo si cerca di definire schedule ammissibili per gli agenti sfruttando le informazioni ricavate nel primo.

In questo lavoro di tesi, supponendo di avere un dettagliato modello stocastico delle dinamiche di un call center per un giorno lavorativo, si formula il problema di scheduling come un modello di ottimizzazione stocastica in cui l'obiettivo e' quello di minimizzare il costo totale degli agenti utilizzati, sotto vincoli non lineari sui livelli di servizio.

In primo luogo, si dimostra come la metodologia a due stadi fornisca soluzioni che sono, in molti casi, lontane dalle soluzioni ottime.

Quindi, si descrivono due possibili approcci risolutivi, alternativi a quello a due stadi, per risolvere il suddetto problema di ottimizzazione.

Il primo e' un approccio di tipo cutting-plane in cui i vincoli non lineari sui livelli di servizio vengono dapprima rilassati e successivamente stimati, utilizzando la simulazione. Qualora la soluzione, correntemente trovata, non dovesse soddisfarli, vengono trovati dei tagli (cioe' vincoli lineari che eliminano dalla regione ammissibile la soluzione corrente senza eliminare nessun altra soluzione ammissibile) da aggiungere alla formulazione, utilizzando, ancora una volta, la simulazione. La procedura termina quando la soluzione finale risulta ammissibile per una simulazione con una fissata lunghezza. Quindi, l'approccio prevede l'applicazione di un algoritmo di ricerca locale sulla migliore soluzione trovata. Questo approccio euristico cerca, muovendosi nell'intorno della soluzione corrente, di migliorarne la qualita' (decrementandone il costo totale).

Il secondo approccio, invece, prevede la definizione e l'implementazione di una procedura di ricerca casuale. E' mostrato come l'implementazione della suddetta metodologia non riesce ad ottenere soluzioni di buona qualita' (quindi basse in costo) soprattutto all'aumentare dello spazio di ricerca e quindi della complessita' della specifica istanza risolta.

Utilizzando alcuni esempi, tratti dalla letteratura e da framework operativi reali, e' mostrato come la procedura di cutting plane fornisca la migliore soluzione di scheduling rispetto alle altre metodologie risolutive sopra descritte (approccio a due stadi e ricerca casuale).

Una sezione specifica viene dedicata all'estenzione della metodologia di cutting plane, introducendo anche il controllo sulla probabilita' di abbandono dei clienti. Alcuni risultati vengono, quindi, analizzati e presentati.

Infine, due approcci risolutivi, ancora oggetto di studio, vengono proposti come possibili alternative all'algoritmo di cutting plane.

Abstract

The aim of this thesis is to develop a methodology for solving the agents scheduling problem for a multi-skill call center.

A multi-skill call center manages different call types distinguished on the base of their specific requirements. Each group of employees (also called agents) is distinguished by the number of different call types (i.e. number of skills) can be handled. The set of rules for assigning the calls to the agents (or viceversa) is named routing. In a typical call center, the inbound calls arrive in a random way accordingly to a sophisticated stochastic process. There are also other random components: the calls duration,the waiting times in queue before the service, the abandonments (i.e. users abandoning the system before the service and after waiting a specific random patience time) and, in some operative contests, also the number of customers trying again for the service after abandoning the system.

In general, the working day is divided into periods (each of about 15 minutes long) for defining agents pauses and detailing the calls trend during the day (specifying the arrival rate). Moreover a shift for an agent is a sequence of working periods (including pauses) and it is defined by the periods in which the agent is available and can handle the calls (i.e. he/she has all the required skills).

The main goal is to guarantee the service quality at minimum cost. The most common service quality measure is the service level, defined as the fraction of the calls whose waiting time in queue is not longer than a fixed threshold. Usually, different measures of service quality are considered: for a given period, for a given call type, for a couple (call type, period) or/and global (i.e. aggregated over all periods and call types).

Considering the forecasts of the calls volumes, the call center managers have to decide (among the other things) how many agents of each type have to be present in the center in each instant of the working day; they have to define the agents shifts; they have, also, to decide about the routing rules. These are all decisions taken under an high level of uncertainty.

The classical problems, related to the managing of the agents, could be divided into two categories: staffing and scheduling problems. In the first case, one has to establish how many agents of each type have to be assigned to each period of the working day in order to satisfy the service levels at minimum cost. In the second case, instead, one has to find the assignment, at minimum cost, of the agents to the different shifts in order to satisfy the service levels. The two problems are not independent from each other: from the final scheduling solution, it is possible to find the staffing one.

Note that, in both cases, the solution feasibility has to be checked using simulation due to the presence of nonlinear constraints (on the service levels, for example). For this last aspect, these two problems belong to the class of simulation optimization problems.

In literature, the classical approach for solving the scheduling problem is a two step one: in the first step a staffing problem is solved for each period; in the second one, instead, feasible schedules for the agents are found considering the staffing information.

In this thesis, supposing to have a detailed stochastic model of a call center dynamics for a working day, the scheduling problem is formulated as a stochastic optimization model with the goal of minimizing the total agents cost, under nonlinear service levels and/or abandonment constraints.

Firstly, it is shown that the two step methodology gives sub-optimal solutions. Secondly, two solving approaches are described for solving the same model. The first one is a cutting plane approach in which the nonlinear service levels and/or abandonments constraints are relaxed and then estimated by simulation. If the solution is not feasible for all of them, then some linear cuts (i.e. linear constraints that remove from the feasible region the current solution but any other feasible one) are added to the formulation. Simulation is again used for finding these cuts. The cuts generation process ends when the final solution is feasible for a simulation with a fixed length. Then the method applies a local search module on the best solution. This heuristic approach tries, exploring the neighborhood of the current solution, to improve its quality (decreasing the total cost). The second approach, instead, defines and implements a random search. It is shown that the implementation of this specific procedure is not able to find good solutions in particular when the search space becomes too large.

Using examples, some taken from literature and some inspired by real life, it is shown how the cutting plane approach gives the best solutions in regard to the others.

A specific section is dedicated to extend this methodology for considering also the control on the abandonments. Some results are presented and analyzed.

Finally, as future work, other two possible solving approaches are proposed.

Chapter 1

Introducing Call Center

Firstly, a brief introduction to the call/contact center world is proposed. Secondly, it is shown how the customer satisfaction can be evaluated. Then, in the remain sub-sections, it is shown how a call center, single-skill and multi-skill, can be modelled using the queue theory and so it is also defined what types of problems the managers should deal with. Two of them are described with more details: the *staffing* problem and the *scheduling* one.

Finally, the general thesis organization is shown.

1.1 Contact and Call Center

Nowadays each organization has a *contact center* through which the customers or users can contact the organization and vice-versa, by telephone, FAX, email, Internet chat, Web and so on (figure 1.1).

For example, in figure 1.2 ((Sharp 2003)) some possible ways, that a customer can use to call a contact center, are shown. Customers, usually, can choose the communication channel. There could be some companies that want to use the old communication systems, but they will gradually be replaced because they will no longer be competitive. The Internet spread has allowed to transform the typical call center in a more general contact center. Call centers are automated service delivery points, full of customer data and products, dedicated to one major objective: providing customers with whatever services or products they require. How does Internet fit into the call center model? It is just one more communication channel for the customer, a channel that call centers need to manage at least as well as traditional communication channels.

In figure 1.3((Sharp 2003)), it is shown that each communication channel has a specific cost. More specifically the figure shows as these costs, related to the use of a communication system to handle a call, vary with the used channel. The contact center becomes *call center* if the communication is only made by telephone. The call center industry is developing a lot. Recent studies show that about 3% of workforce in the United States and Canada works at a call center. Just to have a brief idea about, some of television advertising end with a message that suggests you to call to a call center in order to have more information about the product/service, for example. There are other typical examples of using a call center : to have a pizza delivery service, to rent an

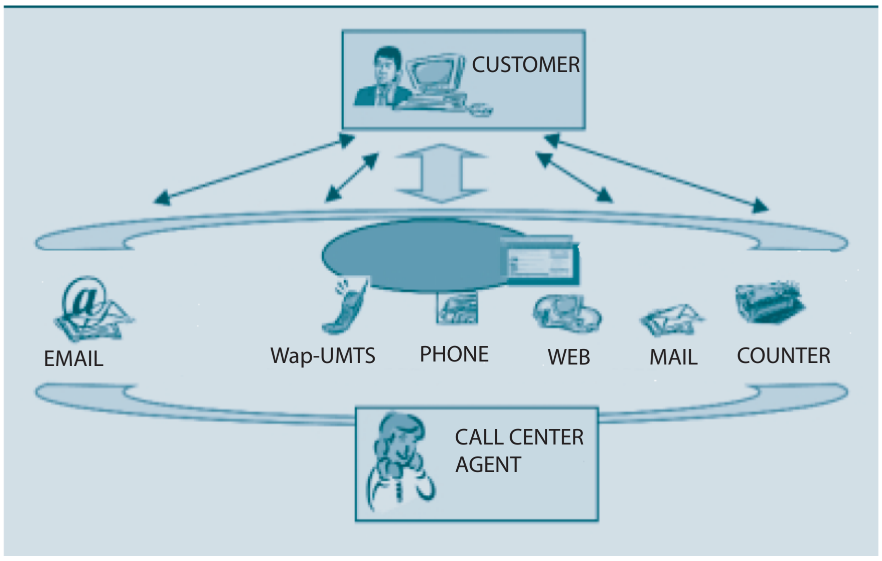

Figure 1.1: Contact Center Architecture

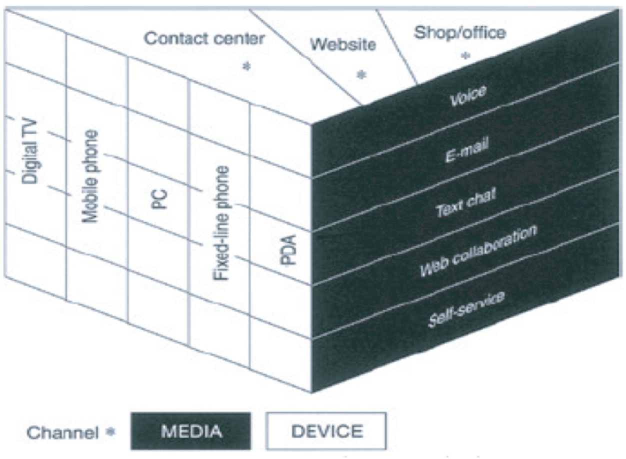

Figure 1.2: Multidimensional customer contact

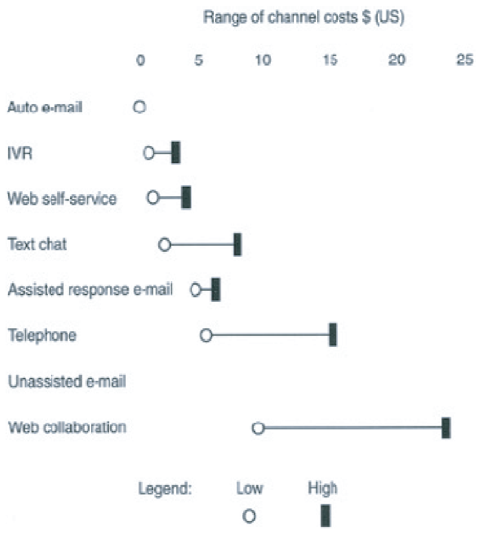

Figure 1.3: Cost comparisons for different media channels

hotel, to have a service on mobile phone and so on.

Then it is not a surprise if most of the operating cost of a call center (around 3/4) is due to the labor costs. For this reason, in general, it is important to schedule in an optimal way the call center employees (also called *agents*), minimizing the costs due to their salaries and satisfying some constraints (usually imposed on the service levels and/or on the abandon probability of the customers). By 2008, various studies predict that: the United States will have over $47,000$ call centers and 2.7 million agents; Europe, Middle Est and Africa together will have $45,000$ call centers and 2.1 million agents; finally Canada and Latin America will have $305,500$ call centers and $730,000$ agents (Akşin et al. (2007)).

Koole and Pot (2006) affirm that the larger call centers employ about from 500 to 2000 agents, there are ten thousands of call centers across the globe, and between 1,500,000 and 2,000,000 agents. It is expected that India, for example, will build up an industry that is worth 17 billion by 2008. In this country, in fact, 100,000 of agents who can work in call centers, graduate each year.

In Mehrotra (1997), a call center is defined as a group of people whose main economic activity is the communication towards the clients or potential ones. The main components that build a call centers are a set of phone lines, of routers, of employees (or *agents*) and computers (figure 1.4((Sharp 2003))).

In particular, all turns around the calls that can be *inbound* (customers towards call center) or *outbound* (call center towards customers). The latter is done, usually, like an advertising for a new product or a new promotion of the company. If a call center handles both types then it is mixed. In any case, all the calls can be classified in regard to their *type* (inbound or outbound for example), the *required service* and their *origin*. This last aspect is important because the

Figure 1.4: A typical Call Center

place from which a call comes specifies also the language to be used during the conversation. This means that, for example, for all the bilingual countries, a skill required for the call center agents could be related to the spoken languages.

An outbound call can be done or directly from an agent who selects a telephone number from an ordered list made in advance or from an automatic dialer (called *automatic dialling unit*) that keeps the phone numbers accordingly to some rules. In the latter case, if there is a successful connection, then the automatic dialing unit forwards the call to the first free agent that can handle it. When all the agents are busy, then either the call is put in queue or it is suppressed.

The inbound calls, instead, arrive to the call center for different reasons and they try to access to the phone lines. If all the phone lines are busy (e.i. there is not a free employee with the correct skills), then it is put in queue and the customer receives a signal that alarms him/her. It is important to note that if the waiting time for a customer is too long, then he/she will decide to close the communication (e.i. *abandon*). Who has to handle these calls is the employee or agent that belong to the *Customer Service Representatives* (CSRs). Also about them, it is possible to have a classification into different groups in regard to the capabilities. For example if the calls can only arrive to the call center from France and England, then a generical agent could be able to speak only in French, or only in English or both (in French and in English). In the first and in the second case, the agent is a specialist, instead in the third case a generalist. If in a call center only one skill is required to the agents, then it is called *single skill* call center, otherwise it is *multiskill*.

1.2 Customer satisfaction and service quality

The customer satisfaction is an efficient instrument to measure the level of the services to the customers, also to verify the performance guaranteed by an external service, in the case in which a service is given from a third entity. The elaboration of the data coming from the customers revelations could be an input to improve and to innovate the services. A bad service could generate two possible cases: it is not noticed or it is not managed by the operator (the customer is not satisfied); the operator notices the bad service and manages it (the customer is satisfied)(figure 1.5).

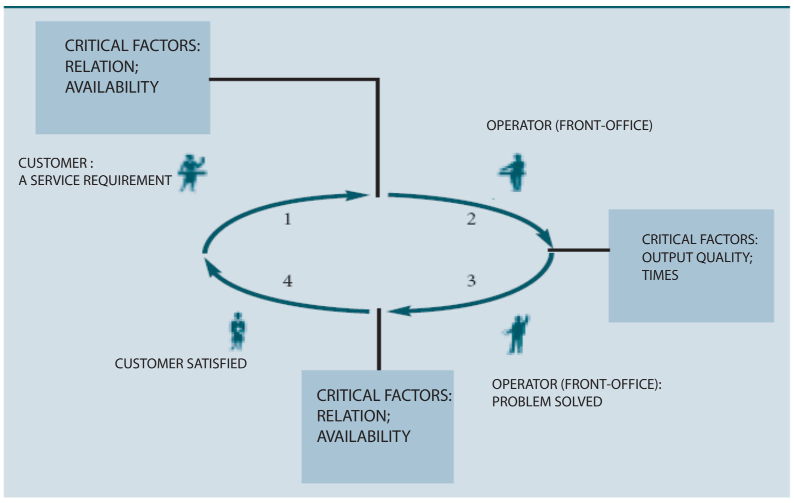

Figure 1.5: The monitoring of the perfermance and of the service quality

The entire call center organization is built around the customer requirements (figure 1.6). The service quality of a call center (how the agents work or equivalently how much the customer satisfaction is) can be measured through the so called *service level* (defined later). It is evident that this measure depends strongly on the objective of the call center. For a call center that has to handle emergency calls then it is important the number of busy phone lines because it is supposed that all the customers have to be served in a reasonable time.

Usually the managers have to balance three important components that drive the quality of the call centers and are shown in figure 1.7: costs, employee satisfaction and service quality. It is clear enough that employee satisfaction and service quality, in particular, are linked to each other. In fact an employee that is satisfied guarantees a good service and it means that the customers are satisfied too.

Another important performance measure, for a call center, is represented by the number of abandons and/or retrials. These are two subjective measures because the customers's point of view is incorporated. Then the customer's patience could be computed and evaluated in terms of the ratio between the number of abandonments and the total number of arrivals (Bassamboo et al. (2004)).

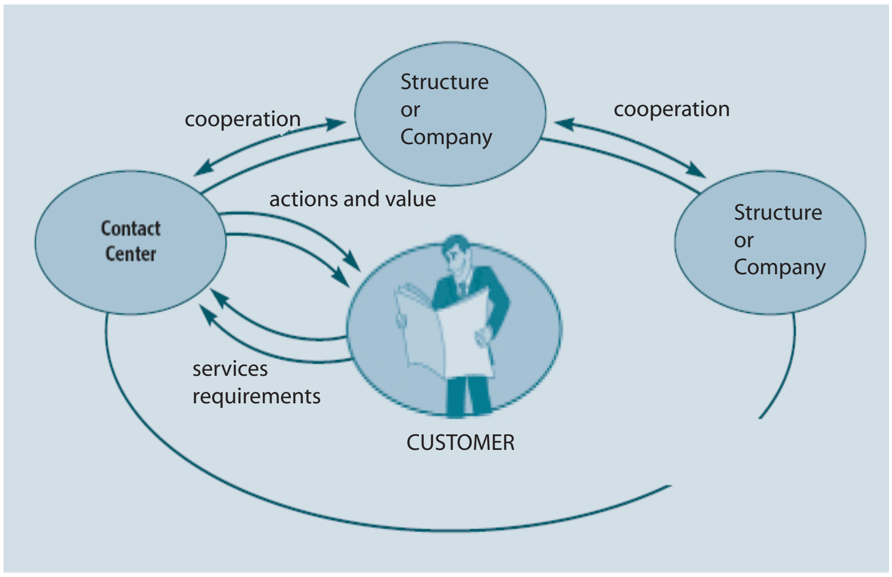

Figure 1.6: The importance of the customer satisfaction

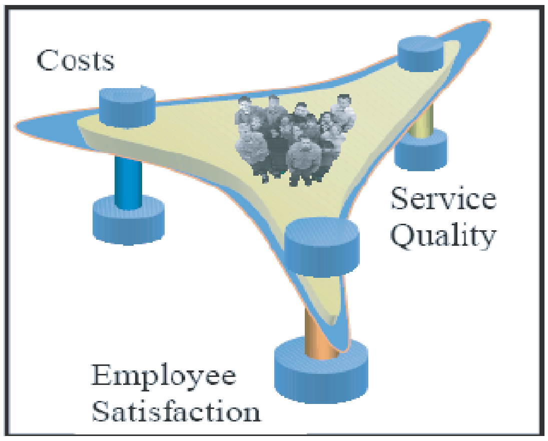

Figure 1.7: the three components that drive the quality of a call center

In particular, one can also define service level as the average waiting time of all served calls that is the sum of waiting times divided by the number of served calls.

Moreover, the service level is the average number of calls served within a time limit called AWT (*Acceptable Waiting Time*). For example 80%, such as they want that at least 80% of the calls are served within 20 seconds.

These two definitions are used in the case in which there are no abandonments. In case of abandonments, instead, one has to define, for example, the service level as the ratio of calls served within AWT and the number of served calls plus the ones that abandon after waiting a time less than AWT. Of course it is one of several possible definitions; some call centers use slightly different definitions, depending on what they do with abandonments.

It is also possible to consider a limit for the total number of abandons. For example,call center managers could require that the total number of abandons is at most equal to a fixed threshold. In real cases, this threshold is between 5% and 20% of total calls.

1.2.1 Some waiting time metrics

Two of the most important waiting time metrics used in the call center environment are: TSF and ASA.

Definition 1.1. *The telephone service factor (TSF) is defined as the percentage of calls that is answered in less than a certain fixed waiting time.*

Definition 1.2. *ASA is defined as the average speed answer.*

The TSF is commonly used as service level metric in call centers, but it is very important to pay attention to its interpretation. For example, 80/20 TSF means that the calls have to be served within 20 seconds (from their arrival) to have a good service quality. An 80/20 TSF means that 80% of the calls receive good service, and 20% bad.

For the unsatisfied call the service level becomes relevant when he-she tries to call again. If the TSF at that moment is again 80/20, then the probability of another bad experience is 0.2, or 20%.

Considering, for example, 2 types of calls for which an 80/20 SL is targeted. Now what happens if 70/20 is obtained on one and 90/20 on the other? Moreover, what if there is the choice, with the same means, between 70/20 and 90/20 or 75/20 and 80/20? The former has a better average SL (assuming an equal load), the latter shows less variance. The answer depends again on how the different call (customer) types are valuated and the nature of the service.

Formula 1.1. *TSF*$= 1 - C(\lambda, \mu, N)e^{-(N-\rho)AWT/\mu}$

where:

- ρ represents the load (see Chapter 2 for a more detailed definition);
- $C(\lambda, \mu, N)$ (*probability of delay*) is the probability that an arbitrary call type finds all agents busy (see the next paragraph).

A particular case for this formula occurs when the number of agents N is lower than the load ρ and TSF becomes equal to 0.

As a direct consequence of this formula:

Formula 1.2. $ASA = \frac{(C(\lambda,\mu,N)\frac{1}{\mu})}{N-\rho}$.

1.3 The importance of the economies of scale in the call center

The number of calls, that usually arrive in a call center considering a fixed time unit, is unpredictable. In a small call center the deviation between the real number of arrivals and the estimated one can increase in a considerable way due to the fact that in general this number is small. In fact one can mathematically express this dependency through a simple formula: assuming that X is the number of calls arriving in a time unit (0,t) and that the arrival process follows a Poisson process whose rate is λ, then the variation coefficient is $\upsilon_X = \frac{1}{\sqrt{\lambda}}$, i.e. the standard deviation divided by expectation. So it is evident that if the rate λ is small (in a small call center), the coefficient υ is large; if, on the contrary, this rate λ is large (large call center) the variation υ is small. This is a very important aspect that implies a simple consideration: in a large call center one is able to predict in a more detailed way the workload in a fixed period due to the fact that the randomness source is very little. Of course, this does not happen in a small call center where the workload forecast becomes hard.

In a single skill large call center environment,the staffing requirements can be approximately equal to the workload prediction.

But in a multi-skill call center, when the size becomes larger, to have agents specialize in specific tasks can be beneficial because, limiting the number of different tasks of an agent decreases the handling time of a job and shorter service times increase the productivity.

In a medium size call center, instead, to have only specialists or only generalists is inefficient. In this last case, it is very crucial to balance the number of specialists with the number of generalists with the main goal to meet the service levels as possible.

It is clear that a good policy helps.

1.4 How is it possible to model a call center?

In this section, a single-skill and a multiskill call center are modelled using the queue theory. The attention will be also focused on the realistic case with abandonments.

1.4.1 A generic queueing model for a call center

Queueing theory was introduced by A.K. Erlang at the beginning of the 20th century and has become one of the most important research themes of Operations Research. In a generic queueing scheme, one has: *clients* (or customers) , *servers* and *queues*. In order to model a call center as a queueing system, the callers become clients, the agents become servers and queues are made up by callers that require a service and that wait for it. In particular the queues represent the meeting point for managers, service providers and customers. The first can use them in order to check and improve the services, instead the customers use them in order to access to the services of the system. For this reason, queueing models are the natural ground to develop tools for the call centers. A possible queueing model is the so-called M/M/N system (also called the *Erlang-C* model). According to it, the calls are characterized by *arrival rate* λ and the N agents, working in parallel, by the *service rate* μ^{-1}. Then, the Erlang-C formula $C(\lambda, \mu, N)$ describes, for example, under the very restrictive assumption of no abandonments, the fraction of customers delayed in queue before receiving the service. But it is a very restrictive model to represent all the call center aspects. In fact, for example, it is assumed, among other things, a *steady-state* environment in which the arrivals occur according to a *Poisson process*, the service durations are exponentially distributed, and customers and servers are statistically identical. But, they do not act independently from each other. Moreover it should be remarked that the strong assumption, for a realistic framework, is that abandonments do not occur. For all these aspects, Erlang-A systems, which can explicitly model the customer patience, were developed. In

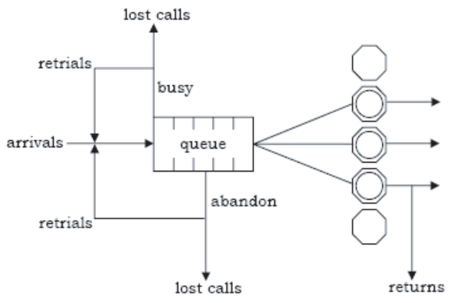

Figure 1.8: A queue model for a call center

figure 1.8, a simple call center model is shown, in which there are only 4 agents and 1 queue. In a realistic framework, the scenario is more complicated than the one in figure 1.8.

A functionality, available in some systems designed to handle and manage large volumes of incoming calls, is the *Automatic Call Distribution* (ACD). Typical applications include customer service desks, telemarketing operations, reservation systems, and so on. An ACD allows efficient distribution of calls to available operators or voice processing options such as voice mail. The customers are represented by "arrivals" such as the inbound calls that require some services. In figure 1.9, an example of communication with a call contact center, through an ACD, is shown.

There are two different cases: in the first one, the call is done without success ("lost call") and the customer, may be, will retrial in another time. This case can occur because the queue has a fixed capacity and it could be "full". In the second one, instead, the call is put in queue because the queue is not busy and there is a free slot for it. The fact that a call is in queue does not guarantee that is served. In fact if the customer waits in queue for a long time, he/she could decide to abandonments and may be to try again in another time. But if the call is served in time, then the specific router will assign it to the first free agent (in a single skill call center), to the first free agent that can handle it (in a multiskill call center). If a customer is unserved, he/she could return in another time

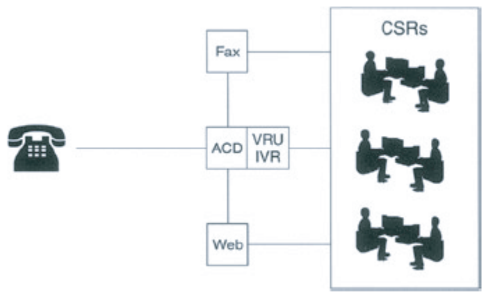

Figure 1.9: Communicating with a call contact center through an ACD

(”returns”). It is easy for the managers to experiment with different situations or scenarios that can occur, for example, varying the number of active trunk lines.

One can generalize considering the flow chart in figure 1.10 as shown in Koole and Pot (2006).

The flow chart is generical and means that the number of customers who abandonments the call center is influenced a lot by the specific routing rule chosen. When a customer finds all the lines busy then he/she can be put in queue or could decide to retry in another moment. If he/she does not retry then he/she is a lost customer. Obviously the call center managers want to limit the number of lost customers meeting the service levels.

Gans et al. (2003a) affirmed that, for most inbound call centers, the management objective is to achieve relatively short mean waiting times and relatively high agent utilization rates.The authors called this environment like Quality and Efficiency Driven (QED) regime. If, in the following, R is the system offered load, in terms of the mean arrival rate times and the mean service time, then the so-called Square-root safety-staffing rule affirms that if R is large enough, staffing the system with $R+\beta\sqrt{R}$ servers (for some parameter β) achieves both short customer waiting times and high server utilization. This rule was observed firstly by Erlang (1923) and then formalized by Halfin and Whitt (1981) for the Erlang-C model. Borst et al. (2004) identified two other operating regimes: the Quality Driven (QD) and the Efficiency Driven (ED) regimes which are rational operating regimes under certain costs structures. In the latter, the server utilization is emphasized over service quality. However, with customer abandonment, this regime can also result in reasonable performance as measured by expected waiting time and fraction of customer abandonment (Whitt (2004e)).

The Erlang C formula has some very relevant properties that one could summarize as in the following:

1. **Lack of robustness**: even for big call centers, with a moderate service level, one, with a relatively limited effort, can increase the SL to an acceptable level. But a higher load, needing an additional agent, can deteriorate

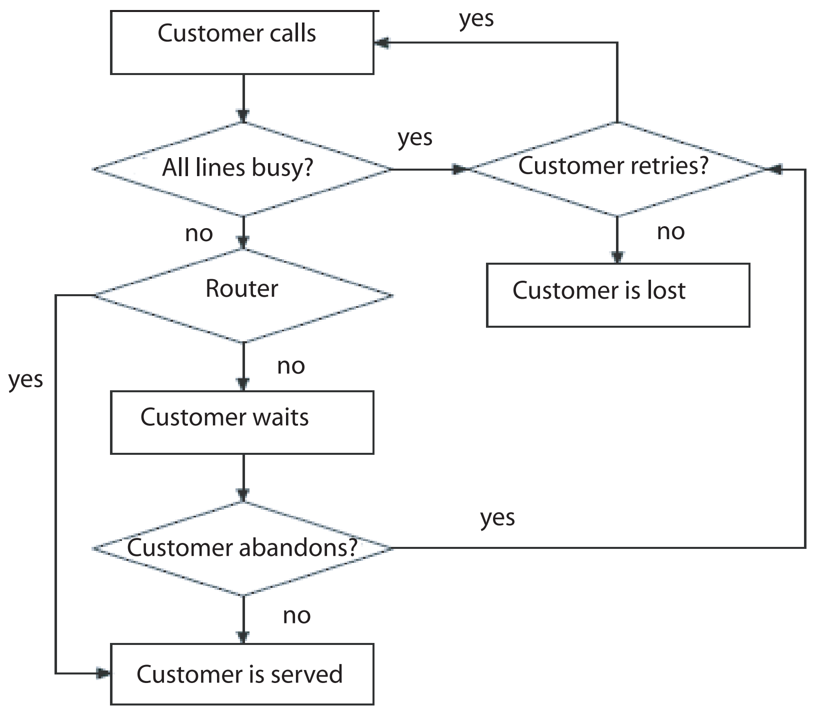

Figure 1.10: A flow chart of a call center customer

the service level considerably. For this reason the formula is very sensitive to the variation of its parameters:λ, μ and N.

2. **Stretching time**: If either 1μ or λ becomes doubled and the other is divided by two, then the load ρ remains the same. This does not mean that the same number of agents is needed to obtain a certain service level. If AWT is multiplied by the same number then the TSF remains the same. The relationship between the ASA and stretching time is more complicated. If time is stretched, and the acceptable waiting time is stretched with it, then the TSF remains the same. Of course, this is just theory, although often the AWT is higher in call centers with long talk times compared to call centers with short talk times. For the ASA the effect of stretching time is simple: the ASA is stretched by the same factor.

3. **Economies of scale**: big call centers work more efficiently.

1.4.2 A queueing model for a multiskill call center

In Section 1.4.1, a generic scenario is shown in order to figure out how to use the queue theory to analyze a call center. But in this thesis, the multiskill call center is analyzed. The scenario is made up by a set of call types, each of them with a particular requirement, and a set of agents, each of them with a particular skill set. It means that only the so-called *generalist* can handle all types of calls, instead the *specialist* can handle only the calls that require his/her skills. Of course they are only two extreme and particular cases. In many large call centers, all the agents are somewhere in between. In figure 1.11 it is supposed

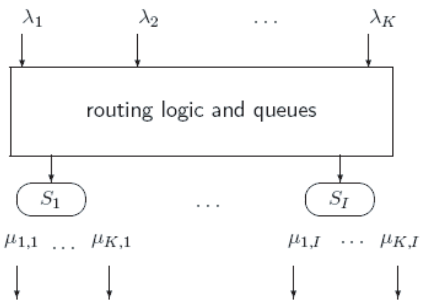

Figure 1.11: A general multiskill call center

to have K call types and I agent types. The λ_k's are the arrival rates for a call type k, assumed exponential distributed; the S_i represent a set of agents whose skill group is i and finally $1/\mu_{k,i}$ is the mean service time required from the skill group i for the call type k. Each inbound call is dispatched accordingly to the router policies to an agent who belongs to a skill group that can handle it. In the multiskill call center, the service rates may depend not only on the agent but also on the call.

In general there are different types of router policies. In particular, in the *static routing*, each inbound call has an ordered list of agent groups to try and only if all are busy, then it put in queue. Again, there could be different scenarios: one queue per agent group, or one queue per call type or one single queue for several call types or a mixture of these (Cezik and L'Ecuyer (2006)). More complicated scenarios could require to add also the priorities used by the agents to serve the inbound calls, for example. Instead in the *dynamic routing* the decisions may depend on the entire state of the system (the current time, the number of calls of each type in service and in the queues, the elapsed service time of the calls in service, etc). In general, due to the fact that the optimal dynamic scheme is too complicated, the routing strategies are selected from a specific set of simpler rules.

The literature is full of studies on the single-skill call center and the related models are based on the Continuous Time Markov Chains (CTMCs). In fact Erlang C and A models are special cases of CTMCs.

But due to the dimension problems (the number of the operations exponentially grows up with the number of call types and agents types), for the multiskill environment, approximated models are used.

1.5 Why simulation?

In this section, it is clarified what simulation is and why it is important to use it, describing two different types of simulation used in call centers.

On the other hand it is also described, in a brief way, the approach used in this work: *simulation-based optimization* (or *optimization via simulation*).

"*A simulation is an imitation of some real thing, state of affairs, or process. The act of simulating something generally entails representing certain key characteristics or behaviors of a selected physical or abstract system*" [Wikipedia]. In a more specific way, simulation in computer science is referred to what happens when a digital computer runs a program describing a particular system.

Sometimes, simulation is used jointly to optimization. In this way,simulations of physical processes are used in conjunction with evolutionary computation to optimize control strategies. "Evolutionary computation" is just one of the many ways doing optimization via simulation.

1.5.1 Steady State Analysis vs Transient Simulation

When a call center model is constructed, then one can analyze its performances making some experiments on it. In particular two types of simulation analysis can be performed: the *steady state* and the *transient.*

Steady State Simulation

In some cases, one can be interested to evaluate the system answer in a long run, such as estimating its performances in an infinite horizon, to verify, for example, if the resources dimension is that to avoid eventual bottleneck decreasing the customer satisfaction. This is a *stationarity simulation* (or *simulation on a infinite horizon* or *steady state* simulation). In these cases, the time limit of the simulation is not fixed a priori but only after executing a run and the results analysis. The reason for this type of simulation can be related to the experiments goal (i.e. the transitory length evaluation correspondent to an initial state of the system) and to the statistical analysis on the performance measures of the system.

In a long-term simulation, it is very important how to get the results. In fact, the infinite horizon simulation has to be truncated and it represents a source of bias in the estimators. In order to reduce the bias, one has to use simulation budget for a single, long replication, and not multiple replications.

When one is interested in controlling the statistical error on the average values of the performance measures, it is necessary to evaluate the so-called *confidence intervals*. In general, it is not simple to estimate the variance on a single long run due to the sample size. To overcome this problem, the total simulation time is divided into batches to get (almost) independent observations. Each batch has a fixed duration s in simulation time units. The set of batches can be the base to determine the interval confidences (*batch means* method).

Transient Simulation

If one wants to evaluate the system answer in a short term, then a *transient simulation* (or *terminating simulation* or *simulation on a finite horizon*) is required. It lasts the necessary time for recording the sequence of the system changes that are related to the sequence of events.

Then, a finite horizon simulation is used to estimate short-term performance measures (for a day, a week, etc). In this case, to compute confidence intervals on performance measures, the simulator performs a determined number n of independent replications. In a replication, the simulator is initialized, the whole horizon is simulated, and statistical observations are collected for each estimated performance measure. During a replication, it is important to remark, that random numbers generators are used but the model is unchanged.

For this reason, when a mathematical model used to describe a real system is studied by simulation, it becomes a *simulation model.*

1.5.2 Simulation-Optimization

As already said in section 1.5, simulation can be used jointly to optimization in order to control and describe some complicate real system. For example, in call center, a such approach (simulation and optimization at the same time) is used in order to solve the scheduling problem where the service levels constraints are too complicate to be computed exactly.

In fact, when the system to be optimized is too complex, usually using simulation and optimization is a natural approach.

In order to fix the idea, one can assume that a generical simulation model is given with n input variables ($x_1 \dots x_n$) and m output variables ($f_1(x) \dots f_m(x)$ or $y_1 \dots y_m$)(fig.1.12) .

Simulation can be linked to optimization to be used efficiently in order to design systems. In fact simulation optimization has the main goal to find the optimal set of values for the input variables, such as the values to be given to X that optimizes the output variables (Y).

A possible simulation optimization model can be represented in figure 1.13 where output is used by an optimization strategy to give feedback during the solution search.

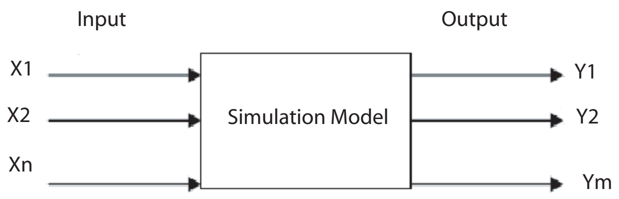

Figure 1.12: Simulation model

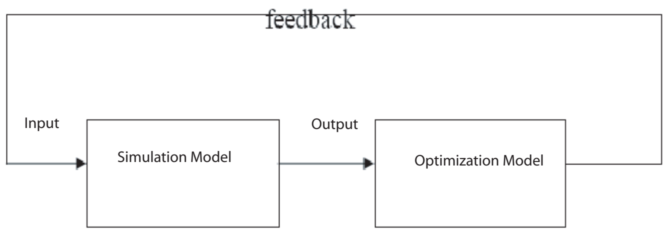

Figure 1.13: Simulation-Optimization model

Nowadays, simulation optimization is an area that is attracting the attention and interest of researches. In fact recently there has been considerable research focused on how to combine simulation and optimization in practice (Fu et al., 2000; April et al., 2001; Ólafsson and Kim, 2001; Fu, 2002). For this reason, simulation optimization can be considered as an active field of research and it is also increasingly being used in practical simulation applications and being incorporated into simulation software tools.

In this work, simulation optimization approach is used to solve the scheduling problem for a multi-skill call center where the simulator evaluates the service levels constraints and the optimizer solves different optimization problems at each step of the algorithm.

1.6 Staffing and Scheduling problem

It is well known that in the call center environment there is a large number of optimization problems because of the large variety of managers objectives. They want to obtain "good solutions" in order to save their budget using as well as possible the demand forecasting and so satisfying the customers (*tactic decision*). On the other hand, there are also *strategic decisions* that employee the managers in searching for new services or improving the already existent ones. But if they are typically long or medium-term decisions, usually the managers have to handle also short term decisions.

Considering a quantitative approach, a call center manager can react to the different situations (too many customers are abandoning or waiting). But the way in which the manager reacts could be not appropriate and could have a different consequence than what his goal is. For this reason, actually, many managers prefer to follow an approach for which a manager could be "*pro-active rather than active*" (Koole and Mandelbaum (2002)).

Due to this last aspect, it is important to define mathematical models that can describe all the managers requirements. The two relevant call center problems are the *staffing* and the *scheduling* one. For both, the working day is, in general, divided into *periods* and each of them has a fixed duration. In this way it is possible to take in account, for example, of the lunch break and the pauses. Moreover, in the two problems, the customer satisfaction can be taken in account through the service levels constrains.

The staffing problem determines the number of agents of each group required for each period in order to meet the service levels constraints.

In the scheduling problem, instead, firstly a set of feasible work schedule (also called *shifts*) is decided and then it is found the number of agents of each type to be assigned to a shift in order to meet the service levels constraints. This problem is complete in the sense that it implicitly decides how many agents will work in each period.

In both problems

1. the *objective function* is to minimize the total cost due to the agents salaries. But if in the staffing problem, an agent cost only depends on his/her type, instead in a scheduling one, it also depends on the shift type to which he/she is assigned. In fact, as shown later, it is important, for the quality of the final solution, to have different type of shifts, where a *shift type* is different from another only for length;

2. the *service levels constraints* are the fraction of calls answered within a certain time limit that, in the long run, exceeds a given threshold. They are defined, in this work, for each call type and period, for each call type (and so grouped by periods), for each period (and so grouped by calls) and finally the global one (grouped by calls and periods). In this thesis, the abandonments constraints are also considered fixing a maximum threshold on this number.

The main goal of this thesis is to solve efficiently the scheduling problem for a multiskill call center. An efficient method, based on some analytical approximations, to solve the staffing problem, for one period and for a multiskill framework, can be found in Avramidis et al. (2006).

Atlason et al. (2004a), Atlason et al. (2004b) have proposed a general methodology to optimize the scheduling of agents in a single-call-type and single-skill call center, under service level constraints. Their method combines simulation with integer programming and cut generation.

Cez̧ik and L'Ecuyer (2006) have designed an iterative cutting plane algorithm on an integer program, for minimizing the staffing costs of a multi-skill call center subject to service-level requirements which are estimated by simulation. Our cutting plane algorithm is an extension of this approach in order to solve the scheduling problem for a multi-skill call center.

1.7 Demand Forecast and Rostering problem

One of the most important issue, when managing a call center, is demand forecast. It is clear that in this context the term "demand" means number of calls arriving to the call center. One can define the call forecasts according to the specific queue or call type associated with the forecast; to the time between the creation of the forecast and the actual time period for which the forecast was created and to the duration of the time periods. For this last aspect, in fact, the forecast could be over a month or over a short period (15, 30, 60 minutes).

Over the years, there were a lot of scientific contributions about this aspect.

For example, Weinberg et al. (2007) proposed a multiplicative effects model for forecasting Poisson arrival rates for short intervals, typically of length 15-, 30-, or 60-minutes, with a one day lead time.

Soyer and Tarimcilar (2007) introduced a new methodology for call forecasting that draws on ideas from survival analysis and marketing models of customer heterogeneity.

Shen and Huang (2007) developed a statistical model for forecasting call volumes for each interval of a given day, and also provided an extension of their core modeling framework to account for intra-day forecast updating.

When one has a well fixed set of available agents to be scheduled for the day or the week, where each agent has a specific set of skills, then the problem becomes a scheduling and rostering problem. This last problem will be not discussed in this thesis. Similiarly, Cez̧ik and L'Ecuyer (2006) proposed a methodology combining linear programming with simulation to determine a schedule. Avramidis et al. (2006) developed search methods using queueing performance approximations in order to produce agent schedules for a multi-skill call center.

1.8 Contact Center Simulation Tools

Due to the increased importance of the contact center industry, it becomes also relevant to develop some software tools in order to simulate its behavior. It is clear that they can help the managers analysis and can improve the performances of these centers.

A simulation tool can be seen like a computer game reproducing exactly the behavior of the call center. As already said, it is relevant to the decisions the managers want to make.

The positive aspect is that one can try everything he/she wants and simulates different scenarios.

There are some software tools already available to the managers: Arena Contact Center Edition, ccProphet and so on. Their main feature is to have a good graphical module that allows the user to analyze in a simpler way the final results. Sometime this graphical representation can be helped by a good animation presenting with a lot of details the situation. But these positive aspects, usually, are paid a lot because of these software are slow and not flexible even if they are very expensive.

The present thesis uses ContactCenters library (Buist and L'Ecuyer (2005) and Buist (2005)) developed in the Prof. L'Ecuyer laboratory.

It is a java library developed on SSJ (L'Ecuyer and Buist (2005)). It presents a strong expressive power and it is GUI-independent. Moreover, the way in which it is developed allows the user the interoperability with other software (statistics, optimization, databases, etc).

Just to have an idea about its performances, one can compare its speed with another good software like Arena. ContactCenters Library is resulted 30 times faster than Arena (in some experiments that have been conducted in the same lab).

The general framework of this library provides building blocks to simulate all types of call centers. Moreover it supports several types of contacts, multiskill, blend, arbitrary dialing and routing policies, various types of arrival processes and so on.

1.9 The general thesis's organization

This thesis is organized as follows:

- firstly, some mathematical formulations are shown (Chapter 2). In particular in this chapter, mathematical formulations are proposed for the staffing and scheduling problems. The staffing ones are presented for better understand the two step approach, for example, even if the main goal is to solve the scheduling problem;

- secondly, the two step approach is presented for solving the scheduling problem. In particular the focus of this section is to show that its solutions could be considerable sub-optimal (Chapter 3);

- in chapter Chapter 4, some lower bounds on the total number of shifts are presented in order to reduce the overstaffing of the final solution without considering a specific methodology for obtaining it;

- then in chapter Chapter 5, the cutting plane methodology and its different versions are described.
- in the chapter Chapter 6 another heuristic approach (Randomized Search) for solving the scheduling problem is proposed, extending the one already defined for a single staffing problem of a multi-skill call center;
- in chapter Chapter 7, an extension of the cutting plane methodology is proposed in order to take into account not only the service level constraints but also the ones on the abandons probability;
- chapter Chapter 8 is dedicated to present some future works, i.e. two different new approaches, as alternative to the ones already described in the previous sections for solving the scheduling problem;
- finally, chapter Chapter 9 reports all the computational results considering call centers of different size.

Chapter 2

Mathematical models

In this section, the mathematical formulation of the staffing and the scheduling problem is presented. For the scheduling problem, in particular, different types of formulations are proposed in order to reach an unique model that can better represent all the constraints. It is important to remember that all these formulations are defined in a multiskill environment.

2.1 A general framework

Whenever possible, in the following, numerical values are not specified to maintain generality. The working day, whose duration is equal to h, is supposed to be divided into some periods, each of them with a fixed length d (usually equal to 15 or 30 minutes or sometimes to 1h). The set of the possible periods is $P = \{1, \ldots, p\}$. Considering some constraints, taken from real call centers, the agents have two different types of pauses: the *lunch break* and the *coffee break*. Dividing the working day into periods allows describing better all the possible feasible schedules (*shifts*) for the agents.

Definition 2.1 (Shift definition)**.** *A shift is a unique time pattern of an agent availability to handle calls.*

It is characterized by: *start period*(time in which the agent starts work); *break period*(time in which the agent has a break) and *end period*(time in which the agent ends work).

For example in figure 2.1, three different types of shifts are shown, assuming that the working day is divided into 15 periods ($P = \{1, \ldots, 15\}$). The first one (shift 1), for example, starts at the period 1, has its first coffee break at period 3, then the lunch break starting at period 7, the second coffee break starting at period 11 and finally it ends at period 13. Each shift, in figure 2.1, has two coffee breaks and one lunch break.

In general the duration of these breaks could vary. For example, it is possible to have three different types of break (the *pre-lunch* (b1), the *lunch*(b2) and *post-lunch* break(b3)) and in general b2 is longer than the other two. The set of the all feasible working schedules is $Q = \{1, \ldots, q\}$.

Due to the fact that the call center is multiskill, then there are other two important sets that should be defined: $N = \{1, \ldots, n\}$ and $T = \{1, \ldots, t\}$ that are the call types and the agent types set respectively.

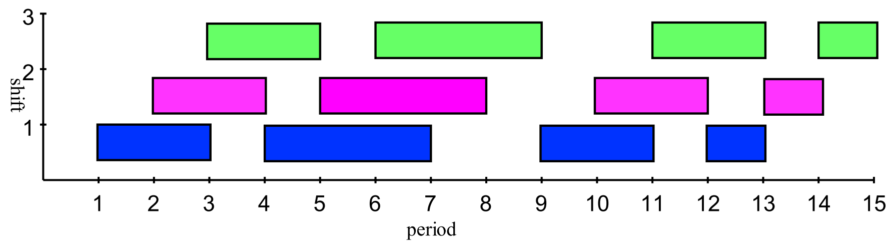

Figure 2.1: An example of three different shifts

The two sets P and Q are related to each other through a block-diagonal matrix $\widetilde{A} = A_1 \ldots A_t$ and the number of blocks is equal to the number of agent types. A generic block:

$$A_k = \begin{pmatrix} a_{11} & . & a_{1q} \\ . & . & . \\ a_{p1} & . & a_{pq} \end{pmatrix}$$

is a binary matrix in which an element a_{ij} is equal to 1 if i is a working period for the shift j, 0 otherwise.

Definition 2.2 (Working period)**.** *A period i is a working period for the shift j if and only if this shift covers this period.*

In figure 2.1, the period 2 is a working period for the shifts 1 and 2 but not for the shift 3. In fact an agent, assigned to shift 3, starts working at period 3.

Later on, $\widetilde{A}$(number) indicates the matrix $\widetilde{A}$ with number columns.

In a multiskill call center, as already said, the most relevant thing is that each agent belongs to a specific *agent group* characterized by a specific *skill set.*

Definition 2.3 (Skill set)**.** *A skill set S_i of an agent type/group i is defined by the list of call types that the agent type i can handle.*

It means that skill set $S_i, \forall i \in T$ can be seen as a subset of the call types set N: $S_i \subseteq N$. A particular case occurs when $S_i \equiv N$, i.e. the agent type i is able to handle all the call types and he/she is a *generalist.*

Moreover $c = (c_{11}, \ldots, c_{1q}, \ldots, c_{t1}, \ldots, c_{tq})^t$ is the cost vector where c_{ij} is the cost of an agent type i assigned to the shift j. If all the shifts have the same length then the cost only depends on the particular agent type and it is modelled as a constant plus an increment of between five to twenty percent for each additional skill (ξ). Even if it is not always allowed in real life,for the final solution quality (as shown later), it is important to have shifts whose length varies and so the cost also depends on the particular shift. The cost vector c and the block-diagonal matrix $\widetilde{A}$ are input data for the scheduling problem.

The *scheduling vector* is $x = (x_{11}, \ldots, x_{1q}, \ldots, x_{t1}, \ldots, x_{tq})^t$ where x_{ij} is the number of agents of type i assigned to the shift j. Moreover the *staffing vector* is $y = (y_{11}, \dots, y_{1p}, \dots, y_{t1}, \dots, y_{tp})^t$ where y_{ij} is the number of agents of type i available in the period j in order to satisfy the service levels constraints. The staffing vector represents the set of intermediate auxiliary variables.

In a general framework, the constraints are imposed on the service levels. In this context, a service level is defined as the fraction of calls whose virtual queue time is no larger than a given constant (in general set to 20 seconds). The virtual queue time is the time a hypothetical customer with infinite patience must wait in queue before beginning service ((Avramidis, Chan, and L'Ecuyer 2006)). It is assumed that they could be expressed for each couple (period, call type), for each period (grouped by call types), for each call type (grouped by periods) and global (grouped by call types and periods). For example the service level for call type j in period p can be defined as:

$$g_{j,p}(y) = \frac{\text{expected \# calls arrived in period } p \text{ with virtual queue time} < \tau_{j,p}}{\text{expected \# calls arrived in period } p}$$

for some constant $\tau_{j,p}$. Similarly the other types of service levels can be also defined and τ_p, τ_j, and τ are the time limits for the aggregate service levels for period p, for call j and overall respectively.

Definition 2.4 (Load)**.** *If λ_{kp} is the arrival rate of the call type k in the period p and μ_{kp} is the service rate of the call type k in the period p, then the load can be defined as $\rho_{kp} = \lambda_{kp}/\mu_{kp}$ for the call type k in the period p.*

To formulate a mathematical program with a finite number of shift types, it is necessary to discretize time. For this thesis's purposes, it is convenient to partition the planning horizon into periods of some length $d > 0$, where period t corresponds to the time interval $[(t-1)d, td]$, and where d is selected so that all the relevant events that affect any agents work status occur exactly at the beginning of some period, i.e., at a time that is an integer multiple of d. These events are,for example,the start and end of shifts and the start and end of any breaks scheduled to occur within the shift. With the planning horizon fixed (e.g., 24 hours) and d decreasing, the number of periods increases, the number of possible shifts increases accordingly, and one may generally expect optimal schedules to exhibit smaller cost (Thompson 1995); this is achieved at the expense of needing to solve larger mathematical programs. In applications, the time discretization used in schedule planning depends primarily on the level of detail at which forecasts of future arrivals are available, because the forecasts are crucial inputs to the mathematical programs supporting scheduling decisions. Typically encountered values of d are 15 minutes to one hour. In this generical framework, it is assumed that a time discretization parameter d has been selected. In this way, time is measured in units of d, unless otherwise said; and, by convention, events occur at the beginning of the stated period.

2.2 Staffing and Scheduling mathematical formulation

In this section mathematical formulations for the staffing and scheduling problems are given. In order to understand, it is important to start formulating the *scheduling problem* (P1).

$$\boxed{\begin{array}{rl} \min & \sum_{i=1}^{t}\sum_{j=1}^{q} c_{i,j}x_{i,j} \\ \text{subject to} & \\ & \tilde{A}x \geq y \\ & g_{i,p}(y) \geq l_{i,p} \qquad \text{for all } i \in N, p \in P \\ & g_p(y) \geq l_p \qquad \text{for all } p \in P \\ & g_i(y) \geq l_i \qquad \text{for all} i \in N \\ & g(y) \geq l \\ & x \geq 0 \qquad \text{and integer} \end{array}} \tag{P1}$$

In (P1) the first group of constraints links the scheduling vector to the staffing one and checks about the schedule feasibility. The other types of constraints are on the service levels. In particular the last type of constraint is on the global service level.

Because these functions g(.) are unknown and too complicated to be estimated exactly, they are approximated via simplified queueing models (Ingolfsson et al. (2003)) or estimated by simulation (Ingolfsson et al. (2003), Atlason et al. (2004a), Cezįk and L'Ecuyer (2006)).

On the other hand the constraints on the service levels are non linear and, also for this reason, the problem (P1) is too complicated to be solved exactly. If the constraints about the schedule feasibility are relaxed, the problem (P1) becomes the so-called *staffing problem* (P2) in which it is assumed that any staffing is admissible. In this case, the cost vector is $c = (c_{1,1}, \ldots, c_{1,p}, \ldots, c_{t,1}, \ldots, c_{t,p})^t$ where c_{ij} is the cost of the agent type i in the period j. It is easy to understand that in the staffing problem any relationship with the scheduling one disappears and then the cost is only related to the agent type and the period in which he/she works.

$$\boxed{\begin{array}{rl} \min & \sum_{i=1}^{t}\sum_{j=1}^{p} c_{i,j}y_{i,j} \\ \text{subject to} & \\ & g_{i,p}(y) \geq l_{i,p} \qquad \text{for all } i \in N, p \in P \\ & g_p(y) \geq l_p \qquad \text{for all } p \in P \\ & g_i(y) \geq l_i \qquad \text{for all } i \in N \\ & g(y) \geq l \\ & y \geq 0 \qquad \text{and integer} \end{array}} \tag{P2}$$

Moreover if one period at a time is considered, the staffing problem becomes the so called *single period staffing problem* (P3) in which the cost vector becomes $c = (c_1, \ldots, c_t)^t$ where c_i is the cost of the agent type i and the decision variables become $y = (y_1, \ldots, y_t)^t$ where y_i is the number of agents of type i. In general it is assumed that the system is in steady state over the given period and the problem formulation becomes:

$$\boxed{\begin{array}{rll} \min & \sum_{i=1}^{t} c_i y_i & \\ \text{subject to} & & \\ & g_i(y) \geq l_i & \text{for all } i \in N \\ & g(y) \geq l & \\ & y \geq 0 & \text{and integer} \end{array}} \tag{P3}$$

The mathematical formulations, proposed so far, are taken from (Cezik and L'Ecuyer 2006).

On (P3), different types of algorithm were proposed. For example, in Cezik and L'Ecuyer (2006) it is solved using a cutting plane methodology, instead in Avramidis et al. (2006) a randomized search algorithm is used where the service level constraints for each call class are represented by an analytical approximation (*Loss-Delay (LD) Approximation*).

In fact, in the next sections of this thesis, some algorithms, to solve the scheduling problem for the multi-skill call center, are presented.

Chapter 3

Two Step Approach

Two Step Approach, in the following called TS, is a general methodology for solving the scheduling problem (P1) of a multiskill call center. In this section, its description is given and at the same time some of its weakness are shown and explained.

3.1 TS:a general methodology

TS is an approach that can be divided into two independent steps:firstly P *single-period staffing problems*(P3) are solved obtaining the staffing vector y^*; then y^* becomes the input data to find a schedule solving the following problem $(P4)$:

$$\boxed{\begin{array}{rl} \min & \sum_{i=1}^{t}\sum_{j=1}^{q} c_{i,j}x_{i,j} \\ \text{subject to} & \\ & Ax >= y^* \\ & x \geq 0 \qquad \text{and integer} \end{array}} \tag{P4}$$

But, even if it is a natural way to solve the problem (P1),in general, it is definitely suboptimal and the suboptimality gap can become significant (e.g.,Jennings et al. (1996)).

In particular, in Avramidis et al. (2007) is shown that the main reason of this suboptimality gap is due to the fact that staffing problems are solved, for each period, independently from each other and from the scheduling one and performing a steady state simulation.

3.2 The formulation of Bhulai et al.

A description of TS can be found in Bhulai et al. (2005).In fact they adopt this methodology for solving the scheduling problem. In particular, they modify the formulation P1 introducing a new type of decision variables that allow modelling agents transfers.

In the following the staffing requirement y for all agent types and for all periods is assumed given. The optimization problem, solved in the second step of this approach, is based on the definition of two sets for each agent type $i \in T$ (see also Avramidis et al. (2007)):

Definition 3.1 (Superset of a skill-set). *A set of the agent types whose skill set is a minimum strict superset of the skill-set of the agent type i is* $S_i^+ = \{j \in T : S_j \supset S_i \wedge \nexists m \in T : S_j \supset S_m \supset S_i\}$

Definition 3.2 (Subset of a skill-set). *A set of the agent types whose skill set is a maximum strict subset of the skill-set of the agent type i is* $S_i^- = \{j \in T : S_j \subset S_i \wedge \nexists m \in T : S_j \subset S_m \subset S_i\}$

Example 3.1. For a specific example with 2 call types $N = \{0,1\}$ and 2 agent types $T = \{0,1\}$ and skill sets $S_0 = \{0\}$(specialist agent) and $S_1 = \{0,1\}$(generalist agent): $S_0^+ = \{1\}$; $S_0^- = \emptyset$; $S_1^+ = \emptyset$; $S_1^- = \{0\}$.

The skill transfer can be easily modelled introducing the vector of new decision variables z, where $z_{p,l,m} \forall p \in P, \forall m \in T, \forall l \in S_m^+ \wedge l \in S_m^-$ is the number of the agents of type l that work as agents of type m in the period p. Then the mathematical formulation, proposed in Bhulai et al. (2005), can be seen as an extension of the well known set covering problem formulation ($P5$) (proposed by Dantzig (1954)) to the multi-skill environment:

$$\boxed{\begin{array}{rll} \min & \sum_{j=1}^{q} c_j x_j & \\ \text{subject to} & & \\ & \sum_{j=1}^{q} a_{pj} x_j \geq y_p & \text{for all } p \in P \\ & x \geq 0 \quad \text{and integer} & \end{array}} \tag{P5}$$

$$\boxed{\begin{array}{rll} \min & \sum_{i=1}^{t} \sum_{j=1}^{q} c_{ij} x_{ij} & \\ \text{subject to} & & \\ & \sum_{j=1}^{q} a_{ij} x_{ij} + \sum_{l \in S_i^+} z_{p,l,i} - \sum_{l \in S_i^-} z_{p,i,l} \geq y_{p,i} & \forall p \in P \wedge \forall i \in T \\ & x \geq 0, z \geq 0 \quad \text{and integer} & \end{array}}$$

(P6)

(P6) can be also written in the following way:

$$\boxed{\begin{array}{rl} \min & \mathbf{c}'\mathbf{x} \\ \text{s.t.} & \\ & Ax + Bz \geq \hat{y} \\ & x \geq 0, z \geq 0 \quad \text{and integer} \end{array}} \tag{P7}$$

The number of nonzero entries in the main constraints (i.e., other than nonnegativity) in (P7) is often used as a measure of problem size (Aykin 1996). For fixed number N of call classes, the worst case occurs when agent types exist with all possible skill combinations: each nonempty subset of $N = \{1, 2, \ldots, n\}$ corresponds to a unique agent type, so there are $T = 2^n - 1$ agent types. Assuming the requirements $y_{p.i}$ are positive for all $p \in P$ and $i \in T$, for an agent type i with skills $\{a_1, \ldots, a_k\}$:$|\mathcal{S}_i^-| = k$. This because all maximal subsets are obtained by removing exactly one of the k skills (they are $\{a_1, \ldots, a_k\} \setminus \{a_i\}$ for $i = 1, 2, \ldots, k$). Similarly, $|\mathcal{S}_i^+| = n - k$, because all minimal supersets are obtained by adding exactly one of the $n - k$ skills the agent does not have.

The number of decision variables

$$\boxed{z_{v,j,i} = \sum_{v=1}^{p}\sum_{i=1}^{t}(|\mathcal{S}_i^-|+|\mathcal{S}_i^+|) = p\sum_{k=1}^{n}\binom{n}{k}(k+n-k) = np\sum_{k=1}^{n}\binom{n}{k} = np(2^n - 1)}$$

To count the number of decision variables $x_{q,i}$, assumptions must be made on the structure of shifts.

In order to solve the staffing problem for each period of the call center, *steady state* simulations are performed to estimate the service levels (Chapter 1). In particular each *single period staffing problem* $P3$ is solved using steady-state simulations and assuming that a period does not affect the next ones. In real life, instead, the decisions made in a period affect the next ones and the simulation is transient then the output measure of performance is defined over a specific interval of time with a specific starting condition and a specific ending condition. Performing a transient simulation one is able to take in account that the decisions taken in a period affect the behavior in the next period.

3.3 The weakness of TS

If the presence of skill-transfer variables generally reduces the optimal cost in (P2) by adding flexibility, compared with the case where no downgrading is allowed, on the other hand, there could be a significant gap between the optimal solution of (P1) and the best solution found for the same problem by the TS.

The following simplified example, taken from Avramidis et al. (2007), illustrates this.

Example 3.2. Let $N = T = P = 3$, and $Q = 1$. The single type of shift covers the three periods. The skill sets are $S_1 = \{1, 2\}$, $S_2 = \{1, 3\}$, and $S_3 = \{2, 3\}$. All agents have the same cost. Suppose that the total arrival process is stationary Poisson with mean 100. This incoming load is equally distributed between call types $\{1, 2\}$ in period 1, $\{1, 3\}$ in period 2, $\{2, 3\}$ in period 3. Any agent can be downgraded to a specialist that can handle a single call type (that belongs to his/her skill set), in any period. In the presence of such specialists, an incoming call goes first to its corresponding specialist if there is one available, otherwise it goes to a generalist that can handle another call type as well. When an agent becomes available he/she serves the call that has waited the longest among those in the queue (if any). The service times are exponential with mean 1, there are no abandonments, and the SL constraints specify that 80% of all calls must be served within 20 seconds, in each time period, on average over an infinite number of days.

If it is assumed that the system operates in steady-state in period 1, then the optimal staffing for that period is 104 agents of type 1. Since all agents can serve all calls, there is an $M/M/s$ queue with $s = 104$, and the global SL is 83.4%, as can be computed by the Erlang-C formula (see Chapter 1). By symmetry, the optimal staffing solutions for the other periods are obviously the same: 104 agents of type 2 in period 2 and 104 agents of type 3 in period 3. Then, the TS gives a solution to (P2) with 104 agents of each type, for a total of 312 agents.

If, instead, (P1) is directly solved , assuming again (as an approximation) that the system is in steady-state in each of the three periods, it gives a feasible solution with 35 agents of type 1, 35 agents of type 2, and 34 agents of type 3, for a total of 104 agents. With this solution, during period 1, the agents of types 2 and 3 are downgraded to specialists who handle only call types 1 and 2, respectively, and the agents of type 1 act as generalists. A similar arrangement applies to the other periods.It is important to note that TS is not able to find this solution because solving a single period staffing problem in the first step, it does not consider that the choices taken in a previous period affect the ones of the next. For example in a period it only choices the agents whose skill set contains exactly the calls arriving in it (i.e. agents of type 1 in period 1); instead solving (P1) directly it is possible that in the period there are also other types of agents that can be used in other periods.

Example 3.3. Suppose now that three additional skill sets $S_4 = \{1\}$, $S_5 = \{2\}$, $S_6 = \{3\}$ are added to the previous example, and that these new specialists cost 6 each, whereas the agents with two skills cost 7. In this case it becomes attractive to use specialists to handle a large fraction of the load, because they are less expensive, and to keep a few generalists in each period to obtain a "resource sharing" effect. It turns out that an optimal staffing solution for period 1 is 2 generalists (type 1) and 52 specialists of each of the types 4 and 5. An analogous solution holds for each period. With these numbers, if downgrading is not possible, TS gives a solution with 6 generalists (2 of each type) and 156 specialists (52 of each type), for a total cost of 978. If downgrading is allowed, then TS finds the following much better solution: 2 agents of type 1 and 52 of each of the types 2 and 3, for a total cost of 742. The reader can easily verify that by appropriate downgrading in each period, this solution can cover the optimal staffing in each period. In fact,the skill transfer works in this way. In period 1: 52 agents of type 2 are downgraded to specialists of type 4 and 52 of type 3 to specialists of type 5. In period 2: 2 agents of type 1 are downgraded to agents of type 5, 52 of type 2 to type 6 and 50 of type 3 to type 5. In period 3: 2 agents of type 1 are downgraded to agents of type 4, 50 of type 2 to type 4 and 52 of type 3 to type 6.

If (P1) is directly solved with these additional skill sets, it gives the same solution as without them; i.e., 104 agents with two skills each, for a total cost of 728. This is again better than with TS, but the gap is much smaller than what we had with only three skill sets.

Example 3.4. Observe that in the previous example, if all the load was from a single call type, there would be a single agent type and the two-step approach would provide exactly the same solution as the optimal solution of (P1). The example illustrates a suboptimality gap due to a variation in the *type* of load.

Another potential source of suboptimality (this one can occur even in the case of a single call type) is the time variation of the total load from period to period. If there is only a global SL constraint over the entire day, then the optimal solution may allow a lower SL during one (or more) peak period(s) and recover an acceptable global SL by catching up in the other periods. To account for this, Bhulai et al. (2007) propose a heuristic based on the solution obtained by their basic two-step approach. Although this appeared to work well in their examples, the effectiveness of this heuristic for general problems is not clear.

Yet another (important) type of limitation that can significantly increase the total cost is the restriction on the set of available shifts. Suppose for example that there is a single call type, that the day has 10 periods, and that all shifts must cover 8 periods, with 7 periods of work and a single period of lunch break after 3 or 4 periods of work. Thus a shift can start in period 1, 2, or 3, and there are six shift types in total. Suppose we need 100 agents available in each period. For this we clearly need 200 agents, each one working for 7 periods, for a total of 1400 agent-periods. If there were no constraints on the duration and shape of shifts, on the other hand, then 1000 agent-periods would suffice.

Example 3.5. This is another simple example that is used in order to show the main weakness of TS. In the following: $N = \{0, 1\}$; $T = \{0, 1, 2\}$; $P = \{0, 1, 2\}$; $Q = \{0\}$ and the skill sets are: $S_0 = \{0, 1\}$; $S_1 = \{0, 2\}$; $S_2 = \{1, 2\}$. Moreover $a_{ij} = 1 \forall i \in P \wedge j \in Q$. The arrival rates are shown in 3.1. The service rates are all equal to 1.

$\lambda =$

	period		
	0	1	2
call type 0	15	0	15
call type 1	15	15	0
call type 2	0	15	15

Table 3.1: call volumes matrix (call type/ period)

All the service times are exponential. The AWT is 20 seconds and $l_p = 0.5 \forall p \in P, l_k = 0.8 \forall k \in N$ and $l = 0.8$. The router policy is based on the agent's preference with FIFO queues. In this case the costs are computed considering the following formula:

Equation 3.1. *Agent Cost*

$$c_{ij} = 1 + (\eta_i - 1)\xi \quad \forall i \in T, j \in Q$$

where η_i represents the number of skills of the agent type i (that is the cardinality of the set S_i where $i \in T$). Assuming that the agent costs are the same, ξ is equal to 1 and η_i is equal to 2 $\forall i \in T$, the costs c_{ij} are equal to 2 $\quad \forall i \in T \wedge j \in Q$. In the following, it is assumed that a "good" scheduling solution (x^*) is already given: $x^*_{00} = 16; x^*_{10} = 17; x^*_{20} = 4$. This solution is experimentally obtained starting from a solution that schedules 15 agents of type 0 and 1 and 0 of type 2 and adding an agent at time (without considering its type because the cost is the same) while the service levels are not reached.

The staffing solution (y^*) associated to this scheduling is: $y^*_{0j} = 16 \ \forall j \in P$; $y^*_{1j} = 17 \ \forall j \in P$ and $y^*_{2j} = 4 \ \forall j \in P$.

Considering TS, instead,the sets S_i^+ and $S_i^- \ \forall i \in T$ are all empty. This means that there is not skill transfer. In the following it is assumed that the P *single-period staffing solutions* of the first stage are obtained using a randomized search algorithm (Avramidis et al. (2006)) because comparing to the one obtained using a cutting plane algorithm (Cezik and L'Ecuyer (2006)) it gives us a better solution: $y_{00} = 28; y_{01} = 6; y_{02} = 14; y_{10} = 5; y_{11} = 6; y_{12} = 19; y_{20} = 4; y_{21} = 25$ and $y_{22} = 4$. Considering the period 0, the greater part of the agents is related to the type 0 because in this period there are only call type 0 and 1 and so the agent type 0 in period 0 constitutes the generalist, in regard to the others that are the specialists (because the skill to handle the calls of type 2 in not used). The same considerations can be extended to the other two periods and agent types. This is because to solve the P single staffing problems, steady state simulations are performed considering that a decision taken in a period doesn't affect the next ones. Obviously, the second stage of TS must adapt the final scheduling solution to the staffing one that is given in input as right sides of the mathematical formulation $P7$. In this particular case in order to satisfy the input data, the problem assigns the maximum number of agents of each type that is required (as it is possible to note considering the scheduling solution: $x_{00} = 28$; $x_{10} = 19$; $x_{20} = 25$). The related staffing solution is:$y_{0j} = 28 \ \forall j \in P$; $y_{1j} = 19 \ \forall j \in P$ and $y_{2j} = 25 \ \forall j \in P$.

Examining it, one can note that there are a lot of useless agents. In fact, supposing to satisfy before the periods 0 and 2, after scheduling the 28 agents of type 0 and the 19 of type 1, it is possible to cover the load in the periods 0 and 2. But these are also sufficient to cover the load of the period 1. In fact in this period $28 - 6 = 22$ and $19 - 6 = 13$ agents of type 0 and 1 are not necessary. Then due to the fact that $S_0 = \{0, 1\}$ and $S_1 = \{0, 2\}$, at the end there are $22 + 13 = 35$ agents that are able to handle to the call types 1 and 2 when only 25 are required in this period. This means that the assignment of agents of type 2 is not necessary (the same argument can be extended to the other cases). These particular considerations are not taken in account by TS because in the formulation $P7$ these type of transfers are not considered. The total number of agents required by this solution is equal to 72 and the final cost is equal to 144. Considering, instead the scheduling solution x^* and the period 0,we have 16 agents of type 0 that are able to handle the call types that arrive in this period. Obviously they are not sufficient because there are 15 calls of type 0 and 15 of type 1 and the lower bound on the number of agents is equal to 30 (remember that the service rate is 1). In fact they are helped by the agents of type 1(who are able to handle the call type 0) and 2 (who are able to handle the call type 1). In this case, there is an implicit transfer of agents that TS is not able to do. In fact,to do this particular agents transfer it is necessary to have a complete vision during the shifts-composition phase . Moreover the final solution x^* assigns less agents of type 2 because the ones of type 0 and 1 are able to handle alone all the call types. The percentage of cost reduction compared with TS is about 48.61%.

Chapter 4

Daily scheduling with fixed shift length: some difficulties

In this section, some difficulties, related to the definition of a set of possible agent shifts, are presented. Considering, in fact, a first possible set of the constraints, it's possible to show the overstaffing of the final solution. In the following all the steps, considered to avoid the overstaffing, are shown and analyzed. Finally, a lower bound on the number of shifts is obtained in order to reduce the overstaffing of the final solution and so its cost. This lower bound is found for the single and multi-skill case.

4.1 Shift constraints

The set of constraints on the structure of shifts is described. Whenever possible, numerical values are not indicated to maintain generality. Each shift must have three distinct breaks, which must occur in the following order: break 1 (pre-lunch), lunch break, and break 3 (post-lunch). Breaks 1 and 3 last l_1 and l_3, respectively; the lunch break lasts l_2. Each shift has length l, including all breaks. Break 1 must occur within a time window $[w_{1,\min},\ w_{1,\max}]$ after the shift start time, i.e., it must occur within the time window $[s + w_{1,\min},\ s + w_{1,\max}]$, where s is the shift start time. The lunch break must occur within an absolute time window. Break 3 must be within a time window $[w_{3,\min},\ w_{3,\max}]$ after the end of the lunch break, i.e., it must occur within the time window $[t_2 + l_2 + w_{3,\min},\ t_2 + l_2 + w_{3,\max}]$, where t_2 is the lunch break start time. Note the difference between lunch break constraints, defined in absolute time, and break 1 and 3 constraints, defined relative to shift start and lunch end, respectively.

Shifts can be represented by the 4-tuple (s, t_1, t_2, t_3) denoting the start period, break 1 start period, lunch break start period, and break 3 start period, respectively. The set of possible shifts is

$$\mathcal{Q} = \{(s, t_1, t_2, t_3) : s \in \mathcal{S},\ t_1 - s \in \mathcal{D}_1,\ t_2 \in \mathcal{P}_2,\ t_3 - (t_2 + l_2) \in \mathcal{D}_3,\ t_3 < s + l\}, \tag{4.1}$$

where: $\mathcal{S}$ is the set of possible shift start periods (without loss of generality, take $\min(\mathcal{S}) = 1$ and define $s_{\max} = \max(\mathcal{S})$); $\mathcal{D}_1 := \{w_{1,\min}, w_{1,\min} + 1, \dots, w_{1,\max}\}$ is the set of possible delays between the shift start and break 1 start; $\mathcal{P}_2$ is the set of periods in which a lunch break may start; and $\mathcal{D}_3 := \{w_{3,\min}, w_{3,\min} + 1, \dots, w_{3,\max}\}$ is the set of possible delays between the lunch break end and the break 3 start.

For any possible shift, the set of periods where break 1 may start is $\mathcal{P}_1 = \mathcal{S} + \mathcal{D}_1$, where $A + B$ denotes the set $\{a + b : a \in A,\ b \in B\}$. Clearly, $\mathcal{P}_1 = \{t_{1,\min},\ t_{1,\min} + 1,\ \dots,\ t_{1,\max}\}$, where $t_{1,\min}1 + w_{1,\min}$ and $t_{1,\max} = s_{\max} + w_{1,\max}$. Similarly, the set of periods where break 3 may start is $\mathcal{P}_3 = \mathcal{P}_2 + \{l_2\} + \mathcal{D}_3 = \{t_{3,\min},\ t_{3,\min}+1,\ \dots,\ t_{3,\max}\}$ where $t_{3,\min} = t_{2,\min} + l_2 + w_{3,\min}$ and $t_{3,\max} = t_{2,\max} + l_2 + w_{3,\max}$. Because of its validity for any possible shift, $\mathcal{P}_1$ is a *global window* for the break-1 start time; and similarly for $\mathcal{P}_3$.

Example 4.1. The call center opens at 8 h (hours) and closes at 17 h. All shifts are exactly 7h 30 m (minutes) long (shift end occurs 7h 30 m after shift start); this suggests that the latest shift start that needs to be considered is 9h 30. Breaks 1 and 3 last 15 minutes; the lunch break lasts 30 minutes. The window for break 1 is [1h 30 m, 2h 15m] after the shift start. The window for the lunch break is between 12h and 13h 30. The window for break 3 is [1h 30 m, 2h 15m] after the end of the lunch break. Shift starts and lunch starts can occur only on the hour or the half hour; that is, possible shift starts are 8h, 8h30, 9h and 9h30, and possible lunch starts are 12h, 12h30, and 13h. Moreover: $T = 1, 2$; $N = 1, 2$; $S_1 = \{1\}$ and $S_2 = \{1, 2\}$.

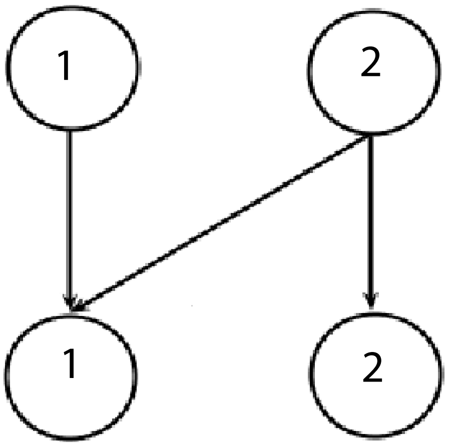

Figure 4.1: N-Design example

As shown in figure 4.1, for the particular configuration, in the following, it will be called as *N-Design Example.* This is a particular example, that usually is presented in literature, because of the agent type 1 is the specialist instead the type 2 is the generalist able to handle both types of call.

Each call has a patience time with an Exponential distribution of mean 20 ($\nu = 20$) for each period and the service rate is equal to 8. For the moment we are assuming that all the shifts have the same length (7h30min) and so also the cost only depends on the agent type (c=$(1, 1.2)$). On the other hand we have also that the load of the call type 1 is 20 and the load of the call type 2 is equal to 5.With time discretization $d = 15$ minutes, we have: $P = 36$; $l = 30$; $l_1 = l_3 = 1$; $l_2 = 2$; $\mathcal{S} = \{1, 3, 5, 7\}$; $\mathcal{D}_1 = \mathcal{D}_3 = \{6, 7, 8\}$; P_1=$\{7, \dots, 15\}$; P_3=$\{25, \dots, 31\}$; $\mathcal{P}_2 = \{17, 19, 21\}$ and $s_{max} = 4$. The number of possible shifts is $Q = 108 - 3 = 105$. To see this, note that without the shift-length constraint $t_3 < s + l$ in (4.1), we would have $Q = 4 \times 3 \times 3 \times 3 = 108$ possible shifts; of these, $(1, t_1, 21, 31)$ for $t_1 \in \{7, 8, 9\}$ exceed the maximum shift length, which leaves $Q = 108 - 3 = 105$ shifts.

4.2 A lower bound on the number of shifts

It is assumed, firstly, a single call class and a single type of agent. In the following, the problem (P5) is simplified to the problem (P4). $|P| = p$ is the total number of periods in the day and for simplicity, it is assumed that the cost of all shifts is 1. A simple lower bound on the optimal cost in (P4), i.e., the minimum number of shifts in a feasible schedule (considering that the maximum shift length is l), is detected.

Proposition 4.1. *The lower bound on the number of shifts in the single-skill formulation (P4) and in the multi skill one (P5) is equal to* $\tilde{c} = \max_{1 \le \bar{p} \le p-l} \tilde{c}(\bar{p})$ $\forall \bar{p} \in P$ *(with* $\bar{p} + l \le p$*).*

Proof: Fix a period $\bar{p} \le l$. To cover the constraints of periods 1 to $\bar{p}$, one needs at least $\max(y_1, y_2, \dots, y_{\bar{p}})$ shifts starting at or before period $\bar{p}$ (and otherwise unspecified). The constraints for periods $\bar{p}+l$ to p must be "covered" by shifts separate from (i.e., in addition to) the first set, because the shifts in the first set start at or before the beginning of period $\bar{p}$ and have maximum length l, so they end at (the beginning of) period $\bar{p} + l$. This implies at least $\tilde{c}(\bar{p}) = \max(y_1, y_2, \dots, y_{\bar{p}}) + \max(y_{\bar{p}+l}, y_{\bar{p}+l+1}, \dots, y_p)$ shifts are needed. Considering all periods $\bar{p}$ (with $\bar{p} + l \le p$) in the same way, at least $\tilde{c} = \max_{1 \le \bar{p} \le p-l} \tilde{c}(\bar{p})$ shifts are necessary.

It is possible to extend the same considerations to the multi-skill case (but one must be more careful because of possible skill substitution (decision variables $z_\bullet$)), finding a lower bound on the number of shifts for the formulation P5. Suppose agent type i has a skill set that is *maximal*, i.e. $\mathcal{S}_i^+ = \emptyset$. Then there exist no skill substitution variables $z_\bullet$ with positive sign in the left side in the constraints in (P5). The earlier argument, applied to period $\bar{p}$, implies we need at least $\tilde{c}_i(\bar{p}) = \max(y_{1,i}, y_{2,i}, \dots, y_{\bar{p},i}) + \max(y_{\bar{p}+l,i}, y_{\bar{p}+l+1,i}, \dots, y_{p,i})$ type-i shifts. Considering all periods $\bar{p}$ in the same way, at least $\tilde{c}_i = \max_{1 \le \bar{p} \le p-l} \tilde{c}_i(\bar{p})$ shifts of agent type i are necessary. A lower bound can obtained for the *total* number of shifts, i.e., regardless of agent type. At least $\tilde{c} = \max_{1 \le \bar{p} \le p-l} \tilde{c}(\bar{p})$ shifts of any agent type are required, where now :

$$\tilde{c}(\bar{p}) = \max\left(\sum_{i=1}^{I} y_{1,i}, \sum_{i=1}^{I} y_{2,i}, \dots, \sum_{i=1}^{I} y_{\bar{p},i}\right) + \max\left(\sum_{i=1}^{I} y_{\bar{p}+l,i}, \sum_{i=1}^{I} y_{\bar{p}+l+1,i}, \dots, \sum_{i=1}^{I} y_{p,i}\right)$$

In the next section,for the multi-skill environment,an example is presented showing that increasing and allowing more shifts flexibility the final cost is reduced. In this example, some shifts of different length are heuristically added in order to avoid the overstaffing of the solution.

One can easy see that relaxing the assumption of fixed shift length, allowing shifts of arbitrary length and assuming agent type i is maximal, ie $S_i^+ = \emptyset$, a lower bound on the number of shifts of type i is $\max\{y_{1,i}, y_{2,i}, \dots, y_{p,i}\}$. In fact there exist no skill-transfer variables with positive sign associated with the constraints for each $y_{p,i} \forall p \in P$. Thus, to cover the constraints of period $\bar{p}$, one needs at least $y_{\bar{p},i}$ shifts of type i. Varying $\bar{p}$ over $1, 2, \dots, p$, the stated bound is obtained.

4.3 A heuristic approach to avoid the overstaffing

As shown in the previous section, it is not impossible to meet the shifts constraints imposed so far, without having an overstaffing on the final solution. In fact, in this last section of this chapter, a heuristic method is presented to avoid overstaffing with the final solution. In particular all the different steps, that build a final staffing solution able to meet better the service level constraints, are described. In the following, the N-Design example will be considered. As already affirmed, to avoid overstaffing, it is important to increase the *shift types*. A *shift type* is characterized by its own length. It means that having shifts with different length defines a set of shift types.

Increasing the shift types

In order to avoid the overstaffing of the final solution,the incident matrix $\tilde{A}(105)$, containing only shifts whose length is equal to 30 periods, will be changed into $\tilde{A}(267)$ adding shifts whose length can be equal to 36, 35 ,. . . ,31 periods, increasing the start frequency. In fact, for example,the shift whose length l is equal to 36 starts at $8:00$ a.m., the ones with l equals to 35 at $8:15$ a.m. and so on until the shift length l equals to 31 starting at $9:15$ a.m..These new shifts can define an interaction between the start periods and the end ones allowing to reduce the overstaffing and so the cost of the final solution (see figure 4.2). In this new solution there is a shift type whose length is equal to $9:00$ a.m. and, in real life, perhaps, it could be not so attractive for an agent.But now the agents cost depends on a particular shift type too and it is directly proportional to its length. In fact,defining $c_{i,l}$ as the cost of an agent of type i assigned to the shift whose length is equal to l, the new cost vector is found considering this general formula: $c_{i,l} = l * c_{i,30}/30$. This means that the cost is not only related to the agent type but also to the shift one.

To reduce the peaks that are present during the pauses, the breaks frequency can be increased and new types of shifts can be added (table 9.11). In this way the total number of shifts becomes 285 ($\tilde{A}(285)$)(figure 4.4).

Just to have a brief idea about the different steps performed in order to define a "good" shifts structure, in figures a comparison between the final solution with 105, 267 and 285 shift types respectively is shown.

In conclusion, to improve the quality of the final staffing solution, it is very important to define shifts that allow reducing the overstaffing meeting better

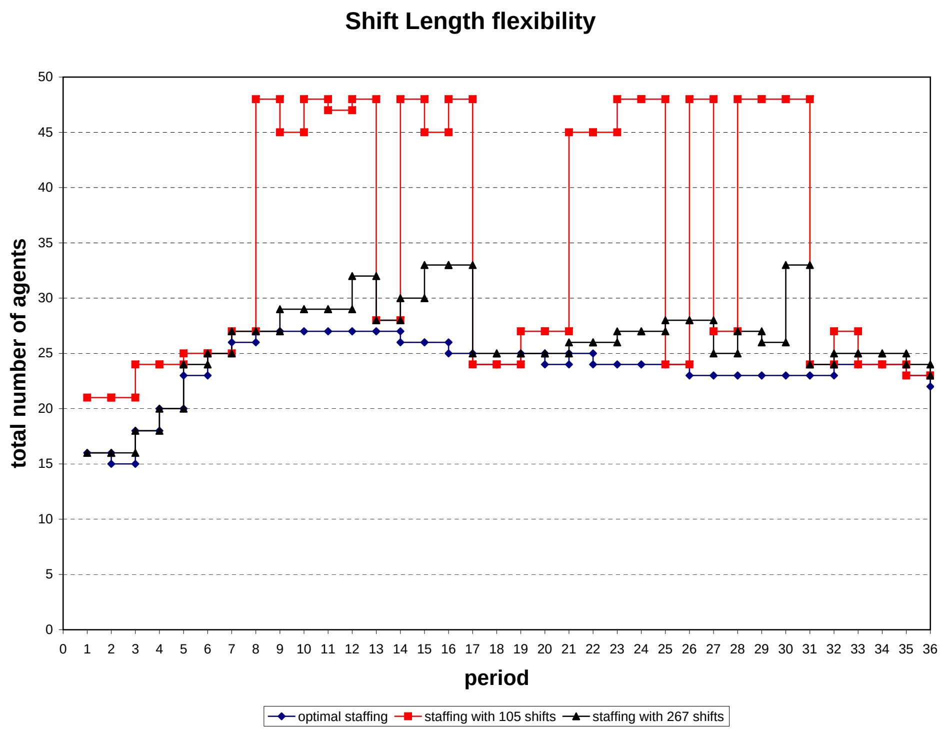

Figure 4.2: staffing solution:$\tilde{A}(105)$vs $\tilde{A}(267)$

Type	length	shift start	break1 start	lunch start	break3 start	final solution
1	7:30	8:00,8:30,9:00,9:30	9:30,11:30	12:00,12:30,13:00	14:00,15:30	10
2	7:45	9:15	10:45,11:15	12:00,12:30,13:00	14:00,15:30	2
3	8:00	9:00	10:30,11:00	12:00,12:30,13:00	14:00,15:30	3
4	8:15	8:45	10:15,10:45	12:00,12:30,13:00	14:00,15:30	2
5	8:30	8:30	10:00,10:30	12:00,12:30,13:00	14:30,16:00	2
6	9:00	8:00	11:00,12:00	13:00,13:30,14:00	14:45,16:15	10
7	6:30	10:00	11:30,12:00	13:00,13:30,14:00	14:15,15:45	2

Table 4.1: Shift types with $\tilde{A}(285)$

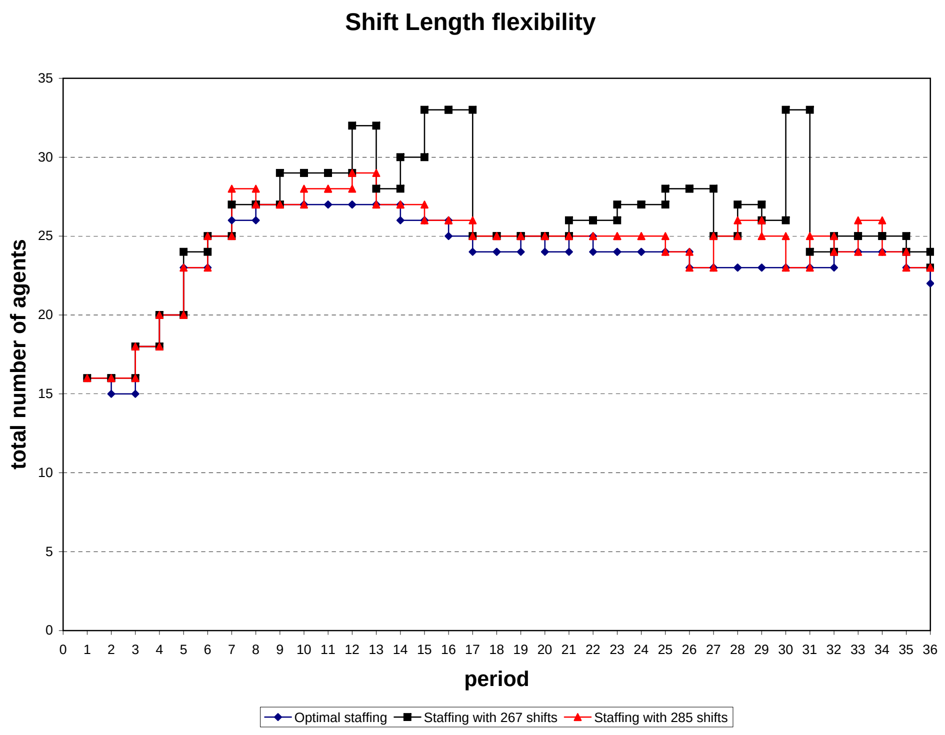

Figure 4.3: staffing solution:$\tilde{A}(267)$vs $\tilde{A}(285)$

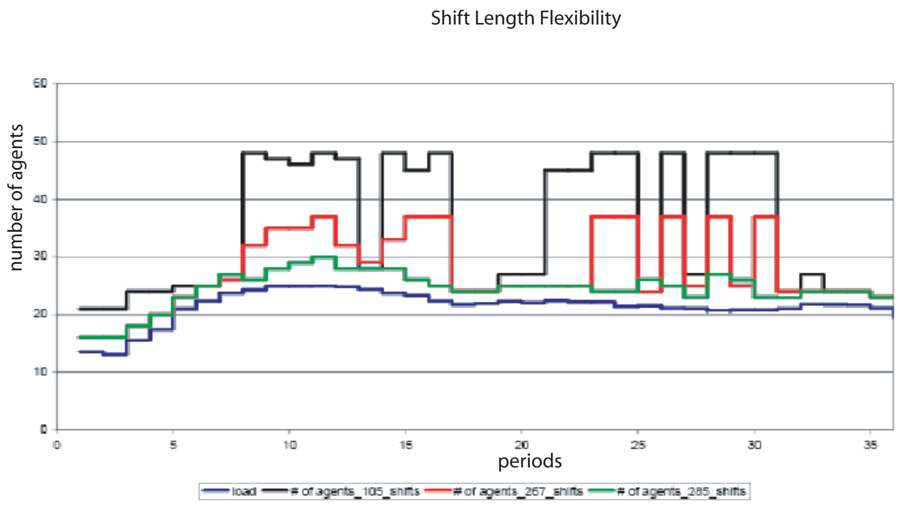

Figure 4.4: shift length flexibility

the service level constraints. Here it has been only proposed a heuristic method that iteratively adds shifts of different length. To improve the quality of the final solution does not only mean to meet better the SL constraints but also to reduce the final cost. In fact in period in which there is a peak in regard to the real load more agents than required are working.

Moreover it is important to remark that these final results on the shifts structure are completely general without considering a specific solution algorithm. The overstaffing on the final solution can be improved simply changing the shifts structure. It will be evident that the different solution methods meet in a different way the SL constraints. The higher the overstaffing, the higher the cost.

Chapter 5

A cutting plane algorithm

In this section, an algorithm, based on the cutting plane methodology, in order to solve the scheduling problem with service level requirements, is presented and described. It is based on the same approach, already proposed in (Cez̧ik and L'Ecuyer 2006), adapted for the scheduling problem and improved for obtaining a high quality solution. Moreover, in the following, only the service levels constraints are taken into account and not the ones on the abandonments (see Chapter 7).

The general idea of this approach relaxes, firstly, the service levels constraints and, at each iteration, solves a linear optimization problem (LP). If the solution satisfies all the service levels constraints then the algorithm ends. Otherwise some cuts are added in LP.

The optimization problem, solved at each iteration, can be integer (IP) or not on the base of the complexity of the problem. In fact when the problem has a small dimension (few periods, agent groups, call types and shifts), then at each iteration a linear integer problem can be solved and the simulation time (required to evaluate the service levels) is dominant on the total CPU time. But when the complexity of the problem grows up then the time required to solve the IP becomes relevant and so it is necessary to solve a LP(relaxing integrality)rounding the final solution. In both cases a local search is performed on the final solution in order to reduce its cost.

5.1 Cutting plane method

In the following, the cutting plane algorithm is called CP. Its scheme is presented in figure 5.1 where all the modules of the application are shown. Firstly, it is important to remark that the proposed algorithm is simulation-based optimization procedure. In fact, at each iteration of the first phase, at the end of the first phase, at each iteration of the local search implemented in the second phase and at the end of it, the simulator is used in order to check about the feasibility of the solution in regard to the service level constraints. The general idea is to replace the problem (P1) by a *sample* version of it, (SP1$_n$), and then replace the nonlinear SL constraints by a small set of linear constraints, in a way that the optimal solution of the resulting *relaxed* sample problem is close to that of (P1).

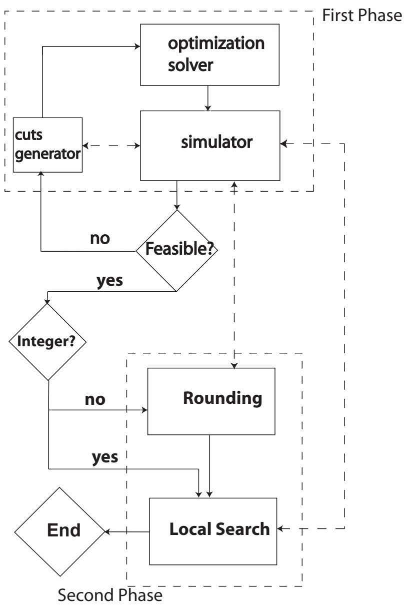

Figure 5.1: Cutting plane scheme

In figure 5.1, the general structure of the algorithm is shown: solving the sample version of the problem $P1$, through the optimization module, provides a staffing solution whose feasibility has to be checked using the simulator module. If it is, then the first stage can end otherwise the cut generator module finds new cuts to be added to the current sample formulation of the original scheduling problem.

In the second step of the procedure, if the final solution is already an integer one (later it is better explained) then the local search module can improve, if it is possible, the quality of the final solution trying to reduce the total cost. If it is not an integer solution then the rounding module, before calling the local search module, rounds the not integer components. This phase ends when it is not possible to improve the quality of the final solution any more.

In what follows, it is shown how the relaxation works, when applied directly to (P1). In fact it should be remarked that it is works in the same way when applied to the sample problem. A version of (P1), in which the SL constraints have been replaced by a small set of linear constraints that does not cut out the optimal solution, is considered. Let $\overline{y}$ be the optimal solution of this (current) relaxed problem. If $\overline{y}$ satisfies all SL constraints of (P1), then it is an optimal solution of (P1) and the procedure can end. The most important property required in the algorithm is that the g functions are concave in y.

Definition 5.1 (Concavity definition)**.** A function $g(y)$ is concave in the domain $[A, B]$, if for all points (a, b) such that $A \leq a$ and $b \leq B$, the following inequality is verified:

$$g(\delta a + (1 - \delta)b) \geq \delta g(a) + (1 - \delta)g(b), 0 \leq \delta \leq 1$$

Definition 5.2 (Subgradient definition (taken from Wikipedia))**.** If $g{:}U \to R$ is a real-valued convex function defined on a convex open set, a vector q, in this space, is called a subgradient at a point y_0 in U if for any y in U one has

$$g(y) - g(y_0) \geq q(y - y_0)$$

If the concavity property holds, then the service level function can be approximated by piecewise linear concave functions, which can be generated as described below.

A violated constraint of (P1) is considered, say $g(\overline{y}) < l$, suppose that g is concave in y for $y \geq \overline{y}$, and that $\overline{q}$ is a *subgradient* of g at $\overline{y}$. Then

$$g(y) \leq g(\overline{y}) + \overline{q}^{'}(y - \overline{y})$$

for all $y \geq \overline{y}$. One wants $g(y) \geq l$, so it is necessary to have

$$l \leq g(y) \leq g(\overline{y}) + \overline{q}^{'}(y - \overline{y}),$$

i.e.,

$$\overline{q}^{'} y \geq \overline{q}^{'} \overline{y} + l - g(\overline{y}). \tag{5.1}$$

Adding this linear *cut inequality* to the constraints removes $\overline{y}$ from the current set of feasible solutions of the relaxed problem without removing any feasible solution of (P1). This last point is shown in Cezik and L'Ecuyer (2006).

Since the g functions are complicated to be evaluated exactly, then they are replaced by a sample average over n independent days, using simulation. Let ω be the sequence of independent uniform random numbers that drives the simulation for those n days. It is assumed ,in the following, that, during the simulation, for each day and for different values of y, the same uniform random numbers are used for the same purpose. Such as:the same ω for all y is used. The *empirical SL* over these n simulated days is a function of the staffing y and of ω. In the following, $\hat{g}_{n,k,p}(y, \omega)$ denotes the empirical service levels for call type k in period p; $\hat{g}_{n,p}(y, \omega)$ the empirical aggregated over period p; $\hat{g}_{n,k}(y, \omega)$ the empirical aggregated for call type k; and $\hat{g}_n(y, \omega)$ the empirical aggregated overall. For a *fixed* ω, these are all deterministic functions of y. Instead of solving directly (P1), a *sample-average approximation* (SP1$_n$), obtained by replacing the functions g in (P1) by their sample counterparts $\hat{g}$, is considered.

It is known that $\hat{g}_{n,k,p}(y)$ converges to $g_{k,p}(y)$ with probability 1 for each (k, p) and each y when $n \to \infty$. In this sense, (SP1$_n$) converges to (P1) when $n \to \infty$.

Let Y^* be the set of optimal solutions of (P1) and it is assumed that no SL constraint is satisfied exactly for these solutions. Let Y_n^* be the set of optimal solutions of (SP1$_n$). Then, the following theorem implies that for n large enough, an optimal solution to the sample problem is also optimal for the original problem.

Theorem 1. *With probability 1, there is an integer $N_0 < \infty$ such that for all $n \geq N_0$, $Y_n^* = Y^*$. Moreover, under mild assumptions on the arrival processes there are positive real numbers α and β such that for all n,*

$$P[Y_n^* = Y^*] \geq 1 - \alpha e^{-\beta n}.$$

(SP1$_n$)is solved by the cutting plane method replacing the functions g by their empirical counterparts. The major practical difficulty is to obtain the subgradients $\bar{q}$. In fact, the functions $\hat{g}$ in the empirical problem are not necessarily concave for finite n, even in the areas where the functions g of (P1) are concave.

5.2 How to initialize the algorithm

At the beginning of the algorithm, all the constraints on the service levels are relaxed. In this way the scheduling formulation maintains only these ones:

$$\sum_{j=1}^{q} a_{pj} x_{ij} + \sum_{l \in S_i^+} z_{p,l,i} - \sum_{l \in S_i^-} z_{p,i,l} \geq y_{p,i} \qquad \forall p \in P \wedge \forall i \in T \quad \text{(c1)}$$

These constraints are maintained in order to model the skill transfer. If one solves the (P1) problem under this condition, the optimal staffing solution is the null vector in which the g function are not concave at all. To avoid this problem, Cezik and L'Ecuyer (2006) add some types of constraints considering a max-flow problem for the particular period p. They impose that the skill supply of agents in p covers a fraction α_{kp} of the load ρ_{kp} of the call type k for all $k \in N$. The generic variable $w(i,j)$ represents the fraction of load of the call type i served by the agent type j. All these variables w are auxiliary and they allow linking the fraction of the load for all call types to the agent requirements (vector y) in a particular period p. Due to the fact there are more than a period, a max-flow network for each of them is considered (see figure 5.2 taken from (Cezik and L'Ecuyer 2006)) and the relative constraints are added. Then the y vectors of each period are related each other by the constraints in (c1).

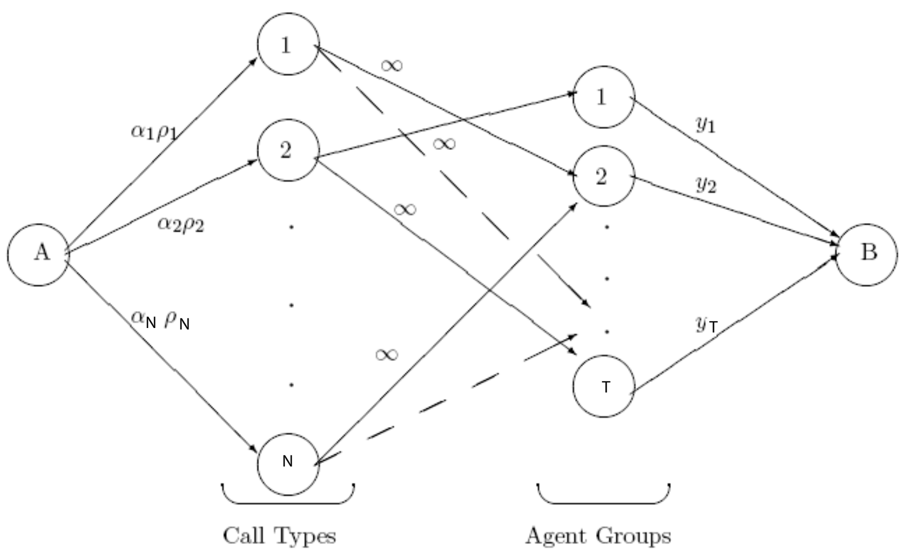

Figure 5.2: An example of the max flow network for a single period

In this way the problem (P8) is solved at the first iteration.

$$\begin{array}{rll} \min & \sum_{i=1}^{t}\sum_{j=1}^{q} c_{ij}x_{ij} & \\ \text{subject to} & & \\ & \sum_{j=1}^{q} a_{pj}x_{ij} + \sum_{l\in S_i^+} z_{p,l,i} - \sum_{l\in S_i^-} z_{p,i,l} \geq y_{p,i} & \forall p\in P \wedge \forall i \in T \\ & \sum_{j=1\dots t\wedge i\in S_j} w_{ij}^p \geq \alpha_{ip}\rho_{ip} \qquad \forall i\in N\wedge \forall p \in P & \\ & \sum_{i=1\dots n\wedge i\in S_j} w_{ij}^p \leq y_{jp} \qquad \forall j\in T\wedge \forall p \in P & \\ & x\geq 0, z\geq 0, y \geq 0 \qquad \text{and integer} & \\ & w \geq 0 & \end{array}$$

(P8)

The formulation (P8) allows finding an initial solution (as shown in Cezik and L'Ecuyer (2006)). The problem (P8) represents the initial model solved at the first iteration of the algorithm and in which, step by step, the linear cuts on service levels are added. It is easy to show that at a generical step u of the algorithm, the scheduling problem, in which the service levels functions ($g(y)$) are replaced by linear cuts ($G(y)$), becomes:

$$\begin{array}{rll} \min & \sum_{i=1}^{t}\sum_{j=1}^{q} c_{ij}x_{ij} & \\ \text{subject to} & & \\ & \sum_{j=1}^{q} a_{pj}x_{ij} + \sum_{l\in S_i^+} z_{p,l,i} - \sum_{l\in S_i^-} z_{p,i,l} \geq y_{p,i} & \forall p\in P \wedge \forall i \in T \\ & \sum_{j=1\dots t\wedge i\in S_j} w_{ij}^p \geq \alpha_{ip}\rho_{ip} \qquad \forall i\in N\wedge \forall p \in P & \\ & \sum_{i=1\dots n\wedge i\in S_j} w_{ij}^p \leq y_{jp} \qquad \forall j\in T\wedge \forall p \in P & \\ & G_{u,k,p}(y) \geq l_{k,p} \forall k\in N \wedge p \in P & \\ & G_{u,p}(y) \geq l_p \forall p \in P & \\ & G_{u,k}(y) \geq l_k \forall k \in N & \\ & G_u(y) \geq l & \\ & x\geq 0, z\geq 0, y \geq 0 \qquad \text{and integer} & \\ & w \geq 0 & \end{array}$$

(P9)

Example 5.1. *A max flow network*

In figure 5.3, in fact, it is shown a max-flow network when, considering a specific period p, $N = \{0, 1\}, T = \{0, 1\}$ and $S_0 = \{0, 1\}$ and $S_1 = \{1\}$.

5.3 A heuristic method to obtain subgradient and feasibility check

Solving (P9) a new staffing solution can be obtained.Then at each iteration of the algorithm, a check about the feasibility is needed.

A generic cut, related to a service level function g, is a linear constraint $G(y) \geq l$ (Cezik and L'Ecuyer (2006)). Considering a current staffing solution $\bar{y}$ and taking a violated constraint, $g(\bar{y}) < l$, for the sample version of the problem (P9), supposing that g is (jointly) concave in y for $y \geq \bar{y}$.

Then $G(\bar{y}) = \bar{q}^t y - \bar{q}^t\bar{y} + g(\bar{y})$ and so, one want $\bar{q}^t y - \bar{q}^t\bar{y} + g(\bar{y}) \geq l$ such as $\bar{q}^t y \geq \bar{q}^t\bar{y} + l - g(\bar{y})$; y is the staffing variables vector; $\bar{q}$ is a sub-gradient of g

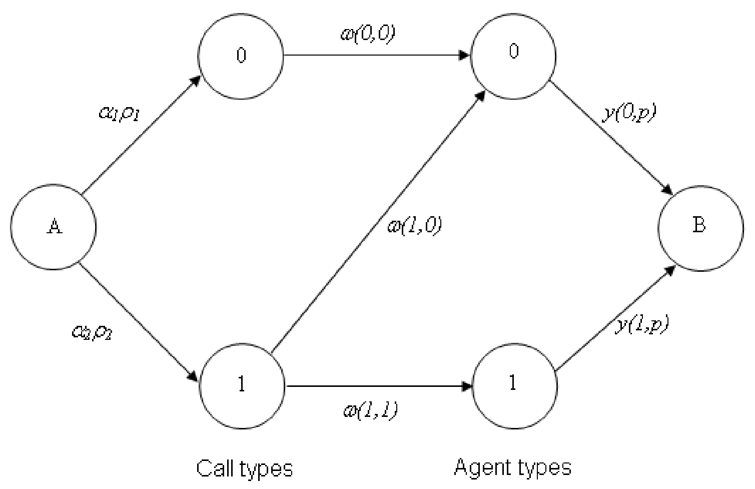

Figure 5.3: A max-flow network example

in the point $\bar{y}$; l is the target service level associated to the one represented by the function $g(y)$. A generic component j of the array $\bar{q}$ is obtained as

Formula 5.1. $\bar{q}_j = (g(\bar{y} + de_j) - g(\bar{y}))/d$

where $d \geq 0$ is an integer value and e_j is a vector such that the j-th component is 1 and the others are equal to 0. So $\bar{y} + de_j$ means to add d agents in the position j (related to a specific couple agent-period of the staffing vector). The computation of the new service level $g(\bar{y} + de_j)$ is done by the simulator. It is important to remark that the unitary vector e is chosen in a different way in regard to the particular function g. In fact:

1. considering a no-grouped service level, let be $\widetilde{c}$ and $\widetilde{p}$ the call and period indices of a violated service level respectively, the sub-gradient $\bar{q}$ has a number of non zero components equal to the number of agent types. In particular, the indices j, in the formula 5.1, are obtained fixing the period to $\widetilde{p}$ and varying the agents type;

2. considering the service level aggregated by period, then only $\widetilde{c}$ is found and the number of non zero components of $\bar{q}$ is equal to $|T| \times |P|$. In particular, the indices j, in the formula 5.1, are obtained varying the period and the agents type;

3. considering the gradient of a service level aggregated by call, then $\widetilde{p}$ is found and the number of non zero components of $\bar{q}$ is equal to $|T|$. In this case the indices j, in the formula 5.1, are obtained fixing the period to $\widetilde{p}$ and varying the agents type;

4. considering the gradient of the global service level, then the indices j, in the formula 5.1, are obtained varying the period and the agents type. In fact the number of non zero components of $\bar{q}$, in this last case, is $|T| \times |P|$.

The value d, i.e. the number of agents to be added, is set to 3 if the service level is less than 0.5, 2 if it is between 0.5 and 0.65, 1 otherwise(Cezik and L'Ecuyer (2006)).

In the cuts generation process more priority is given to the global service level. For this reason, the check on the service levels starts from the global one: if it is not satisfied, then the cut related to it is found and added. Otherwise the ones aggregated by period are considered and if at least one is not satisfied then all the cuts related to the violated service levels of this category are computed and added. If all of the service levels aggregated by period are satisfied then the ones aggregated by call are considered. Finally if all of the service levels aggregated by call are satisfied, then the global ones are checked.

5.4 An integer problem or a linear one?

As it is possible to see in the figure 5.1, adding the linear cuts at each iteration of the method could define a new integer linear problem too complicated to be solved, in particular when the complexity grows up. In the following, the complexity of the problem is related to the number of variables and constraints. In real life call center scenario, one has a lot of call types, agent types and periods and it implies a very complex framework. To have a complicated scenario implies very long CPU time to solve an IP and sometimes Cplex (the optimization tool used in this work) ends with a typical message of "out of memory".In fact,as well known, Cplex implements a Branch and Bound algorithm to solve the IP problems and growing up the complexity of the model, the branch and bound tree depth increases so much that it is not possible to explore it in a reasonable time. The message "out of memory" appears when the number of sub-problems to be explored has became considerable. To avoid this problem, instead to wait for a long time to have a solution and sometimes without result, a LP problem is solved in regard to an IP and then a rounding method is performed on the solution. It is evident that the computational time is consistent but less than the situation in which an IP is solved. In fact it is important to remark that the rounding procedure is called not only at the end of the method to give an integer output solution, but at each step of the algorithm, when the simulation module is called.

At each time a feasibility check on the current solution is required, the rounding method is called because the simulator needs an integer input solution in any case. It means that it is important to define a "good" approximation module in the general application in order to reduce noise in the final solution.

5.4.1 Rounding module

In order to recover an integer solution, a threshold τ between 0 and 1 is selected. The approximation module rounds up (to the next integer) the real numbers whose fractional part is larger than τ and it truncates (round down) the other ones. So it memorizes the cumulated amount of truncation and whenever it exceeds 1, it is reset to 0 and one agent of the currently considered type is added in the solution.

We use two rounding algorithms: during the cut generation process we fixe the value of τ in order to save time owing to the fact that the rounding is required at each iteration. After the cut generation phase, we round the solution in order to obtain an integer one. In this second rounding we dynamically adapt τ finding its optimal value in the interval $[0.1, 1.0]$.

The rounding module is called only if the solution has at least one non-integer component. It usually happens whenever the problem test is characterized by a complex scenario. For this reason, in the following, two different versions of the same algorithm are presented: $CP - IP$ and $CP - LP$. The former is run when at each step it is possible to solve an integer problem; the latter, instead, when the integer constraints are relaxed and a linear problem is solved.

5.5 Local search

After adding enough linear cuts, the final solution could be:

1. infeasible for (P1) (because of random noise, especially if n is small);
2. feasible but suboptimal for (P1) (because one of the cuts may have removed the optimal solution of (P1) from the feasible set of ($\mathrm{SP1}_n$)).

In order to improve the solution to ($\mathrm{SP1}_n$), a local search is performed around it, still using the same n and the same random numbers.

This local search has two phases.

In phase 1 (*remove phase*), the cost is hopefully reduced iteratively removing one shift at a time, until either none of the possibilities is feasible or a time limit is reached. For the CP-LP version, firstly the solution is round to integer by using a threshold τ as explained earlier. Starting with $\tau = 0.1$ and the value of τ is increased by 0.01 successively until a feasible solution is obtained.

Phase 2 ,(*switch phase*), attempts to reduce the cost by iteratively considering a *switch move* in which it tries to replace an agent/shift pair by another one with smaller cost. In particular: it chooses in a random way two couples (*removeAgent* rA, *removeShift* rS) and (*addAgent* aA,*addShift* aS). An agent of type rA is removed from the shift rS and an agent of type aA is added in the shift aS. Obviously the two couples are chosen in a way that the final cost is reduced and so such that $c_{rA,rS} > c_{aA,aS}$.This phase ends when a time limit is reached, or when a maximum number of consecutive moves without improvement is reached.

When at the end of the CP algorithm, the final solution is not feasible at all for the original scheduling problem (P1), before performing any type of moves (in the local search module), an *ADD phase* is performed starting adding agents with the smaller costs. This is performed until a feasible solution is obtained. In particular, the agent type that is added and the related shift is chosen in the following way:

1. if there is at least one service level grouped by call j that is not satisfied, then the agent type is the less expensive able to handle the calls whose type is j;
2. otherwise, if there are no service levels grouped by call that are violated, then the agent type less expensive but more busy (i.e. whose occupancy is maximum) is chosen.

On the other hand, the shift type is chosen in this way: it is the shift that covers more periods in which there is the service level more violated. In particular:

1. the calls whose service level is more violated are found;
2. the periods, with the maximum arrival rate of the calls, found in the previous step,are found;
3. between the shifts that cover the maximum number of these periods, the final shift, in which the agent is added, is chosen according to an uniform distribution.

5.6 CP:Some improvements

In this section of this chapter, some improvements are presented in regard to two phases of the CP algorithm.

The fist one is the rounding phase calling when a LP is solved at each step of the procedure instead of IP; the second one is the local search procedure calling at the end of the algorithm in order to improve the quality of the final solution decreasing its cost (when it is possible).

In the following, in order to refer this new version, the names CP_{new} and $CP - LP_{new}$ are used.

Moreover at the end of the section, it's showed a little adjustment to the algorithm for detecting more feasible solutions, increasing the targets of the service levels.

5.6.1 Rounding module: some improvements

As one can well note considering the description of the algorithm CP, one of the most critical aspect is in solving CP-LP when an approximation is used to obtain, from a not integer solution, an integer one.

This approximation is needed in two situations:

- during the cut generation process, after solving the LP problem, in order to check about the feasibility of the obtained solution;
- after the cut generation phase, in order to give an integer solution to the module of local search.

Moreover, an important parameter for the rounding module is τ.

Higher values of it (i.e. higher than 0.5) allow having less expensive solutions (that with more probability will be infeasible) because with more probability a no integer component will be approximated to the previous integer one. For example, if the value of τ is set to 0.7 then it means that all the components whose integer part less than 0.7 will be approximated to the previous integer value. On the contrary, lower values of τ (i.e. less than 0.5) allow having more expensive solutions but with that are able to satisfy more service levels constraints at the same time. It means that choosing the parameter τ in the right way is fundamental for having good performances of the rounding module.

On the other hand, as one can figure out, the rounding procedure performed in the first phase can not be more time consuming because it is run at each iteration of the CP algorithm. For this reason, it is run with a τ value fixed a-priori and chosen using the input file.

The rounding before the local search module,instead, is run only one time and at the same time it is more predominant on the final solution quality. For this reason, it is run with a variable value of τ.

In the first version of CP, the τ parameter linearly varies starting from a value of 0.5 and incremented by 0.01 until the integer solution remains feasible. But this procedure could be more inefficiency because it is experimentally observed that the optimal values of τ are the ones in the range $[0.6, 0.7]$. This last aspect implies a number of iterations of the rounding procedure that could pass from 10 to 20. In order to improve (in efficiency) the rounding procedure, a binary search is performed (that is faster) replacing the linear one.

Assuming, for example, to perform the binary search in the interval $[0, 0.1]$, the maximum number of iterations is 7.

In particular, the new scheme of the algorithm becomes the following one:

```
Algorithm Rounding
1.  double low = 0.0;
2.  double high = 1.0;
3.  double bestApprox = high;
4.  while ((high − low) > 0.01)
5.     do
6.        double mid = (low + high)/2;
7.        round solution with τ=mid
8.        Check feasibility
9.        if (!feasible(sl))
10.          then low = mid + 0.01;
11.          else
12.                high = mid − 0.01;
13.                bestApprox = mid;
14. round solution with τ= bestApprox
```

This change could give better solutions using less CPU time budget.

5.6.2 Local Search Procedure: some improvements

Another important aspect of CP is the local search procedure.

It has been noted that not always the final solution is feasible after a check using a long simulation ($n_* = 50000$ days). This happens because the local search is performed with fewer number of days n_1. In fact, in the original version of CP (presented and described in this chapter), the local search is performed using a value of n equal to the one used during the cutting plane phase to check the feasibility of the solution. When it ends, one can be sure that the solution is feasible for a simulation of n days. But considering that the final feasibility check is performed using a simulation of $n* = 50000$ days, the original local search cannot guarantee that this final solution is feasible for this last check too.

On the other hand, one can not perform the local search with long simulations because otherwise this procedure will become too much time consuming. For this last aspect, in the new version of CP, the local search starts from a parametric value of n_1 that is not necessarily chosen equal to the number of days simulated during the cut generation phase. This value is set to n_2 (an input value) and it is incremented by 50% at a time until the obtained solution, after the local search, is feasible for a simulation with $n_3 = max(n, 500)$ or if a time limit is reached. We have:

```
Algorithm Local Search
Input: scheduling_solution x,int n2, n
1.  boolean longFeasible=false;
2.  n1=n2;
3.  n3= max(500,n);
4.  repeat
5.        simulate(x,n1);
6.        if (!feasible())
7.           then addAgents(x,n1);
8.        removeAgents(x,n1);
9.        switchAgents(x,n1);
10.       simulate(x,n3);
11.       longFeasible=feasible();
12.       n1+=0.5*n1;
13.       if (time-limit-reached())
14.          then break;
15. until ((longFeasible))
```

In this algorithm:

`simulate(x,n1)` means that the solution x is simulated with a number of days n1;

`addAgents(x,n1)`, `removeAgents(x,n1)`, `switchAgents(x,n1)` means respectively `add/remove/switch` agents considering the solution x and the check for the feasibility is performed with a number of days equal to n1. After switching, it is sure that the solution is feasible for n1.

Moreover if the time limit is reached and the `longFeasible` variable is `false`, the method ends with a solution that is, in any case, feasible for a simulation with n1 days.

5.6.3 Increasing the target service levels

When the problem to be solved becomes too complex (more agent types, call types, period, shifts), the chance that the algorithm finds non feasible solutions increases due to the noise of simulation.

In a lot of cases, the infeasibility is not high and so it is acceptable. The call center managers could accept low infeasibility gaps to have lower cost solutions, considering that, due to the forecast uncertainty, in real cases, these violations on the service levels could never occur.

However, for having only solutions that respect all the service levels constraints, we apply a simple adjustment to the algorithm.

Before starting the algorithm, the service level targets are increased (they are the right and sides of the optimization problem $P9$). The entire algorithm is executed with the incremented targets, but, before performing the long simulation, they are decreased to the original ones in order to check its feasibility considering only the targets imposed by the call center manager.

In this way, the entire algorithm is run with tighter constraints on the service levels and it should preserve the final solution by the noise of the simulation.

Naturally if all these changes (on the rounding, on the local search and on the service level targets) guarantee better solutions in quality, on the other hand, they imply higher computational times.

Chapter 6

Randomized search(RS)

In this chapter a heuristic approach to solve the scheduling problem $P1$ for a multi-skill call center is proposed. It is a *randomized search*, in the following called RS (Randomized Search Algorithm), and it is a relatively straightforward adaptation of the one already proposed for solving the single period staffing problem in a multi-skill call center (Avramidis et al. (2006)).

The approach is divided in two stages. In the first one, some moves are applied to the current solution in order to improve it and its feasibility is checked using an *evaluator* of the service levels (Loss-delay (LD) approximation, section 6.1). In this step a Randomized Neighborhood Search is performed and it ends after a finite amount of work with an incumbent that is locally optimal. Also in the second stage, some moves are applied to the current solution in order to reduce the final cost but here the simulation is used as the service level constraints evaluator. This stage is needed whenever the LD approximation error in estimating the service levels may be considerable.

6.1 Loss-delay (LD) approximation

The loss-delay approximation (LD) is a service level evaluation method for each call class of the call center.It is used in the first stage of the randomized search in order to check about the solution feasibility. In fact this method evaluates the service levels for a single period, while we need evaluating all periods. For this reason, as said later, it will be used for evaluating the service levels period by period and a solution will be feasible if and only if there are no violated service levels (of all periods). In this section we shall describe, briefly, the LD approximation for a single period (Avramidis et al. (2006)).

It was already formulated in Avramidis et al. (2006) for solving a single period staffing problem and was inspired by the approximation of Koole and Talim (Koole and Talim (2000)). Respect to Koole and Talim (2000) in the LD approximation the queueing of calls that cannot be served immediately is introduced . The call classes flowing into each station [1] are classified into one of two types: if the station is the last one on the call's routing list, then the call is of *delay* type, otherwise it is of the *loss* type.

[1]A station is the set of all agents of the same type

For each station Avramidis et al. (2006) approximate the call-overflow rate of the loss type calls and the stationary delay distribution for the delay type calls.

In this section the key aspects of this technique are summarized (for details refer to Avramidis et al. (2006)).

Let's be:

- λ_j, $j \in N$ arrival rates;
- $\mu_{i,j}$, $i \in T$, $j \in N$ service rates;
- τ_j, $j \in N$ wait times;
- $\mathcal{R} := \{\mathcal{R}_j, j \in N\}$ the routing information . For each call class j ($j \in N$),$\mathcal{R}_j = \{R_j(1), R_j(2), ..., R_j(m_j)\}$ (where $\mathcal{R}_j \subseteq T$) is the set of stations able to handle it and m_j is the number of station able to handle call class j. When a call of class j arrives, it is assigned to an agent in the first station in the list $\mathcal{R}_j$ that has an available agent.
- $r(i, j)$ $i \in \mathcal{R}_j$ the rank of station i in the list $\mathcal{R}_j$ of calls type j, so that $R_j(r(i, j)) = i$;
- $p(i, j) := R_j(r(i, j) - 1)$, whenever $r(i, j) > 1$, the index of the station immediately preceding station i in the routing of calls of type j.

The approximated key objects are the rate of arrivals into station i of call type j: $\gamma_{i,j}$, for $i \in T$, $j \in \mathcal{S}_i$. If $r(i, j) = 1$, they are the arrival rates λ_j; if $r(i, j) > 1$ they are overflows from station $p(i, j)$ into station i. The approximate overflow rates satisfy the following system of nonlinear equations:

$$\lambda_{i,j} = \lambda_{p(i,j),j} B_{p(i,j)}, \ \ \texttt{if} \ \ r(i, j) > 1, j \in \mathcal{S}_i, i \in T$$

This system says that the arrival rate of call type j to station i has to be equal to the arrival rate of call type j to station $p(i, j)$ times the blocking probability at that station $B_{p,j}$.

Other computed probabilities are:

- the blocking probability B_i for stations i;
- the virtual queue time tail probability D_i for stations i;

The service level of class j is one minus the product of the fraction of such calls that arrive to the last station in the routing, $\gamma_{L_j,j}/\lambda_j$, times the virtual queue time tail probability for that station, D_{L_j}, as stated by the following relation:

$$(\alpha_j)_{j=1}^{n} = 1 - \tfrac{\gamma_{L_j,j} D_{L_j}}{\lambda_j}, j \in N$$

Avramidis et al. (2006) defined the crossed rounding as follows:

Definition 6.1. crossed routing occurs if and only if there exists call classes $j_1 \neq j_2$ and agent types $i_1 \neq i_2$ such that $r(i_1, j_1) < r(i_2, j_1)$ and $r(i_2, j_2) < r(i_1, j_2)$, that is, any call of type j_1 that flows into station $i - 2$ has overflowed from station i_1 and any call of type j_2 that flows into station i_1 has overflowed from station i_2 (the overflow can be direct of not).

If the routing is not crossed, then an ordering of the stations, such that call overflows occur only along increasing order, exists. When the routing is crossed, proceeding as above is impossible because (by definition) there exist stations i_1 and i_2 that mutually affect each others input flows. Avramidis et al. (2006) propose an iterative solution method inspired from Koole and Talim (2000).

Avramidis et al. (2006) formulates a sufficient condition for convergence of the algorithm. Of course, it should be clarified that it does not necessarily converge to a feasible or optimal solution.

6.2 The first phase

The evaluator, used in the first stage, is the LD-approximation. In order to follow the steps of this first phase, it is very useful this definition:

Definition 6.2. *LD- feasible solution*
A solution x is LD-feasible if all the periods are LD-feasible (the evaluator declares it as feasible for P3), otherwise it is LD-infeasible.

`Initialization.` Like for all heuristic methods, this phase matters a lot on the quality of the final solution. For this reason a lot of different methods to initialize the search have been considered. In all the methods, the initial staffing, for each period (i.e. the number of agent of each type that have to be in the period), is calculated as an LD-feasible solution of the staffing problem for that period (Avramidis et al. (2006)). In the following, a brief description of these different methods is presented in order to give an idea about their own performance for the scheduling problem. In particular the initialization methods differ only for the way to obtain an initial scheduling solution from the P staffing ones as one can easily see in the following.

Best shift method. All the agents of a particular type that must be in a period are assigned to the less expensive shift that covers this period. If the shifts have the same cost, then all the agents of a type that must be in a period are assigned to the first shift that covers it. In particular, supposing that the current period is p and the agent type is i, it is possible to define the set:

$$\text{Shift}_i^p = \{j | i \in Q \wedge a_{p,j} = 1\}$$

If the shifts have different lengths, the cost of the agent i varies with the length of the shift:

$$j^* = \text{argmin}_{j \in Shift_i^p} \; c_{i,j}$$
$$x_{i,j^*} += y_{p,i}$$

Example 6.1. In the following example it is assumed that there are 2 call types $N = \{0, 1\}$ and 2 agent types $T = \{0, 1\}$. Moreover the two skill sets are: $S_0 = \{0\}$ and $S_1 = \{0, 1\}$. There are three shifts ($Q = \{0, 1, 2\}$) and three periods ($P = \{0, 1, 2\}$) in total such that:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

For this reason the costs vary (table 6.1). Moreover, in order to show how this

	shift 0	**shift 1**	**shift 2**
agent type 0	3	2	2
agent type 1	6	5	5

Table 6.1: cost matrix (agent type/ shift)

	period 0	**period 1**	**period 2**
agent type 0	10	5	6
agent type 1	7	8	9

Table 6.2: staffing matrix (agent type/ period)

method works, it is also assumed that the staffing requirements are the ones indicated in table 6.2. Firstly it is considered the case $p = 0$, then the less expensive shift covering it is 1 for both agent types and so it means: $x_{01} = 10$ and $x_{11} = 7$.

Then $p = 1$ and again the less expensive shift covering it is 1 and so:$x_{01} = 15$ and $x_{11} = 15$.

Finally $p = 2$ and the less expensive shift is 2: $x_{02} = 6$ and $x_{12} = 9$.

The starting scheduling solution is reported in table 6.3.

	shift 0	**shift 1**	**shift 2**
agent type 0	0	15	6
agent type 1	0	15	9

Table 6.3: scheduling matrix (agent type/ shift)

Random Method. All the agents of a type that must be in a period are assigned to a shift that is chosen randomly among the ones that cover it. It is possible to describe this method, supposing, again, that the current period is p and the agent type is i:

$$x_{i,j} += y_{p,j}$$
$$\texttt{where } j \in \text{Shift}_i^p$$

Round-robin method. All the agents of a type that must be in a period are distributed on all shifts that cover it. In other words, supposing to be in a particular period p, the method slides the list of shifts that cover it in a circular way and inserts an agent at a time until the number of the agents that has to stay in p is equal to zero. In figure 6.1, this method is summarize supposing that the current period is p and the agent type is i.

Example 6.2. Considering the example 6.1, in the following it is shown how the *uniform* method works. It is important to remark that, in order to simplify and to allow the reader to follow all the steps, it is shown the method for a particular couple (period p, agent type i): $p = 0$ and $i = 0$. [2]

[2]In the following, we are assuming that the method *head*, applied to a list, returns its first element, instead *tail* the last one.

```
Algorithm Round-robin
1.  $UL_i^p = \{j \in Q \wedge A_{p,j} = 1\}$;
2.  $j$ =head($UL_i^p$);
3.  while ($staffing_{p,i} \neq 0$)
4.      do
5.          $x_{i,j} += 1$;
6.          $staffing_{p,i} --$;
7.          if (j=$tail(UL_i^p)$)
8.              then $j$ =head($UL_i^p$);
9.              else
10.                 $j$ =next($UL_i^p$);
```

Figure 6.1: Round-robin method scheme

Firstly, as reported in figure 6.1, the following set is built: $UL_0^0 = \{0, 1\}$. Then the shift j is chosen as $j = head(UL_0^0) = 0$ and 1 agent of type 0 is assigned to the shift 0: $x_{00} = 1$; the corresponding staffing is decreased by 1 becoming $y_{00} = 9$. In the next step, $j = 1$ and 1 agent of type 0 is assigned to the shift 1: $x_{01} = 1$; the staffing becomes $y_{00} = 8$. This is done until there are 0 agents to be assigned and at the end of the procedure, the reader can simply verify that: $x_{00} = 5$ and $x_{01} = 5$. It means that the 0 agents required in the period 0 are uniformly distributed in the shifts of type 0 and 1.

Extending the same considerations (mutatis mutandis), one can obtain the starting scheduling matrix for this particular example applying the Round-robin method.

Complex method: the staffing matrix rows are obtained solving the staffing problems for each period. Let's suppose to be in the row p (the staffing vector of the period p called v). For each agent type a, all the agents of this type, already scheduled in the period p, are computed (summing all the agents of type a that are in the shifts that cover the period p). If this number is less than v_a, the method adds agents in order to satisfy the staffing requirements. These agents are added in a shift that covers the period p and at the same time has the maximum coverage.

Definition 6.3. *The shift coverage*

Let be p the current period and j a shift index. We can define as coverage of j the number of periods p' (from $p + 1 \ldots |P|$) that it covers.

In figure 6.2, this method is summarize supposing that current period is p and agent type is i.

Example 6.3. Again the general scenario presented in the example 6.1 is considered and applied to this particular method.

As in the example 6.2, this procedure is shown for a particular couple (period p, agent type i): $p = 0$ and $i = 0$.

Firstly, the coverage of the shift 0 and the shift 1 (the shift 2 does not cover the period p) are defined: $C_0^0 = 2$ and $C_1^0 = 1$. The shift whose coverage is maximum is 0. It means that, following the general steps presented above: $sum_0^0 = 0$ and $x_{00} = 10$.

Mutatis mutandis, one can easily find the complete scheduling matrix.

```
Algorithm Complex
1.  $C_j^p =\mid \{p^{'} \mid A_{p',j} = 1 \wedge A_{p,j} = 1, p^{'} = p+1,\dots,p\} \mid;$
2.  $\mathtt{sum}_i^p = \sum_{j=1,|A_{p,j}=1}^{|Q|} x_{i,j};$
3.  if $(\mathtt{sum}_i^p < staffing_{p,i})$
4.      then $\jmath^* = \text{argmax}_{j\in Q} C_j^p;$
5.          $x_{i*,j*} += staffing_{p,i} - \mathtt{sum}_i^p;$
```

Figure 6.2: Complex method scheme

Fifth method: a copy of the staffing matrix $staffing$ is defined and it is called $staffing1$. For each period p and agent type a, the following procedure is performed: until $staffing1_{p,a}$ is greater than 0, an agent of type a is assigned to a particular shift that covers the period p, chosen in the following way:

1. the excess staffing for each period p' and for each agent type j is found as:

$$upLoad_{p',j} = currentStaffing_{p',j} - staffing_{p',j};$$

   $currentStaffing$ is the staffing matrix containing the agents scheduled until this iteration.

2. for each shift s that covers the period p, the surplus of agents of type a, $supLoad_{a,s}$, in all the periods covered by s, is found summing $upLoad_{p',a}$ for each period p' covered by s:

$$supLoad_{a,s} = \sum_{p'\in P|A_{p',s}=1} upLoad_{p',a} \ \forall s \in Q|A_{p,s} = 1$$

3. the shift s^* is chosen such that: $s^* = \text{argmin}_{s=1,\dots,Q \wedge A_{p,s}==1} supLoad_{a,s};$

4. $scheduling_{s*,a}$ is increased by one and $staffing1_{p,a}$ is decreased by one.

Example 6.4. Considering the example 6.1, it is possible to see how this method works. The particular couple (period p, agent type i), chosen to show the steps, is: $p = 0$ and $i = 0$.

Firstly the matrix $UpLoad$ is computed and is reported in table 6.4, being $currentStaffing$ a matrix whose all elements are equal to 0.

	period 0	**period 1**	**period 2**
agent type 0	-10	-5	-6
agent type 1	-7	-8	-9

Table 6.4: UpLoad matrix (agent type/ period)

Then $SupLoad_{00}$ and $SupLoad_{01}$ can be obtain being shift 0 and shift 1 the only one that cover the period 0: $SupLoad_{00} = -21$ and $SupLoad_{01} = -15$. Then $s* = 0$ and 1 agent of type 0 is assigned to the shift $s* = 0$: $x_{00} = 1$. To this scheduling solution corresponds the following staffing one, reporting the only components not equal to 0: $currentStaffing_{00} = 1$, $currentStaffing_{01} = 1$ and $currentStaffing_{02} = 1$.

	period 0	**period 1**	**period 2**
agent type 0	-9	-4	-5
agent type 1	-7	-8	-9

Table 6.5: UpLoad matrix (agent type/ period)

The next iteration:

The matrix *UpLoad* in table 6.5 is obtained considering that the current staffing is the one reported above. Then $SupLoad_{00}$ and $SupLoad_{01}$ can be obtain: $SupLoad_{00} = -18$ and $SupLoad_{01} = -13$. Then $s* = 0$ and 1 agent of type 0 is assigned to the shift $s* = 0$: $x_{00} = 2$. To this scheduling solution corresponds the following staffing one, reporting the only components not equal to 0: $currentStaffing_{00} = 2$, $currentStaffing_{01} = 2$ and $currentStaffing_{02} = 2$.

In order to obtain the scheduling solution, one can continue applying the same considerations to the other cases. It is evident that this method is applied only in the case in which the elements of *UpLoad* are > 0 because of it tries to associate agents to shifts in a way that the load is low.

In the section related to experiments, one can easily see that the *round-robin method* gives the best solution.

`Moves.` After finding an initial solution, some moves are applied to the current one in order to explore its neighborhood in a way that it is possible to improve it. Two different types of moves are performed.

Remove move: applied to the current solution , if it is possible, it returns another solution whose cost is reduced by removing agents (figure 6.3). The most important parameter of this procedure is r: the number of agents that it tries to reduce at a particular iteration. Initially set to the maximum number of agents in each shift $max_{i,j}x_{i,j}$, during the procedure, it is decreased by one if the current value doesn't produce any LD-feasible solution. Only when it becomes zero, the method ends. At this point, it is important to remark that a scheduling solution is LD-feasible if all the periods are LD-feasible. Considering the scheduling matrix x, in a generic iteration, the candidates to be reduced are only the elements $x_{i,j} \geq r$. Among all of these candidates, only a sub-set is chosen according to some parameters in order to reduce the computational time. The candidates in the sub-set are randomly processed and between the LD-feasible solutions, obtained reducing $x_{i,j}$ by r, the one that gives the best cost reduction is chosen. The stop criterion of this procedure is verified either when the time limit is reached or when the value of r becomes zero. In a variant of the algorithm, the sub-set of candidates is not sorted in a random way but by increasing cost of the pair <agent-shift>. In fact ,in this way, the most expensive agents are always taken. But, comparing to the first approach, it does not give any improvement on the cost of the final solution.

Switch move: it tries to reduce the solution cost performing some agents moves, where the moves are related to the pairs <agent-shift>. Each iteration of this procedure is characterized by: a pivot (a_-, s_-) and a parameter $\Phi \geq 0$. The neighborhood, in a generic iteration, is made up by all the solutions obtained removing Φ agents of type a_- by the shift s_- and adding Φ agents of a type a_+ in a shift s_+. In other words, the method performs the following operations: $x_{a_-,s_-} = x_{a_-,s_-} - \Phi$ and $x_{a_+,s_+} = x_{a_+,s_+} + \Phi$. The number of agents ($\Phi$), to

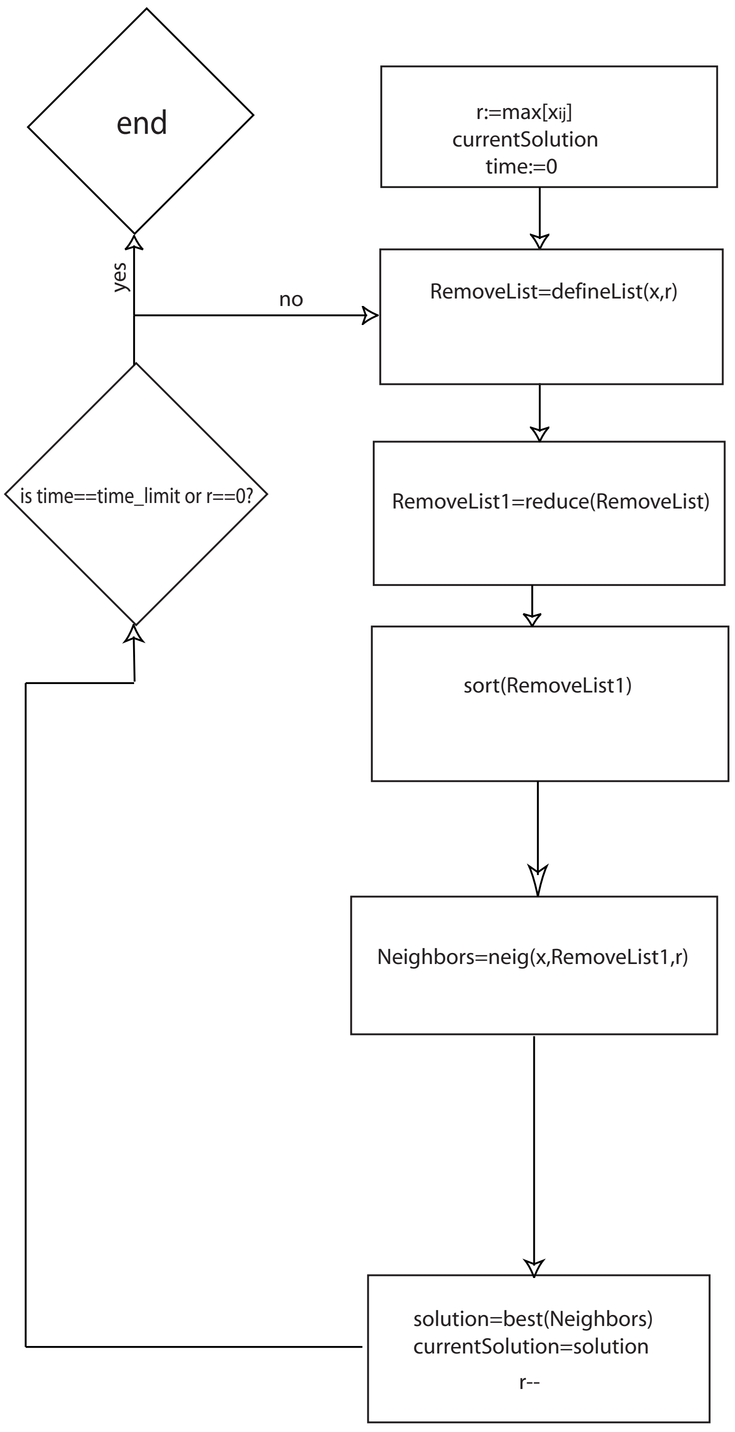
end
r:=max[xij]
currentSolution
time:=0
yes
no
RemoveList=defineList(x,r)
RemoveList1=reduce(RemoveList)
is time==time_limit or r==0?
sort(RemoveList1)
Neighbors=neig(x,RemoveList1,r)
solution=best(Neighbors)
currentSolution=solution
r--

Figure 6.3: Remove move

be initially switched, is randomly chosen among the variables $x_{a,s}$. The new pair (a_+, s_+) is chosen such that $c(a_+, s_+) < c(a_-, s_-)$ in order to obtain a cost reduction. A pivot (a_-, s_-) and a size Φ are considered infeasible if the neighborhood, characterized by the relative move, is empty (*infeasibility1*) or it is made up by LD-infeasible solutions (*infeasibility2*). In a generic switch move iteration, the candidate pivots, belonging to the so called *Pivots List*, are chosen as the elements (a, s) of the scheduling matrix x such that $x_{a,s} \geq \Phi$. From all of these candidates, the couples (a', s') are removed such that the move $(\Phi^{'}, (a', s'))$ with $\Phi' \leq \Phi$ is not feasible, obtaining a new *Pivots List1*. The pivot (a_-, s_-) is chosen in a random way from *Pivots List1*. The move $(\Phi, (a_-, s_-))$ can be in the case of *infeasibility1* if its neighborhood is empty and so it is tagged as a infeasible move. Otherwise, the *neighborhoodList* is defined as the list of the neighbors of (a_-, s_-) and sorted in a random way. Among the elements in *neighborhoodList*, the one, that generate the LD-feasible solution with the maximum cost reduction, is chosen. If there are no LD-feasible neighbors, the case of *infeasibility2* occurs and so the move is tagged as not-feasible.

The parameter Φ is reduced by 1 after each switch moves and the procedure ends when it becomes 0. Figure 6.4 describes the algorithm to compute the neighborhood of a pivot (a_-, s_-).

```
Algorithm NEIGHBORHOOD
1.  $\Phi \leftarrow \texttt{random}(\{x_{a,s}, a = 1, \ldots, |T|, s = 1, \ldots, |Q|\})$;
2.  $Pivots = \{(a, s) \mid x_{a,s} \geq \Phi \wedge a = 1, \ldots, |T|, s = 1, \ldots, |Q|\}$;
3.  $Pivots1 = Pivots \setminus \{a^{'}, s^{'}) \in Pivots \mid ((a^{'}, s^{'}), \Phi')$ is-not-feasible, $\Phi^{'} \leq \Phi\}$;
4.  $(a_-, s_-) \leftarrow \texttt{random}(Pivots1)$;
5.  $Neib_{(a_-,s_-)} = \{(a, s) \mid c(a, s) < c(a_-, s_-)\}$;
```

Figure 6.4: Computing the set of cost-reducing neighbors of candidate (a_-, s_-): $Neib_{(a_-,s_-)}$

6.3 The second phase

This phase is necessary because of the LD-approximation, in stage 1, can introduce some errors and so the solution may be infeasible or far from the optimal one(Avramidis et al. (2006)).

This is relevant, in particular, in the scheduling problem, in which the LD-feasibility check is applied independently period by period.

The service levels evaluator in this stage is the simulator. At the end of stage 1, two different cases can occur:

1. the solution is SIM-feasible (see Avramidis et al. (2006) for more details about);

2. the solution is not SIM-feasible: it has to be modified in order to become SIM- feasible, calling *SIMAdd procedure.*

6.3.1 SimAdd procedure

This procedure adds agents until the SIM-feasibility is reached. The agent type a^* that has to be added and the shift s^* in which it is added are chosen on the base of the unsatisfied service levels.

Firstly the pair (a^*, s^*) is found on the base of the period grouped service levels. The call type m^* with the maximum service level violation, obtained summing on all periods the differences among the target service level and the reached one, is selected. Then an agent of type a^* is added in a shift s^*, maximizing the fraction of the agent-type's time spent serving call m^*, until all the no global SL are satisfied. Defining:

1. $\widetilde{s}_{\bar{p},m,a}$ as the service rate of the call type m for the agent type a in the period $\bar{p}$ computed by the simulator;

2. $s_{\bar{p},m,a}$ as the correspondent service rate given by the call center parameters;

3. $\alpha_{\bar{p},m}$ as the estimate of the simulator for the service level of the call type m in the period $\bar{p}$;

4. $l_{\bar{p},m}$ as the target service level of the call type m in the period $\bar{p}$;

5. $Sum(\bar{p}, a) = \sum_{m|m \in Sa} \widetilde{s}_{\bar{p},m,a}/s_{\bar{p},m,a}$

it is possible to have (a^*, s^*)=argmax$_{a\in T, s\in Q} \sum_{\bar{p}|A_{\bar{p}s}=1} \widetilde{s}_{\bar{p},m^*,a}/(s_{\bar{p},m^*,a} * Sum(\bar{p}, a))$.

If all the service levels aggregated per period are satisfied, the global service level is considered in the following manner: iteratively an agent type a^* in a shift s^* is added such that (a^*, s^*) maximizes the occupancy-to-cost ratio over all the shifts, i.e. (a^*, s^*) =argmax$_{a\in T, s\in Q} \sum_{\bar{p}|A_{\bar{p}s}=1} Sum(\bar{p}, a)/(schedule_{a,j} * c(a, j))$

The SimAdd procedure is executed until all the service levels are not satisfied.

6.3.2 SimRemove procedure

This procedure is applied both the solution is already Sim-feasible and it is not (and so *SimAdd* procedure is called). Calling the *SimRemove* procedure, the solution cost is tried to be reduced removing an agent at each time.

In this case the agent type to be removed and the shift from which it is removed are chosen on the base of the service levels and the agents occupancy.

In a generic iteration the candidates list is given by the couples (a, s) such that $x_{a,s} \geq 0$. For each candidate, the *slack* is computed in the following manner:

$$\widetilde{\chi}_{a,s} = \sum_{\bar{p}|A_{\bar{p},s}=1} \sum_m \widetilde{w}_m * (\alpha_{\bar{p},m} - l_{\bar{p},m})$$

where

$$w_m = \sum_{\bar{p}|A_{\bar{p},s}=1} \frac{\widetilde{s}_{\bar{p},m,a}}{s_{\bar{p},m,a}}$$

and so

$$\widetilde{w}_m = \frac{w_m}{\sum_m w_m}$$

then

$$\chi_{(a,s)} = \widetilde{\chi}_{(a,s)} * c(a,s)$$

The candidates list π is sorted by decreasing values of $\chi_{(a,s)}$ and for each candidate $\pi(i) \equiv (a,s)$ an agent from $x_{a,s}$ is removed. If the obtained solution is SIM-feasible, $\pi(i)$ is removed from the candidate list if $x_{a,s} = 0$ and the slacks are again computed; otherwise $\pi(i)$ is removed and the removed agent is reinserted and then the procedure can consider the next candidate in the sorted list. The iteration ends if and only if the candidate list is empty.

Figure 6.5 illustrates the entire RS algorithm.

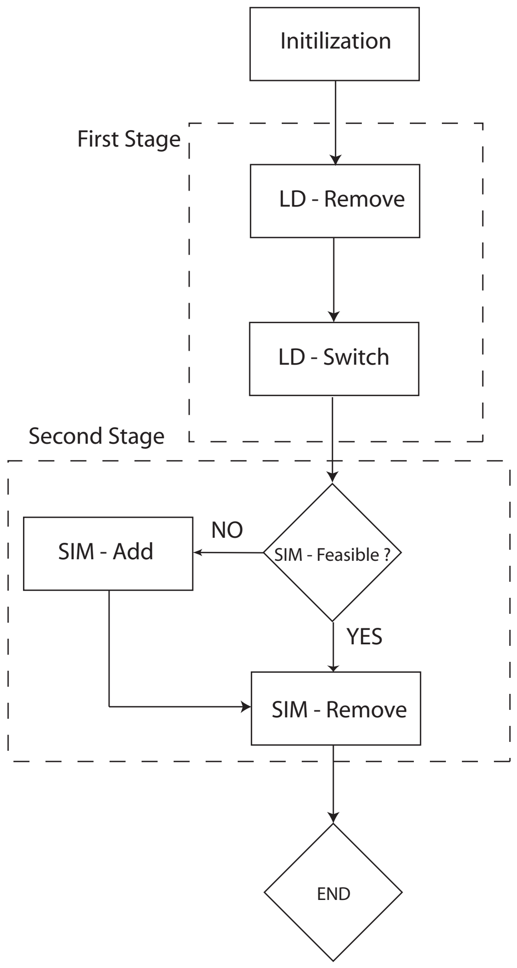

Figure 6.5: Remove move

Chapter 7

Optimization with constraints on the abandonments

In this chapter, a version of the problem P1 is proposed in order to take in account the abandonments.

The main goal is to show that other types of constraints than those on the service levels can be handled as well.

In fact in real call centers, there are usually constraints on the service levels, on the abandonments, on the server's utilization for each server and so on. In the following, we show an example of this extension to handle the abandonments and the service levels constraints at the same time.

7.1 Definition for abandonments in a call center

Usually, in fact, a call center manager is very interested in controlling the number of abandonments. Having a high number of abandonments is a clear measure of the quality of services that the center offers. It means that something is wrong in managing the center and so something should be changed. It is evident that controlling the service levels, trying to reach the targets, it is a first method to improve the quality of services. But it should be remarked that it is not the only one.

It is very difficult to model the customer's abandonments due to the variability of the factors that influence them. For example some callers decide to abandon as soon they are put in queue if all the lines are busy; some of them, instead, decide to abandon only after a time is passed.

An interesting factor is the psychology of abandonment and the ways to influence it. There is, in fact, a set of primitives, in operational queueing models, related to the behavior of the customers who have to be served. In literature, particular attention is given to patience and abandonment. In any case, one has to decide (among the other things) how the so-called *patience time* has to be represented.

The customer decide to abandon only on the base of time spent in queue.But why do people abandon at entering the queue? They have no patience at all?

A possible way used to explain abandonments is based on the following criterion: ”people make their decision on the time that they expect that they still have to wait” (Koole 2005). In fact, some customers abandon at entering the queue because they expect their waiting time surpasses the time that they accept to wait.

In order to figure out, it is important to remark that usually a customer abandons because of he/she does not know a priori his/her waiting time. Remaining in queue teaches to a customer about his/her own waiting time.((Koole 2005)).

One can mathematically model the *impatience function* through out the cumulative distribution function F that represents the distribution function of the impatience time. Its corresponding *hazard rate function*, assuming that F has a density $f = F'$, is

$$h(t) = \frac{f(t)}{(1-F(t))}, t \geq 0.$$

In fact $h(t)dt$ can be considered like the probability that a customer, who has already waited for t units of time, will abandon within the next dt units, i.e. during $(t, t + dt]$. For this reason , the hazard rate $h(t)$ represents a natural dynamic depiction of (im)patience, as it evolves while waiting. It is very important to remark that, usually, the call center managers want to provide a sort of priority to a/some particular class of customers.

In general, high priority customers could be either more or less impatient, and the hazard rate can be monotone or not, depending on the call center.

In this case, the impatience of priority customers lies strictly below that of regular customers, so that the high priority customers emerge as (stochastically) more patient (less impatient) than regular customers. Second, the impatience functions of both types of customers are not monotone (see figure 7.1 from Gans et al. (2003b)).

To be able to find a ”good” model to represent abandonments is one of the most important goal for managing a call center.

The first model of impatience was developed by Palm [123] in the 1940s estimating the inconvenience of waiting. Introducing an *inconvenience* function $I(t), t \geq 0$, he defined its derivative as *irritation* of the customer: $dI(t) = c * t^l * dt, t \geq 0$ and it is proportional to the hazard function and the patience distribution is Weibull.

Zohar et al. (2002) consider the following linear relationship between the abandonment fraction and expected delay, considering for patience that is exponentially distributed with parameter θ:

$$P_{Abandonment} = \theta * E[wait]$$

In order to verify this relationship, there is an empirical support shown in figure 7.2 taken from Brown et al. (2005a). In fact, in this figure, it is shown, using some experiments, as the relationship between the fraction abandoning and the average waiting time (in second) is supported by a linear trend.

Through this plot, Zohar et al. (2002) tried to represent the effect of customer learning. In fact, the customers should learn using the experience and consequently adjust their expectations concerning delay. Only in this way, they

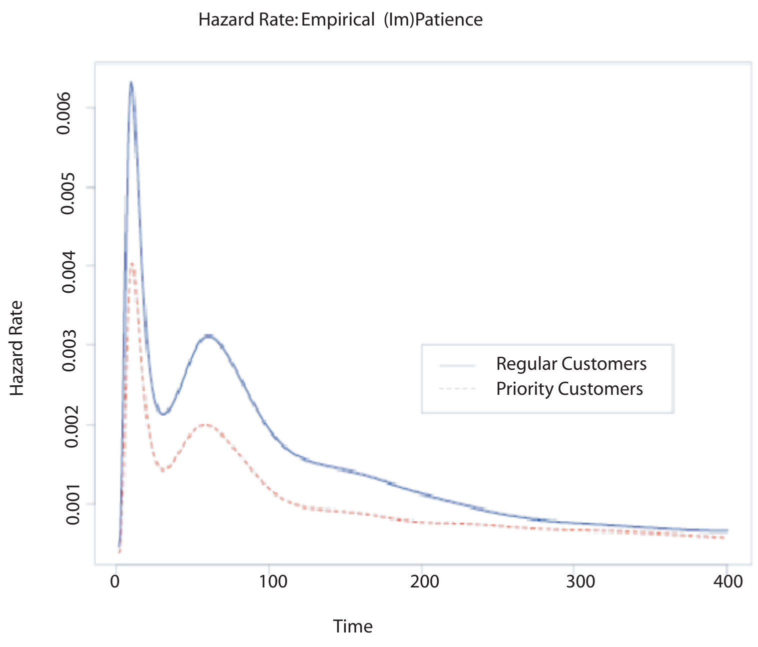

Figure 7.1: Impatience Functions of Regular and Priority Customers

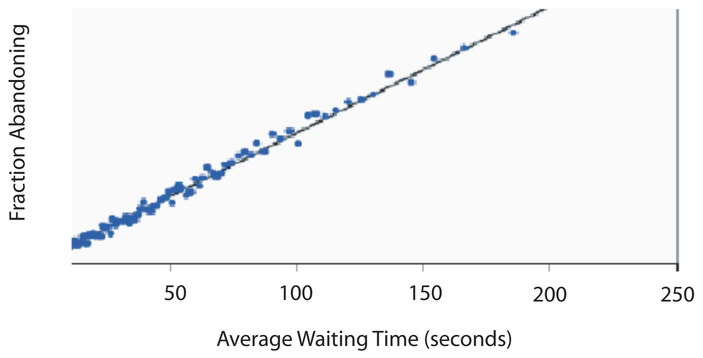

Figure 7.2: Empirical Relationship Between Abandonment and Delay

can become more patient. This effect tends to flatten the observed slope of plots (7.2).

7.2 An extension of the cutting plane algorithm for controlling the abandonment probability

As already shown in the previous section of this chapter, it is very important to control the abandonment probability of the customers in order to obtain good performance of the system. In this section, it is shown how to control the abandonment using the cutting plane algorithm.

Firstly, it is important to clarify that three different types of statistics on the abandonments exist:

- *ABANDONMENT-RATIO*: Represents the fraction of the expected number of contacts having left the system without service over the total expected number of arrivals.
- *ABANDONMENTRATIO-AFTER-AWT*: Represents the fraction of the expected number of contacts having left the system without service and after a waiting time greater than or equal to the acceptable waiting time, over the total expected number of arrivals.
- *ABANDONMENTRATIO-BEFORE-AWT*: Represents the fraction of the expected number of contacts having left the system without service and without waiting the acceptable waiting time, over the total expected number of arrivals.

In the new version of CP (i.e. the one plus the abandonments), constraints on all of these three statistics are imposed. At the first time, changing the scheduling problem P1 and imposing new constraints means also to change the call center parameters considering so far. In fact, now, there are also new three targets:

- abandonParams
- abandonParamsBeforeAWT
- abandonParamsAfterAWT

They are represented by matrices. All of them have dimension equal to $(N+1)\times(P+1)$ (where N is the number of call types and P the number of periods as already said). The generical element of a matrix (i,j), $(i \in N, j \in P)$ is the upper bound imposed on the abandonment probability of the corresponding type (abandon, abandon-Before-AWT, abandon-After-AWT) for the call i in the period j. Moreover the row whose index is $N+1$ represents the parameters grouped by call type; the column $P+1$, instead, the ones grouped by period; the last element in position $((N+1),(P+1))$ represents the global parameter.

At this point, let $a_{i,j}$ be the abandonment probability of the call type i in the period j (supposing a generic type of statistic among the three ones). Then the generical constraint imposed has the following expression:

$$a_{i,j} \leq u_{i,j}$$

where $u_{i,j}$ is the upper bound decided by the user.

Due to the fact that the functions $a_{\cdot,\cdot}$ are non linear at all, then it is necessary to find, also for them, linear approximations $\widehat{a}_{i,j}$ using the simulator. The corresponding constraint is

$$\widehat{a}_{i,j} \leq u_{i,j}$$

7.2.1 How to find the linear cuts

As already remarked in the Chapter 5, the cutting plane methodology replaces the non linear constraints on the service levels by linear cuts that are found using the simulator. In this chapter, this method is extended in order to control the abandonment probability too. It means that the non linear constraints on the abandonment probability are replaced by linear equations using again the simulator. In order to understand well how it works, let's suppose that $\overline{y}$ is the current staffing solution found at a generical iteration of the algorithm and let's suppose that it does not satisfy a generical constraint on the abandonment probability:

$$a_{i,j}(\overline{y}) > u_{i,j}$$

It is important to remark that the $a_{\cdot,\cdot}$ are functions of the staffing solution. In order to replace this equation by a linear cut, adopting the same methodology, already described for the original scheduling problem P1 and using the simulation, it is possible to find the subgradient $\overline{q}$ such that:

$$A(\overline{y}) \doteq \overline{q}^t \times y - \overline{q}^t \times \overline{y} + a_{i,j}(\overline{y}) \leq u_{i,j} \tag{7.1}$$

It could be considered the simple adoption of the same algorithm to this new version of the scheduling problem that controls at the same time the abandonment probability too.

But, as already remarked in the chapter 5, an important property that the non linear function, giving the abandonment probability, must satisfy is the concavity. In fact only in this specific case, it is sure that the found subgradient is the right one.

So the mathematical equation 7.1 has to be replaced by a new one:

$$1 - A(\overline{y}) = 1 - \overline{q}^t \times y - \overline{q}^t \times \overline{y} + a_{i,j}(\overline{y}) \tag{7.2}$$

In order to understand the 7.2, it is important to note that increasing the number of agents working at the call center, the abandonment probability follows a trend shown in figure 7.3.

In figure 7.3, it is shown a reasonable behavior: increasing the number of agents working at the call center, the abandonment probability, starting from 1 (no agents at the call center), decreases until reaching low values. On the other hand, in order to guarantee the concavity property of the functions, the inverse of the abandonment probability is plot too. In fact in figure 7.3, the black curve and the red one represent the function $A(\overline{y})$ and $1 - A(\overline{y})$ respectively.

Firstly, one can re-write 7.1, in a generic expression as in the follow:

$$A(\overline{y}) = \overline{q}^t \times y - \overline{q}^t \times \overline{y} + a(\overline{y}) \leq u \tag{7.3}$$

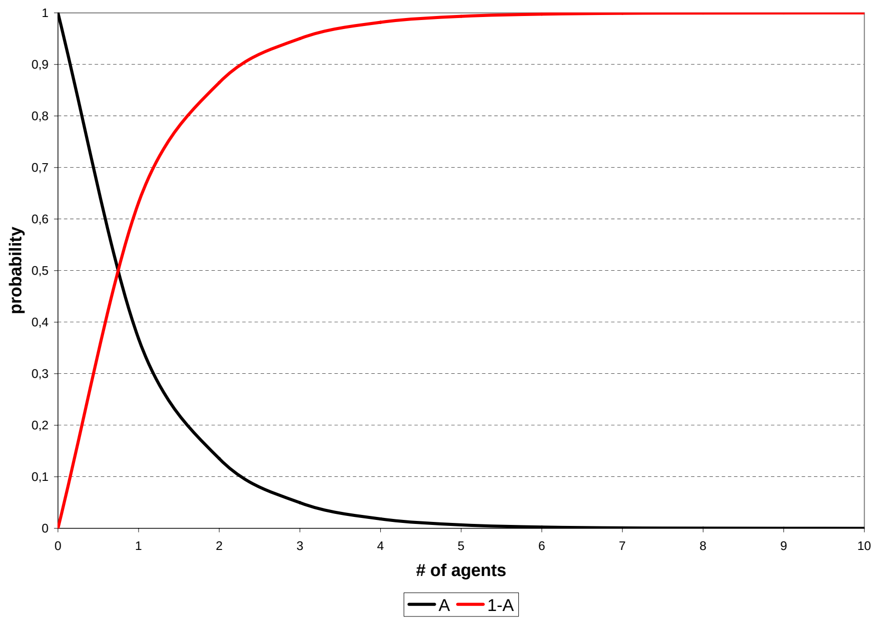

Figure 7.3: The abandonment probability

In order to obtain the 7.2, it is necessary to manipulate 7.3:

$$-A(\overline{y}) = -\overline{q}^t \times y + \overline{q}^t \times \overline{y} - a(\overline{y}) \geq -u \tag{7.4}$$

Then

$$1 - A(\overline{y}) = 1 - \overline{q}^t \times y + \overline{q}^t \times \overline{y} - a(\overline{y}) \geq 1 - u \tag{7.5}$$

Finally

$$\overline{q}^t \times y \leq -a + \overline{q}^t \times \overline{y} + u \tag{7.6}$$

And so 7.6 represents the cut to be added.
A generic component of the subgradient is estimated in the following way:

$$\tilde{q}_i = \frac{a(\overline{y} + d \times e_j) - a(\overline{y})}{d} \tag{7.7}$$

To choose d and e_j, one can follow the same considerations done with the service levels constraints. In this case, two aspects are important:

- the priority used for adding cuts: the cuts related to the service levels and the ones to the abandonment probability are examined separately. At each iteration cuts of both types can be added;
- during the execution of the algorithm, cuts of all the three abandonment types are added.

For the computational results, see the related section of this thesis.

Chapter 8

New solution approaches

This chapter is the final part of this thesis in which some possible solving approaches, that in any case will be future works, are analyzed and presented in order to solve the scheduling problem for a multi-skill call center.

In particular some of them are improvements of the CP algorithm described so far while others are completely new.

Some of these new approaches were suggested by Prof. Koole [1] from the University of Amsterdam (Netherlands) starting from a *Max-flow problem* formulation in order to generalize to the scheduling one.

In the following, two different solution approaches are proposed for solving the scheduling problem of a multiskill call center.

The former is a new solution method described in section 8.3. The latter, instead, is a cutting plane approach initialized using a different max flow problem (section 8.4).

8.1 The Max-flow problem

The *Max-flow problem* is the problem to find a flow from a source to a destination that satisfies the so called *balance equation*(see definition 8.1) and maximum and minimum capacity constraints on each arc (see also (Schoen 2006) for more details).

Definition 8.1. *The balance equation*
For any transient node of the network the input flow has to be equal to the output one; for the source node its output flow has to be equal to the input flow into the destination one.

In this problem the *circulating flow* amount is a variable and its definition is given in 8.2.

Definition 8.2. *Circulating Flow*
Assuming that in the network there are only a source and a destination and the other nodes are transient, the *circulating flow* represents the sum of the output flows from the source node.

[1] one can find info at this web page Ger Koole

For the balance equation, the circulating flow is equal to the input flow into the destination node. If in the network there are more than one source node then one can easily report himself to the previous situation inserting a *dummy node* linked to each source node by arches whose maximum capacity is infinity (the same thing for the destinations).

In general, one can formulate the max-flow problem in this way:

Let be V the nodes set and $E \subseteq V \times V$ the arches set of the network. Moreover, $s \in V$ is the source node and $t \in V$ is the destination one. $Cmin_{ij}$ represents the minimum capacity of the arc $(i,j) \in E$ (usually equal to 0) and $Cmax_{ij}$ the maximum capacity of the arc $(i,j) \in E$. The variables of the problem are of two types:

- $f_{ij} : \forall (i,j) \in E$ represents the flow on the edge (i,j);
- val represents the flow value.

Instead the constraints are of two types:

- the balance equations:

  $$\sum_{(v,j)\in E} f_{v,j} - \sum_{(i,v)\in E} f_{i,v} = b_v \forall v \in V$$

  where b_v is equal to val if $v = s$; $-val$ if $v = t$; 0 otherwise;
- Bounds on the flow on each arc:

  $$Cmin_{i,j} \leq f_{i,j} \leq Cmax_{i,j} \forall (i,j) \in E$$

The objective is to maximize the flow value. In this way the general max-flow formulation can be presented in 8.1

$$\boxed{\begin{array}{c} max\ val \\ \sum_{(v,j)\in E} f_{v,j} - \sum_{(i,v)\in E} f_{i,v} = b_v \forall v \in V \\ Cmin_{i,j} \leq f_{i,j} \leq Cmax_{i,j} \forall (i,j) \in E \end{array}} \qquad 8.1$$

This problem has a lot of applications in real life. For example, to find the maximum water quantity in a distribution network or the analysis of the urban traffic flows or the emergency exits design of public buildings.

If the minimum and maximum capacities on the arches are integer, then all the feasible solutions, in particular the optimal one, will be integer too.

8.2 The scheduling problem as a maximum flow problem

In this section, it is assumed that:

- N is the set of call types
- $b(i)$ is the demand for type $i \in N$
- $y(S)$ is the supply of agents whose skill set $S \subset N$

In order to formulate the scheduling problem for a multi-skill call center, it is important to add to the previous assumptions the following ones:

- Q, the set of shifts;
- P, the set of time periods;
- $g_t(S)$, the number of generalists necessary to deal with the traffic from skill set $S \subset N$ during period t, according to some appropriate traffic model (e.g., Erlang A), and some SL definition;
- $a_{tj} = 1$ if an agent with shift j works during period t, 0 otherwise;
- $c(i, S)$, the cost of an agent having shift i and skill set S;
- $x(i, S)$, the number of agents having shift i and skill set S (the decision variable);
- p, a parameter indicating the amount of efficiency loss due to multi-skilling compared to having only fully x-trained agents (e.g., $p = 1.05$);
- q, a parameter indicating the amount of x-training (e.g., $q = 0.25$).

The shift scheduling algorithm proposed by Koole and implemented and analyzed in this thesis works as follow:

Schema 8.1.

1. *choose some values of p and q;*
2. *solve the MPP below which gives values for x;*
3. *simulate this schedule;*
4. *adapt p and q accordingly;*
5. *repeat from 2 until optimum found.*

The MPP is as follows:

$$\begin{array}{ll}
\min & \sum_{j\in Q}\sum_{S:S\subset N} c(j,S)x(j,S) \qquad (1) \\
& \text{subject to} \\
& \sum_{j\in Q} a_{tj}x(j,S) = y_t(S) \qquad \forall S\subset N, t\in P \qquad (2) \\
& \sum_{S':S'\cap S\neq\emptyset} y_t(S') \geq p g_t(S) \qquad \forall S\subset N, t\in P \qquad (3) \\
& \sum_{S':S'\cap S\neq\emptyset, S'\not\subset S} y_t(S') \geq q g_t(S) \qquad \forall S\subset N, S\neq N, t\in P \qquad (4) \\
& x(j,S)\geq 0 \text{ and integer} \qquad \forall j\in Q, S\subset N \qquad (5)
\end{array}$$

(MPP)

The (2) can be modified replacing = by ≥ if it is better from an optimization point of view.

Another interesting option could be to replace $\sum_{j\in Q} a_{tj}x(j,S) \geq y_t(S)$ by

$$\sum_{j\in Q} a_{tj}x(j,S) = y_t(S) + z_t(S)$$

One can add a constraint of the form $\sum_t z_t(S) \geq z(S)$, with $z(S)$ represents the time that agents with skill set S are in total needed for other work such as responding to emails. In fact, a new single-period max-flow can be formulated for this.

Furthermore it is natural to add constraints of the form $\sum_{j\in Q'} x(j,S) \leq e(S)$ or $= e(S)$, where $e(S)$ is the number of agents with skill set S and a contract that allows them to do the shifts in the set Q'.

In the following sections, two different approaches, for solving the scheduling problem for a multiskill call center, are presented and described. The former is the *original approach* as presented in schema 8.1. The latter is an *hybrid* one in the sense that it starts solving MPP and adds cuts as found in CP algorithm.

8.3 The original approach

In order to implement the approach presented in section 8.2 and described in schema 8.1, the variables $x(i,S)$ (the number of agents having shift j and skill set S) have to be interpreted as the x_{ij} introduced in §2. Then they have to be seen as the number of agents of type i assigned to the shift j. At the same time, $c(i,S)$ (the cost of an agent having shift j and skill set S) has to be interpreted as c_{ij} (the cost of an agent type i assigned to the shift j) introduced in §2.

The values of $g_t(S)$ (i.e. the number of generalists necessary to deal with the traffic from skill set $S \subset N$ during period t) become $g_t(i)$, $i \in T$ (where T is the set of agents types). It means that they assume the following meaning: the number of generalists necessary to deal with the traffic from skill set $S_i \subset N$ during period t. Moreover they are computed using the Erlang formula (Erlang-A or Erlang-C if there are or not abandons respectively) as in the follow:

$$g_t(i) = \sum_{j\in S_i} s_{jp}$$

where s_{jp} represents the number of agents needed to handle the calls of type j in period p computed using one of the Erlang formulas accordingly .

In practise it is supposed that the number of generalists, necessary to deal with the traffic from skill set S_i, is equal to the sum of agents needed to handle all the call types in the set S_i.

The values of p and q are adapted if the current solution does not respect all the service level constraints. In this case, the adaptation procedure increases p and q by a fixed value (0.05 in the performed tests (see chapter Chapter 9 for more details)) starting from a value that is an input data to the algorithm.

At the end of the procedure, the Local Search is performed in order to improve the quality of the solution, if it is possible, and it is the same of the one already described in chapter Chapter 5.

In the example 8.1,one can see as the approach works.

Example 8.1. In the following the example 3.2 will be considered. Using the Erlang-C formula and observing the particular structure of the shifts and of the skill sets, it is possible to affirm that the optimal solution has 35 agents of type 1, 35 2 type agents and 34 3 type agents. In this way, in total this example has 104 agents.

Considering the formulation (MPP), in particular the constraints in which there are the two parameters p and q, it is possible to write:

$$\begin{array}{l}
y(0,0)+y(1,0)+y(2,0)\geq pg_0(0)\\
y(0,1)+y(1,1)+y(2,1)\geq pg_0(1)\\
y(0,2)+y(1,2)+y(2,2)\geq pg_0(2)\\
y(0,0)+y(1,0)+y(2,0)\geq pg_1(0)\\
y(0,1)+y(1,1)+y(2,1)\geq pg_1(1)\\
y(0,2)+y(1,2)+y(2,2)\geq pg_1(2)\\
y(0,0)+y(1,0)+y(2,0)\geq pg_2(0)\\
y(0,1)+y(1,1)+y(2,1)\geq pg_2(1)\\
y(0,2)+y(1,2)+y(2,2)\geq pg_2(2)
\end{array}$$

$$\begin{array}{l}
y(1,0)+y(2,0)\geq qg_0(0)\\
y(1,1)+y(2,1)\geq qg_0(1)\\
y(1,2)+y(2,2)\geq qg_0(2)\\
y(0,0)+y(2,0)\geq qg_1(0)\\
y(0,1)+y(2,1)\geq qg_1(1)\\
y(0,2)+y(2,2)\geq qg_1(2)\\
y(0,0)+y(1,0)\geq qg_2(0)\\
y(0,1)+y(1,1)\geq qg_2(1)\\
y(0,2)+y(1,2)\geq qg_2(2)
\end{array}$$

The optimal values of the parameters p and q allow to write the following constraints:

$$\begin{array}{l}
y(0,0)+y(1,0)+y(2,0)\geq 96.28\\
y(0,0)+y(1,0)+y(2,0)\geq 48.97\\
y(0,0)+y(1,0)+y(2,0)\geq 48.97\\
y(0,1)+y(1,1)+y(2,1)\geq 96.28\\
y(0,1)+y(1,1)+y(2,1)\geq 48.97\\
y(0,1)+y(1,1)+y(2,1)\geq 48.97\\
y(0,2)+y(1,2)+y(2,2)\geq 96.28\\
y(0,2)+y(1,2)+y(2,2)\geq 48.97\\
y(0,2)+y(1,2)+y(2,2)\geq 48.97\\
y(1,0)+y(2,0)\geq 69.6\\
y(1,1)+y(2,1)\geq 35.4\\
y(1,2)+y(2,2)\geq 35.4\\
y(0,0)+y(2,0)\geq 35.4\\
y(0,0)+y(1,0)\geq 35.4\\
y(0,1)+y(2,1)\geq 69.6\\
y(0,1)+y(1,1)\geq 35.4\\
y(0,2)+y(2,2)\geq 35.4\\
y(0,2)+y(1,2)\geq 69.6
\end{array}$$

As one can easily see, only some of these constraints are dominant:

$$\begin{aligned}
&y(0,0) + y(1,0) + y(2,0) \geq 96.28\\
&y(0,1) + y(1,1) + y(2,1) \geq 96.28\\
&y(0,2) + y(1,2) + y(2,2) \geq 96.28\\
&y(1,0) + y(2,0) \geq 69.6\\
&y(1,1) + y(2,1) \geq 35.4\\
&y(1,2) + y(2,2) \geq 35.4\\
&y(0,0) + y(2,0) \geq 35.4\\
&y(0,1) + y(2,1) \geq 69.6\\
&y(0,2) + y(2,2) \geq 35.4\\
&y(0,0) + y(1,0) \geq 35.4\\
&y(0,1) + y(1,1) \geq 35.4\\
&y(0,2) + y(1,2) \geq 69.6
\end{aligned}$$

In this way, this method gives a solution with 35 agents of each type (in total 105). The next phase of Local Search has eliminated one agent allowing to obtain a final solution with 104 agents in total.

8.4 The hybrid approach

Using the model (MPP), it is possible to design a new algorithm that uses, at the same time, the schema 8.1 and the general idea of CP presented in Chapter 5.

As already remarked, a very important phase in CP is the initialization one. In the original version of CP (presented in Chapter 5), it is performed solving a max flow problem that allows starting with a number of agents that, in each period, is a given fraction of the load.

The hybrid approach represents a possible variation of the initialization phase of CP and it can be obtained replacing the max flow problem (defined in Chapter 5) with MPP. This means that in this new method, the starting problem is MPP while in the next phases cuts are added found as described in Chapter 5. It is important to note that in this approach the two parameters p and q are not changed during the execution. In fact they remain fixed and are set experimentally. This happens because in this case the max-flow is only a tool in order to initialize the cutting plane approach and to find an initial solution.

On the other hand, it has to be noted that in this hybrid approach the z variables (representing the skill transfer) are not used at all.

The schema 8.1 shows the hybrid approach.

Note that the initialization procedure used in the cutting plane algorithm (Chapter 5) is completely different form the one used in this approach, even though both are using a max-flow problem.

In particular, the max flow problem of the cutting plane procedure (described in Chapter 5) assigns, for each call type, a number of agents that can handle it greater than equal to a fraction of its load.

The max flow problem, described in this section, instead, assigns, for each agent type a and period p, a number of agents that depends on the number of generalist necessary to handle the calls that belong to the skill set S_a.

```
Algorithm Hybrid
Input: boolean IP
1.  Initialization phase:$x$=solve MPP;
2.  while (!$x$ feasible)
3.      do
4.          add cuts (see Chapter 5 for more details)to MPP;
5.          $x$= solve MPP;
6.          if (!IP)
7.              then rounding on $x$;
8.  Local search on $x$;
```

Figure 8.1: the general scheme of the Hybrid Approach

Chapter 9

Computational results

In this section, it's shown an overview about the computational results obtained considering all the methods and algorithms described in detail in the previous chapters.

In the following, after selecting some significant problem tests, some final comparisons, among the three different solution approaches described in Chapter 3, Chapter 5 and Chapter 6, are proposed.

The problem tests will be partially taken from real life ,inspired by Bell Canada [1] that, as already said, is a telephonic company that supports, in part, this research. This is done in order to have an idea about the performances of all the methods considering realistic scenarios. In fact, as already remarked in Chapter 4, increasing the number of call types, agent types and periods, the complexity of the scheduling problem is increased too.

This chapter is organized as in the follow:

- firstly, it is studied and analyzed the improved version of the cutting-plane methodology already described in section 5.6. This is done in order to understand how CP_{new} is able to get more feasible solutions in regard to the original version of the method. It is shown how the new rounding (for the instances that require solving LP) and the new local search can improve the quality (in terms of feasibility and cost) of the final solutions. In particular this analysis will be done on a real instance that requires both rounding and local search during the algorithm execution. In this way, one can see the benefits of both new modules;

- secondly, comparisons between CP (original version) and the new one (CP_{new},described in section 5.6) are presented and analyzed;

- then, considering some problem tests, it is shown the behavior of the different methods used to initialize the Randomized Search algorithm. In fact in the section Chapter 6, it is affirmed that the best initialization method, by experiments, is the so called *round-robin method.* In this specific section some results are shown in order to support it (section 9.3).

[1]Bell Canada is a great telephonic company that for 127 years is serving Canadians communication needs. Its home page is Bell Canada

- so, comparisons are shown among the three methods: TS, CP_{new} and RS considering the same scenarios and looking not only to the cost of the final result but also to its quality (see section 9.4);
- then the results, controlling, directly, the abandonment probability (method described in Chapter 7), are presented;
- finally, also some results obtained using the two new approaches presented in Chapter 8 are shown;

9.1 CP_{new}:the feasibility problem

As problem instances become larger and more complex, there is a definite possibility that cutting plane algorithm would return a set of low-cost, but only near-feasible solutions.

While this may be acceptable in some practical settings, it is nonetheless annoying to be unable to provide the call center manager with a solution that meets all his/her requirements.

A simple and attractive way of tackling this problem consists in slightly increasing the right-hand side value of the service level constraints when applying the algorithm , except, obviously, for the final long simulation that is used to check the feasibility of solutions.In fact the long simulation is performed considering the original targets.

This idea is tested on a large size call center, using first a value of 0.81 and then of 0.82 as target SL for all periods.

Moreover these tests are combined with experiments on the value of the threshold τ that is used for rounding continuous solutions to integer ones in $CP - LP$ version.

The rationale for investigating different values of τ is that the rounding procedure used in $CP-LP$ introduces a heuristic element in what would otherwise be an exact procedure and that selecting the best value for this threshold is far from obvious.

The example (Big_{52}),analyzed in the following, is characterized by: 20 call types ($N = \{1, \ldots, 20\}$) and 35 agent types ($T = \{1, \ldots, 35\}$). The skill sets are shown in table 9.1. The patience time of each call is exponentially distributed with mean $\nu = 10$ for each period. For this big example, the service levels grouped by call type and the global one are only considered with the following targets: $l(p) = 0.8 \forall p \in P$ and $l = 0.8$.

The general setting is characterized by some particular aspects briefly described in the following:

- the call center opens at 8:00 AM and closes at 5:00 PM;
- the working day is divided into $P = 36$. Each of them has a length equal to 15-minutes (unless otherwise specificated);
- each shift must include 3 breaks: *pre-lunch*, *lunch*, and *post-lunch* breaks. The first and the third breaks are 15-minute long (one period). The lunch break is 30-minute long (2 periods);

- shifts can vary in length between 6.5 hours (26 periods) and 9 hours (36 periods) according to pre-defined patterns (see Table 9.11), thus yielding 285 shift types;

- calls arrive according to a piecewise Poisson process;

- the *routing policy* is an agents' preference-based router (Buist and L'Ecuyer (2005));

- All service times are exponential with service rate $\mu = 8$ calls per hour. Patience times have a mixture distribution: the patience is 0 with probability 0.001, and with probability 0.999, it is exponential with rate 0.1 per minute;

- For most instances, the *aggregate* service level constraints for each period are only considered. These require that at least 80% of all the calls received during the period has to be answered within 20 seconds (i.e., $\tau_p = 20$ seconds and $l_p = 0.8$ for each p). The satisfaction of these constraints implies that the global constraint with $\tau = 20$ seconds and $l = 0.8$ is automatically satisfied, but still this is explicitly required, because this constraint plays a key role in the cutting-plane algorithm. In some cases, also *disaggregate* SL constraints, for each couple (call type k, period p) with $\tau_{k,p} = 20$ seconds and $l_{k,p} = 0.5$ for all k and p, are imposed.

The formula used in order to compute agents' costs takes in account both the number of skills in the agent's skill set and the length of the shift being worked.

This formula is as follows:

Formula 9.1. $c_{ij} = (1 + (\eta_i - 1)*\xi) * \frac{l_j}{30} \quad \forall i \in T, j \in Q \qquad \textit{(9.1)}$

where l_j is the length (in periods) of shift type j, 30 is the number of periods in a "standard" 7h30-shift, η_i is the cardinality of S_i, $\forall i \in T$, and ξ is an instance-specific parameter representing the cost associated with each agent skill. For this example, the parameter ξ is equal to 0.1.

All the results are obtained performing a number of replications nr equal to 8.

Moreover, two CPU time budgets are examined: 5 and 10 hours. Due to the complexity of the scenario, the $CP - LP_{new}$ is run.

The working day starts at 8:00 AM and ends at 9:00 PM and the total number of periods is equal to 52. All the shifts have a fixed length of 7.5 hours, thus yielding a total of 123 different shifts (considering also shifts starting at 1:00 PM and 1:30 PM in order to cover all the periods).

The arrival rates are plotted in Figure 9.1.

All 48 runs performed for each of the two target values (0.8 and 0.81) turned out to be feasible for the 50,000-day simulation!

The results can be summarized in tables in which the meaning of each column is reported in the following:

- n:number of days, simulated during the cuts generation process, to check about the feasibility of the solution;

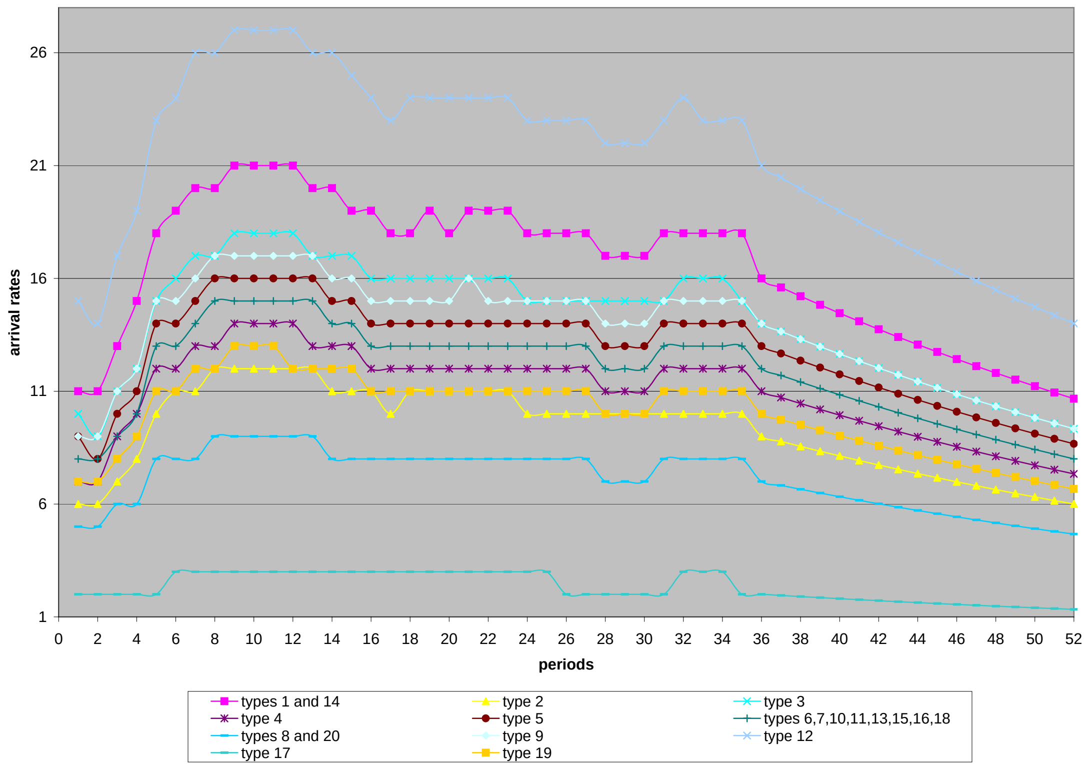

Figure 9.1: Larger instances: arrival rates

Skill	Agent types
1	10, 12, 14, 16, 35
2	1, 3, 5, 7, 9, 17
3	3, 11, 13, 18
4	1, 4, 10, 19, 35
5	2, 6, 7, 9, 20
6	4, 8, 11, 12, 21
7	4, 5, 9, 11, 13, 14, 22
8	1, 3, 4, 5, 9, 23
9	4, 5, 8, 12, 13, 24
10	4, 7, 9, 11, 25, 35
11	3, 8, 10, 13, 14, 26
12	1, 4, 6, 9, 14, 27
13	7, 8, 12, 14, 28
14	1, 5, 6, 13, 29
15	4, 9, 11, 30, 35
16	1, 5, 10, 31
17	2, 3, 12, 13, 32
18	1, 7, 11, 14, 33
19	2, 5, 7, 11, 12, 13, 34
20	2, 4, 6, 8, 15, 35

Table 9.1: The skill sets for the larger instances

- preR : solution at the end of the cuts generation phase and before the second type of rounding;
- finalA : value of τ at the end of the second type of rounding (in the second type of rounding the value of approx is adapted using the simulation);
- postR : the cost of the solution after the second type of rounding;
- postLS : the cost of the solution after the local search (i.e. final solution);
- worstSL : the worst sl (for each run the worst sl is the one that is smaller than the target);
- % : the violation in percent of the Worst sl ((target -worstsl)/target);
- IT : number of iterations performed by the cut generation process;
- minF : the best solution cost found considering all the runs;
- medF: median value on the all costs obtained with all runs;
- max violation: the maximum violation (in percent) on all the runs;
- min pre R: minimum cost value before the second type of rounding on all the runs;
- min post R: minimum cost value after the second rounding on all the runs;

- med post R: median of all the cost values after the second type of rounding on all the runs.

Moreover , in the following, $b1$ and $b2$ are, respectively, the smaller and the larger CPU time budget.

The tables 9.2, 9.3 and 9.4 show the final results with the target on service levels equal to 0.8 and varying the threshold τ for the rounding module from 0.5, to 0.6 and to 0.7.

The tables 9.5, 9.6 and 9.7 show the final results with the target on service levels equal to 0.81 and varying the threshold τ for the rounding module from 0.5, to 0.6 and to 0.7.

Finally, the tables 9.8, 9.9 and 9.10 show the final results with the target on service levels equal to 0.82 and varying the threshold τ for the rounding module from 0.5, to 0.6 and to 0.7.

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	122.36	0.60	129.2	134.10	0.798	0.800	0.26	40						
	300	130.41	0.58	137.8	131.50	0.793	0.800	0.90	64						
	300	133.66	0.64	145.5	135.50	0.799	0.800	0.12	41						
	300	125.22	0.64	137.4	131.70	0.800	0.800	0.00	73						
b1	300	128.81	0.55	133.9	134.50	0.794	0.800	0.71	52	130.80	133.60	0.899	122.36	129.20	135.65
	300	124.43	0.60	137.9	133.10	0.795	0.800	0.62	84						
	300	128.81	0.55	133.9	134.50	0.794	0.800	0.71	52						
	300	125.72	0.60	131.9	130.80	0.797	0.800	0.42	41						
	400	128.46	0.64	145.9	137.20	0.800	0.800	0.00	36						
	400	130.94	0.74	151.4	135.00	0.796	0.800	0.49	25						
	400	122.00	0.71	141.4	135.20	0.797	0.800	0.41	14						
	400	132.32	0.60	139.5	133.50	0.800	0.800	0.04	90						
b2	400	139.60	0.60	153.0	139.00	0.794	0.800	0.80	51	130.50	137.65	0.799	116.03	139.50	149.30
	400	132.59	0.60	147.2	138.30	0.800	0.800	0.03	45						
	400	132.71	0.68	157.4	139.00	0.800	0.800	0.00	38						
	400	116.03	0.88	160.3	138.10	0.799	0.800	0.15	8						

Table 9.2: Results without increment of SL and $\tau = 0.5$

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	118.92	0.66	131.8	135.40	0.790	0.800	0.75	47						
	300	120.90	0.60	133.0	132.30	0.800	0.800	0.18	48						
	300	121.19	0.56	132.0	130.90	0.800	0.800	0.00	33						
	300	125.50	0.77	160.2	131.70	0.790	0.800	0.84	40						
b1	300	121.00	0.68	143.0	132.70	0.790	0.800	0.92	63	130.90	132.50	0.924	118.92	131.80	136.90
	300	120.53	0.68	139.1	141.30	0.800	0.800	0.59	54						
	300	123.85	0.58	134.7	132.20	0.800	0.800	0.00	58						
	300	122.80	0.68	139.4	132.70	0.800	0.800	0.03	65						
	400	116.35	0.79	152.9	142.10	0.800	0.800	0.00	10						
	400	122.10	0.63	141.1	130.40	0.800	0.800	0.00	65						
	400	123.04	0.66	142.6	134.40	0.795	0.800	0.57	25						
	400	116.21	0.71	141.6	137.60	0.799	0.800	0.17	9						
b2	400	124.06	0.63	135.8	134.20	0.793	0.800	0.92	116	130.40	134.30	0.915	116.21	135.80	141.35
	400	127.69	0.66	140.5	131.70	0.800	0.800	0.00	27						
	400	117.31	0.72	139.5	131.60	0.796	0.800	0.46	22						
	400	127.92	0.66	144.5	136.20	0.800	0.800	0.00	27						

Table 9.3: Results without increment of SL and $\tau = 0.6$

Moreover, in all of the previous tables, one can easily see that increasing the CPU time budget, from $b1$ to $b2$, the $minpostR$ and the $medpostR$ are increased too, even if we get a better final solution. It seems to underline the importance of the local search procedure.

It is possible to summarize all the final results by box-plots (see figures 9.2, 9.3 and 9.4).

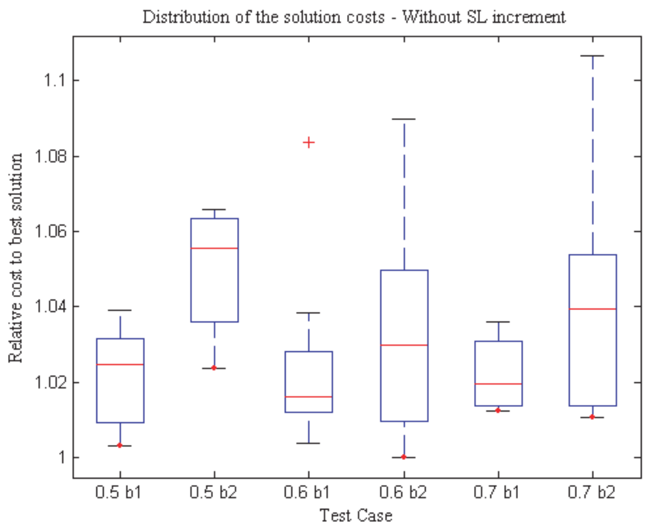

Figure 9.2: No increment on the target service level, varying τ

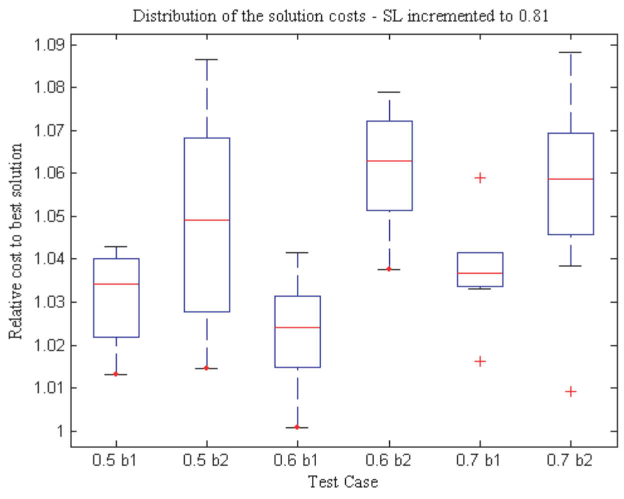

Figure 9.3: Incrementing the target service level to 0.81, varying τ

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	113.56	0.72	136.9	134.20	0.796	0.800	0.49	41						
	300	122.95	0.64	138.9	135.10	0.798	0.800	0.22	45						
	300	112.11	0.66	135.8	132.80	0.800	0.800	0.00	52						
	300	116.50	0.68	136.2	132.30	0.798	0.800	0.25	47						
b1	300	115.97	0.72	142.3	132.00	0.800	0.800	0.03	40	132.00	132.95	0.914	112.11	135.80	137.90
	300	114.96	0.71	141.1	132.10	0.800	0.800	0.00	74						
	300	113.43	0.66	136.6	134.60	0.794	0.800	0.69	26						
	300	112.74	0.80	163.4	133.10	0.793	0.800	0.91	58						
	400	117.64	0.69	150.9	135.90	0.794	0.800	0.70	105						
	400	117.53	0.68	147.6	138.90	0.797	0.800	0.39	108						
	400	117.58	0.71	141.9	135.50	0.800	0.800	0.00	70						
	400	114.41	0.82	160.4	144.30	0.795	0.800	0.64	13						
b2	400	113.32	0.76	149.2	132.40	0.799	0.800	0.17	15	131.80	135.55	0.697	113.32	136.80	144.75
	400	114.03	0.68	138.8	131.80	0.798	0.800	0.28	29						
	400	115.99	0.69	139.2	132.00	0.800	0.800	0.00	43						
	400	118.69	0.58	136.8	135.60	0.797	0.800	0.37	70						

Table 9.4: Results without increment of SL and $\tau = 0.7$

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	127.07	0.58	133.2	135.90	0.800	0.800	0.00	50						
	300	127.36	0.58	134.9	135.10	0.800	0.800	0.00	30						
	300	127.00	0.66	137.8	132.10	0.800	0.800	0.00	64						
	300	127.03	0.58	133.2	134.20	0.800	0.800	0.00	56						
b1	300	124.89	0.63	129.8	134.60	0.800	0.800	0.00	62	132.10	134.85	0.000	122.94	128.80	133.15
	300	126.61	0.60	128.8	132.30	0.800	0.800	0.00	65						
	300	122.94	0.58	130.9	135.40	0.800	0.800	0.00	113						
	300	128.30	0.53	133.1	136.00	0.800	0.800	0.00	64						
	400	137.31	0.64	154.6	141.70	0.800	0.800	0.00	53						
	400	131.39	0.69	158.3	135.20	0.800	0.800	0.00	14						
	400	124.43	0.71	145.8	138.90	0.800	0.800	0.00	15						
	400	126.69	0.60	141.1	134.90	0.800	0.800	0.00	40						
b2	400	122.38	0.72	143.2	139.70	0.800	0.800	0.00	10	132.30	136.80	0.000	118.76	141.10	150.90
	400	134.87	0.68	156.2	133.10	0.800	0.800	0.00	88						
	400	118.76	0.76	158.5	138.40	0.800	0.800	0.00	21						
	400	132.22	0.60	147.2	132.30	0.800	0.800	0.00	48						

Table 9.5: Results with target SL equals to 0.81 and $\tau = 0.5$

At the first time, one can note that increasing the target service level to 0.81 allows having, at the end of the algorithm, final solutions that are completely feasible satisfying all the service levels constraints.

On the other hand, it is not useful to increment this target to 0.82 because it means to have at the end more expensive solutions without improving their feasibility any more.

Moreover, these results show that the value selected for τ seems to have a slight, but not critical impact, on the quality of the solutions. In fact, it seems to be much more important to make sure that the runs, that are performed, produce feasible solutions, to have a larger set to choose from. For this reason, in the following sections, when the CP_{new} algorithm is used, the results, with no increment of the target service level and increasing it to 0.81, are shown fixing the threshold of τ to the value of 0.5, for the rounding procedure during the cut generation process. This corresponds rounding the not integer components to the nearest integer one.

In conclusion, these tests have clearly showed that modifying only τ was not sufficient to consistently obtain feasible solutions, since more than half of the runs, with target 0.8 (tables 9.2, 9.3 and 9.4), returned infeasible solutions. However, they allow finding an even cheaper feasible solution with a cost 130.4 (obtained, seeing table 9.3, with a budget of 10 h). This again highlights the

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	121.98	0.60	133.0	133.40	0.800	0.800	0.00	69						
	300	129.80	0.61	138.7	133.70	0.800	0.800	0.00	53						
	300	118.78	0.63	140.9	132.50	0.800	0.800	0.00	43						
	300	122.43	0.61	137.9	132.20	0.800	0.800	0.00	56						
b1	300	123.64	0.66	146.6	135.80	0.800	0.800	0.00	51	130.50	133.55	0.000	118.78	128.30	139.80
	300	119.54	0.72	144.1	134.50	0.800	0.800	0.00	57						
	300	118.83	0.88	167.7	134.50	0.800	0.800	0.00	61						
	300	120.93	0.58	128.3	130.50	0.800	0.800	0.00	65						
	400	123.80	0.79	162.6	140.70	0.800	0.800	0.00	14						
	400	134.03	0.76	160.5	138.60	0.800	0.800	0.00	23						
	400	122.71	0.63	143.6	135.30	0.800	0.800	0.00	43						
	400	119.19	0.87	167.9	138.60	0.800	0.800	0.00	11						
b2	400	131.52	0.60	136.5	139.70	0.800	0.800	0.00	104	135.30	138.60	0.000	119.19	136.50	147.00
	400	121.74	0.69	150.4	138.20	0.800	0.800	0.00	14						
	400	122.29	0.71	141.9	136.00	0.800	0.800	0.00	32						
	400	121.07	0.69	140.3	139.90	0.800	0.800	0.00	40						

Table 9.6: Results with target SL equals to 0.81 and $\tau = 0.6$

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	113.98	0.68	141.2	135.80	0.800	0.800	0.00	81						
	300	115.42	0.66	135.4	134.90	0.800	0.800	0.00	86						
	300	117.45	0.72	143.8	138.10	0.800	0.800	0.00	57						
	300	115.50	0.66	133.6	132.50	0.800	0.800	0.00	43						
b1	300	115.94	0.76	147.8	135.00	0.800	0.800	0.00	64	132.50	135.20	0.000	110.90	133.60	142.50
	300	125.01	0.72	149.5	135.40	0.800	0.800	0.00	63						
	300	110.90	0.71	145.0	134.70	0.800	0.800	0.00	42						
	300	119.60	0.63	140.9	135.80	0.800	0.800	0.00	47						
	400	118.32	0.66	146.3	137.30	0.800	0.800	0.00	86						
	400	122.04	0.72	154.3	138.00	0.800	0.800	0.00	34						
	400	111.38	0.87	162.9	138.10	0.800	0.800	0.00	9						
	400	124.04	0.72	153.1	139.80	0.800	0.800	0.00	30						
b2	400	116.94	0.85	163.0	141.90	0.800	0.800	0.00	22	131.60	138.05	0.000	111.38	135.50	151.85
	400	119.50	0.68	135.5	131.60	0.800	0.800	0.00	86						
	400	122.48	0.66	150.6	135.40	0.800	0.800	0.00	107						
	400	116.69	0.66	141.7	139.10	0.800	0.800	0.00	75						

Table 9.7: Results with target SL equals to 0.81 and $\tau = 0.7$

stochastic nature of the overall procedure, which cannot be avoided considering the significant amount of noise in the simulations.

9.2 CP vs CP_{new}: some comparisons

In this section, some comparisons are shown in order to present and analyze the differences between the two solving approach: CP and CP_{new}.

In particular, this section wants to shown how the new rounding and local search work on different call center sizes.

The general setting of the instances is characterized by some particular aspects briefly described in the following:

- the call center opens at 8:00 AM and closes at 5:00 PM;
- the working day is divided into $P = 36$. Each of them has a length equal to 15-minutes (unless otherwise specificated);
- each shift must include 3 breaks: *pre-lunch*, *lunch*, and *post-lunch* breaks. The first and the third breaks are 15-minute long (one period). The lunch break is 30-minute long (2 periods);

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	129.15	0.63	140.3	136.30	0.800	0.800	0.00	63						
	300	128.12	0.58	136.5	132.50	0.800	0.800	0.00	52						
	300	128.53	0.56	135.9	133.70	0.800	0.800	0.00	75						
	300	128.14	0.58	135.6	137.50	0.800	0.800	0.00	111						
b1	300	127.30	0.56	138.0	136.00	0.800	0.800	0.00	52	132.50	136.30	0.000	126.75	130.10	137.25
	300	130.32	0.61	139.8	138.70	0.800	0.800	0.00	71						
	300	126.75	0.52	130.1	137.10	0.800	0.800	0.00	57						
	300	128.12	0.63	140.5	136.30	0.800	0.800	0.00	43						
	400	124.14	0.72	153.3	140.40	0.800	0.800	0.00	17						
	400	138.85	0.63	151.50	139.50	0.800	0.800	0.00	61						
	400	133.92	0.77	161.20	138	0.800	0.800	0.00	17						
	400	129.89	0.69	151.90	139.80	0.800	0.800	0.00	16						
b2	400	130.94	0.76	158.00	144.20	0.800	0.800	0.00	22	133.90	139.65	0.000	108.83	137.20	152.60
	400	108.83	0.87	159.90	134.60	0.800	0.800	0.00	23						
	400	128.70	0.66	144.70	141.20	0.800	0.800	0.00	31						
	400	136.36	0.56	137.20	133.90	0.800	0.800	0.00	60						

Table 9.8: Results with target SL equals to 0.82 and $\tau = 0.5$

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	123.97	0.66	137.5	135.80	0.800	0.80	0.00	53						
	300	124.14	0.64	139.2	133.00	0.800	0.80	0.00	75						
	300	123.94	0.61	140.9	135.50	0.800	0.80	0.00	107						
	300	122.94	0.68	143.6	136.00	0.800	0.80	0.00	47						
b1	300	117.66	0.76	152.7	136.30	0.800	0.80	0.00	47	133.00	135.65	0.000	117.66	137.50	140.05
	300	128.57	0.56	138.5	135.50	0.800	0.80	0.00	53						
	300	124.80	0.68	147.3	133.20	0.800	0.80	0.00	37						
	300	123.97	0.66	137.5	135.80	0.800	0.80	0.00	53						
	400	128.53	0.60	142.3	139.00	0.800	0.800	0.00	89						
	400	132.84	0.63	142.5	138.50	0.800	0.800	0.00	86						
	400	139.17	0.63	162.4	142.10	0.800	0.800	0.00	64						
	400	123.64	0.66	146.3	141.80	0.800	0.800	0.00	25						
b2	400	130.78	0.72	162.8	152.10	0.800	0.800	0.00	12	138.50	141.95	0.000	123.64	142.30	156.05
	400	128.11	0.77	167.3	145.50	0.800	0.800	0.00	30						
	400	130.72	0.74	164.1	150.50	0.800	0.800	0.00	9						
	400	123.70	0.74	149.7	140.00	0.800	0.800	0.00	68						

Table 9.9: Results with target SL equals to 0.82 and $\tau = 0.6$

- shifts can vary in length between 6.5 hours (26 periods) and 9 hours (36 periods) according to pre-defined patterns (see Table 9.11), thus yielding 285 shift types;
- calls arrive according to a piecewise Poisson process;
- the *routing policy* is an agents' preference-based router (Buist and L'Ecuyer (2005));
- All service times are exponential with service rate $\mu = 8$ calls per hour. Patience times have a mixture distribution: the patience is 0 with probability 0.001, and with probability 0.999, it is exponential with rate 0.1 per minute;
- For most instances, the *aggregate* service level constraints for each period are only considered. These require that at least 80% of all the calls received during the period has to be answered within 20 seconds (i.e., $\tau_p = 20$ seconds and $l_p = 0.8$ for each p). The satisfaction of these constraints implies that the global constraint with $\tau = 20$ seconds and $l = 0.8$ is automatically satisfied, but still this is explicitly required, because this constraint plays a key role in the cutting-plane algorithm. In some cases,

Budget	n	preR	finalA	postR	postLS	worstSL	target	%	IT	minF	medF	max violation	min pre R	min post R	med post R
	300	116.02	0.64	135.3	138.60	0.800	0.800	0.00	50						
	300	122.75	0.66	144.5	138.20	0.800	0.800	0.00	70						
	300	114.63	0.69	139.3	135.10	0.800	0.800	0.00	68						
	300	122.78	0.68	144.7	135.30	0.800	0.800	0.00	39						
b1	300	124.99	0.79	154.5	139.10	0.800	0.800	0.00	44	135.10	137.50	0.000	114.63	135.30	140.00
	300	118.10	0.71	140.7	136.80	0.800	0.800	0.00	84						
	300	115.40	0.63	137.6	139.50	0.800	0.800	0.00	41						
	300	114.63	0.69	139.3	135.10	0.800	0.800	0.00	68						
	400	116.81	0.77	156.4	144.50	0.800	0.800	0.00	27						
	400	120.54	0.77	157.9	139.30	0.800	0.800	0.00	32						
	400	115.82	0.80	146.6	136.10	0.800	0.800	0.00	11						
	400	122.07	0.72	148.6	134.60	0.800	0.800	0.00	49						
b2	400	118.78	0.76	146.7	137.00	0.800	0.800	0.00	50	134.60	137.75	0.000	115.82	138.40	147.65
	400	121.95	0.64	138.4	134.90	0.800	0.800	0.00	76						
	400	123.84	0.74	150.6	138.50	0.800	0.800	0.00	65						
	400	117.17	0.69	138.4	138.70	0.800	0.800	0.00	39						

Table 9.10: Results with target SL equals to 0.82 and $\tau = 0.7$

also *disaggregate* SL constraints, for each couple (call type k, period p) with $\tau_{k,p} = 20$ seconds and $l_{k,p} = 0.5$ for all k and p, are imposed.

To compute agents's costs, (9.1) is used.

In order to assess the performance of the proposed algorithms, a number of problem instances will be solved with the various approaches. It is important to remark that all of these general instances have been constructed in such a way as to mimic the operations of real call centers.

Type	length	shift start	break1 start	lunch start	break3 start
1	7h30	8h,8h30,9h,9h30	9h30,11h30	12h,12h30,13h	14h,15h30
2	7h45	9h15	10h45,11h15	12h,12h30,13h	14h,15h30
3	8h	9h	10h30,11h	12h,12h30,13h	14h,15h30
4	8h15	8h45	10h15,10h45	12h,12h30,13h	14h,15h30
5	8h30	8h30	10h,10h30	12h,12h30,13h	14h30,16h
6	9h	8h	11h,12h	13h,13h30,14h	14h45,16h15
7	6h30	10h	11h30,12h	13h,13h30,14h	14h15,15h45

Table 9.11: Description of shift types (285 types)

9.2.1 Computational results

The original cutting plane algorithm is marked with CP when ,at each iteration, an integer problem is solved and $CP - LP$ is the cutting plane algorithm in which, at each iteration, a no-integer linear problem is solved. In this last case, the *rounding module* is called, not only at the end of the method, but, also, at each time the simulator has to check about the feasibility (see Chapter 5 for more details). It remarks again the importance to have a good rounding module. The same considerations can be applied to CP_{new}: CP_{new} and $CP - LP_{new}$.

In order to solve both versions, $Cplex$9.0 is used as the optimization tool.

In total, three different sizes of call center are considered:

1. a little size, called *N-Design* example;

2. a medium one called *MS* example;

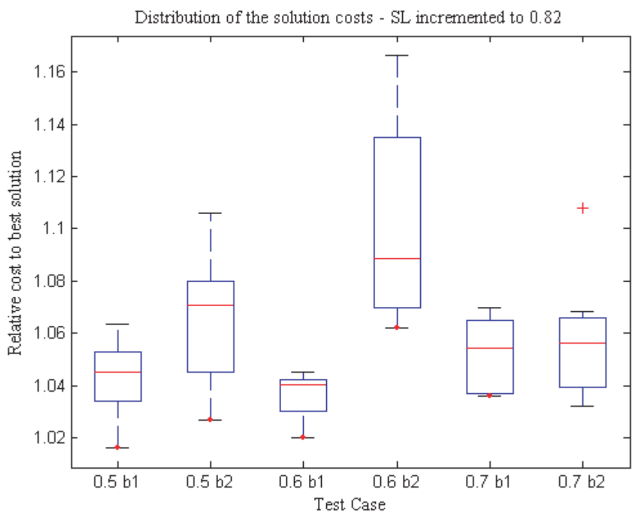

Figure 9.4: Incrementing the target service level to 0.82, varying τ

3. a biggest one, called *Big* example.

About *MS* example, two versions are analyzed: *MS1* with only the global service level and the ones grouped by call type; *MS2* with all the service level constraints. This is done in order to analyze the methods considering the same general scenario and changing only the constraints on the service levels. Moreover, in real call centers, in general, the most relevant types of service levels constrains are the only ones considered in *MS1*. It gives the opportunity to test again the algorithms on more realistic scenarios.

About the *Big* example, in order to analyze different scenarios, there is a first version (Big_{36}) defined on a working day starting at 8 am and ending at 5 pm (i.e. the number of the periods is equal to 36) and a second one (Big_{52}) stretching the working day until 9 pm (i.e. the number of the periods is equal to 52) (it is the same problem already described in section 9.1).This is done in order to show that the performances of the algorithms, in any case, do not depend on the particular shifts structure.

The length of a period is 15 minutes in all the examples and (9.1) is used to compute the agent costs.

For each problem test and each solving method, nr replications are performed, due to the random behavior of the methods, with different CPU time budgets as shown in the final tables.

A critical parameter is n, i.e. the number of days, simulated during the cuts generation phase, to check the feasibility. An higher value of n allows having, at the end, more accurate and stable results, as it is possible to note considering the numerical results. It is called $minRep$ in the CP methods.

The CP methods (original and new version) are also characterized by another parameter: the minimum number of replications ($minRepCP$) used by the simulator during the gradient computation in the cuts generation phase (generally chosen equal to 10% of $minRep$) (see Cezik and L'Ecuyer (2006)). Due to the link between *minRep* and *minRepCP*, in the tables, n will be set to the first one (unless otherwise said).

In the following *empirical optimum* is considered like the true optimum.

Definition 9.1. The *empirical optimum* is the lowest-cost solution generated by either algorithm (over all replications and CPU time budgets) that passes an optimality test constituted by a long simulation (in which the number of simulated days is equal to 50000)

All the results, for each example, are summarized in a whole table with the following columns:

1. *case* that is an index of specific CPU time budget;
2. CPU_{avg} that is the average CPU time per replication;
3. *Min Cost* and *Med Cost* that is the minimum cost and the median one on nr replications respectively.

It is also reported the percentage of solutions that are:

1. feasible (P^*);
2. feasible and within 0.5% and 1% of the empirical cost ($P^*_{0.5}$ and P^*_1);
3. feasible and within 0.5% and 1% of the empirical optimum ($P_{0.5}$ and P_1) without considering the feasibility.

Finally estimates (G) of the expected maximum normalized constraint violation (in percent) are computed as indicated in formula 9.2.

Formula 9.2. $G = 100max\{\frac{l-g(x^*)}{l}, \frac{l_{(.)}-g_{(.)}(x^*)}{l_{(.)}}\}$ *(9.2)*

where (.) depends on the particular type of service level constraint.

About the service levels grouped by periods, the *critical call class* is obtained following the formula 9.3.

Formula 9.3. $(.) = j^* = argmax_{j\in N}\{l_j - g_j(x^*)\}$ *(9.3)*

About the service level grouped by calls, the *critical period* is individualized considering the formula 9.4.

Formula 9.4. $(.) = p^* = argmax_{p\in P}\{l_p - g_p(x^*)\}$ *(9.4)*

About the service level for each call and period, the critical couple (*critical period*, *critical call class*) is obtained by the formula 9.5.

Formula 9.5. $(.) = (p^*, j^*) = argmax_{p\in P, j\in N}\{l_{p,j} - g_{p,j}(x^*)\}$ *(9.5)*

These functions are computed using a long simulation (in which the number of simulated days is equal to 50000) and whenever x^* is an infeasible solution.

Finally, for all the tests executed running CP_{new} or $CP - LP_{new}$, two subcases are analyzed: no increment on the target service level and incrementing the target service level to 0.81. In both cases the threshold τ is fixed to 0.5 in order to have a rounding, during the cut generation process, more accurate. These considerations are done on the base of the final results reported in section 9.1. In particular, in the following tables, $CP_{new}(0.8)$ and $CP_{new}(0.81)$ mean respectively CP_{new} without increment on the target service level and CP_{new} incrementing the target service level to 0.81.

Small size call center: N-Design example

In the following, considering CP_{new} and CP, the N-Design example (see example 4.1 for more details) is analyzed.

This instance has $N = 2$ call types and $T = 2$ agent types, with $S_1 = \{1\}$ and $S_2 = \{1, 2\}$. Agent costs are computed by setting the parameter ξ equals to 0.2 in (9.1). Arrival rates for the two call types are plotted in Figure 9.5.

In this small example, the CPU time budget is varied: case 1 corresponds to $3m$; case 2 to $15m$; case 3 to $30m$ and case 4 to $1h$.

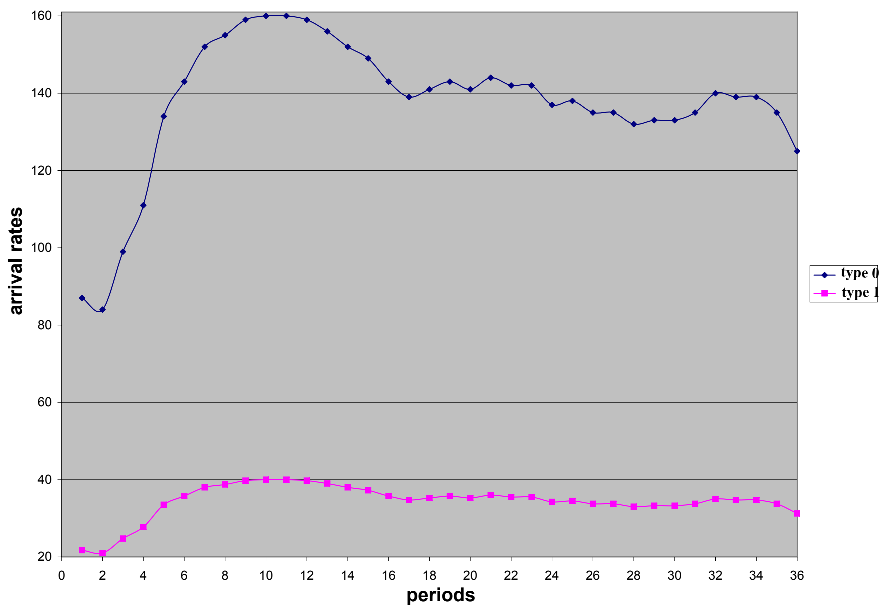

Figure 9.5: Small center: arrival rates

Observing the table 9.12, one can easily see that passing from the original version of CP to the new one (not considering for a moment the target service level), the cost of the best solution is increased a little bit. This is a natural and reasonable behavior because in order to obtain final solutions that are totally feasible (i.e. that respect all the service level constraints), the final cost is increased. In particular this gap becomes thinner increasing the CPU time budget. But it is very important to remark that already with a very short CPU time budget equal to $3m$, the both versions of CP_{new} find a final solution that is totally feasible. In fact, it is possible to note that all the G values are equal to 0. Considering, instead, the comparison between $CP_{new}(0.8)$ and $CP_{new}(0.81)$, there is a little increasing of the objective function in the second case and on the other hand for this example one can not note a real benefit to run the algorithm with a target equal to 0.81, due to the fact that with 0.8 there are all feasible solutions.

In figures 9.6 and 9.7, comparisons between CP and CP_{new} are shown by box-plots. In these figures, one can note that, increasing the CPU time budget, the obtained solutions are more stable (i.e. the box-plots height is shorter).

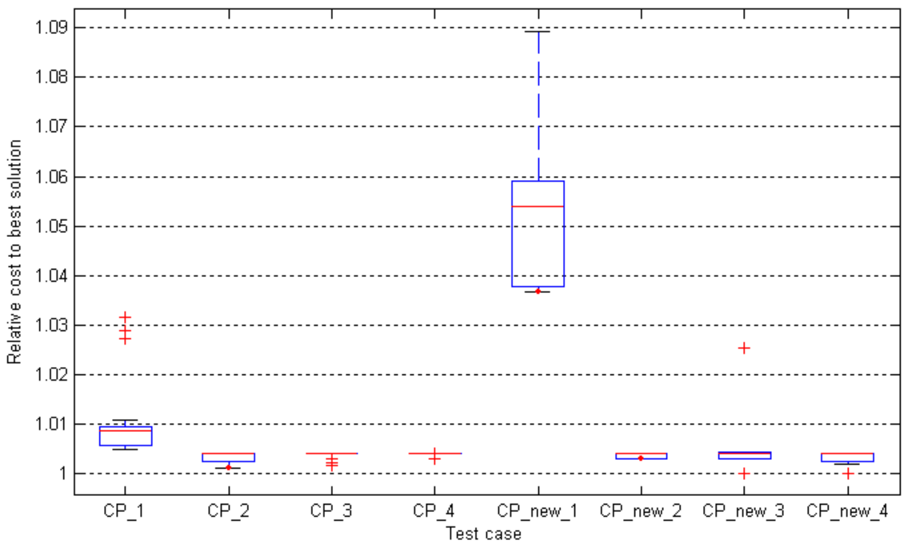

Figure 9.6: N-Design Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.8

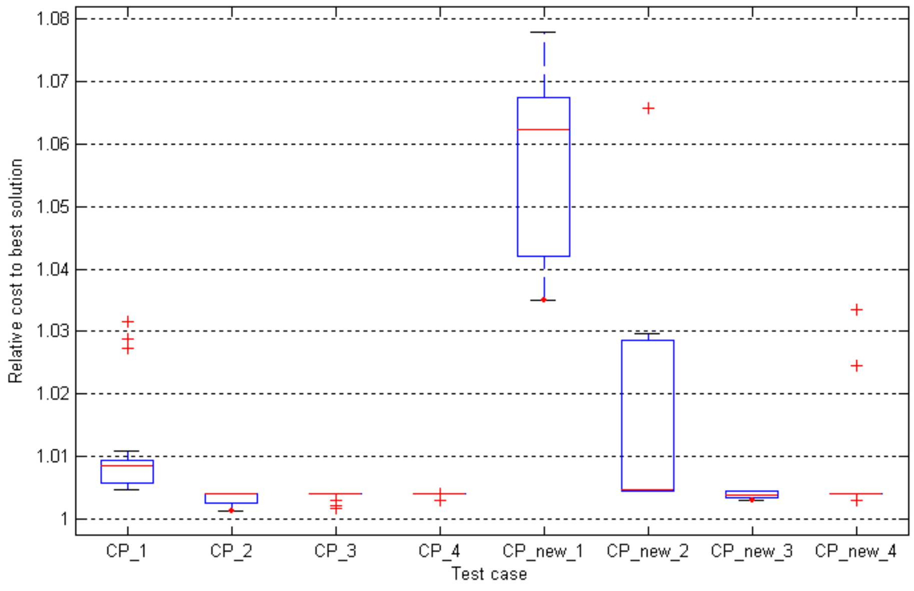

Figure 9.7: N-Design Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.81

Case	Alg	n	CPU_{avg}	Min cost	Med cost	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{call}	G_{period}	$G_{period,call}$
	CP	120	2m25s	35.19	35.33	12.5	21.89	53.13	81.25	65.62	0	0.17	0.59
1	CP_{new} (0.8)	120	2m16s	36.31	36.91	0	0	0	0	100	0	0	0
	CP_{new} (0.81)	120	2m36s	36.25	37.21	0	0	0	0	100	0	0	0
	CP	1500	14m57s	35.07	35.17	68.75	100	68.75	100	68.75	0	0	0.37
2	CP_{new} (0.8)	1500	15m14s	35.13	35.17	100	100	100	100	100	0	0	0
	CP_{new} (0.81)	1500	15m33s	35.18	35.19	62.5	62.5	62.5	62.5	100	0	0	0
	CP	3000	28m31s	35.09	35.17	53.12	100	53.12	100	53.12	0	0	0.36
3	CP_{new} (0.8)	1600	29m34s	35.03	35.17	75	87.5	75	87.5	87.5	0	0	0.15
	CP_{new} (0.81)	1600	30m28s	35.13	35.16	100	100	100	100	100	0	0	0
	CP	7800	60m2s	35.13	35.17	62.5	100	62.5	100	62.5	0	0	0.27
4	CP_{new} (0.8)	2000	61m34s	35.03	35.17	100	100	100	100	100	0	0	0
	CP_{new} (0.81)	2000	60m27s	35.13	35.17	78.13	78.13	78.13	78.13	100	0	0	0

Table 9.12: N-Design Example: solution quality with CP and CP_{new},varying the CPU time budget

Two medium size call centers: $MS1$ and $MS2$

As already done for the previous example, also for this one, the results with two different targets on the service levels are shown.

In the medium-sized instances, there are $N = 5$ call types and $T = 15$ agent types. Skill sets are displayed in Table 9.13.

Moreover the CPU time budgets considered are the follows: 15m, 30m and finally 1h.

Skill	Agent types
1	1, 6, 9, 10, 11, 14, 15
2	2, 7, 9, 10, 12, 13, 14, 15
3	3, 8, 13, 15
4	4, 11, 12, 13, 14, 15
5	5, 6, 7, 8, 10, 11, 12, 14, 15

Table 9.13: Medium-sized center: skill sets

The parameter ξ used to compute agent costs in (9.1) is now equal to 0.1. Arrival rates for all call types are plotted in Figure 9.8.

Case	Alg	n	CPU_{avg}	Min cost	Med cost	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{period}
	CP	600	14m22s	20.9	21.48	0	0	0	0	62.50	0.46
1	CP_{new} (0.8)	300	14m15s	18.33	18.85	0	0	0	0	87.50	0.51
	CP_{new} (0.81)	300	15m18s	17.92	18.76	0	0	0	0	100	0
	CP	1000	24m39s	20.83	21.40	0	0	0	0	75	0.69
2	CP_{new} (0.8)	600	26m38s	17.56	18.43	0	0	0	0	75.00	0.41
	CP_{new} (0.81)	600	28m40s	17.34	18.69	0	0	13	13	100	0
	CP	2000	43m36s	20.19	21.69	0	0	0	0	75	0.22
3	CP_{new} (0.8)	1000	44m46s	17.36	17.66	0	0	25	25	87.50	0.02
	CP_{new} (0.81)	1000	59m46s	17.22	18.27	12.50	12.50	12.50	12.50	100	0

Table 9.14: *MS1* Example: solution quality with CP and CP_{new},varying the CPU time budget

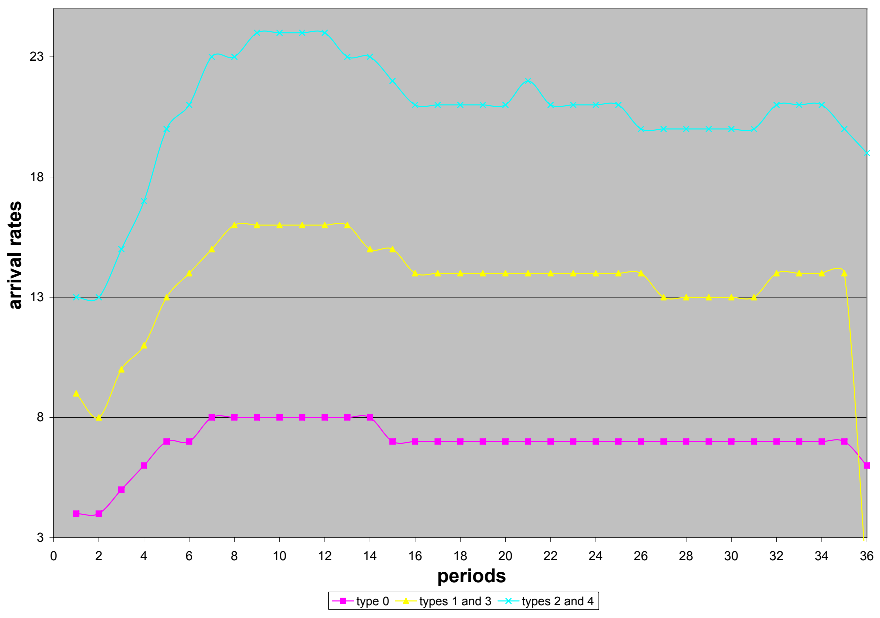

Figure 9.8: Medium-sized center: arrival rates

Analyzing tables 9.14 and 9.15, one can see that the final cost obtained by CP is higher than the one given by the new version. This does not happen with the previous example. In fact, when the complexity of the problem become larger, then a LP is solved at each step of the cutting plane procedure and not an IP. It means that with medium and large size call center, the final solutions are affected by the improvements on the rounding method too (as described in Chapter 5). This last aspect, again, underlines the importance of defining a good rounding method because it can affect a lot on the quality of the final solution in terms of its cost.

Moreover, another aspect that affects the quality of the final solution is the CPU time budget. In fact one can note that increasing it, the final cost is reasonable reduced. This happens in a very relevant way considering the results obtained with $CP_{new}(0.8)$.

In both methods MS1 and MS2, the best result is obtained by $CP_{new}(0.81)$ where the solution has the lower cost and at the same time satisfies all the service level constraints. In particular, analyzing better tables 9.14 and 9.15, it is possible to note that $CP_{new}(0.81)$ is able to give solutions totally feasible with all the CPU time budgets.

Moreover, as it will be explained later, it is possible to note that in MS2 the final cost is reasonable increased.

In figures 9.9,9.10, 9.11 and 9.12 the all the results are plotted by box-plots.

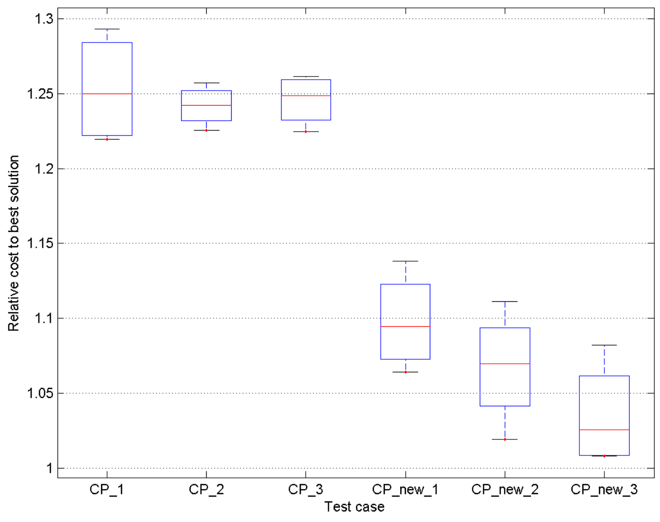

Figure 9.9: MS1 Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.8

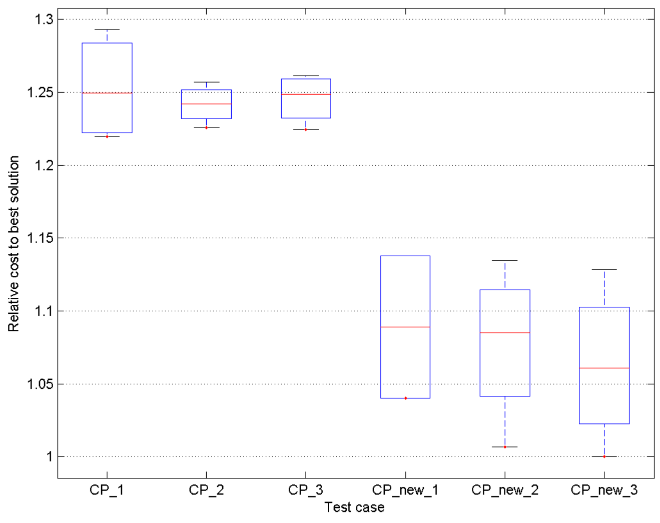

Figure 9.10: MS1 Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.81

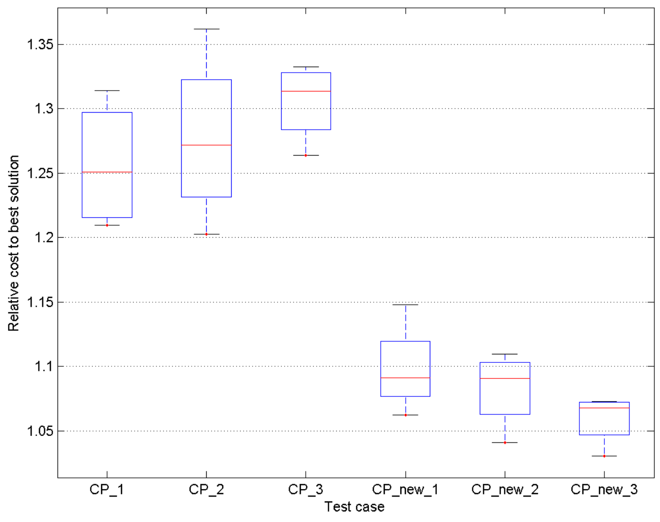

Figure 9.11: MS2 Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.8

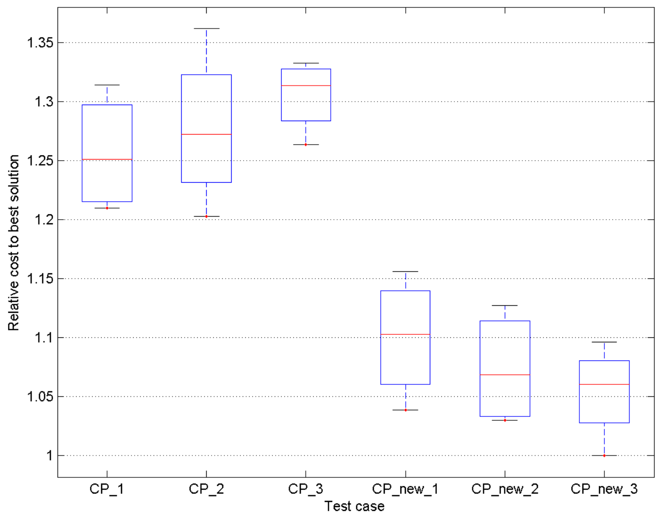

Figure 9.12: MS2 Example: Boxplots of optimality gap (CP vs CP_{new} with target 0.81

Case	Alg	n	CPU_{avg}	Min cost	Med cost	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{call}	G_{period}	$G_{period,call}$
1	CP	400	14m51s	21.04	22.56	0	0	0	0	87.50	0	0.82	4.55
	CP_{new} (0.8)	400	13m48s	18.48	18.98	0	0	0	0	87.50	0	0.58	0
	CP_{new} (0.81)	400	15m33s	18.07	19.18	0	0	0	0	100	0	0	0
2	CP	400	28m14s	20.92	22.48	0	0	0	0	75	0	0.86	0.12
	CP_{new} (0.8)	400	29m11s	18.11	18.97	0	0	0	0	87.50	0	0.54	0
	CP_{new} (0.81)	400	30m07s	17.91	18.59	0	0	0	0	100	0	0	0
3	CP	400	59m50s	21.04	22.70	0	0	0	0	75	0	0.69	0
	CP_{new} (0.8)	400	58m40s	17.92	18.57	0	0	0	0	75.00	0	0.28	0
	CP_{new} (0.81)	400	60m34s	17.39	18.44	13	13	13	13	100	0	0	0

Table 9.15: *MS2* Example: solution quality with CP and CP_{new},varying the CPU time budget

Two large size call center: Big_{36} and Big_{52}

In this section, two large size call centers will be analyzed in order to study and to describe the performances of the two versions of the cutting plane algorithm in presence of realistic frameworks (see section 9.1 for all the details about their general framework).

They (Big_{36} and Big_{52}) are executed solving a LP instead of IP due to their complexity as done with the both versions of MS example. It means that the results will underline both the benefits of the improvement on the local search module and on the rounding one (as happens with the medium size call center).

In this sub-section, one can see not only the different behavior of CP and CP_{new} on the same problem but also some comparisons between the two versions.

Case	Alg	n	CPU_{avg}	Min cost	Med cost	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{period}
1	CP	800	4h57m	79.74	79.91	0	0	0	0	100	0
	CP_{new} (0.8)	400	4h48m	82.02	82.20	0	0	0	0	25	0.25
	CP_{new} (0.81)	400	5h11m	81.97	82.89	0	0	0	0	100	0
2	CP	950	9h49m	80.09	80.54	0	0	0	0	50	0.07
	CP_{new} (0.8)	500	9h37m	78.87	81.80	25	25	25	25	87.5	0.15
	CP_{new} (0.81)	500	10h09m	79.79	81.51	0	0	0	0	87.5	0.11

Table 9.16: Big_{36} Example: solution quality with CP and CP_{new},varying the CPU time budget

In table 9.16, the comparisons, for the Big_{36} problem test, between the original version of CP and the new one, are reported considering two different CPU time budgets due to the complexity of the scenario: $5h$ and $10h$. It is worth noting that with a CPU time budget equal to $5h$, the original version of the method gives a result that is less expensive and at the same time satisfies all the service level constraints. Increasing the CPU time budget, instead, the new version of the method, with a target service level equal to 0.8, gives a solution whose cost is equal to 78.87 that represents the so called *empirical optimum* (i.e. a solution with a low cost and at the same time feasible for all the service level constraints). On the other hand the difference, imposing a target equal to 0.8 or 0.81, is evident analyzing, for example, their results with both CPU time budgets: in any case the second version, in fact, is able to reduce the infeasibility gap. In particular,with the first budget, $CP_{new}(0.81)$ gives all feasible solution on the contrary of $CP_{new}(0.8)$ that presents, over all the replications, a little infeasibility for the SL grouped by period(0.25%). Considering the longer CPU time budget, $CP_{new}(0.8)$ gives, again, an infeasibility gap that is higher than $CP_{new}(0.81)$ by 0.03% but in any case it gives the best feasible solution over all the methods, replications and budgets. This last fact shows that the cutting plane methodology is affected by the noise of the simulation that can be only reduced but not eliminated completely.

Moreover, for this specific problem test, the original version of CP gives solutions that are either feasible (see $5h$) or with a little infeasibility gap (see $10h$). In any case, at the end, the new version of the methodology gives the best solution, low in cost and feasible, for which one can see the effects of the new local search and the benefits of the new way to round the not integer components of the scheduling solution (as explained in details in chapter 5).

In figure 9.13, the different behaviors are shown by box-plots.

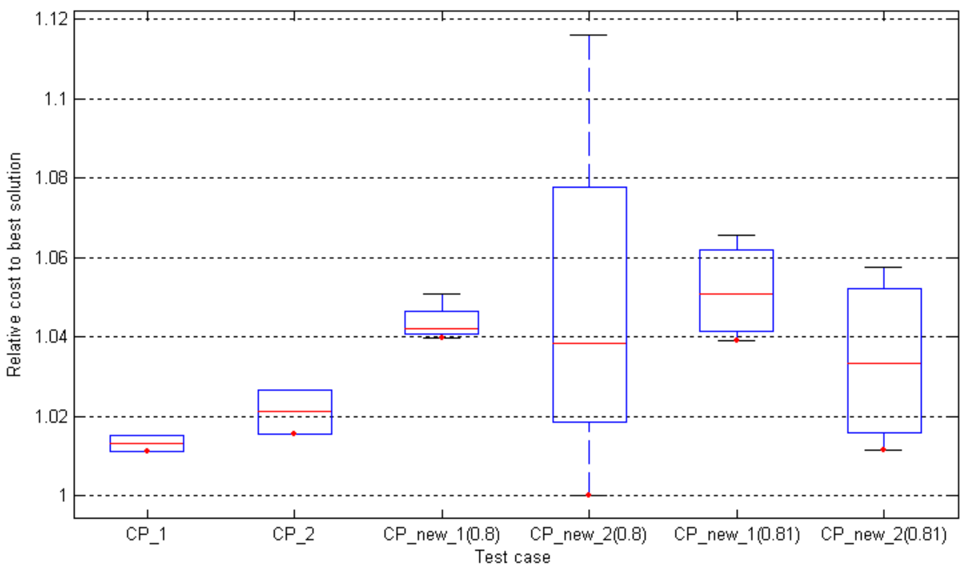

Figure 9.13: Big_{36} Example: Boxplots of optimality gap (CP vs CP_{new})

In table 9.17, the comparisons are shown for the Big_{52} example.

At the first time, one can easily note that CP is not able to give a solution less expensive and feasible at the same time as happens in Big_{36}. This underlines again as the noise of simulation affects the final results. In this case, the empirical optimum is equal to 131.7 given by $CP_{new}(0.8)$ with $5h$ CPU time budget. For all the versions, it is possible to underline that, increasing the budget there is only a small increase of the final cost.

As a reasonable consequence, again $CP_{new}(0.81)$ gives solutions that are, a little bit, higher in cost than $CP_{new}(0.8)$ but that are completely feasible

Case	Alg	n	CPU_{avg}	Min cost	Med cost	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{period}
1	CP	300	5h	136.2	137.5	0	0	0	0	50	0.41
	CP_{new} (0.8)	300	4h56m	130.8	133.6	25	50	25	50	12.5	0.54
	CP_{new} (0.81)	300	5h09m	132.1	134.85	25	25	25	25	100	0
2	CP	400	9h42m	139.2	139.9	0	0	0	0	0	0.47
	CP_{new} (0.8)	400	9h48m	133.50	137.7	0	0	0	0	50	0.44
	CP_{new} (0.81)	400	10h11m	132.30	136.8	0	0	0	0	100	0

Table 9.17: Big_{52} Example: solution quality with CP and CP_{new},varying the CPU time budget

already with a $5h$ CPU time budget.

In figure 9.14, the results are summarize by box-plots.

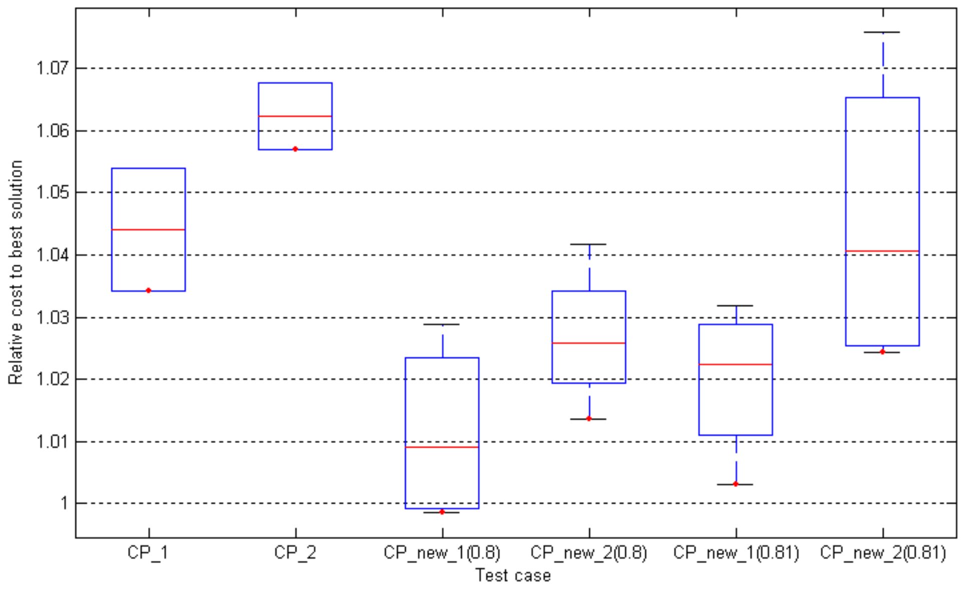

Figure 9.14: Big_{52} Example: Boxplots of optimality gap (CP vs CP_{new})

9.3 Some tests on Randomized Search initialization

In the following, considering some different versions of the problem test called as *N-Design*, it is shown the behavior of Randomized Search (RS) changing its own initialization procedure. In fact as already anticipated, RS gives different final solutions, whose quality and cost is different changing the initialization procedure.

In this section one can easily see how the final solution, with this solving approach, depends from the starting point chosen to initialize it. In fact its quality affects the future search of the method.

In the following, it will be considered only a common scenario, even if some different tests have been performed in order to figure out about the initialization method of this procedure.

9.3.1 The common scenario and its different versions

The N-Design example (see example 4.1)is considered in the following with $P = 17$ periods. Again the period length is equal to 30 minutes and in total there are 9 shifts. Each of them has 14 periods with 2 coffee-break and 1 lunch-break.

The agents costs are obtained implementing the formula 3.1 imposing $\xi = 0.2$.

The service times are all exponential and their average is equal to 4 minutes. The patience times are equal to 0 with probability 0.001 and with probability 0.999 are exponential with rate 0.1 per minute.

The different versions obtained by this generical scenario are distinguished on the basis of two factors:

- the arrival rate distribution of the two call types:
  1. *asl*: the call type 1 has an arrival rate equal about to 4 times the one of the call type 2 in all periods;
  2. *asl2*: the call type 2 has an arrival rate equal about 4 times the one of call type 1 in all periods;
  3. *eql*: the two call types have the same arrival rate in all periods.
- the routing:
  1. *npr*:the routing does not consider the priority and the agent type 2 can receive both the two call types without preferences;
  2. *pr*: the routing considers the priority and the agent type 2 gives more priority to the call type 2.

Then the total number of examples (considering the different versions described so far) is equal to 6 named as in the follow: npr.asl, npr.asl2, npr.eql, pr.asl, pr.asl2, pr.eql.

The service levels are equal to 80/20 for the ones grouped by periods and the global one. The others are equal to 50/20.

9.3.2 Results

Considering the chapter 5 in which *RS* has been presented in detail and considering the general scenario introduced in section 9.3.1, in the following some results are presented. It is very important to underline that all of these final solutions have been obtained performing 10 simulation runs and taken for each different initialization procedure the best one.

As shown in table 9.18, the best initialization method has been the so called *uniform* one. In fact it gives the best compromise between objective function

Problem name	Random	bestShift	Uniform	Complex	Fifth
npr.asl	22.77	**20.53**	21.65	25.67	22.40
npr.asl2	27.35	**24.36**	26.97	29.96	28.19
npr.eql	34.53	35.65	**34.21**	43.77	38.83
pr.asl	**19.41**	20.53	20.91	25.67	22.40
pr.asl2	24.73	24.36	**24.17**	29.96	28.19
pr.eql	33.04	35.65	**32.95**	43.77	38.83

Table 9.18: RS initialization procedure: cost comparisons (in bold the best solution for each example)

value and the computational time required to find it (see figure 9.15). For this reason, in order to compare RS with CP_{new} and TS, in the following, results will be obtained initializing it by *uniform* method.

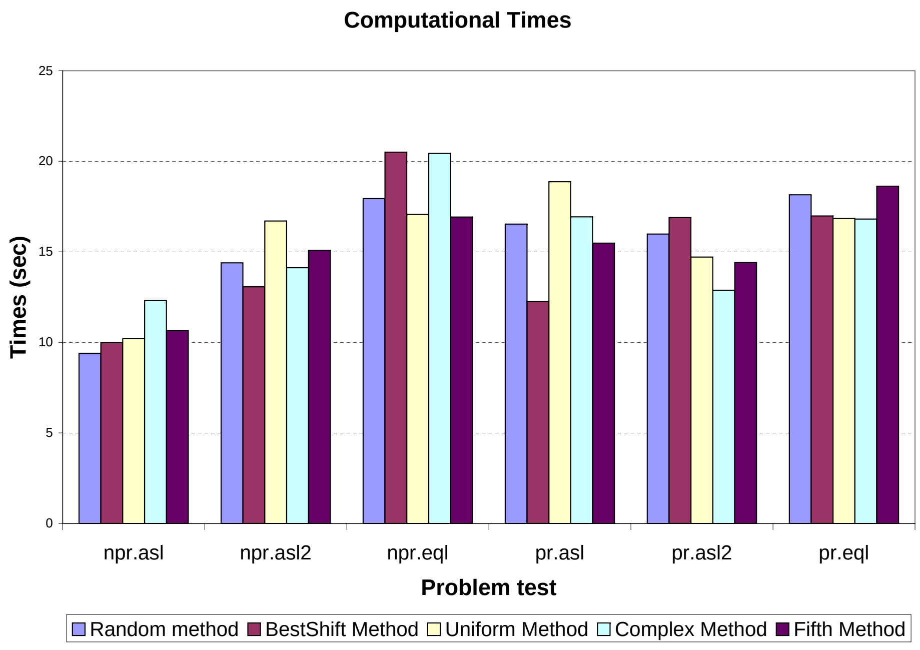

Figure 9.15: Computational times varying the RS initialization methods

9.4 Comparisons: CP_{new}, TS and RS

In this section, a comparison among CP_{new}, TS and RS is analyzed and studied in order to assess the efficient behavior of the cutting plane algorithm improved by a new local search and rounding (as already said in Chapter 5 and in section 9.1).

Again, three different sizes of call center are considered and they are the same already introduced in the section 9.2.

9.4.1 Computational Results

A small size call center: N-Design

For the general framework already introduced in section 9.2.1, in table 9.19 some comparisons (CP_{new}, TS and RS) are shown.

In the following, it should be underlined that less attention is given to RS (because of it is not able to give competitive results in term of final costs), instead much more attention is given to the two approaches: CP_{new} and TS.

It is, in fact, very clear that for all the performed simulation runs, RS gives the worst solutions (considering that the optimality gap in regard to the empirical optimum is very consistent). In any case, it should be noted that increasing the budget, RS has a reasonable behavior,because the final cost is reduced, even if it is completely affected by the random search. The little increasing passing from $15m$ to $30m$ is a reasonable consequence of the random nature of the method. Moreover, considering RS method, one has to take in account that if the method is able to obtain good results solving a single-period staffing problem (Avramidis et al. (2006)), in the case of the scheduling problem, its performances decrease because of the search space becomes larger. In fact, it has not only to decide the number of agents of each type in each period, but also to establish the assignment agent-shift in a set of $|Q|$ possibilities.

About CP_{new} it is important to underline that an IP is solved at each iteration and so at the end one can easily see the only benefits of the local search module and not of the rounding one. This method gives solutions that are almost always feasible and most of them are very good. The only case in which they are more expensive than the ones by TS, but all the TS solutions are infeasible, is when $n = 120$. It could indicate that the value chosen for the parameter n might be too small. In fact, at this point, one has to remember the meaning of this parameter: it is the number of days simulated during the cuts generation process to check the feasibility of the current solution. On the other hand, the results obtained for the three larger values of n are quite similar and setting n to 1500 is probably sufficient for this small center.

Considering again the table 9.19, one can note that not only the parameter n is reported but also the values for the parameter n_2 that is set as an input value for the method. This controls the local search module in the sense that, as already explained in chapter 5, it defines the starting value of the parameter n_1 that represents the number of simulated days used during an iteration of the local search module.

At the end of the cutting plane method, accordingly to the considerations of chapter 5, each iteration of the local search ends with a feasibility check of $max(500, n)$ simulated days.It means that in the first case (n=120) these simulations are of 500 days, in the other, they are, respectively, of 1500, 1600 and 2000. The module of local search ends when either the solution is feasible or the time limit is reached. The long simulation, performed at the end of the entire cutting plane module (including the local search one), has always a length of 50000 days.

For all the three solving methods, $nr = 32$ replications are performed.

It is not a surprise that TS method finds out that all runs fail to find a feasible solution, even though constraint violations were always inferior to 1%.

Moreover, all the solutions by TS are much more expensive than the ones obtained by CP_{new} (with a large enough value of n).

On the other hand, CP_{new} has a reasonable behavior because of increasing the value of the parameter n, passing from a budget to another one, it gives less expensive solutions.

Case	Algorithm	n	n_2	CPU_{avg} sec.	Min cost	Med cost	P_1^*	P^*	G_{period}	$G_{\text{call,period}}$
1	CP_{new}	120	50	136	36.31	36.91	0	100	0	0
	TS	120		162	35.60	35.60	0	0	0.81	0
	RS	350		180	47.02	50.05	0	50	0	0.32
2	CP_{new}	1500	800	914	35.13	35.17	100	100	0	0
	TS	1500		898	35.59	35.59	0	0	0.94	0
	RS	700		900	42	46.27	0	100	0	0
3	CP_{new}	1600	1000	1774	35.03	35.17	75	87	0	0.15
	TS	2800		1753	35.67	35.67	0	0	0.79	0
	RS	4100		1645	43.47	50.91	0	62.5	0.33	0
4	CP_{new}	2000	1000	3694	35.03	35.17	100	100	0	0
	TS	6000		3453	35.67	35.67	0	0	0.79	0
	RS	8000		3304	43	50.56	0	100	0	0

Table 9.19: N-Design: results obtained with CP_{new}, TS and RS for different CPU time budgets

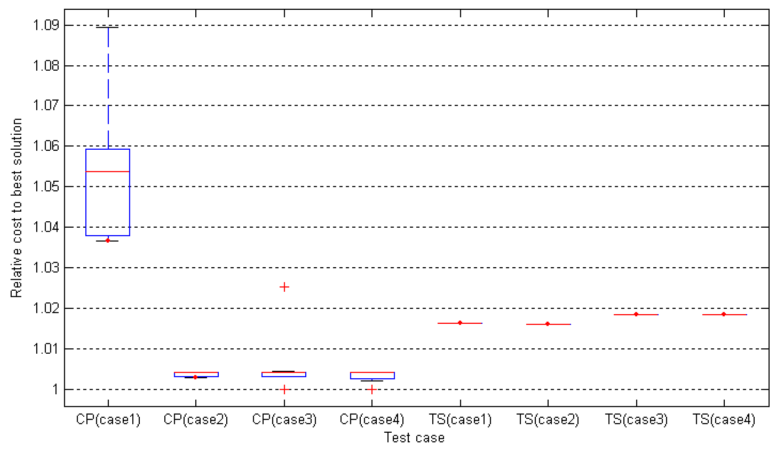

Figure 9.16: Small center: box-plots of the relative cost distribution

In figure 9.16, all the results are summarized, by box-plots, taking into account only the comparison between CP_{new} and TS.

In table 9.20, the best scheduling solutions, by CP_{new} and TS, are compared.

Algorithm	Agent type	Shift type						
		1	2	3	4	5	6	7
CP_{new}	1	6	3	3	1	1	8	2
	2	3	0	0	1	1	2	0
TS	1	6	1	3	1	2	7	1
	2	4	1	0	1	0	3	1

Table 9.20: Small center: scheduling solutions

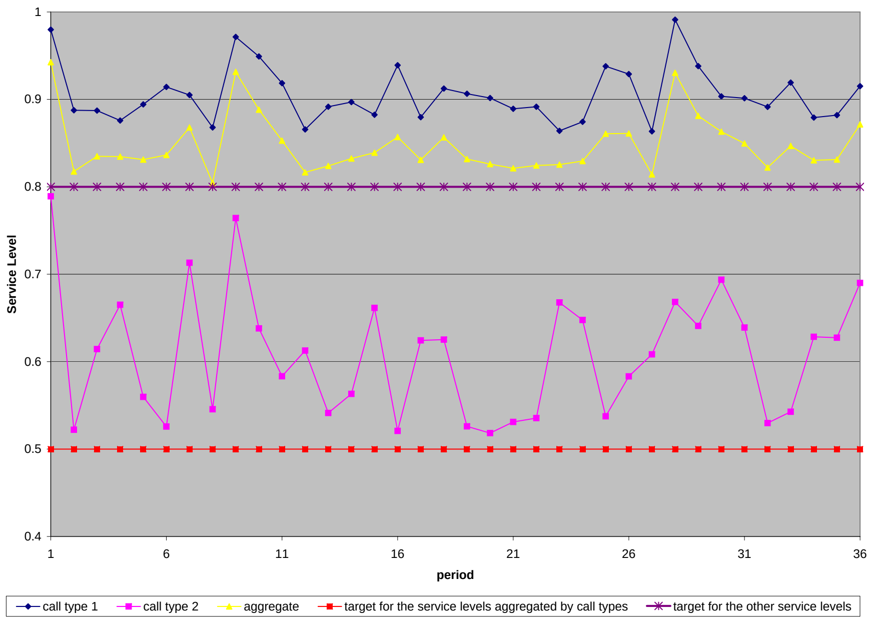

Figure 9.17: Small center: service levels by period

In particular, both methods return solutions with the same number of agents (31), most of which are specialists (type 1). They differ only slightly in terms of the duration of the shifts scheduled, but CP_{new} uses more specialists than TS (24 vs 21) and it guarantees that CP_{new} obtains a cheaper solution.

In figure 9.17, the service levels, obtained by the best run of CP_{new}, are plotted. One can see: a wide variation of the SL throughout the day and that calls of type 1 have much better SL than those of type 2. This is due to the fact that the type 1 calls can be answered by less expensive specialists, while type 2 calls must be handled by generalists. This remarks that it is often necessary to include call-type specific SL constraints (either over the whole day or for each period) in the problem formulation, if one is interested in controlling the service levels of each call type.

Two medium size call centers: MS1 and MS2

Considering the framework already described in the section 9.2 for these two problems, CP_{new} is compared to TS and RS.

Again it should be remarked that for these tests, CP_{new} solves at each iteration a LP. It means that one can note the effects of the local search and the rounding method.

Case	Algorithm	n	n_2	CPU$_{\text{avg}}$ sec.	Min cost	Med cost	P_1^*	P^*	G_{period}
1	$CPnew$	300	100	855	18.33	18.85	0	87	0.51
	TS	1200		897	21.83	21.83	0	100	0
	RS	200		850	22.51	25.87	0	25	0.64
2	$CPnew$	600	100	1598	17.56	18.43	0	75	0.41
	TS	2400		1774	21.72	21.72	0	100	0
	RS	1000		1794	24.76	26.38	0	75	0.35
3	$CPnew$	1000	400	2686	17.36	17.66	25	87	0.02
	TS	3000		2793	21.69	21.69	0	100	0
	RS	3500		3600	23.74	26.04	0	100	0

Table 9.21: $MS1$: results obtained with $CPnew$,TS and RS for different CPU time budgets

For each of the two versions of the problem, $nr = 8$ replications are performed for the CPU time budgets of 15, 30 and 60 minutes.

As already said in the previous example, RS is not able to give good results. In particular, for this framework, one can note that increasing the complexity of the problem (i.e. number of call types, agent types, periods and shifts), the randomized search has a search space larger and it is imply more difficulties to find a good solution.

The results for $MS1$ and $MS2$ are summarized in tables 9.21 and 9.24.

From these tables, one can conclude that most replications of CP_{new} return low-cost feasible solutions for both variants, but the cost variation, between TS and CP_{new} solutions, is more pronounced than for the small center.

The quality of solutions also increases significantly with n, which emphasizes the importance of performing long enough simulations in order to obtain good results. In fact, as already remarked in the previous sections, having a large value of n means performing more accurate simulations, during the cut generation process, to check feasibility. But, as happened, increasing n more than a particular threshold could not give benefits.

Contrary to what was observed for the small center, TS always finds feasible solutions, but their cost is much higher than the cost of CP solutions; in fact, the optimality gap,respect to the best solution found, is over 25% for $MS1$ and close to that value for $MS2$.

These conclusions remark that large suboptimality gaps found by TS occur in realistic call center settings as in artificial examples.

The comparison between CP_{new} and TS is also summarized by box-plot in figure 9.18 for MS1 example.

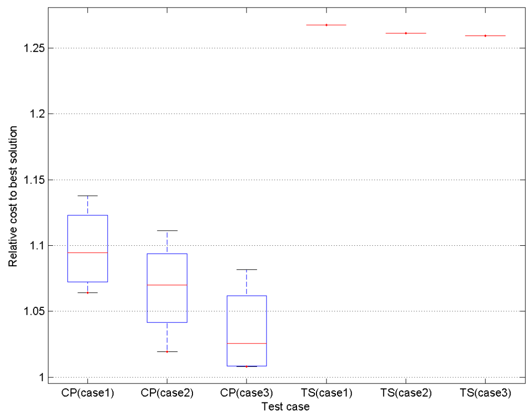

Figure 9.18: $MS1$: boxplots of the relative cost distribution

As done with the simple example, one can compare the best scheduling solutions obtained for $MS1$ with CP and TS (tables 9.22 and 9.23).

	shift types							total
	1	2	3	4	5	6	7	
agents CP	4	1	1	0	2	4	2	14
agents TS	5	1	1	2	1	6	1	17

Table 9.22: $MS1$: scheduling solutions

	agent types														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CP	1	0	1	1	1	0	1	2	0	1	1	1	2	0	2
TS	0	0	3	0	0	1	2	2	0	1	1	1	2	1	3

Table 9.23: $MS1$: total number of agents of each type in the scheduling solutions

One observes that CP_{new} not only uses significantly less agents (14 vs 17), but also that these agents have on average a smaller number of skills (2.50 vs 2.76) and work shorter hours (more than 10 minutes less a day, on average).

The results for MS2 are summarized in table 9.24. In figure 9.19 these are plotted by box-plots.

Comparing the cost of the best solutions obtained by CP_{new} for $MS1$ and $MS2$, the solution obtained for $MS1$ is cheaper (as already said in section 9.2), which is exactly what was to be expected since $MS1$ turns out to be a relaxation of $MS2$. This last conclusion seems to remark that the increase in cost, incurred when imposing the disaggregate SL constraints, is rather marginal.

The SL by period for these solutions are plotted in figure 9.20. They clearly show that the overall SL undergoes significant variations throughout the day.

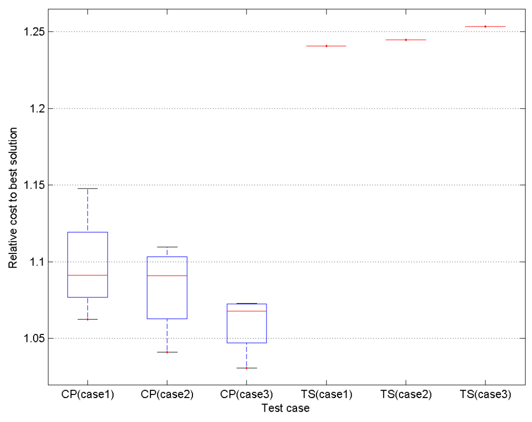

Figure 9.19: $MS2$: boxplots of the relative cost distribution

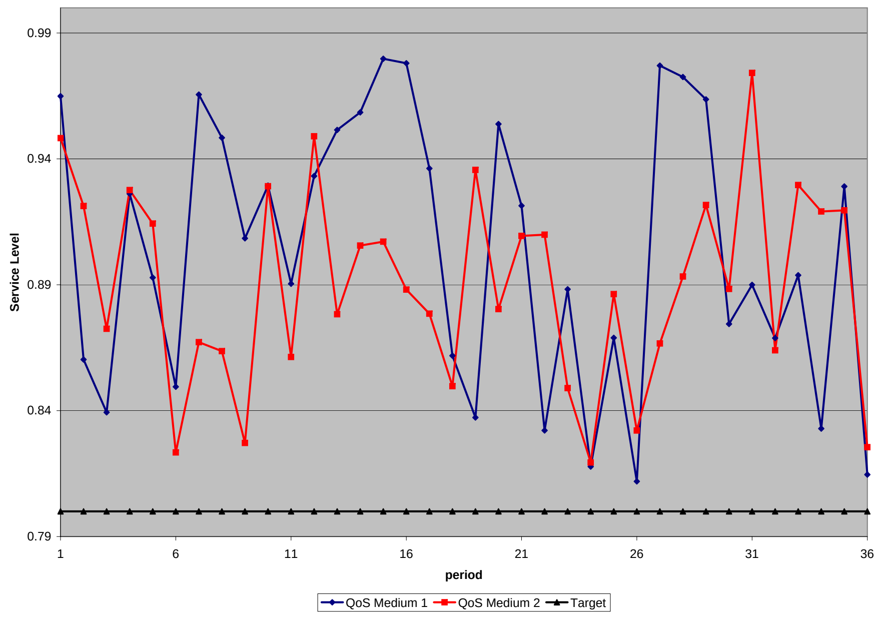

Figure 9.20: Service level by period for $MS1$ and $MS2$

Case	Algorithm	n	n_2	CPU$_{\text{avg}}$ sec.	Min cost	Med cost	P_1^*	P^*	G_{period}	$G_{\text{call,period}}$
1	CP-LP	400	100	828	18.48	18.98	0	87	0.58	0
	TS	600		876	21.58	21.58	0	100	0	0
	RS	800		870	24.62	25.03	0	100	0	0
2	CP-LP	400	100	1751	18.11	18.97	0	87	0.54	0
	TS	1500		1677	21.65	21.65	0	100	0	0
	RS	1000		1226	23.67	24.44	0	75	0.14	0
3	CP-LP	400	100	3520	17.92	18.57	0	75	0.28	0
	TS	3000		3212	21.80	21.80	0	100	0	0
	RS	3000		3281	22.95	24.26	0	87.50	0.05	0

Table 9.24: $MS2$: results obtained with CP_{new}, TS and RS varying the CPU time budget

Moreover, the two patterns observed are quite different, which highlights the importance of imposing SL constraints by type of calls.

Similar results, shown for MS1, are observed for $MS2$ in tables 9.25 and 9.26.

	shift types							total
	1	2	3	4	5	6	7	
agents CP	4	0	1	1	0	6	2	14
agents TS	3	1	1	2	1	7	2	17

Table 9.25: $MS2$: scheduling solutions

	agent types														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CP	0	1	1	0	1	1	0	3	1	0	1	1	2	1	1
TS	0	0	2	0	0	2	2	3	0	1	1	1	2	1	2

Table 9.26: $MS2$: total number of agents for each type in the scheduling solutions

Two large size call centers: Big_{36} and Big_{52}

As done in the previous sub sections, here some comparisons, among the three solving methods, are shown.

In this case, it seems that the RS method is able to give good solutions, better than TS. This result remark, again, the stochastic behavior of the method.

In any case, CP_{new} gives the best results over all the two other methods.

The general scenario that is considering in this section is the more realistic one and, again, CP_{new} solves LP instead of IP.

For this larger problem, CP_{new} has difficulty finding a feasible solution with the smaller computing budget (the first case of the table 9.27). In fact, only two

Case	Algorithm	n	n_2	CPU$_{\text{avg}}$ min.	Min cost	Med cost	P_1^*	P^*	G_{period}
1	$CP-LP$	400	50	288	82.02	82.20	0	25	0.25
	TS	1500		299	96.08	96.08	0	100	0
	RS	2000		294	87.08	94.71	0	75	0.47
2	$CP-LP$	500	50	577	78.87	81.80	25	87	0.15
	TS	2400		598	102.87	102.87	0	100	0
	RS	5000		555	87.41	95.47	0	75	0.03

Table 9.27: Big_{36}: results obtained with $CP-LP$ and TS for different CPU time budgets

runs out of 8 succeed in finding feasible solutions to the problem, even though constraint violations might not be severe. This situation is clearly alleviated by allotting more CPU time (the second case of the table 9.27). With more time, CP_{new} finds better solutions. It should be remarked that, as in the previous problem test, the final result is affected by the rounding module and, at the same time, by the local search one.

TS always finds feasible solutions within the allotted computing budget, but the solutions are on average 20% more expensive than those obtained with the cutting-plane algorithm. This confirms the initial observations regarding the poor performance of TS. Surprisingly, increasing the CPU budget does not improve the quality of the results obtained with TS, returning inferior solutions. This is, mainly, because the method obtains a different staffing solution in the first step; while this solution might track more closely the call arrival curve, it ends up leading to a poorer scheduling solution. A close examination of the relative cost distributions of solutions highlights the fact that, in the case of TS, all replications produce identical, and largely suboptimal, solutions.

These conclusions can be summarized by box-plots in figure 9.21.

In table 9.28, a comparison, between the best scheduling solution obtained by CP_{new} and the two step approach, is presented. This analysis explains the lower cost of the cutting-plane algorithm solution. In fact, resorting to slightly more agents with long shifts, CP_{new} is capable of covering the demand with only 52 agents compared to 62 for TS.

	shift types							total
	1	2	3	4	5	6	7	
agents CP	17	1	4	7	3	17	3	52
agents TS	25	4	4	4	2	17	6	62

Table 9.28: Big_{36}: scheduling solutions

Some comparisons can be done considering the variant of the Big_{36} problems with more periods (52) and they are reported in table 9.29.

On this version, a single one of the 8 solutions found by $CP-LP$ (case 1), was declared feasible by the 50,000-day simulation, but it is an excellent solution with a cost of 131.7 obtaining with the first CPU time budget.

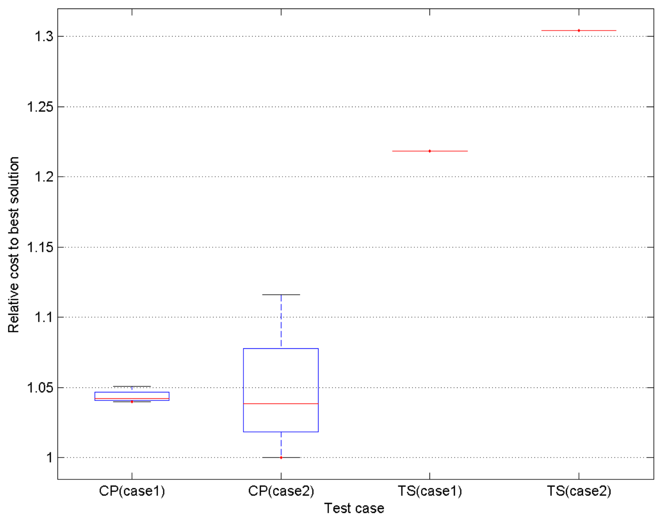

Figure 9.21: Big_{36}: boxplots of the relative cost distribution

Case	Algorithm	n	n_2	CPU$_{\text{avg}}$ minutes	Min cost	Med cost	P_1^*	P^*	G_{period}
1	$CP-LP$	300	50	296	130.8	133.6	12	12	0.54
	TS	1500		262	156.1	156.1	0	100	0
	RS	2000		263	158.3	165.55	0	25	0.23
2	$CP-LP$	400	50	588	133.5	137.7	0	50	0.44
	TS	1800		542	156.1	156.1	0	100	0
	RS	4500		580	161	173.5	0	0	0.78

Table 9.29: Big_{52}: results obtained with $CP-LP$ and TS for different CPU time budgets

With more CPU time budget, CP returned 4 feasible solutions out of 8 runs, but these turned out to be inferior to the one found in 5 hours, probably due to simulation noise.

Overall, these results emphasizes the importance of performing several trial runs when using this type of approach.

All the solutions returned by TS were declared feasible, but they are significantly more expensive, with a cost of 156.1. This is once more an example of the large suboptimality gaps produced by the TS method.

Table 9.30 reports the best solutions found by the cutting-plane algorithm and TS. In the table, agent types are regrouped by cost (i.e. to group them by number of skills, since in this variant, because all agents work 7.5-hour shifts, agent costs depend only upon their skill sets). The table gives the total number of agents of each group (each cost) in the solution. It is worth noting that the cutting plane approach employees only 96 agents compared to 107 for TS.

agent type	cost	$CP - LP$	TS
4	1.8	1	1
1,5,9,11,13	1.6	12	32
7,12,14	1.5	29	18
2,3,8,35	1.4	15	33
6,10	1.3	24	23
15,. . .,34	1.0	15	0
total number		96	107
total cost		131.7	156.1

Table 9.30: Big_{52}: a summary of the best feasible solutions found by $CP - LP$ and by TS

Moreover, several of the agents in the CP_{new} solution are specialists, while TS uses a large number of expensive generalists with 7 skills. On the contrary of TS, CP employees 15 agents of type 1.

These two factors combined motivate the large difference in cost.

Considering the final results on Big_{52}, one can easily understand the main motivation for investigating this specific variant of the problem. In fact, accordingly, it was possible to verify that the cutting plane algorithm gives good solutions in presence of instances with a different shift structure. It again confirms that the cutting-plane algorithm performances do not depend on the particular shift structure definition and for this reason it can be considered a general approach.

9.5 Adding constraints on abandonment ratios

In this paragraph, some tests are proposed introducing also the control on the abandonment probability, running the algorithm proposed in Chapter 7.

In Chapter 7, the three main performance measures (ABANDONMENTRATIO,ABANDONMENTRATIO-AFTER-AWT and ABANDONMENTRATIO-BEFORE-AWT) have been introduced and they have been controlled during the algorithm execution adding some particular cuts in the general formulation of the scheduling problem.

In order to clarify better, the meaning of these three measures of performance is reported:

- *ABANDONMENT-RATIO*: Represents the fraction of the expected number of contacts having left the system without service over the total expected number of arrivals.
- *ABANDONMENTRATIO-AFTER-AWT*: Represents the fraction of the expected number of contacts having left the system without service and after a waiting time greater than or equal to the acceptable waiting time, over the total expected number of arrivals.
- *ABANDONMENTRATIO-BEFORE-AWT*: Represents the fraction of the expected number of contacts having left the system without service and

without waiting the acceptable waiting time, over the total expected number of arrivals.

These measures, as the service levels, could be not aggregate (for each period and call type), aggregate by period, aggregate by call type and finally global (aggregate by period and call type at the same time).

In order to control their level, on all of these measures, an upper bound is imposed to the expected number of abandonments and this means to add some additional constraints to the original problem(Chapter 7). This limit has been set, in all the tests, to the value of 0.01. Such as at most 1% of the total expected number of arrivals can leave the system without service and without considering the waiting time (i.e. it could be less or greater than AWT).

9.5.1 Numerical Results

In the following, some results are shown taking into account the previous considerations.

In particular one of the already described problem tests will be considered and analyzed: N-Design example. In this example, there are only two call types ($N = \{1, 2\}$) and two agents types ($T = \{1, 2\}$), such that $S_1 = \{1, 2\}$ and $S_2 = \{1\}$. The total number of periods is equal to 36 ($P = \{1, \ldots, 36\}$) and the total number of shifts is equal to 285.

In this case the value of nr is equal to 8 for each of the different CPU time budget:$1h30m$,$2h30m$ and $3h30m$.

In particular the CPU time budgets are imposed greater than the ones of the version without constraints on the abandonments because of considering the abandonments that increase a lot the general complexity of the problem in terms, also, of computational times.

In fact the total number of constraints is increased a lot: there are $(|N|+1)*(|P|+1)$ constraints more than the previous version (i.e. without considering the abandonment probability).

For this specific case, for example, the original scheduling problem has $(3)*(37) = 111$ constraints more than its previous version.

In table 9.31, the final results, for the different CPU time budgets, are reported where the three measures are indicated, respectively, by A-R, A-A-AWT and A-B-AWT. In particular, for each of these measures and for each of CPU time budgets,the final abandonments ratios are shown. Note that even though A-A-AWT and A-B-AWT could be redundant (because they are included by the first one A-R), introducing them makes a difference in the optimization algorithm.

Firstly, it is worth nothing that the empirical optimum is found by a CPU time budget of $1h30m$ and a value of n equal to 2000. Its value is 49.41.

Increasing the CPU time budget, one can note that the abandonments ratios are reasonable decreased. In particular, focusing the attention on the unique no-zero values (column A-R: $A_{call,period}$), increasing the budget, its value is considerable decreased. The phenomena that passing from $n = 4000$ to $n = 8000$, its value passes from 0 to 8.84, is due to the stochastic nature of the algorithm.

It is important to remark that the abandonments probability control is put in the new version of the cutting plane algorithm (CP_{new}) and, in this case, the

IP is run and, as in the previous tests, a final feasibility test is executed with a 50,000-days simulation.

Again, increasing the budget, one can note a little increase on the min cost value due to the stochastic nature of the approach. What one has to consider is that, reasonably, the median cost is decreased arriving to the littlest value of 51.72 obtained by the case 3.

Again, increasing the budget, one can note a little increase on the min cost value. On the other hand the median cost is decreased arriving to the littlest value of 51.72 obtained by the case 3. So, one can affirm that the little increment of the min cost value is due only to the stochastic nature of the method.

Considering the table 9.31, one can easily note that the final cost is increased comparing it with the table 9.19. It is a reasonable result because, considering the abandonments probability too, the number of constraints is increased in the scheduling problem. As known, increasing the number of constraints implies having a smaller feasible region and so it is not possible to find a solution that is better in regard to the first case (without constraints on the abandonments). If one compares the final two costs, with the same number of n (=2000), it happens that the problem with few constraints finds a solution whose cost is 35.03, while this new version finds a cost equal to 49.41. Examining, again, the table 9.31, it is possible to remark that the violations on the abandonments probability (whose targets are all equal to 0.01) occur for the call type 2 that can be handled only by the agent type 1. This agent type is a generalist in fact he/she is able to handle both calls types, thus he/she is naturally slower than the one of type 2 than can handle only the call type 1.

In figures 9.22, 9.23 and 9.24, the trends, of the best feasible solution, are shown for each call type, considering the imposed targets (all equal to 0.01). One can observe an increment of the abandonments during the break periods and the extreme ones (at the beginning and at the end of the working day). This is a natural behavior due to the inferior number of agents during these periods.

						A-R
Budget	n	CPU_{avg}	Min Cost	Med Cost	P^*	$A_{call,period}$
1	2000	1h30m	49.41	52.40	87.5	14.89
2	4000	2h30m	50.47	51.87	100	0
3	8000	3h30m	49.67	51.72	87.5	8.84

Table 9.31: N-Design example: results with the abandonments probability control

It is important to underline that the table 9.31 only shows, for each CPU time budget, the non zero maximum violation percentage. The other cases have value equal to zero and it means no violation. In the figure 9.26 the service levels are shown.

The figure 9.25 shows the abandonments ratios of the best solution found by the cutting plane algorithm (described in Chapter 5) without controlling the abandonments. As we can easily note, almost all the abandonments ratios (of all call types) are greater than the 1%. This means that, without imposing specific cuts on the abandonments, we cannot have a directly control on them. So even though we have a good service levels, there could be an abandonment

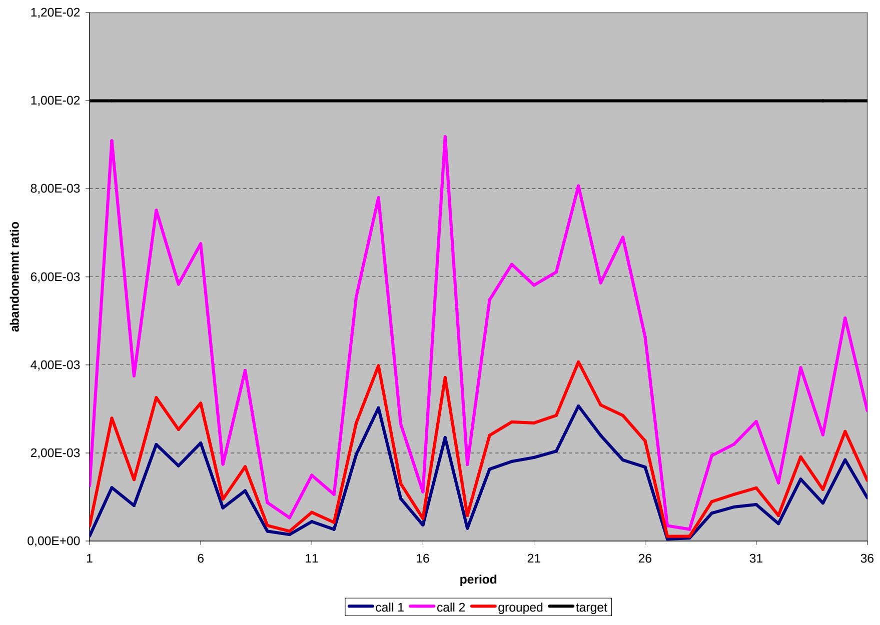

Figure 9.22: Abandonment Ratio

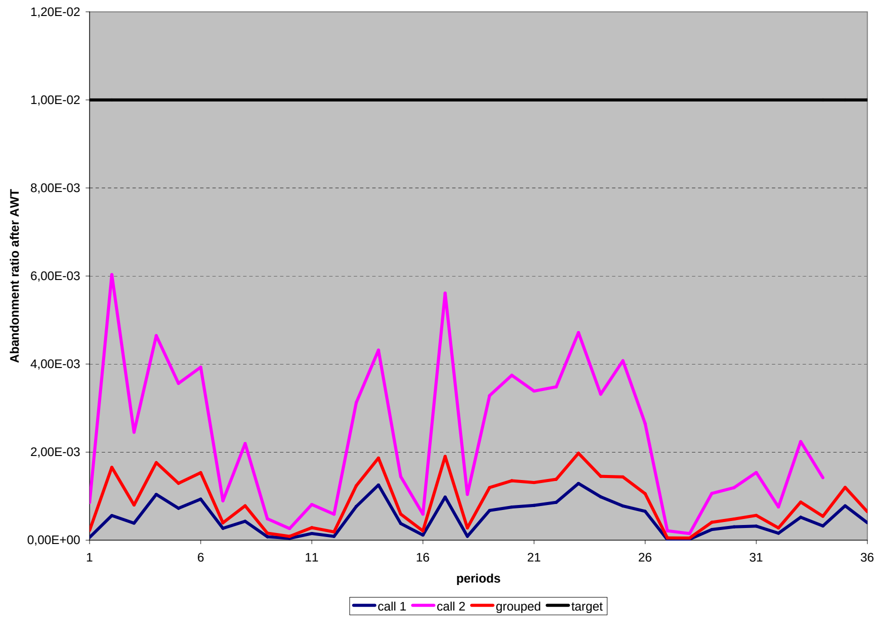

Figure 9.23: Abandonment Ratio after AWT

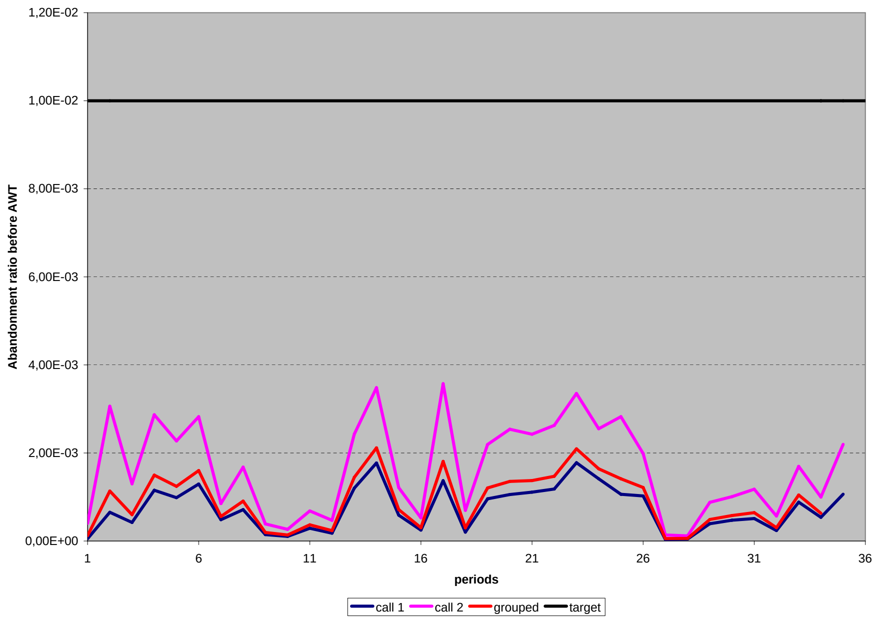

Figure 9.24: Abandonment Ratio before AWT

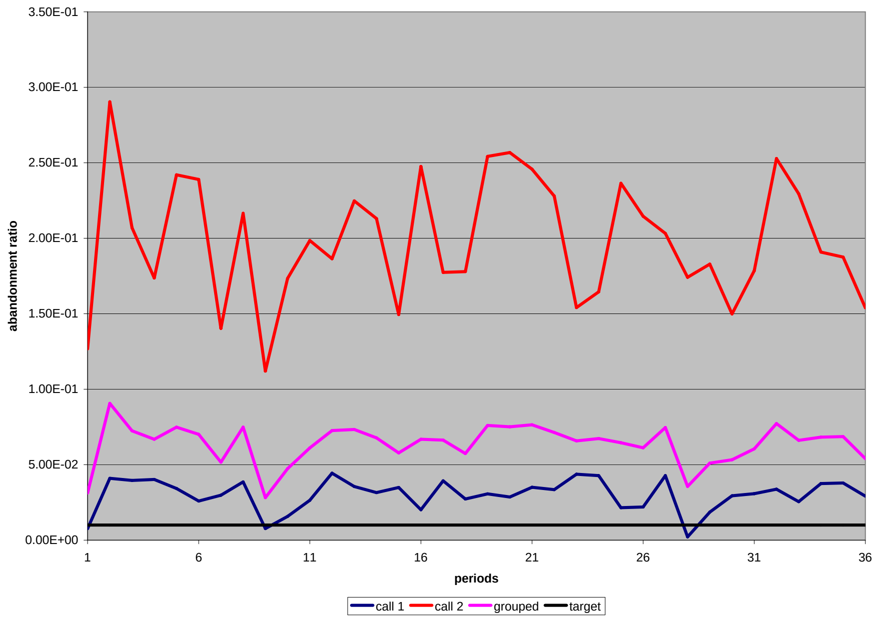

Figure 9.25: Abandonment ratios without cuts on the abandonments

ratio not equal to zero.

As expected, the service levels are increased (in regard to the version of the problem without abandonments, figure 9.17). In fact, imposing others constraints, the final solution has a greater number of agents and so a better service level. This is can be explained also considering that to reach the main goal of reducing the number of abandonments, a manager has to use more agents and so the average waiting time is reduced consequently.

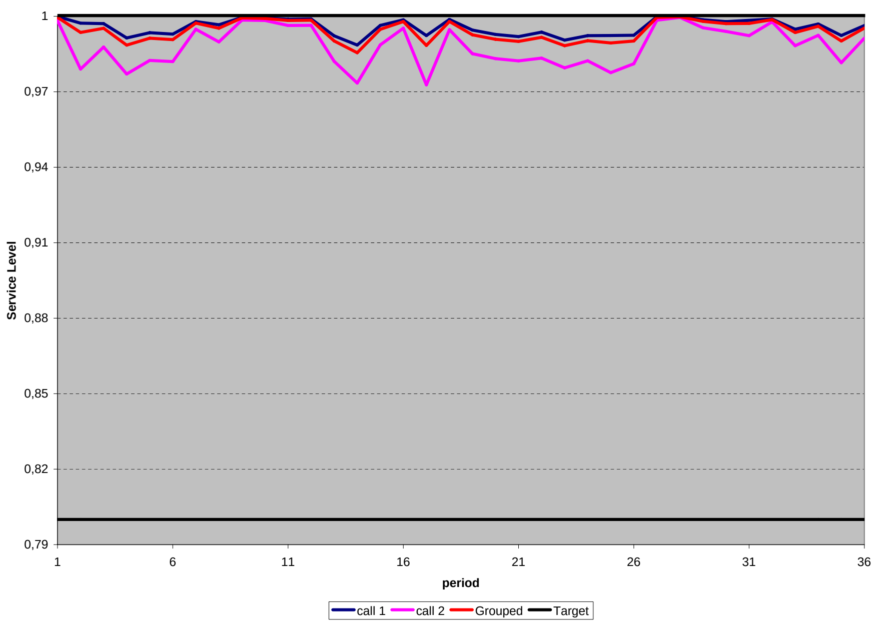

Figure 9.26: Service levels by periods with 1% abandonment ratio

In particular, as a reasonable consequence, the service levels of 1 call type is higher than the 2 call type because of it can be handled by both agent types.

Moreover, what one can observe is that adding the constraints on abandonments increases the cost by a very large percentage. As we can see in figure 9.26 the service levels are increased to a value very close to 1.

It is possible to assume, in the following, to vary the percentage of abandonment. In fact, observing the results in table 9.31, we can note to have high costs and service levels very close to 1. This implies that imposing maximum abandonment ratio equal to 1% is very unrealistic. For this reason, we perform other tests imposing maximum ratio of abandonments equal to 2% and 5%.

In tables 9.32 and 9.33, we can note that not only the final costs are lower than the ones presented in table 9.31, but we, also, get all feasible solutions.

In figures 9.27 and 9.28, the service levels, related to the best solution, found in both cases, are shown. We can note, comparing them to the ones obtained in figure 9.26, that the service levels are farer from 1. This is a reasonable behavior because we have less tight abandonments constraints and so the solutions have less agents.

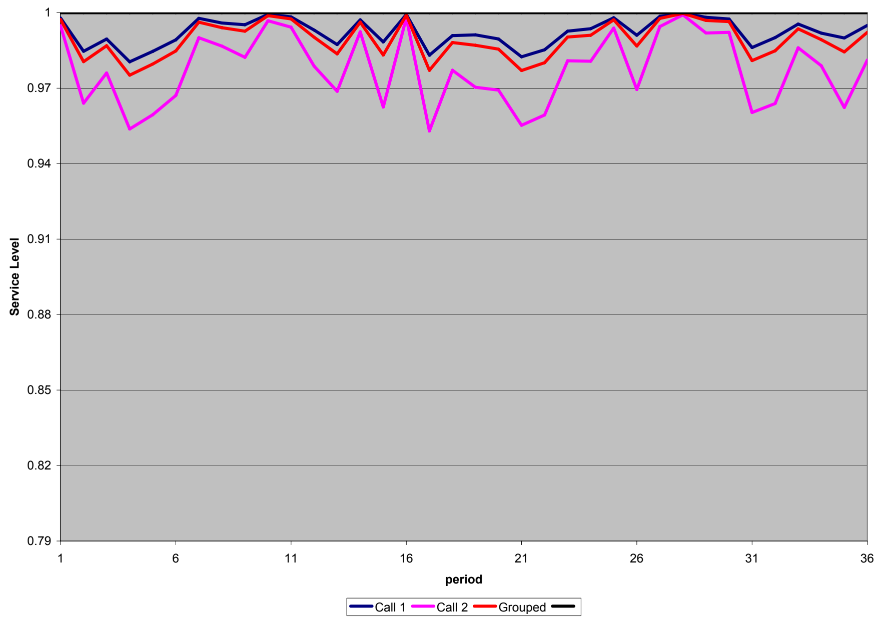

Figure 9.27: Service levels by periods with 2% abandonment ratio

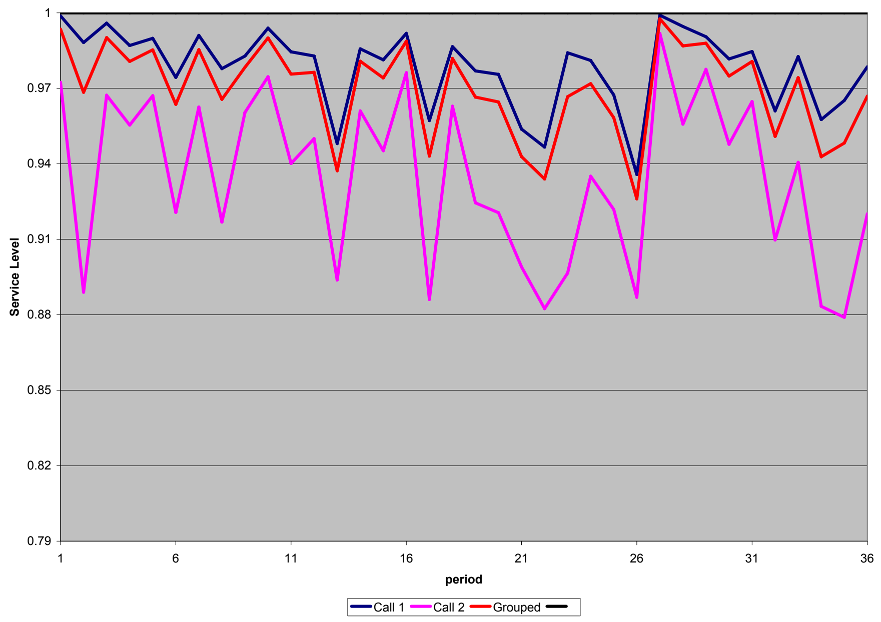

Figure 9.28: Service levels by periods with 5% abandonment ratio

						A-R
Budget	n	CPU_{avg}	Min Cost	Med Cost	P^*	$A_{call,period}$
1	2000	1h15m	48.49	49.39	100	6.95
2	4000	2h28m	49.49	49.73	100	0.80
3	8000	3h33m	47.49	49.04	100	0

Table 9.32: N-Design example: results with the maximum abandonment ration equal to 2%

						A-R
Budget	n	CPU_{avg}	Min Cost	Med Cost	P^*	$A_{call,period}$
1	2000	1h10m	46.44	49.60	100	4.60
2	4000	2h35m	45.66	49.61	100	2.59
3	8000	3h38m	43.31	48.47	100	0

Table 9.33: N-Design example: results with the maximum abandonment ration equal to 5%

9.6 Other Comparisons

Some comparisons are, finally, presented for CP_{new}, the Original Approach and the Hybrid one (called later $A1$ and $A2$) proposed in chapter 8. Remembering that these are two other possible way for solving the scheduling problem for a multi skill call center, we consider two examples already introduced in the previous sections: *N-Design* and *MS2*.

This section represents, in any case, future work for researching in new innovative algorithms.

9.6.1 N-Design and MS2

Case	Algorithm	n	CPU$_{\text{avg}}$	Min	Med	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{call}	G_{period}	$G_{\text{call,period}}$
1	CP_{new}	1600	1774	35.03	35.17	28.125	28.125	75	87.5	87.5	0	0	0.15
	A1	2000	1703	35.42	36.05	0	0	0	0	62.5	0	0.49	0.03
	A2	600	1879	34.53	34.96	25	62.5	37.5	87.5	37.5	0	0.86	0.50
2	CP_{new}	2000	3694	35.03	35.17	43.75	43.75	43.75	43.75	100	0	0	0
	A1	3000	3634	35.99	36.16	0	0	0	0	62.5	0	0.38	0.09
	A2	1100	3596	34.48	34.98	75	87.5	75	87.5	75	0	0.09	0

Table 9.34: $N - design$ results

Table 9.34 remarks the fact that CP_{new} is capable of finding the best solution (with both CPU time budgets) considering not only the final cost but also its feasibility.

In fact A2 is capable of finding a better solution in regard to the one of CP_{new}, but with more difficulties it ends with a totally feasible solution (even though the infeasibility gap is very low).

Algorithm	Agent type	Shift type						
		1	2	3	4	5	6	7
CP_{new}	1	6	3	3	1	1	8	2
	2	3	0	0	1	1	2	0
A1	1	8	0	4	2	0	9	2
	2	4	0	0	0	1	2	0
A2	1	8	2	2	2	1	8	1
	2	2	0	1	0	1	2	1

Table 9.35: *N-Design*: scheduling solutions

In particular, the approach named $A1$ (the original approach proposed in Chapter 8) does not give us good solutions because it is based on an iterative updating of some parameters and it is not possible to have a fixed rule for setting/updating them.

On the other hand, instead, the Hybrid Approach ($A2$) gives us good solutions because it solves the MPP problem only at the beginning but later it adopts a cutting plane algorithm for reaching more feasible solutions. In fact, its final solutions are almost equal to the ones found by CP_{new}.

The little cost decrement, passing from CP_{new} to $A2$, is due to a different agents arrangement to the shifts. For example, observing the table 9.35, one can see that the total number of 1 and 2 agents types is the same for both methods.

The final results, reported in table 9.34, suggest an important aspect: increasing the CPU time budget, CP_{new} is able to find more feasible solutions. For example, looking at the results with budget $b2$, CP_{new} finds all feasible solutions in the performed runs.

Case	Algorithm	n	CPU$_{\text{avg}}$	Min	Med	$P^*_{0.5}$	$P_{0.5}$	P^*_1	P_1	P^*	G_{call}	G_{period}	$G_{\text{call,period}}$
1	CP_{new}	400	1751	18.11	18.97	0	0	0	0	87.5	0	0.53	0
	A1	6000	1623	23.53	25.61	0	0	0	0	87.5	0	0.54	0
	A2	900	1881	21.40	21.84	0	0	0	0	100	0	0	0
2	CP_{new}	400	3520	17.92	18.57	0	0	0	0	75	0	0.28	0
	A1	8000	3394	23.53	25.76	0	0	0	0	100	0	0	0
	A2	900	3754	21.73	22.04	0	0	0	0	100	0	0	0

Table 9.36: $MS2$ results

	shift types							total
	1	2	3	4	5	6	7	
agents CP	4	0	1	1	0	6	2	14
agents A1	9	1	1	0	1	8	1	21
agents A2	4	3	0	0	1	8	1	17

Table 9.37: $MS2$: scheduling solutions

	agent types														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CP	0	1	1	0	1	1	0	3	1	0	1	1	2	1	1
A1	1	2	3	4	4	1	2	3	0	0	0	1	0	0	1
A2	1	0	0	1	1	0	1	2	2	0	1	2	5	1	0

Table 9.38: $MS2$: total number of agents for each type in the scheduling solutions

Examining *MS2* results and observing table 9.36, it is possible to see that CP_{new} finds better solutions (considering only the cost) but they present a little infeasibility.

The reduced costs of CP_{new} is due to the fact that it assigns, in general, few agents compared to the other methods (as shown in table 9.38).

The conclusions of this section (in particular with these two examples) should not lead us to think that both methods have to be discarded. They, in fact, represent future work on which one can define more adjustments of their implementation in order to reach (or to improve) CP_{new} performances.

Chapter 10

Conclusions

The main contribute of this PhD thesis has been the designing and implementation of a methodology for solving the scheduling problem for a multi-skill call center under the service levels constraints.

Firstly, we have showed the main weakness of a two step approach, a typical methodology proposed in literature, for solving the above problem. We have also proposed some concrete examples showing the optimality gap of the final solution obtained by this approach.

Secondly, we have detected some lower bounds on the number of shifts in the single-skill and in the multi-skill call center. Then, we have performed a series of numerical experiments to quantify, empirically, the impact of the shift length flexibility provided by considering a rich set of shift types.

Combining the new type of constraints, introduced in Bhulai et al. (2007), in order to model the skill-transfer in a multi-skill environment, with the typical constraints on the service levels, we have proposed a new scheduling problem formulation. Then we have designed and implemented a cutting plane algorithm to solve this problem overcoming the weakness of a two step approach. This methodology, extending the one already proposed in Cezik and L'Ecuyer (2006) for the single-skill call center, has been tested on a set of problem tests. We have showed that, even if the use of common random numbers reduces the simulation noise (or variance) significantly, there is still a fair amount of randomness in the solution provided by the algorithm. For example, a source of noise for the algorithm is the simulation length. In fact, due to the fact that the estimation of each subgradient requires a lot of simulations, the simulation length has to be kept short. In any case, one could run the algorithm a few times (e.g., overnight) and, considering a few solutions, retains the best.

We have also showed that, by slightly perturbing the SL targets, it is possible to overcome some of the problems related to the simulation noise. In fact, by numerical experiments, we have showed that the probability to obtain more feasible and high quality solutions increases.

In order to improve the quality of the final solution, we have designed also a local search module for decreasing the total cost exploring its neighborhood performing moves of removing and switching agents.

As another possible approach, for solving the problem, we have proposed a Randomized Search method (extending the one already designed for the single-period staffing problem in Avramidis et al. (2006)). But, by testing it on a set of instances, we have showed that the particular implementation of the method does not work well for the scheduling problem in particular when the search space becomes too large.

Comparing the three different methodologies, we have showed that the cutting plane one is able to obtain low cost solutions in reasonable computational times.

For this reason, we have also extended this approach for solving a variant of the scheduling problem in which not only the service levels constraints are imposed but also the ones on the abandonments probability.

Finally, we proposed two new solving methods taking into account different ways to model the scheduling problem by mathematical formulations.

REFERENCES

Aguir, M. S., O. Akşin, F. Karaesmen, and Y. Dallery. 2004a. The impact of retrials on call center performance. *OR Spectrum* 26 (3): 353–376.

Aguir, M. S., O. Akşin, F. Karaesmen, and Y. Dallery. 2004b. On the interaction between retrials and sizing of call centers. Technical report, Department of Industrial Engineering, Koç University.

Akşin, O., and P. Harker. 2001a. Capacity sizing in the presence of a common shared resource: Dimensioning an inbound call center. Technical report, The Wharton School, Philadelphia.

Akşin, O. Z., M. Armony, and V. Mehrotra. 2007. The modern call-center: A multi-disciplinary perspective on operations management research. *Working Paper*.

Akşin, O. Z., and P. T. Harker. 2001b. Modeling a phone center: Analysis of a multichannel, multiresource processor shared loss system. *Management Science* 47 (2): 324–336.

Andrews, B., and S. M. Cunningham. 1995. L.L. Bean improves call-center forecasting. *Interfaces* 25:1–13.

Armory, M., and C. Maglaras. 2004a. Contact centers with a call-back option and real-time delay information. *Operations Research*. To appear.

Armory, M., and C. Maglaras. 2004b. On customer contact centers with a call-back option: Customer decisions, sequencing rules, and system design. *Operations Research*. To appear.

Atar, R., A. Mandelbaum, and M. I. Reiman. 2004. Scheduling a multi class queue with many exponential servers: Asymptotic optimality in heavy traffic. *Annals of Applied Probability* 14:1084–1134.

Atlason, J., M. A. Epelman, and S. G. Henderson. 2003. Using simulation to approximate subgradients of convex performance measures in service systems. In *Proceedings of the 2003 Winter Simulation Conference*, 1824–1832: IEEE Press.

Atlason, J., M. A. Epelman, and S. G. Henderson. 2004a. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research* 127:333–358.

Atlason, J., M. A. Epelman, and S. G. Henderson. 2004b. Optimizing call center staffing using simulation and analytic center cutting plane methods. manuscript.

Avaya Communications 2001, September. Blending: The changing color of contact center productivity. `http://www1.avaya.com/enterprise/who/docs/predictivedialing/resources.html`.

Avramidis, A. N., W. Chan, and P. L'Ecuyer. 2006. Staffing multi-skill call centers via search methods and a performance approximation. Submitted.

Avramidis, A. N., A. Deslauriers, and P. L'Ecuyer. 2004. Modeling daily arrivals to a telephone call center. *Management Science* 50 (7): 896–908.

Avramidis, A. N., M. Gendreau, P. L'Ecuyer, and O. Pisacane. 2007. Optimizing daily agent scheduling in a multiskill call center. Submitted.

Avramidis, A. N., and P. L'Ecuyer. 2005. Modeling and simulation of call centers. In *Proceedings of the 2005 Winter Simulation Conference*, 144–152: IEEE Press.

Aykin, T. 1996. Optimal shift scheduling with multiple break windows. *Management Science* 42 (4): 591–602.

Baker, J. R., and K. E. Fitzpatrick. 1986. Determination of an optimal forecast model for ambulance demand using goal programming. *Journal of the Operational Research Society* 37 (11): 1047–1059.

Bapat, V., and E. B. Pruitte Jr.. 1998. Using simulation in call centers. In *Proceedings of the 1998 Winter Simulation Conference*, Volume 2, 1395–1399.

Bartholomew, D., A. Forbes, and S. McLean. 1991. *Statistical techniques for manpower planning*. 2nd ed. Wiley.

Bassamboo, A., J. M. Harrison, and A. Zeevi. 2004. Design and control of a large call center: Asymptotic analysis of an LP-based method. Manuscript, Graduate School of Business, Stanford University.

Bechtold, S. E., and L. W. Jacobs. 1990. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science* 36 (11): 1339–1351.

Belacel, N., P. Hansen, and P. L'Ecuyer. 2001, juillet. Rapport d'étape pour le projet LUB: gestion optimale d'un centre d'appels en environement "blend". Rapport fourni à Bell, 28 pages.

Bell, S., and R. Williams. 2001. Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. *Annals of Applied Probability* 11:608–649.

Berman, O., and R. C. Larson. 2000. A queueing control model for retail services having backroom operations and cross-trained workers. Technical report, Massachusetts Institute of Technology. Preprint.

Bhulai, S. 2004. Dynamic routing policies in multi-skill call centers. Technical report, Technical report 2004-11, Free University, Amsterdam.

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

Bhulai, S., G. Koole, and G. Pot. 2005. Simple methods for shift scheduling in multi-skill call centers. Technical report, Technical Report WS 2005-10, Free University, Amsterdam.

Bhulai, S., G. Koole, and G. Pot. 2007. Simple methods for shift scheduling in multi-skill call centers. Technical report, Free University, Amsterdam.

Bianchi, L., J. Jarrett, and R. C. Hanumara. 1998. Improving forecasting for telemarketing centers by ARIMA modeling with intervention. *International Journal of Forecasting* 14:497–504.

Borst, S., A. Mandelbaum, and M. Reiman. 2004. Dimensioning large call centers. *Operations Research* 52:17–34.

Borst, S. C., R. J. Boucherie, and O. J. Boxma. 1998, 31. ERMR: a generalised equivalent random method for overflow systems with repacking. In *742*, 18. ISSN 1386-3711: Centrum voor Wiskunde en Informatica (CWI).

Borst, S. C., and P. Serri. 2000. Robust algorithms for sharing agents with multiple skills. manuscript.

Brandt, A., and M. Brandt. 1999a. On the $M(n)/M(n)/s$ queue with impatient calls. *Performance Evaluation* 35 (1-2): 1–18.

Brandt, A., and M. Brandt. 1999b. A two-queue priority system with impatience and its application to a call center. *Methodology and Computing in Applied Probability* 1:191–210.

Brandt, A., M. Brandt, G. Spahl, and D. Weber. 1997. Modeling and optimization of call distribution. In *Proceedings of the 15th International Teletraffic Conference*, ed. V. Ramaswani and P. E. Wirth, 133–144: Elsevier Science.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. 2005a. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association* 100:36–50.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. 2005b. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association* 100:36–50.

Buist, E. 2005. Outils de simulation en Java pour les centres de contact. Master's thesis, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada.

Buist, E., and P. L'Ecuyer. 2005. A Java library for simulating contact centers. In *Proceedings of the 2005 Winter Simulation Conference*, 556–565: IEEE Press.

Call Center News Service 2001. Call center statistics. `http://www.callcenternews.com/`.

Ceẓik, M. T., and P. L'Ecuyer. 2006. Staffing multiskill call centers via linear programming and simulation. *Management Science*. To appear.

Chan, W. 2006. Optimisation stochastique pour l'affectation du personnel polyvalent dans un centre d'appels téléphoniques. Master's thesis, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada.

Channouf, N., P. L'Ecuyer, A. Ingolfsson, and A. N. Avramidis. 2007. The application of forecasting techniques to modeling emergency medical system calls in Calgary, Alberta. *Health Care Management Science* 10. To appear.

Chen, B. P. K., and S. G. Henderson. 2001. Two issues in setting call center staffing levels. *Annals of Operations Research* 108:175–192.

Chevalier, P., R. A. Shumsky, and N. Tabordon. 2003. Overflow analysis and cross-trained servers. *International Journal of Production Economics* 85:47–60.

Chevalier, P., R. A. Shumsky, and N. Tabordon. 2004. Routing and staffing in large call centers with specialized and fully flexible servers. Technical report, Simon Graduate School of Business, University of Rochester.

Chevalier, P., and J. Van den Schrieck. 2005. Optimizing the staffing and routing of small-size hierarchical call centers. Preprint.

Chlebus, E. 1997. Empirical validation of call holding time distribution in cellular communications systems. In *Proceedings of the 15th International Teletraffic Congress*, 1179–1188: Elsevier.

Chokshi, R. 1999. Decision support for call center management using simulation. In *Proceedings of the 1999 Winter Simulation Conference*, Volume 2, 1634–1639: IEEE Press.

Cleveland, B., and J. Mayben. 1997. Call center management—on fast forward. Preprint. Call Center Press.

Dantzig, G. B. 1954. A comment on Edie's "traffic delays at toll booths". *Operations Research* 2 (3): 339–341.

Deslauriers, A. 2003. Modélisation et simulation d'un centre d'appels téléphoniques dans un environnement mixte. Master's thesis, Department of Computer Science and Operations Research, University of Montreal, Montreal, Canada.

Deslauriers, A., J. Pichitlamken, P. L'Ecuyer, A. Ingolfsson, and A. N. Avramidis. Markov chain models of a telephone call center in blend mode. *Computers and Operations Research*. To appear.

E-Comm 2005, July. Quarterly service report. Available from `http://www.ecomm.bc.ca/corporate/publications/esr_final.pdf`.

Erdogan, G., E. Erkut, and A. Ingolfsson. 2006. Ambulance deployment for maximum survival. Technical report. Available from `http://www.bus.ualberta.ca/aingolfsson/working_papers`.

Franx, G. J., G. Koole, and A. Pot. 2006. Approximating multi-skill blocking systems by hyper-exponential decomposition. *Performance Evaluation* 63:799–824.

Fukunaga, A., E. Hamilton, J. Fama, D. Andre, O. Matan, and I. Nourbakhsh. 2002. Staff scheduling for inbound call centers and customer contact centers. *Eighteenth National Conference on Artificial intelligence*:822–829.

Gans, N., G. Koole, and A. Mandelbaum. 2003a. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management* 5:79–141.

Gans, N., G. Koole, and A. Mandelbaum. 2003b. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management* 5:79–141.

Gans, N., and Y.-P. Zhou. 2001, September. A call-routing problem with service-level constraints. Technical report, The Wharton School, University of Pennsylvania, Philadelphia. Preprint.

Gans, N., and Y.-P. Zhou. 2002. Managing learning and turnover in employee staffing. *Operations Research* 50:991–1006.

Gans, N., and Y.-P. Zhou. 2004. Overflow routing for call center outsourcing. Preprint.

Garnett, O., and A. Mandelbaum. 2000. An introduction to skill-based routing and its operational complexities. manuscript.

Garnett, O., A. Mandelbaum, and M. Reiman. 2002. Designing a call center with impatient customers. *Manufacturing and Service Operations Management* 4 (3): 208–227.

Goldberg, J. B. 2004. Operations research models for the deployment of emergency service vehicles. *EMS Management Journal* 1:20–39.

Green, L. V., and P. Kolesar. 1991. The pointwise stationary approximation for queues with nonstationary arrivals. *Management Science* 37 (1): 84–97.

Green, L. V., and P. Kolesar. 1997. The lagged PSA for estimating peak congestion in multiserver Markovian queues with periodic arrival rates. *Management Science* 43:80–87.

Green, L. V., and P. J. Kolesar. 2004. Improving emergency responsiveness with management science. *Management Science* 50:1001–1014.

Grossman Jr., T. A., S. L. Oh, T. R. Rohleder, and D. A. Samuelson. 2000. Call centers. In *The Encyclopedia of Operations Research and Management Science* (second ed.)., ed. S. I. Gass and C. M. Harris, 73–76. New York: Kluwer Academic Publishers.

Gulati, S. 2001. Call center scheduling technology evaluation using simulation. In *Proceedings of the 2001 Winter Simulation Conference*, Volume 2, 1438–1442.

Gunes, E., and R. Szechtman. 2005. A simulation model of a helicopter ambulance service. In *Proceedings of the 1992 Winter Simulation Conference*, 951–957: IEEE Press.

Halfin, S., and W. Whitt. 1981. Heavy-traffic limits for queues with many exponential servers. *Operations Research* 29:567–588.

Harris, C., K. Hoffman, and P. Saunders. 1987. Modeling the irs telephone taxpayer information system. *Operations Research* 35:504–523.

Harrison, J. M., and M. J. Lopez. 1999. Heavy-traffic resource pooling in parallel-server systems. *Queueing Systems* 33:339–368.

Harrison, J. M., and A. Zeevi. 2004. Dynamic scheduling of a multi-class queue in the Halfin-Whitt heavy-traffic regime. *Operations Research* 52:243–257.

Harrison, J. M., and A. Zeevi. 2005. A method for staffing large call centers based on stochastic fluid models. *Manufacturing and Service Operations Management* 7 (1): 20–36.

Henderson, S., and A. Mason. 1998. Rostering by iterating integer programming and simulation. In *Proceedings of the 1998 Winter Simulation Conference*, Volume 1, 677–683.

Henderson, S. G., and A. J. Mason. 2004. Ambulance service planning: simulation and data visualization. In *Handbook of Operations Research and Health Care Methods and Applications*, ed. F. Sainfort, M. L. Brandeau, and W. P. Pierskalla, 77–102. Boston: Kluwer Academic.

Henderson, S. G., A. J. Mason, I. Ziedins, and R. Thomson. 1999. A heuristic for determining efficient staffing requirements for call centres. Technical Report 594, School of Engineering, University of Auckland, NZ.

Hoffman, K. L., and C. M. Harris. 1986. Estimation of a caller retrial rate for a telephone information system. *European Journal of Operational Research* 27 (2): 207–214.

Ingolfsson, A., E. Akhmetshina, S. Budge, Y. Li, and X. Wu. 2005. A survey and experimental comparison of service level approximation methods for non-stationary $M/M/s$ queueing systems. Technical report, School of Business, University of Alberta, Edmonton, Alberta, Canada.

Ingolfsson, A., E. Cabral, and X. Wu. 2003. Combining integer programming and the randomization method to schedule employees. Technical report, School of Business, University of Alberta, Edmonton, Alberta, Canada. Preprint.

Ingolfsson, A., E. Erkut, and S. Budge. 2003. Simulation of single start station for Edmonton EMS. *Journal of the Operational Research Society* 54:736–746.

Jennings, O. B., A. Mandelbaum, W. A. Massey, and W. Whitt. 1996. Server staffing to meet time-varying demand. *Management Science* 42 (10): 1383–1394.

Jiménez, T., and G. Koole. 2004. Scaling and comparison of fluid limits of queues applied to call centers with time-varying parameters. *OR Spectrum* 26:413–422.

Jones, S. A., M. P. Joy, and J. Pearson. 2002. Forecasting demand of emergency care. *Health Care Management Science* 5:297–305.

Jongbloed, G., and G. Koole. 2001. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry* 17:307–318.

Kamenetsky, R. D., L. J. Shuman, and H. Wolfe. 1982. Estimating need and demand for prehospital care. *Operations Research* 30:1148–1167.

Kelley, J. 1960. The cutting plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics* 8(4):703712.

Klungle, R. 1999. Simulation of a claims call center : a succss and a failure. In *Proceedings of the 1999 Winter Simulation Conference*, Volume 2, 1648–1652.

Koole, G. 2001. Mathematical modeling of call centers. Technical report, Department of Stochastics, Vrije Universiteit, Amsterdam.

Koole, G. 2003. Redefining the service level in call centers. Technical report, Department of Stochastics, Vrije Universiteit, Amsterdam.

Koole, G. 2004. A formula for tail probabilities of Cox distributions. *Journal of Applied Probability* 41 (3). To appear.

Koole, G. 2005. Call center mathematics. In preparation.

Koole, G., and A. Mandelbaum. 2002. Queueing models of call centers: An introduction. *Annals of Operations Research* 113:41–59.

Koole, G., and A. Pot. 2005. Approximate dynamic programming in multi-skill call centers. In *Proceedings of the 2005 Winter Simulation Conference*: IEEE Press.

Koole, G., and A. Pot. 2006. An overview of routing and staffing algorithms in multi-skill customer contact centers. Submitted version.

Koole, G., A. Pot, and J. Talim. 2003. Routing heuristics for multi-skill call centers. In *Proceedings of the 2003 Winter Simulation Conference*, 1813–1816: IEEE Press.

Koole, G., and J. Talim. 2000. Exponential approximation of multi-skill call centers architecture. In *Proceedings of QNETs*, 23/1–10.

Koole, G., and H. J. van der Sluis. 1998. An optimal local search procedure for manpower scheduling in call centers. Technical report, Vrije Universiteit, Amsterdam.

Koole, G., and H. J. van der Sluis. 2003. Optimal shift scheduling with a global service level constraint. *IIE Transactions* 35:1049–1055.

Kort, B. 1983. Models and methods for evaluating customer acceptance of telephone connections. In *GLOBECOM '83*, 706–714: IEEE.

Larson, R. C. 1974. A hypercube queuing model for facility location and redistricting in urban emergency services. *Computers & Operations Research* 1 (1): 67–95.

Larson, R. C. 1975. Approximating the performance of urban emergency service systems. *Operations Research* 23 (5): 845–868.

L'Ecuyer, P. 2006. Modeling and optimization problems in contact centers. In *Proceedings of the Third International Conference on Quantitative Evaluation of Systems (QEST'2006)*, 145–154. University of California, Riversdale: IEEE Computing Society.

L'Ecuyer, P., and E. Buist. 2005. Simulation in java with ssj. In *Proceedings of the 2005 Winter Simulation Conference*, 611–620: IEEE Press.

L'Ecuyer, P., and E. Buist. 2006. Variance reduction in the simulation of call centers. In *Proceedings of the 2006 Winter Simulation Conference*, 604–613: IEEE Press.

Mabert, V. A. 1985. Short interval forecasting of emergency phone call (911) work loads. *Journal of Operations Management* 5 (3): 259–271.

Mandelbaum, A. 2003. Call centers: Research bibliography with abstracts. Downloadable from `http://iew3.technion.ac.il/serveng/References/references.html`.

Mandelbaum, A. 2006. Call centers: Research bibliography with abstracts. Version 7, downloadable from `http://iew3.technion.ac.il/serveng/References/references.html`.

Mandelbaum, A., W. Massey, and M. I. Reimann. 1998. Strong approximations for Markovian service networks. *Queueing Systems* 30:149–201.

Mandelbaum, A., and M. I. Reimann. 1998. On pooling in queueing networks.

Management Science 44:971–981.

Mandelbaum, A., and N. Shimkin. 2000. A model for rational abandonments from invisible queues. *Queueing Systems: Theory and Applications* 36 (1-3): 141–173.

Mandelbaum, A., and A. L. Stolyar. 2004. Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized c-mu rule. *Operations Research* 52:836–855.

Mandelbaum, A., and S. Zeltyn. Service engineering in action: The Palm/Erlang-A queue, with applications to call centers. Manuscript, downloadable from `http://iew3.technion.ac.il/serveng/References/references.html`.

Mason, A. J., D. M. Ryan, and D. M. Panton. 1998. Integrated simulation, heuristic and optimization approaches to staff scheduling. *Operations Research* 46 (2): 161–175.

McConnell, C. E., and R. W. Wilson. 1998. The demand for prehospital emergency services in an aging society. *Social Science & Medicine* 46 (8): 1027–1031.

Mehrotra, A., K. E. Murphy, and M. A. Trick. 2000. Optimal shift scheduling: A branch-and-price approach. *Naval Research Logistics* 47 (3): 185–200.

Mehrotra, V. 1997, October. Ringing up big business. *ORMS Today* 24 (4): 18–24.

Mehrotra, V. 1999, August. Call center simulation enhances planning and performance. *Call Center Management Review*.

Mehrotra, V., and J. Fama. 2003. Call center simulation modeling: Methods, challenges, and opportunities. In *Proceedings of the 2003 Winter Simulation Conference*, 135–143: IEEE Press.

Mieghem, J. A. V. 1995. Dynamic scheduling with convex delay costs: the generalized *cmu* rule. *Annals of Applied Probability* 5:809–833.

Mieghem, J. A. V. 1998. Investment strategies for flexible resources. *Management Science* 44:1071–1078.

Milner, P. C. 1997. Ten-year follow-up of arima forecasts of attendances at accident and emergency departments in the trent region. *Statistics in Medicine* 16:2117–2125.

Moeller, A. 2004. Obstacles to measuring emergency medical service performance. *EMS Management Journal* 1:8–15.

National Fire Protection Association 2002. NFPA 1221: Standard for the installation, maintenance, and use of emergency service communication systems.

Ormeci, E. L. 2004. Dynamic admission control in a call center with one shared and two dedicated service facilities. *IEEE Transactions on Automatic Control*. to appear.

Palm, C. 1943. Intensitätsschwankungen im fernsprechverkehr. *Ericsson Technics* 44:1–189.

Pichitlamken, J., A. Deslauriers, P. L'Ecuyer, and A. N. Avramidis. 2003. Modeling and simulation of a telephone call center. In *Proceedings of the 2003 Winter Simulation Conference*, 1805–1812: IEEE Press.

Repede, J. F., and J. J. Bernardo. 1994. Developing and validating a decision support system for location emergency medical vehicules in Louisville, Kentucky. *European Journal of Operational Research* 75:567–581.

Ridley, A. D., M. C. Fu, and W. A. Massey. 2003. Fluid approximations for a priority call center with time-varying arrivals. In *Proceedings of the 2003*

Winter Simulation Conference, 1817–1823: IEEE Press.

Rockafellar, R. 1970. Convex analysis. *Princeton, NJ: Princeton University Press*.

Saltzman, R. M., and V. Mehrotra. 2001. A call center uses simulation to drive strategic change. *Interfaces* 31 (3): 87–101.

Samuelson, D. A. 1999. Call attempt pacing for outbound telephone dialing systems. *Interfaces* 29 (5): 66–81.

Schoen, F. 2006. *Modelli di ottimizzazione per le decisioni*. 1nd ed. Pogetto Leonardo-Esculapio.

Sharp, D. 2003. *Call center operation: Design, operation, and maintenance*. 1nd ed. Digital Press.

Shen, H., and J. Huang. 2007. Interday forecasting and intraday updating of call center arrivals. *Working Paper*.

Shumsky, R. A. 2004. Approximation and analysis of a call center with flexible and specialized servers. *OR Spectrum* 26:307–330.

Soyer, R., and M. Tarimcilar. 2007. Modeling and analysis of call center arrival data: A bayesian approach. *Management Science, forthcoming*.

Spaite, D. W., E. A. Criss, T. D. Valenzuela, and J. Guisto. 1995. Emergency medical service systems research: Problems of the past, challenges of the future. *Annals of Emergency Medicine* 26 (2): 146–152.

Steckley, S. G., S. G. Henderson, and V. Mehrotra. 2004. Service system planning in the presence of a random arrival rate. submitted.

Swersey, A. J. 1994. The deployment of police, fire and emergency medical units. In *Handbooks in Operations Research and Management Science*, ed. S. M. Pollock, M. Rothkopf, and A. Barnett, Volume 6, 151–190. New York: North-Holland.

Tanir, O., and R. J. Booth. 1999. Call center simulation in Bell Canada. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. A. Farrington, H. B. Nemhard, D. T. Sturrock, and G. W. Evans, 1640–1647. Piscataway, New Jersey: IEEE Press. Available on line via `www.informs-cs.org`.

Thompson, G. M. 1995. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science* 41 (4): 595–607.

Tych, W., D. J. Pedregal, P. C. Young, and J. Davies. 2002. An unobserved component model for multi-rate forecasting of telephone call demand: The design of a forecasting support system. *International Journal of Forecasting* 18:673–695.

van Dijk, N. M. 2000. On hybrid combination of queueing and simulation. In *Proceedings of the 2000 Winter Simulation Conference*, Volume 1, 147–150.

Wallace, R. B., and W. Whitt. 2005. A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management* 7 (4): 276–294.

Weinberg, J., L. D. Brown, and J. R. Stroud. 2007. Bayesian forecasting of an inhomogeneous poisson process with applications to call center data. *Journal of the American Statistical Association, forthcoming*.

Whitt, W. 1999a. Dynamic staffing in a telephone call center aiming to immediately answer all calls. *Operations Research Letters* 24:205–212.

Whitt, W. 1999b. Improving service by informing customers about anticipated delays. *Management Science* 45 (2): 192–207.

Whitt, W. 1999c. Partitioning customers into service groups. *Management Science* 45 (11): 1579–1592.

Whitt, W. 2004a. A diffusion approximation for the G/GI/n/m queue. *Operations Research* 6:922–941.

Whitt, W. 2004b. Engineering solution of a basic call-center model. *Management Science*. To appear.

Whitt, W. 2004c. Fluid models for many-server queues with abandonments. *Operations Research*. To appear.

Whitt, W. 2004d. Sensitivity of performance in the Erlang a model to changes in the model parameters. manuscript.

Whitt, W. 2004e. Staffing a call center with uncertain arrival rate and absenteeism. manuscript.

Zhu, Z., M. A. McKnew, and J. Lee. 1992. Effects of time-varied arrival rates: an investigation in emergency ambulance service systems. In *Proceedings of the 1992 Winter Simulation Conference*, 1180–1186: IEEE Press.

Zohar, E., A. Mandelbaum, and N. Shimkin. 2002. Adaptive behavior of impatient customers in tele-queues: Theory and emperical support. *Management Science* 48:566–583.