CHALMERS
UNIVERSITY OF TECHNOLOGY

Optimal Scheduling of Customer Service Personnel in a Call Centre

Master's thesis in Engineering Mathematics and Computational Science

ARBER VOKSHI

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Optimal Scheduling of Customer Service Personnel in a Call Centre

Arber Vokshi

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2023

Optimal Scheduling of Customer Service Personnel in a Call Centre

Arber Vokshi

Supervisors: Ann-Brith Strömberg, Caroline Granfeldt
Examiner: Ann-Brith Strömberg

Master's Thesis 2023
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Optimal Scheduling of Customer Service Personnel in a Call Centre

Arber Vokshi
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This project has studied the ability to schedule staff in customer service at Telia Sverige AB using a well-known optimisation algorithm called 'column generation'. Two mathematical models have been constructed, one for obtaining the amount of shifts required to satisfy the given demand and to which the column generation algorithm is applied, the other for distributing those shifts among employees. Also, one way of generating shifts is presented in detail, involving repeatedly solving a shortest path problem for a weighted, directed graph. The objective of this project was to see how well a more optimisation-based approach compares to the already in-use system at the company. After obtaining the results it was concluded that an optimisation-based approach works quite well for problems of this nature, but that additional real-world parameters need to be taken into account before the model can be used in a call centre.

Keywords: Mathematical optimisation, column generation, scheduling problem, call centre optimisation, shortest path problem.

Acknowledgements

I wish to thank Professor Ann-Brith Strömberg for her excellent tutoring throughout this project, as well as deepening further my knowledge in the field of mathematical optimisation. I also thank PhD. student Caroline Granfeldt for her much needed assistance during the early stages of the project with introducing me to the programming language Julia and also the Gurobi solver. Finally, I thank Hannes Jansson, Product Manager at Telia Sverige AB, for his faith in this project, his guidance regarding the currently in-place systems for staffing management, as well as for providing the necessary data that made this project possible.

My sincere thanks to all of you, hopefully our paths will cross again in the future.

Arber Vokshi, Borås, October 16, 2023

Contents

1

Introduction

In our current era, where technology and digitisation have grown larger than ever before, many businesses have become quite large as compared to a few decades ago. This raises the need to implement a way of maintaining effective communication channels between organisations and their customers, so that desirable public relations are upheld. A large portion of the business-to-customer communication is in the form customer service via call centres. With increasing customer expectations and growing competition, call centres face the challenge of managing resources efficiently to ensure high-quality customer service while minimising operational costs. Mathematical models derived through optimisation theory have proven quite effective at addressing problems of this nature, and have made it possible to improve efficiency, while minimising costs, in many different businesses.

This thesis considers, specifically, a scheduling problem for a call centre at Telia Sverige AB. For telecom-companies in particular, the most significant aspect that differentiates one organisation from another is the quality of their customer service. This is because most of the time, quite similar services in (usually) similar price ranges are offered and since the same technologies and hardware are used for providing this there is also little difference in performance. Therefore, it is of utmost importance that the customer service holds a high enough standard so that customers will, in this case, choose Telia over their competitors. A largely contributing factor to the general customer satisfaction is the company's ability to provide the help needed as quickly and efficiently as possible. One way to achieve this is by making sure that the time a customer has to wait for support is minimised. Hence it is important that the work-schedules of customer-support personnel is optimised toward an as good balance as possible between demand and supply of support. The aim of this thesis is to present, and solve, mathematical models describing the problem at hand, and also provide a schedule for every given employee so that a 'demand-supply' balance is achieved. The best solution found should then be compared to the results generated by the current system at the company to decide whether an optimisation based approach is a worthwhile investment.

1.1 Problem definition

The task is to find an optimal, or close to optimal, allocation of staff at Telia such that under- and over staffing is minimised, with respect to defined constraints. This section includes a description of the current system at the company, the structure of the shifts, and finally, a suggestion of a new system based on optimisation modelling.

1.1.1 Current system

In order to fully grasp the problem it is beneficial to understand how the current system at the company works and what is already being done. Telia offers quite a wide range of services, which naturally results in customer support being divided into different categories to cover these. Each support category is called a ‘queue’ and depending on a given agent’s training and experience they may be assigned one or several (in some cases all) queues, also called ‘competencies’. The vast majority of customer support is conducted via telephone calls from customers describing their problem and receiving direct help. There are also some chat and e-mail cases, but these are quite few relative to the number of calls. Every telephone used for taking calls from customers is directly linked to a computer software called ‘ACE’, which every agent logs in to (on a work-computer) at the start of their shift. When a customer calls the support-line they are met by an automated answering machine that places them into an appropriate queue that is best suited for the matter at hand. Since every agent uses a personal log-in to ACE, the system knows which queues they are assigned to and distributes calls accordingly. Furthermore, ACE salvages data from every telephone in the office such as total number of calls (as well as accepted/unaccepted calls) for a given time interval, length of every call (including a recording of the call itself), time between calls (at the end of each call an agent has some time to write a few notes that may be of use in the future), when and how long breaks/lunch breaks are taken, etc.

Relevant data that is gathered by ACE gets fed into the scheduling software called ‘Calabrio WFM’. The data then goes through a validation process that removes values that deviate significantly from historical data. After the validation procedure, the validated data can be used to predict the number of future calls by assuming that they arrive according to a Poisson process. The prediction is inserted into the ‘Erlang A’ formula in order to calculate agent demand. Details of the Erlang A definition is out of scope for this project and is thus omitted; the interested reader is referred to Erlang A in [1], along with the references therein. In addition to the forecast number of calls, the formula uses parameters such as 'Service Level' (percentage of calls answered), ´Maximum Occupancy' (maximum percentage of time spent handling calls) and 'Customer abandonment rate' (percentage of calls abandoned by customers before being answered), in order to obtain a more accurate estimate. Other things to account for are holidays, sick-leave, training, meetings, etc. which is done by further increasing the demand of staff by use of a 'Shrinkage factor'. The cost per hour is assumed to be the same for all agents, regardless of experience. For more details regarding how Calabrio specifically uses the Erlang formula to derive agent demand, see [2].

1.1.2 Shifts

Customer support at Telia, the setting studied in this project, requires staff Monday to Friday from 8:00 a.m. to 4:00 p.m. For a full time employee the requirement is 160 hours of work per planning period, which is four weeks; this averages to 40 hours per week but there is generally no weekly work-hour requirement (one can, for example, work 39 hours one week and 41 hours another week within the current planning period). The shifts can be varied quite freely, however, they are subject to some constraints (or often reoccurring patterns) which are the following:

- Shifts $\geq$ 6h in length should include one break in the morning, one lunch break, and one break in the afternoon.
- Shifts $\leq$ 4h should include only one break and no lunch break.
- Shifts $\geq$ 4h 15min but $\leq$ 5h 45min should include one break in the morning and one lunch break.
- The time between breaks should be at least 1h 15min, but no longer than 3h.
- The first break cannot start earlier than 45min from shift start, but no later than 2h 45min.
- The final 45min of a shift must include no breaks.
- Shifts can be 3h the shortest and 8h the longest.

All shifts can vary in start time during the day as long as they remain within opening hours, but an agent can work no more than one shift per day. Currently, all possible combinations of shifts are generated and distributed among agents so that the demand over the planning period is met. The smallest time-unit used when measuring data (e.g. length of call or demand) and constructing schedules are 15-minute time steps, meaning a shift is only valid if it can be divided into 15-minute parts. Normally, a break is 15 minutes long (one time step) and lunch break 30–45 minutes (2–3 time steps). The 30 minute lunch breaks primarily occur for the shifts that are $\geq$ 4h 15min but $\leq$ 5h 45min in length - the longer shifts usually have longer lunch breaks.

1.1.3 A suggestion of a new system based on optimisation modelling

The Erlang A model has proven to work quite well in many cases and as suggested in [3, Ch. 7], using Erlang A is generally a wise decision when scheduling a call centre. However, as with any model or system, some drawbacks are naturally present, for example the model tends to consistently underestimate the amount of agents needed [3, Ch. 7].

What if a more optimisation based approach were to be taken instead? Could a suitable optimisation model be better (or at least as good) at predicting how much staff is actually needed, and perhaps also distribute shifts among agents more efficiently?

1.2 Previous work

Scheduling problems have been prevalent for quite some time and is highly relevant in any form of business with employees, and so, there have been many research studies conducted regarding this topic that present many different ways of attacking this type of problem.

Cordone et al., 2011, presented in their article *Optimization of Multi-skill Call Centers Contracts and Work-shifts* [4] a mixed-integer model that minimises a weighted sum of employee under- and overstaffing, and lacking-/surplus of skills of the employees, both of which are also weighted in relation to their respective importance. Their model takes various aspects into consideration, such as contract types and lunch breaks, as well as making sure that a good balance of contracts is achieved. The solution method implemented is based on solving the linear relaxation of the model and fixing some of the relevant integer variables each iteration, until an integer solution is found.

Bard and Purnomo, 2005, have in their paper *Preference scheduling for nurses using column generation* [5], presented a model for scheduling nurses, and then solving it using column generation. Column generation has proven to be quite powerful when solving scheduling problems, which is also suggested in [5], where the authors in most cases were able to find solutions to problems with up to 100 nurses and planning horizons up to 6 weeks in minutes. Simiar methods have also been used in [6], where Zhao et al., 2018, *A Column Generation Based Algorithm for Airline Cabin Crew Rostering Problem*, have proposed a column generation-based algorithm in order to solve a rostering problem for airline crew, where the subproblems used to generate columns each iteration takes the form of a shortest path problem in a weighted graph. Their algorithm has been put to use in a major airline in China, with good performance.

Another agent scheduling problem, rather similar to this project, is considered by Um-in and Tharmmaphornphilas, in their conference paper *Agent Scheduling of Call Center Using Decomposition Technique* [7], where they essentially have split up the problem into two parts. The first part considers the minimum amount of agents required to satisfy the predetermined demand over a one month-period. The second part uses the solution from the first part as input in order to assign shifts to individual agents. Having this splitting allows for evaluating the possibility that perhaps not all agents that are available are actually needed (or worth hiring, in regards to cost) to cover the demand. This was in fact the case in [7], where there were a total of 179 agents available, but only 162 were required to reach the optimal solution.

As shown in [4] and [7] for example, there are ways of solving scheduling problems that do not directly make use of the column generation scheme, depending on the specific problem. However due to the general nature of scheduling problems, for example that their formulation tend to be mixed integer, as well as the number of variables easily reach a large amount, it comes as no surprise that column generation is one of the more reoccurring strategies in the literature, since it has been proven to work well when solving large integer-constrained linear problems [8], and seems to yield satisfactory results when used in problems of this nature.

In the journal article *Preventive maintenance scheduling of multi-component systems with interval costs* [9] authored by Gustavsson et al., 2014, a scheduling problem of the maintenance of components in a system is presented as a shortest path problem in a directed graph, with the objective of minimising the sum of all set-up and interval costs. It turns out that this specific model is quite applicable to the subproblem-part of this project in the column generation method due to the interchangeability between the system components and the available agents, as well as component maintenance and agent breaks, see Section 3.2.2 for more details.

This project will take on the approach of splitting the problem into two parts in a similar way as was done in [7], so that the amount of required agents versus available agents can be evaluated (see Sections 3.2.1 and 3.2.3). For the column generation subproblem, the idea of solving a shortest path problem in a weighted graph presented in [6], applied as a generalisation to the PMSPIC method presented in [9] is implemented as a method of generating columns for each iteration in the column generation scheme (see Section 3.2.2).

1.3 Outline

The outline of this text is as follows: Following the introduction in Chapter 1, Chapter 2 presents relevant theory regarding mathematical optimisation, particularly focused on linear programming. Chapter 3 presents the mathematical models, as well as the methods used to solve the scheduling problem at hand. Furthermore, Chapter 4 includes all tests and results and compares some of them to the corresponding results at the company. Finally, Chapter 5 provides conclusions and discussions about the results and concludes by suggesting a few topics of further research.

2

Theory

This chapter aims to present relevant theory so that the reader is equipped with the necessary tools in order to follow the process described in the rest of the text. If the reader is already familiar with optimisation theory, LP duality, the column generation method and multi-objective optimisation, this chapter can be skipped. Most of the content in this chapter comes from [10], [11] and [12, Ch. 6] - the interested reader is referred to those texts for more details regarding these topics. Section 3.2.2 includes rather rudimentary levels of graph-theory and will therefore not be included here, the reader is referred to the following Britannica article [13] for a brief summary of relevant topics in graph-theory.

2.1 Mathematical optimisation, linear programming and LP duality

The word 'optimum' means 'the ultimate ideal', hence the term 'optimise' refers to finding the 'ultimate state' of the matter in question. Optimisation is a field of study in applied mathematics that strives to find the optimum to a given problem by constructing and solving a mathematical model. Such a model normally consists of

- **Sets**: unique sets of objects relevant for the problem, e.g., the set of all cities, vehicle types, or people.
- **Parameters**: data given by the problem formulation, usually constants such as the price of vehicle types, or distances between cities.
- **Objective**: a function that defines the goal, e.g. to minimise the total cost or to maximise profit.
- **Decision variables**: something that can be varied (or decided upon), so that the objective reaches its optimum. The variable values that yield the optimum are called *the optimal solution*.

- **Constraints**: expressions that limit the combinations of possible values that the variables can take.

Generally, an optimisation problem can be formulated as to

$$
\begin{aligned}
\text{minimise} \quad & f(\boldsymbol{x}), && (2.1a)\\
\text{subject to} \quad & g_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \dots, m, && (2.1b)\\
& \boldsymbol{x} \in X, && (2.1c)
\end{aligned}
$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ denotes the objective function (in this case to be minimised), X denotes the region (or set) of allowed $\boldsymbol{x}$-values, also called a *feasible* set, and g_i denotes the constraints that restrict the feasible set further. Thus the optimal solution $\boldsymbol{x}^* \in X$ that, per definition, also satisfies (2.1b) is feasible and yields the minimum value of f, which is $f(\boldsymbol{x}^*)$ (also denoted f^*).

Sometimes the problem (2.1) may prove to be too hard to solve directly due to the constraints being overly restrictive. One option is then to *relax* some or all of the constraints in (2.1b) so that the feasible set becomes larger, and thus it will be easier to find an optimal solution:

$$
\begin{aligned}
f^* &:= \underset{\boldsymbol{x}}{\operatorname{minimum}} \{ f(\boldsymbol{x}) \mid \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}^m;\ \boldsymbol{x} \in X \} \equiv \underset{\boldsymbol{x}}{\operatorname{minimum}} \{ f(\boldsymbol{x}) \mid \boldsymbol{x} \in S \}, && (2.2a)\\
f^*_{\text{relax}} &:= \underset{\boldsymbol{x}}{\operatorname{minimum}} \{ f(\boldsymbol{x}) \mid \boldsymbol{x} \in S_{\text{relax}} \}, && (2.2b)
\end{aligned}
$$

where $f, f_{\text{relax}} : \mathbb{R}^n \mapsto \mathbb{R}$, $\boldsymbol{g} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $X \subset \mathbb{R}^n$, $m, n \in \mathbb{Z}_+$, and $S := \{\boldsymbol{x} \in X \mid \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}^m\}$.

Theorem 1 (The relaxation theorem [10])**.** *Assume that $f_{relax}(\boldsymbol{x}) \leq f(\boldsymbol{x})$ holds for all $\boldsymbol{x} \in S$, and $S \subseteq S_{relax}$. The following holds*

(i) ***[relaxation]*** *f_{relax} is a relaxation of f i.e. the inequality $f^*_{relax} \leq f^*$ holds.*

(ii) ***[infeasibility]*** *If* (2.2b) *is infeasible then* (2.2a) *is also infeasible.*

(iii) ***[optimal relaxation]*** *If* (2.2b) *has an optimal solution $\boldsymbol{x}^*_{relax}$, such that $\boldsymbol{x}^*_{relax} \in S$ and $f_{relax}(\boldsymbol{x}^*_{relax}) = f(\boldsymbol{x}^*_{relax})$ hold, then $\boldsymbol{x}^*_{relax}$ is also optimal in* (2.2a)*.*

Proof. (i): Because $S \subseteq S_{\text{relax}}$ and $f_{\text{relax}} \leq f(\boldsymbol{x})$ holds for all $\boldsymbol{x} \in S$, we get

$$
f^*_{\text{relax}} = \min_{\boldsymbol{x} \in S_{\text{relax}}} f_{\text{relax}}(\boldsymbol{x}) \leq \min_{\boldsymbol{x} \in S} f_{\text{relax}}(\boldsymbol{x}) \leq \min_{\boldsymbol{x} \in S} f(\boldsymbol{x}) = f^* \qquad (2.3)
$$

and the result follows.

(ii): Follows from $S \subseteq S_{\text{relax}} = \emptyset$.

(iii): By insertion into (2.3) we get

$$f^*_{\text{relax}} = f_{\text{relax}}(\boldsymbol{x}^*_{relax}) = \min_{\boldsymbol{x} \in S_{\text{relax}}} f_{\text{relax}}(\boldsymbol{x}) = \min_{\boldsymbol{x} \in S} f_{\text{relax}}(\boldsymbol{x}) = f(\boldsymbol{x}^*_{\text{relax}}) = f^* \tag{2.4}$$

and the result follows. ■

Definition 1 (Linear program)**.** *A* ***linear program*** *(LP) is a special subset of optimisation models that have the following general notation*

$$\begin{aligned} &\textit{minimum} && \boldsymbol{c}^\mathsf{T}\boldsymbol{x}, && \text{(2.5a)}\\ &\textit{subject to} && A\boldsymbol{x} \geq \boldsymbol{b}, && \text{(2.5b)}\\ & && \boldsymbol{x} \in X_{LP}, && \text{(2.5c)} \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n$*,* $A \in \mathbb{R}^{m \times n}$*,* $\boldsymbol{b} \in \mathbb{R}^m$*, and* m*,* $n \in \mathbb{Z}_+$*. The set* X_{LP} *is a non-empty and bounded polyhedron.*

Definition 2 (Lagrangean duality applied to a linear program)**.** *Consider* (2.5)*, we relax the affine (linear) constraints in* (2.5b) *so that an easier problem is obtained. The relaxed constraints are added to the objective with a cost vector* $\boldsymbol{u}$ *such that the act of breaking one or more constraints is penalised:*

$$\begin{aligned} &\underset{\boldsymbol{x}}{\textit{minimum}} && \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}(\boldsymbol{b} - A\boldsymbol{x}), && \text{(2.6a)}\\ &\textit{subject to} && \boldsymbol{x} \in X_{LP}. && \text{(2.6b)} \end{aligned}$$

Let $\boldsymbol{x}^*(\boldsymbol{u})$ *be optimal in* (2.6)*, rewriting we get the* ***Lagrangean dual function*** $h_{LP} : \mathbb{R}^m \mapsto \mathbb{R}$

$$\begin{aligned} h_{LP}(\boldsymbol{u}) = \min_{\boldsymbol{x} \in X_{LP}} \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}(\boldsymbol{b} - A\boldsymbol{x}) &= \boldsymbol{b}^\mathsf{T}\boldsymbol{u} + \min_{\boldsymbol{x} \in X_{LP}} \boldsymbol{x}^\mathsf{T}(\boldsymbol{c} - A^\mathsf{T}\boldsymbol{u})\\ &= \boldsymbol{b}^\mathsf{T}\boldsymbol{u} + \boldsymbol{x}^*(\boldsymbol{u})^\mathsf{T}(\boldsymbol{c} - A^\mathsf{T}\boldsymbol{u}). \end{aligned} \tag{2.7}$$

The ***Lagrangean dual problem*** *is defined as follows:*

$$h^*_{LP} \;:=\; \underset{\boldsymbol{u} \in \mathbb{R}^m_+}{\textit{maximum}} \quad h_{LP}(\boldsymbol{u}). \tag{2.8}$$

Let z^*_{LP} *be the optimal objective value to* (2.5)*.*

Theorem 2 (Weak duality [10])**.** *The inequality* $h_{LP}(\boldsymbol{u}) \leq z^*_{LP}$ *holds for any* $\boldsymbol{u} \in \mathbb{R}^m_+$*.*

Proof. Let $\boldsymbol{x}$ be feasible in (2.5), then

$$h_{LP}(\boldsymbol{u}) = \min_{\boldsymbol{y} \in X_{\text{LP}}} \{\boldsymbol{c}^\mathsf{T}\boldsymbol{y} + \boldsymbol{u}^\mathsf{T}(\boldsymbol{b} - A\boldsymbol{y})\} \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{u}^\mathsf{T}(\boldsymbol{b} - A\boldsymbol{x}) \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x},$$

where the first inequality comes from the inclusion $\boldsymbol{x} \in X_{LP}$ and the second due to $A\boldsymbol{x} \geq \boldsymbol{b}$ and $\boldsymbol{u} \geq \boldsymbol{0}$. ■

Remark. *Every optimal solution $\boldsymbol{x}^*$ to (2.5) is also feasible and so it follows that $h_{LP}(\boldsymbol{u}) \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^* = z^*_{LP}$.*

Definition 3 (Convex set)**.** *The set $S \in \mathbb{R}^n$ is convex if*

$$\begin{matrix} x_1, x_2 \in S \\ \lambda \in [0,1] \end{matrix} \implies \lambda x_1 + (1-\lambda)x_2 \in S. \tag{2.9}$$

Definition 4 (Convex hull)**.** *The convex hull of a finite set $V = \{v^1, \ldots, v^k\}$ is defined as*

$$conv\, V := \left\{ \lambda_1 v^1 + \ldots + \lambda_k v^k \,\middle|\, \lambda_1, \ldots, \lambda_k \geq 0,\ \sum_{i=1}^{k} \lambda_i = 1 \right\}, \tag{2.10}$$

hence the convex hull is defined by the set of all convex combinations of the k points.

Definition 5 (Extreme points)**.** *An extreme point to a convex set is a point that cannot be described by a convex combination of any other points in the set. Hence a convex set is defined by its extreme points.*

Remark. *Because the set X_{LP} is a polyhedron, it is also convex. If we let $\boldsymbol{x}^q$ denote an extreme point to X_{LP} and Q_{LP} an index set for all such points, we can express X_{LP} in the following manner utilising the convex hull notation:*

$$X_{LP} = \underset{q \in Q_{LP}}{conv}\ \{\boldsymbol{x}^q\} := \left\{ \boldsymbol{x} \in \mathbb{R}^n \,\middle|\, \sum_{q \in Q_{LP}} \boldsymbol{x}^q \lambda_q;\ \sum_{q \in Q_{LP}} \lambda_q = 1;\ \lambda_q \geq 0,\ q \in Q_{LP} \right\}. \tag{2.11}$$

Theorem 3 (Strong duality [10])**.** *Let z^*_{LP} and h^*_{LP} be the optimal values of (2.5) and (2.8) respectively. It holds that*

$$z^*_{LP} = h^*_{LP}. \tag{2.12}$$

Proof.

$$
\begin{aligned}
h^*_{LP} &= \max_{\boldsymbol{u}} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + \min_{q \in Q_{LP}} \left\{ (\boldsymbol{c} - A^\top \boldsymbol{u})^\top \boldsymbol{x}^q \right\} \middle| \boldsymbol{u} \in \mathbb{R}^m_+ \right\} \\
&= \max_{\boldsymbol{u},\nu} \left\{ \boldsymbol{b}^\top \boldsymbol{u} + \nu \middle| (A\boldsymbol{x}^q)^\top \boldsymbol{u} + \nu \leq \boldsymbol{c}^\top \boldsymbol{x}^q,\ q \in Q_{LP};\ \boldsymbol{u} \in \mathbb{R}^m_+;\ \nu \in \mathbb{R} \right\} \\
&= \min_{\boldsymbol{\lambda}} \left\{ \sum_{q \in Q_{LP}} (\boldsymbol{c}^\top \boldsymbol{x}^q)\lambda_q \middle| \sum_{q \in Q_{LP}} (A\boldsymbol{x}^q)\lambda_q \geq \boldsymbol{b};\ \sum_{q \in Q_{LP}} \lambda_q = 1;\ \lambda_q \geq 0, q \in Q_{LP} \right\} \\
&= \min_{\boldsymbol{x},\boldsymbol{\lambda}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \middle| A\boldsymbol{x} \geq \boldsymbol{b};\ \boldsymbol{x} = \sum_{q \in Q_{LP}} \lambda_q \boldsymbol{x}^q;\ \sum_{q \in Q_{LP}} \lambda_q = 1;\ \lambda_q \geq 0, q \in Q_{LP} \right\} \\
&= \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \middle| A\boldsymbol{x} \geq \boldsymbol{b};\ \boldsymbol{x} \in X_{LP} \right\} = z^*_{LP}
\end{aligned} \tag{2.13}
$$

■

2.1.1 Basic solutions and complementary slackness

Consider the LP

$$
\begin{aligned}
\text{minimum} \quad & \boldsymbol{c}^\top \boldsymbol{x}, & (2.14a) \\
\text{subject to} \quad & A\boldsymbol{x} \geq \boldsymbol{b}, & (2.14b) \\
& \boldsymbol{x} \geq 0. & (2.14c)
\end{aligned}
$$

From now on, we refer to (2.14) as the primal problem (P) and its corresponding dual problem (D).

Definition 6 (Basic solution)**.** *A basic solution $\bar{\boldsymbol{x}}$ is a solution that satisfies the following:*

- $A\bar{\boldsymbol{x}} = \boldsymbol{b}$.
- *The columns of A corresponding to non-zero components of $\bar{\boldsymbol{x}}$ are linearly independent.*

Definition 7 (Basic feasible solution)**.** *A* ***basic feasible solution*** *(BFS) is a basic solution that is feasible i.e. it also satisfies $\bar{\boldsymbol{x}} \geq 0$.*

Remark. *The non-zero components of $\bar{\boldsymbol{x}}$ are referred to as basic variables $\boldsymbol{x}_B$ and, conversely, the zero components non-basic variables x_N, in fact*

$$
\bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{pmatrix},\ A = (B \quad N),\ A\bar{\boldsymbol{x}} = B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b} \implies \bar{\boldsymbol{x}} = \begin{pmatrix} B^{-1}\boldsymbol{b} \\ \boldsymbol{0}^{n-m} \end{pmatrix}. \tag{2.15}
$$

Theorem 4 (Complementary slackness [11]). *Let $\boldsymbol{x}$ be a feasible solution to (P) and $\boldsymbol{u}$ a feasible solution to (D). Then*

$$\left.\begin{array}{l}\boldsymbol{x} \text{ optimal to (P)}\\ \boldsymbol{u} \text{ optimal to (D)}\end{array}\right\} \Longleftrightarrow x_j(c_j - A_{\cdot j}^\mathsf{T}\boldsymbol{u}) = 0, \quad j = 1, \ldots, n, \tag{2.16}$$

where $A_{\cdot j}$ is the j'th column of A.

Proof. If $\boldsymbol{x}$ and $\boldsymbol{u}$ are feasible then

$$\boldsymbol{c}^\mathsf{T}\boldsymbol{x} \geq (A^\mathsf{T}\boldsymbol{u})^\mathsf{T}\boldsymbol{x} = \boldsymbol{u}^\mathsf{T}A\boldsymbol{x} = \boldsymbol{b}^\mathsf{T}\boldsymbol{u}. \tag{2.17}$$

Furthermore, by weak- and strong duality $\boldsymbol{x}$ and $\boldsymbol{u}$ are optimal if and only if $\boldsymbol{c}^\mathsf{T}\boldsymbol{x} = \boldsymbol{b}^\mathsf{T}\boldsymbol{u}$, meaning that in fact

$$\boldsymbol{c}^\mathsf{T}\boldsymbol{x} = (A^\mathsf{T}\boldsymbol{u})^\mathsf{T}\boldsymbol{x} \Longleftrightarrow \boldsymbol{x}^\mathsf{T}(\boldsymbol{c} - A^\mathsf{T}\boldsymbol{u}) = 0. \tag{2.18}$$

Since $\boldsymbol{x} \geq \boldsymbol{0}^n$ and $A^\mathsf{T}\boldsymbol{u} \leq \boldsymbol{c}$ we see that $\boldsymbol{x}^\mathsf{T}(\boldsymbol{c} - A^\mathsf{T}\boldsymbol{u}) = 0$ is equivalent to each term in the sum being zero and we are done. ■

2.2 Column generation

One issue that could present itself when dealing with LP models is that they may become too large to handle i.e. the amount of variables are quite many, which might result in a high computational complexity. A way of tackling this problem is to realise that all variables do not necessarily contribute to the optimal solution; only the basic variables do, the rest (non basic variables) are zero. This means that in fact only a subset of variables are needed to find the solution and so, given a feasible solution to start with, one can iteratively generate new variables (or *columns*) that improve the objective until this is no longer possible and we have found the optimal solution. The method is called *Column generation* and is a widely used technique in order to appropriately attack a large problem.

Consider the model (2.5), which can also be expressed as

$$z^*_{LP} = \text{minimum} \quad \sum_{j=1}^{n} c_j x_j, \quad (MP), \tag{2.19a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} \boldsymbol{a}_j x_j \geq \boldsymbol{b}, \tag{2.19b}$$

$$\boldsymbol{x} \in X_{LP}, \tag{2.19c}$$

where $\boldsymbol{a}_j \in \mathbb{R}^m$ and $\boldsymbol{b} \in \mathbb{R}^m$. We assume that n is 'large'. We call (2.19) the *Master problem* (MP) and thus form the *Restricted master problem* (RMP) by restricting the number of columns in the MP to $n_r \leq n$, in which we end up with

$$
\begin{aligned}
z_r^* = \text{minimum} \qquad & \sum_{j=1}^{n_r} c_j x_j, \qquad (RMP), && \text{(2.20a)}\\
\text{subject to} \qquad & \sum_{j=1}^{n_r} \boldsymbol{a}_j x_j \geq \boldsymbol{b}, && \text{(2.20b)}\\
& \boldsymbol{x} \in X_{LP}, && \text{(2.20c)}
\end{aligned}
$$

which is a *restriction* of the MP, thus $z_r^* \geq z_{LP}^*$, with $n_r \leq n$. Let the optimal solution to the RMP be $\boldsymbol{x}_r^*$, then the dual optimal solution, by complementary slackness (see Theorem 4), is $\boldsymbol{u}_r^* = (\boldsymbol{c}_B^\mathsf{T} B^{-1})^\mathsf{T}$ where $B \subseteq A$ are the columns corresponding to the optimal basic feasible solution. New columns are generated by choosing one that gives a negative reduced cost on the basic variables (i.e. will improve the objective value). The reduced costs is computed as: $\bar{c}_j = c_j - \boldsymbol{u}^\mathsf{T} \boldsymbol{a}_j$ for $j \notin B$. Once there are no variables left that give a negative reduced cost we have enough of them to obtain the optimal solution and can thus discard the rest.

Let P be the restricted set of columns used and $B \subseteq P$ be an optimal basis for this restricted problem with optimal dual variable $\boldsymbol{u}_r^*$. In essence, the column generation algorithm repeatedly solves the subproblem:

$$
\bar{c}_p = \underset{j \notin P}{\text{minimum}} \; \{c_j - \boldsymbol{u}_r^{*\mathsf{T}} \boldsymbol{a}_j\} \tag{2.21}
$$

and adds the optimal column (if there exists one) $\begin{pmatrix} c_p \\ \boldsymbol{a}_p \end{pmatrix}$ such that $\bar{c}_p < 0$, and $P := P \cup \{p\}$. A summary of the column generation method is given in Algorithm 1. More detailed information can be found in [8].

Algorithm 1: Column generation

1. Define the set $P := \emptyset$.
2. Find a feasible solution to (2.19), and add the columns used in the solution to P.
3. Solve (2.20), using the columns in P.
4. Obtain dual vector $\boldsymbol{u}_r^*$, of optimal solutions to the LP-dual of (2.20).
5. Using $\boldsymbol{u}_r^*$, find the column with the smallest negative reduced cost and add it to P.
6. Repeat steps 3.–5. until there are no columns with negative reduced cost, or other termination criteria have been reached.

2.3 Multi-objective optimisation for linear programs

First introduced by Vilfredo Pareto at the end of the 19th century, multi-objective optimisation concerns the problem to find an as good of a compromise as possible between two or more conflicting objectives. This section will, briefly, introduce this concept for *linear* programs specifically.

Consider the following LP

$$\begin{aligned}
\text{minimum} \quad & C\boldsymbol{x}, && (2.22a)\\
\text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}, && (2.22b)\\
& \boldsymbol{x} \geq 0, && (2.22c)
\end{aligned}$$

where $C \in \mathbb{R}^{p\times n}$, $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $p, m, n \in \mathbb{Z}_+$. The feasible set is $X := \{x \in \mathbb{R}^n : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\}$.

Definition 8 (Pareto optimality)**.** *Let $\bar{\boldsymbol{x}} \in X$ be a feasible solution to* (2.22)*.*

1. *$\bar{\boldsymbol{x}}$ is **weakly** efficient if there is no $\boldsymbol{x} \in X$ such that $C\boldsymbol{x} < C\bar{\boldsymbol{x}}$.*
2. *$\bar{\boldsymbol{x}}$ is efficient, or **Pareto optimal**, if there is no $\boldsymbol{x} \in X$ such that $C\boldsymbol{x} \leq C\bar{\boldsymbol{x}}$.*

Remark. *Since* (2.22) *is linear it is also convex, thus if $\bar{\boldsymbol{x}} \in X$ is optimal to* (2.22)*, it is per definition also Pareto optimal if* (2.22) *is nondegenerate.*

It is possible to find *all* Pareto optimal solutions to (2.22) by considering a weighted sum of the objective functions, as is presented in Theorem 5.

Theorem 5 (Pareto optimal solutions [12, Thm. 6.6])**.** *A feasible solution $\bar{\boldsymbol{x}} \in X$ is Pareto optimal if and only if there exists a $\boldsymbol{\lambda} \in \mathbb{R}^p_+$ such that*

$$\boldsymbol{\lambda}^\mathsf{T} C\bar{\boldsymbol{x}} \leq \boldsymbol{\lambda}^\mathsf{T} C\boldsymbol{x}, \tag{2.23}$$

for all $\boldsymbol{x} \in X$.

Definition 9 (Pareto front)**.** *The **Pareto front** is the set of all Pareto optimal values to* (2.22)*.*

3

Mathematical model

This chapter contains the mathematical models used to define the problem described in Section 1. We start by defining the relevant sets, parameters and variables. Then, we construct the master problem and define the subproblems for calculating reduced costs in the column generation method. Finally, the section concludes with the second part of the mathematical model, which assigns the column generated shifts to the available agents. When defining the models, we do not consider different forms of employment, meaning that any agent can in theory work as much, or as little, as the constraints allow.

3.1 Sets, parameters and variables

We define the following:

Sets

- Agent $i \in \mathcal{I}$ (set of all agents).
- Shift $j \in \mathcal{J}$ (set of all shifts).
- Competence $k \in \mathcal{K}$ (set of all competencies).
- Day $\ell \in \mathcal{L}$ (set of all days in the planning period).
- Time step $h \in \mathcal{H}$ (set of all time steps in a day).
- Break $b \in \mathcal{B}$ (set of all breaks in any shift).

Since the set of all agents, $\mathcal{I}$, and the set of all competencies, $\mathcal{K}$, are closely intertwined it is wise to partition $\mathcal{I}$ into smaller sets, $\mathcal{I}_k$, $k \in \mathcal{K}$, which denote the set of all agents with competence k. By using this notation, we can exclude the index k on some of the variables and consequently end up with a smaller problem that is possibly less computationally expensive to solve. Note also that generally, the intersection $\mathcal{I}_1 \cap \ldots \cap \mathcal{I}_k$ is nonempty and the union $\mathcal{I}_1 \cup \ldots \cup \mathcal{I}_k = \mathcal{I}$. Conversely, we can define $\mathcal{K}_i$ to be the set of all competencies possessed by agent $i \in \mathcal{I}$.

Parameters

$d_{\ell hk} :=$ number of incoming calls day $\ell \in \mathcal{L}$, time step $h \in \mathcal{H}$, that require competence $k \in \mathcal{K}$.

$T_{hk} :=$ average time between the start of a call and the start of the next call during time step $h \in \mathcal{H}$, for competence $k \in \mathcal{K}$.

$$a_{hj} := \begin{cases} 1 & \text{if time step } h \in \mathcal{H} \text{ is covered by shift } j \in \mathcal{J}, \\ 0 & \text{otherwise.} \end{cases}$$

$B_b :=$ length of break $b \in \mathcal{B}$ (measured in amount of time steps h).

Variables

$$x_{ij\ell} := \begin{cases} 1 & \text{if agent } i \in \mathcal{I}_k,\ k \in \mathcal{K}, \text{ works shift } j \in \mathcal{J} \text{ on day } \ell \in \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

$y^{\mathrm{u}}_{\ell hk} :=$ number of lacking staff, day ℓ, at quarter h, with competence k.

$y^{\mathrm{o}}_{\ell hk} :=$ number of abundant staff, day ℓ, at quarter h, with competence k.

3.2 Model definition

The task is to minimise under- and overstaffing, hence these terms need to be more closely defined. We can define 'perfect' staffing as when the number of available agents per time step is exactly equal to the demand; in this case no call has to wait, and over-/understaffing would simply be defined as when we no longer achieve equality between demand and availability. The problem with this definition is that in many cases where understaffing is not very severe it may be more profitable to let customers wait in queue for a certain time, instead of scheduling more agents. Assuming the time it takes to handle a call (including after-call work and time to receive and answer the next call), given by T_{hk}, is less than the time required to handle the incoming calls (it makes no sense to require e.g. a call that takes 30 minutes to handle be handled in 15 minutes), we can compute the least number of agents needed to handle the amount of incoming calls $d_{\ell hk}$ within the specified time frame F_{hk} i.e. we define the *demand*, $D_{\ell hk}$, as:

$$D_{\ell hk} := \frac{d_{\ell hk} \cdot T_{hk}}{F_{hk}}, \tag{3.1}$$

with $F_{hk} \geq T_{hk}$. Clearly, we achieve zero waiting time when $F_{hk} = T_{hk}$. If we want to add some 'breathing room' between calls, we can simply increase T_{hk} by a small amount. Normally, F_{hk} is set to 900 seconds at the company, which we will also use for our results in Chapter 4, so that they will be more comparable. As for T_{hk}, the average value per competence measured over the entire year of 2022 will be used; this might lead to a slight inaccuracy in the results since T_{hk} tends to start relatively low in the morning, to then systematically increase towards the end of the day.

Because understaffing is less desirable than overstaffing, it will be penalised more in the objective so that the solution will prioritise minimising understaffing. We penalise by introducing constants P_1 and P_2 where $P_1 > P_2 > 0$.

We obtain the following mixed integer-linear optimisation model:

$$\begin{aligned}
\text{min.} \quad & \sum_{k\in\mathcal{K}}\sum_{\ell\in\mathcal{L}}\sum_{h\in\mathcal{H}} P_1 y^{\text{u}}_{\ell hk} + P_2 y^{\text{o}}_{\ell hk}, && && \text{(3.2a)}\\
\text{s.t.} \quad & \sum_{i\in\mathcal{I}_k}\sum_{j\in\mathcal{J}} a_{hj}x_{ij\ell} + y^{\text{u}}_{\ell hk} \geq D_{\ell hk}, && \forall h\in\mathcal{H}, k\in\mathcal{K}, \ell\in\mathcal{L}, && \text{(3.2b)}\\
& \sum_{i\in\mathcal{I}_k}\sum_{j\in\mathcal{J}} a_{hj}x_{ij\ell} - y^{\text{o}}_{\ell hk} \leq D_{\ell hk}, && \forall h\in\mathcal{H}, k\in\mathcal{K}, \ell\in\mathcal{L}, && \text{(3.2c)}\\
& \sum_{j\in\mathcal{J}} x_{ij\ell} \leq 1, && \forall i\in\mathcal{I}_k, k\in\mathcal{K}, \ell\in\mathcal{L}, && \text{(3.2d)}\\
& y^{\text{u}}_{\ell hk},\, y^{\text{o}}_{\ell hk} \geq 0, && \forall h\in\mathcal{H}, k\in\mathcal{K}, \ell\in\mathcal{L}, && \text{(3.2e)}\\
& x_{ij\ell} \in \{0,1\}, && \forall i\in\mathcal{I}, j\in\mathcal{J}, \ell\in\mathcal{L}. && \text{(3.2f)}
\end{aligned}$$

Explanation:
(3.2a): minimise the sum of under- and overstaffing, where we penalise understaffing more.
(3.2b): the amount of additional agents needed in order to no longer be understaffed should be no less than the difference between demand and agent availability.

(3.2c): same as (3.2b) but for overstaffing.
(3.2d): an agent can work a maximum of one shift per day.

One important thing to consider is the trade-off in accuracy of results and overall model complexity. The model (3.2) takes into account practically everything about the problem that was presented in this as well as previous sections, but the question is whether the current complexity is justified, or if possibly a simpler model could provide just as good, or good enough, results.

Firstly, P_1 and P_2 are parameters that have a nontrivial definition, thus deriving their best values mathematically could prove quite cumbersome. The only way to reasonably find values that might give at least acceptable results is by empirical means.

Secondly, having both under- and overstaffing makes the problem significantly more difficult so solve and perhaps unnecessarily because there is already a predefined upper limit on the number of agents that are actually available and if we were to instead introduce a cost of hiring agents we would automatically give an incentive to also minimise overstaffing without explicitly including it in the constraints. Furthermore, considering only understaffing also removes the need for having to define P_1 and P_2 and we can therefore omit them entirely.

Applying the simplifications mentioned to the model in (3.2) gives us the following simplified model

$$\begin{aligned}
\text{min.} \quad & \sum_{k\in\mathcal{K}}\sum_{\ell\in\mathcal{L}}\sum_{h\in\mathcal{H}} y^{\mathrm{u}}_{\ell hk}, && && (3.3a)\\
\text{s.t.} \quad & \sum_{i\in\mathcal{I}_k}\sum_{j\in\mathcal{J}} a_{hj}x_{ij\ell} + y^{\mathrm{u}}_{\ell hk} \geq D_{\ell hk}, && \forall h\in\mathcal{H}, k\in\mathcal{K}, \ell\in\mathcal{L}, && (3.3b)\\
& y^{\mathrm{u}}_{\ell hk} \geq 0, && \forall h\in\mathcal{H}, k\in\mathcal{K}, \ell\in\mathcal{L}, && (3.3c)\\
& x_{ij\ell} \in \{0,1\}, && \forall i\in\mathcal{I}, j\in\mathcal{J}, \ell\in\mathcal{L}. && (3.3d)
\end{aligned}$$

3.2.1 The restricted master problem

If we want to not only distribute shifts among the currently available agents but also attempt to minimise (if possible) the amount of agents needed in the first place, which might be of interest if one aims to cut down on some monetary spending, we will have to modify our current model (3.3) even further. First we create a minimisation model similar to (3.3) but instead of deciding whether an agent works or not we introduce an integer variable $w_{jk\ell} \geq 0$ that tells us how many agents with competence k need to be assigned a certain shift j on day ℓ. Furthermore we need to include a cost for $w_{jk\ell}$ in the objective so that assigning an agent to a shift actually comes at a cost, and also an upper bound to make sure $w_{jk\ell}$ does not surpass the amount of available agents of a specific competence, as well as overall in the day.

From Section 1.1.2, we see that shifts can vary between three and eight hours in length, with specific constraints depending on its length. In order to separate the constraints somewhat, so that the subproblem definition in Section 3.2.2 is facilitated, as well as allowing a clearer model formulation in (3.4), we define three different types of shifts. The first type are the *short* shifts, which are $\geq$ 3h and $\leq$ 4h in length. Second type are the *medium* shifts, $\geq$ 4h 15min and $\leq$ 5h 45min in length. Third type are the *long* shifts, $\geq$ 6h and $\leq$ 8h in length. It follows from Section 1.1.2 that all short shifts will have exactly one 15min break, medium shifts will have one 15min break and then another 30min lunch break and finally, long shifts will have one 15min break, one 45min lunch break, and then another 15min break. Thus in addition to the sets, parameters and variables defined in Section 3.1, we also define the following sets:

Shift type $q \in \{s, m, l\}$ (set of all shift types, s = short, m = medium, l = long),

$\mathcal{J}^q :=$ set of shifts of type $q \in \{s, m, l\}$, $\mathcal{J}^s \cup \mathcal{J}^m \cup \mathcal{J}^l = \mathcal{J}$,

and variables:

$w_{jk\ell} :=$ number of agents with competence k, that need to be assigned shift j, day ℓ.

Because our model determines only how many agents should work and says nothing about which shift and/or day individual agents are assigned, no union or work environment rules of the sort need to be taken into account. This means that there will be no constraints spanning more than one given day and the solution can therefore be obtained on a day to day basis, rather than having to consider the entire planning period $\mathcal{L}$. Thus the ℓ-index can be omitted and so for every $\ell \in \mathcal{L}$ we formulate the model:

$$\text{min.} \qquad c_1 \sum_{k\in\mathcal{K}} \sum_{h\in\mathcal{H}} y^{\text{u}}_{hk} + c_2 \sum_{k\in\mathcal{K}} \sum_{q\in\{s,m,l\}} \sum_{j\in\mathcal{J}^q} \left(\sum_{h\in\mathcal{H}} a_{hj} + B^q_{tot} \right) w_{jk}, \tag{3.4a}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{j\in\mathcal{J}} a_{hj} w_{jk} + y^{\mathrm{u}}_{hk} \geq D_{hk}, && \forall h \in \mathcal{H}, k \in \mathcal{K}, && \text{(3.4b)}\\
& \sum_{j\in\mathcal{J}} w_{jk} \leq |\mathcal{I}_k|, && \forall k \in \mathcal{K}, && \text{(3.4c)}\\
& \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} w_{jk} \leq |\mathcal{I}|, && && \text{(3.4d)}\\
& y^{\mathrm{u}}_{hk} \geq 0, && \forall h \in \mathcal{H}, && \text{(3.4e)}\\
& w_{jk} \in \mathbb{Z}^{+}, && \forall j \in \mathcal{J}, k \in \mathcal{K}, && \text{(3.4f)}
\end{aligned}
$$

Explanation:
(3.4a): minimise the cost of being understaffed, summed with the cost of the total availability of agents.
(3.4b): the availability summed with the amount of understaffing a certain time step should be greater than the demand for that time step.
(3.4c): amount of agents with competence $k \in \mathcal{K}$ should not exceed the maximum amount for that competence.
(3.4d): total amount of agents should not exceed maximum amount of available agents.

where B^q_{tot} is the total length of paid breaks in shift type $q \in \{s, m, l\}$, c_1 is the cost of being understaffed and c_2 the cost per time step of hiring an agent. The ratio between c_1 and c_2 is of great importance since the model will choose which variables are prioritized in the minimisation based on whether it costs more to add a shift or to be understaffed. Once the optimal solutions have been found for every day $\ell \in \mathcal{L}$, they will be merged to form $y^u_{\ell hk}$ and $w_{jk\ell}$, respectively. Because the amount of possible shifts are quite large[1], we will apply column generation to the LP-relaxed version of (3.4) and once the optimal solution to the LP is found the integer requirements will be reinstated and (3.4) solved with the new set of shifts. We obtain the master problem (see Section 2.2) by relaxing the integer requirements on $w_{jk\ell}$. Let $\mathcal{J}' \in \mathcal{J}$ be the restricted set of shifts, thus the restricted master problem is

[1]Considering that the shift lengths, along with their constraints, can all be expressed in terms of the total amount of time steps $\mathcal{H}$, we can deduce that the total amount of short shifts have a complexity of $O(\mathcal{H}^3)$. This is due to the fact that the amount of time steps in the shift depends on $\mathcal{H}$, the shift itself can start and end in a certain amount of ways which also depends on $\mathcal{H}$, and the break can also move a certain amount of time steps, also dependent of $\mathcal{H}$. Hence, utilising the multiplication principle of combinatorics, the total number of short shifts is $O(\mathcal{H}^3)$. Similarly, the amount is $O(\mathcal{H}^4)$ for medium shifts and $O(\mathcal{H}^5)$ for long shifts.

$$
\begin{aligned}
\text{min.} \quad & c_1 \sum_{k\in\mathcal{K}} \sum_{h\in\mathcal{H}} y^{\mathrm{u}}_{hk} + c_2 \sum_{k\in\mathcal{K}} \sum_{q\in\{s,m,l\}} \sum_{j\in\mathcal{J}'^q} \left(\sum_{h\in\mathcal{H}} a_{hj} + B^q_{tot} \right) w_{jk}, && && (3.5a)\\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}'} a_{hj} w_{jk} + y^{\mathrm{u}}_{hk} \geq D_{hk}, && \forall h \in \mathcal{H}, k \in \mathcal{K}, && (3.5b)\\
& \sum_{j\in\mathcal{J}'} w_{jk} \leq |\mathcal{I}_k|, && \forall k \in \mathcal{K}, && (3.5c)\\
& \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}'} w_{jk} \leq |\mathcal{I}|, && && (3.5d)\\
& y^{\mathrm{u}}_{hk}, w_{jk} \geq 0, && \forall h \in \mathcal{H}, j \in \mathcal{J}', k \in \mathcal{K}. && (3.5e)
\end{aligned}
$$

3.2.2 Application of column generation

In this section, together with Sections 3.2.2.1, 3.2.2.2 and 3.2.2.3, we will present the reduced cost subproblems (see Section 2.2) that are used in the column generation method to find the column with the smallest negative reduced cost each iteration. One problem however, is that the definition of the columns themselves is not trivial. Clearly, since we are restricting the set of shifts $\mathcal{J}$ in (3.5), the columns should be the shifts that are added each iteration, but how do we create the shifts? One way that may come to mind is to simply store all possible shifts in a list and then pick from that list every iteration based on the reduced costs of the shifts. One will probably realise however that the total number of shifts will grow large quite quickly and will inevitably give rise to memory issues (see Footnote 1), especially for larger sets $\mathcal{H}$. Another way is to construct, and solve, a shortest path problem by letting the opening hours define a graph and then creating a shift by traversing the graph from start to end. As mentioned in Section 1.2, the PMSPIC model is applicable to this problem, as we will see in this section.

The parameter a_{hj}, first introduced in Section 3.1, forms a matrix of dimension $|\mathcal{H}| \times |\mathcal{J}'|$, $\mathcal{J}' \subseteq \mathcal{J}$, in which $\mathcal{J}'$ will be extended for every generated column by adding a_{hj}-values, with $j \notin \mathcal{J}'$. From (2.21), we know that the objective of the column generation subproblem should be to minimise the cost of adding a column (or shift in this case), subtracted with the optimal value of the dual variable multiplied with that shift. In this context, the shift cost depends simply on its length - all shifts have the same cost per time step (which we have already defined as c_2 in (3.4)), thus their length is the only thing that dictates their total cost. The optimal dual variable, which we denote as $u^*_{hk\ell}$, corresponds to the respective constraint (3.5b). Let the length of shift $j \in \mathcal{J}$ be denoted by p_j, the objective will be to minimise $c_2 \cdot p_j - \sum_{h\in\mathcal{H}} u^*_{hk\ell} \cdot a_{hj}$.

As we saw in Section 1.1.2, there are clearly constraints that also need to be considered, in addition to the objective function, when constructing the column generation subproblem. The subproblem will be constructed as a shortest path problem for a weighted directed graph and there will be one subproblem defined for each shift type (short, medium, long), meaning a total of three different subproblems. To start, we consider a discretisation of the timeline that denotes the opening hours, as shown in Figure 3.1, where the nodes depict the time steps $h \in \mathcal{H}$. For this particular scheduling problem, the total number of time steps is 32 i.e. $|\mathcal{H}| = 32$, and the time step length is 15 minutes for all time steps, thus the call centre is open for a total of eight hours. From now on we will refer to time in terms of time steps rather than hours and minutes.

For the model considered in [9], one more timeline, parallel to the one in Figure 3.1 is needed to illustrate the complete graph, which is depicted in Figure 3.2. One unit of flow starts at time step 0 on the bottom timeline, with flow balance being preserved all the way to $\mathcal{H} + 1$ in the upper timeline, and the cheapest path from node 0 in the lower- to $\mathcal{H} + 1$ in the upper timeline is found. The path traverses from the lower- to the upper timeline when preventive maintenance occurs, as described in further detail in [9].

Figure 3.1: Illustration of the discrete timeline for the call centre opening hours, read from left to right. The nodes depict the equal sized time steps.

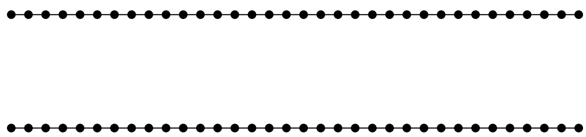


Figure 3.2: Illustration of the graph for the model presented in [9]. The nodes are represented by two indices: (h, n), where h is the time step, and n is the timeline.

The problem with this particular project however, is that shifts can start and end at different times of the day and thus need to include two extra parallel time lines. Also there are breaks in the shifts, of which some are of different lengths and quantity, that need to be taken into account in the formulation, hence the generalisation mentioned in Section 1.2.

We now define the set of *arcs*:

$$\mathcal{A} := \{(s,h)|\; 0 \leq s < h \leq H+1;\; h,s \in \mathbb{Z}_+\},$$

and the set of (parallel) timelines:

$$\mathcal{N} := \{0,1,\ldots,5\}.$$

The chosen *path* through the graph will be a subset of $\mathcal{A}$. In order to differentiate between the arcs that are included in the optimal path from those that are not we also define the following variable:

$$v_{sh}^{n} := \begin{cases} 1 & \text{if arc } (s,h) \in \mathcal{A}, \text{ in timeline } n \in \mathcal{N}, \text{ is included in the path,} \\ 0 & \text{otherwise.} \end{cases}$$

An illustrated example of a graph with a completed path depicting a short shift is given in Figure 3.3. Note that the total number of timelines are 4 for the short shifts specifically, for medium- and long shifts they are 5 and 6 respectively due to the extra breaks (see Section 3.2.1 for the shift definitions). The particular shift depicted in Figure 3.3 is 16 time steps long, starts at step 8, has one break during step 14 and ends at (also including) step 23. We can see from the diagonal dashed line between timeline 1 and 2 that a break has been taken, as well as for how long. If the path is optimal in the sense that it minimises the reduced cost, which we have defined earlier in this section, it will be added in the form of a shift to $\mathcal{J}'$. This will be done for every shift type, for every iteration of the column generation algorithm.

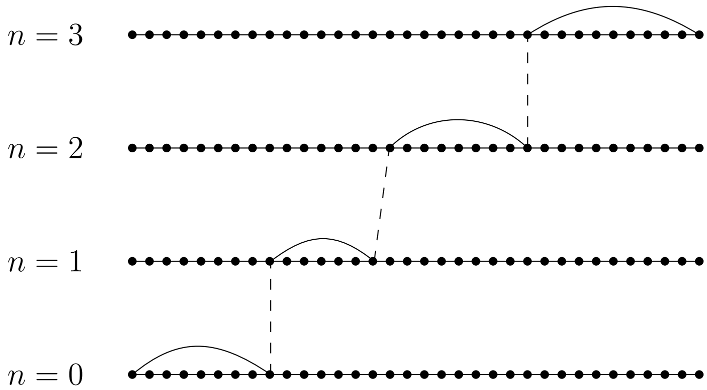


Figure 3.3: Illustrated example of a "short" shift where the lines $n = 0, \ldots, 3$ represent timelines, the nodes are represented by (h, n) as in Figure 3.2. The arcs in the graph go from node (s, n) to node (h, m), such that $0 \leq s < h \leq \mathcal{H} + 1$ for all n and m such that $0 \leq n \leq m \leq 3$ (for short shifts, but the same principle applies to all shift types). The dashed lines between timelines are for visual clarity.

3.2.2.1 Subproblem for generating short shifts

This section will present the mathematical model for the subproblem used to generate short shifts, and is based on the discussion in Section 3.2.2.

Let p denote the shift length, given by the specific time step where the final arc in the optimal path starts (shift ends), subtracted with the time step where the first arc ends (shift starts). This can equivalently be expressed as $p := \sum_{s=0}^{H} s \cdot v^3_{s,H+1} - \sum_{h=1}^{H+1} h \cdot v^0_{0,h} + 1$. The '+1' at the end comes from the fact that we actually include the time step where the final arc ends (see the short shift example in Figure 3.3). Recall also from Section 3.1 that we denoted the break lengths by B_b, $b \in \mathcal{B}$. Inserting this notation into the context of Section 3.2.1 it follows that $\mathcal{B} = \{1, 2, 3\}$ and ordering the breaks by length we get that $B_1 = 1$, $B_2 = 2$, $B_3 = 3$, or, in words, the set of all breaks consists of three breaks total, each numbered 1, 2 and 3, where break 1 is one time step long, break 2 is two time steps long, and break 3 is three time steps long. Furthermore, the subproblem needs to be computed for every competence $k \in \mathcal{K}$ for the current day $\ell \in \mathcal{L}$, giving us $O(|\mathcal{K}|)$ subproblems to solve every iteration[2].

Hence, for every $\ell \in \mathcal{L}$, the model for generating a short shift is then expressed as

[2]The master problem in (3.4), as mentioned in Section 3.2.1, is solved for all $\ell \in \mathcal{L}$ and thus is solved $O(|\mathcal{L}|)$ times, followed by $O(|\mathcal{K}|)$ subproblems for every iteration of the column generation.

$$
\begin{aligned}
\text{min.} \quad & c_2 p - \sum_{h=1}^{H} u_{hk\ell} a_h, && (3.6a)\\
\text{s.t.} \quad & a_h = \sum_{n=1}^{3} \sum_{s=0}^{h} \sum_{r=h+1}^{H+1} v_{sr}^{n} - \sum_{s=0}^{h-1} \sum_{r=h}^{H+1} v_{sr}^{3}, && \forall h \in \mathcal{H}, \quad (3.6b)\\
& \sum_{h=1}^{H+1} v_{0h}^{0} = 1, && (3.6c)\\
& \sum_{s=0}^{h-1} v_{sh}^{n} = \sum_{r=h+1}^{H+1} v_{hr}^{n+1}, && \forall n \in \{0,2\}, h \in \mathcal{H}, \quad (3.6d)\\
& \sum_{s=0}^{h-1} v_{sh}^{1} = \sum_{r=h+B_1+1}^{H+1} v_{h+B_1,r}^{2}, && \forall h \in \mathcal{H}, \quad (3.6e)\\
& \sum_{s=h-3}^{h-1} \left(v_{sh}^{1} + v_{sh}^{2}\right) = 0, && h = 3, \dots, H+1, \quad (3.6f)\\
& \sum_{n=1}^{2} v_{sh}^{n} = 0, && 0 \le s < h \le 2, \quad (3.6g)\\
& 12 \le p \le 16, && (3.6h)\\
& v_{sh}^{n} \in \{0,1\}, && \forall n \in \{0,\dots,3\}, \quad (3.6i)\\
& && (s,h) \in \mathcal{A}.
\end{aligned}
$$

Explanation:
(3.6a): minimise total cost of chosen path.
(3.6b): translate the arcs included in the optimal path into the variables a_h so that the reduced cost can be computed in (3.6a). A shift is then obtained through a_h.
(3.6c): start the 'flow' at node $(0, 0)$.
(3.6d)–(3.6e): flow balance between timelines.
(3.6f): break cannot start earlier than 3 time steps from shift start, and must occur earlier than 3 time steps from shift end.
(3.6g): remove the arcs that were left out in (3.6f).
(3.6h): shift length cannot be shorter than 12 time steps and not longer than 16 time steps.

3.2.2.2 Subproblem for generating medium shifts

For the medium shift subproblem, most of the reasoning is the same as for the short shifts in Section 3.2.2.1. The difference is that there is an extra 2 time step long lunch break included and as such, one extra timeline is needed in the graph.

Also, since lunch breaks are not paid, we will have to subtract 2 time steps from the shift length, thus $p = \sum_{s=0}^{H} s \cdot v^4_{s,H+1} - \sum_{h=1}^{H+1} h \cdot v^0_{0,h} - 1$.

For every $\ell \in \mathcal{L}$, the model for generating a medium shift is expressed as

$$\begin{aligned}
\text{min.} \quad & c_2 p - \sum_{h=1}^{H} u_{hk\ell} a_h, && && \text{(3.7a)} \\
\text{s.t.} \quad & a_h = \sum_{n=1}^{4} \sum_{s=0}^{h-1} \sum_{r=h}^{H+1} v^n_{sr} - \sum_{s=0}^{h-1} \sum_{r=h}^{H+1} v^4_{sr}, && \forall h \in \mathcal{H}, && \text{(3.7b)} \\
& \sum_{h=1}^{H+1} v^0_{0h} = 1, && && \text{(3.7c)} \\
& \sum_{s=0}^{h-1} v^n_{sh} = \sum_{r=h+1}^{H+1} v^{n+1}_{hr}, && \forall n \in \{0,3\}, h \in \mathcal{H}, && \text{(3.7d)} \\
& \sum_{s=0}^{h-1} v^n_{sh} = \sum_{r=h+B_n+1}^{H+1} v^{n+1}_{h+B_n,r}, && \forall n \in \{1,2\}, h \in \mathcal{H}, && \text{(3.7e)} \\
& \sum_{s=h-3}^{h-1} \left(v^1_{sh} + v^3_{sh}\right) = 0, && h = 3, \dots, H+1, && \text{(3.7f)} \\
& \sum_{s=h-5}^{h-1} v^2_{sh} = 0, && h = 5, \dots, H+1, && \text{(3.7g)} \\
& \sum_{s=0}^{h-11} v^1_{sh} = 0, && h = 11, \dots, H+1, && \text{(3.7h)} \\
& \sum_{s=0}^{h-12} v^2_{sh} = 0, && h = 12, \dots, H+1, && \text{(3.7i)} \\
& \sum_{n=1}^{3} v^n_{sh} = 0, && 0 \leq s < h \leq 2, && \text{(3.7j)} \\
& \sum_{n=2}^{3} v^n_{sh} = 0, && 0 \leq s < h \in \{3,4\}, && \text{(3.7k)} \\
& 17 \leq p \leq 23, && && \text{(3.7l)} \\
& v^n_{sh} \in \{0,1\}, && \forall n \in \{0,\dots,4\}, && \text{(3.7m)} \\
& && (s,h) \in \mathcal{A}. &&
\end{aligned}$$

Explanation:
(3.7a): minimise total cost of chosen path.
(3.7b): translate the arcs included in the optimal path into the variables a_h so that

the reduced cost can be computed in (3.7a). A shift is then obtained through a_h.
(3.7c): start the 'flow' at node $(0,0)$.
(3.7d)–(3.7e): flow balance between timelines.
(3.7f): first break cannot start earlier than 3 time steps from shift start, and the last break must occur earlier than 3 time steps from shift end.
(3.7g): second break cannot start earlier than 5 time steps from the end of the preceding break.
(3.7h): first break cannot start later than 11 time steps from shift start.
(3.7i): second break cannot start later than 12 time steps from the end of the preceding break.
(3.7j)–(3.7k): remove the arcs that were left out in (3.7f)–(3.7i).
(3.7l): shift length cannot be shorter than 17 time steps and not longer than 23 time steps.

3.2.2.3 Subproblem for generating long shifts

Long shifts have one extra break, as well as a 3 time step lunch break, hence there will be yet another extra timeline compared to the medium shifts in Section 3.2.2.2, and the shift length is given by $p = \sum_{s=0}^{H} s \cdot v^5_{s,H+1} - \sum_{h=1}^{H+1} h \cdot v^0_{0,h} - 2$.

For every $\ell \in \mathcal{L}$, the model for generating a long shift is expressed as

$$\text{min.} \quad c_2 p - \sum_{h=1}^{H} u_{hk\ell} a_h, \tag{3.8a}$$

$$\text{s.t.} \quad a_h = \sum_{n=1}^{5} \sum_{s=0}^{h-1} \sum_{r=h}^{H+1} v^n_{sr} - \sum_{s=0}^{h-1} \sum_{r=h}^{H+1} v^5_{sr}, \qquad \forall h \in \mathcal{H}, \tag{3.8b}$$

$$\sum_{h=1}^{H+1} v^0_{0h} = 1, \tag{3.8c}$$

$$\sum_{s=0}^{h-1} v^n_{sh} = \sum_{r=h+1}^{H+1} v^{n+1}_{hr}, \qquad \forall n \in \{0,4\}, h \in \mathcal{H}, \tag{3.8d}$$

$$\sum_{s=0}^{h-1} v^n_{sh} = \sum_{r=h+B_1+1}^{H+1} v^{n+1}_{h+B_1,r}, \qquad \forall n \in \{1,3\}, h \in \mathcal{H}, \tag{3.8e}$$

$$\sum_{s=0}^{h-1} v^2_{sh} = \sum_{r=h+B_3+1}^{H+1} v^3_{h+B_3,r}, \qquad \forall h \in \mathcal{H}, \tag{3.8f}$$

$$\sum_{s=h-3}^{h-1} \left(v^1_{sh} + v^4_{sh}\right) = 0, \qquad h = 3, \dots, H+1, \tag{3.8g}$$

$$\sum_{s=h-5}^{h-1} \left(v_{sh}^2 + v_{sh}^3\right) = 0, \qquad h = 5, \dots, H+1, \quad (3.8h)$$

$$\sum_{s=0}^{h-11} v_{sh}^1 = 0, \qquad h = 11, \dots, H+1, \quad (3.8i)$$

$$\sum_{s=0}^{h-12} \left(v_{sh}^2 + v_{sh}^3\right) = 0, \qquad h = 12, \dots, H+1, \quad (3.8j)$$

$$\sum_{n=1}^{4} v_{sh}^n = 0, \qquad 0 \le s < h \le 2, \quad (3.8k)$$

$$\sum_{n=2}^{3} v_{sh}^n = 0, \qquad 0 \le s < h \in \{3, 4\}, \quad (3.8l)$$

$$24 \le p \le 32 \quad (3.8m)$$

$$v_{sh}^n \in \{0, 1\}, \qquad \forall n \in \mathcal{N}, (s, h) \in \mathcal{A}. \quad (3.8n)$$

Explanation:
(3.8a): minimise total cost of chosen path.
(3.8b): translate the arcs included in the optimal path into the variables a_h so that the reduced cost can be computed in (3.8a). A shift is then obtained through a_h.
(3.8c): start the ‘flow’ at node $(0, 0)$.
(3.8d)–(3.8f): flow balance between timelines.
(3.8g): first break cannot start earlier than 3 time steps from shift start, and the last break must occur earlier than 3 time steps from shift end.
(3.8h): second and third break cannot start earlier than 5 time steps from the end of the preceding break.
(3.8i): first break cannot start later than 11 time steps from shift start.
(3.8j): second and third break cannot start later than 12 time steps from the end of the preceding break.
(3.8k)–(3.8l): remove the arcs that were left out in (3.8g)–(3.8j).
(3.8m): shift length cannot be shorter than 24 time steps and not longer than 32 time steps.

3.2.3 Distributing the shifts

Letting the optimal solution to the restricted master problem (3.4) be $(y_{hk\ell}^{u*}, w_{jk\ell}^*)$, we wish to use the information from $w_{jk\ell}^*$ as input to a model that distributes the generated shifts among agents. To do so, we further define the following variables:

$$z_i := \begin{cases} 1 & \text{if agent } i \in \mathcal{I} \text{ works at least one shift in the planning period } \mathcal{L}, \\ 0 & \text{otherwise,} \end{cases}$$

$S_{ikjh\ell} :=$ percentage of time step $h \in \mathcal{H}$ agent $i \in \mathcal{I}$, assigned shift $j \in \mathcal{J}$, spends on answering calls that require competence $k \in \mathcal{K}$, on day $\ell \in \mathcal{L}$,

together with a large enough constant $M := |\mathcal{J}| \cdot |\mathcal{L}|$, we obtain the following model that minimises the total number of individual agents needed to cover all the shifts that result from the model in (3.5):

$$\begin{aligned}
\text{min.} \quad & \sum_{i \in \mathcal{I}} z_i, && && \text{(3.9a)} \\
\text{s.t.} \quad & \sum_{\ell \in \mathcal{L}} \sum_{j \in \mathcal{J}_\ell} x_{ij\ell} \leq M z_i, && \forall i \in \mathcal{I}, && \text{(3.9b)} \\
& \sum_{i \in I_k} S_{ikjh\ell} \geq a_{hj} w^*_{jk\ell}, && \forall h \in \mathcal{H}, j \in \mathcal{J}, k \in \mathcal{K}, \ell \in \mathcal{L}, && \text{(3.9c)} \\
& \sum_{k \in \mathcal{K}_i} S_{ikjh\ell} = a_{hj} x_{ij\ell}, && \forall h \in \mathcal{H}, i \in \mathcal{I}, j \in \mathcal{J}_\ell, \ell \in \mathcal{L}, && \text{(3.9d)} \\
& \sum_{j \in \mathcal{J}_\ell} x_{ij\ell} \leq 1, && \forall i \in \mathcal{I}, \ell \in \mathcal{L}, && \text{(3.9e)} \\
& z_i, x_{ij\ell} \in \{0, 1\}, && \forall i \in \mathcal{I}, j \in \mathcal{J}_\ell, \ell \in \mathcal{L}, && \text{(3.9f)} \\
& S_{ikjh\ell} \geq 0, && \forall h \in \mathcal{H}, i \in \mathcal{I}, j \in \mathcal{J}_\ell, k \in \mathcal{K}, \ell \in \mathcal{L}. && \text{(3.9g)}
\end{aligned}$$

Explanation:
(3.9a): minimise the amount of agents working at least one shift.
(3.9b): if agent $i \in \mathcal{I}$ is assigned a shift on any day during the planning period $\mathcal{L}$ it should equate to z_i being equal to one for that agent, zero otherwise.
(3.9c): ensure the demand of availability is met by the total time distribution among agents with competence $k \in \mathcal{K}$.
(3.9d): the total time distribution of agent $i \in \mathcal{I}$, with competence $k \in \mathcal{K}$, that is assigned shift $j \in \mathcal{J}$, during time step $h \in \mathcal{H}$, day $\ell \in \mathcal{L}$, should sum up to one if the agent works (is available) at that time.
(3.9e): an agent can work a maximum of one shift per day.

Once an integer solution, (z_i^*, $x_{ij\ell}^*$, $S_{ikjh\ell}^*$), to (3.9) has been procured it will be possible to obtain a schedule for all agents (to get an *optimal* integer solution to

(3.9), one would have to encase the column generation method into a Branch and Price framework, see Section 5.1). Note that since (3.4) was solved for every ℓ we get a different set of column generated shifts each time, hence why we have denoted the set of all shifts as $\mathcal{J}_\ell$ in (3.9).

The variable $S_{ikjh\ell}$ is a necessity to the model in (3.9) because, for this specific problem, agents can have more than one competence. This means that if (3.9c) and (3.9d) were not included it would imply that agents can handle demand from different competencies simultaneously and the results would therefore be excessively optimistic.

3.2.4 Model definition summary

This section summarises everything presented and discussed in Section 3.2, along with its subsections 3.2.1–3.2.3 in Algorithm 2.

Algorithm 2: Generating agent schedules

1. Relax integer requirements on the model in (3.4).
2. Restrict the set of available shifts by defining $\mathcal{J}' \subset \mathcal{J}$. This set can either be chosen randomly or with a suitable heuristic.
3. Using Algorithm 1, find the optimal LP-solution to (3.4) and let $\mathcal{J}_\ell$ be the set of shifts with which the optimal solution was found, for every day ℓ in the planning period $\mathcal{L}$.
4. Reinstate the integer requirements and solve (3.4) with the set of shifts $\mathcal{J}_\ell$ obtained in step 3., for every $\ell \in \mathcal{L}$.
5. Use the solutions from 4., together with the shifts $\mathcal{J}_\ell$, $\ell \in \mathcal{L}$, as input to the model in (3.9) and solve it.

4

Tests and results

In this chapter the tests and results of this project are presented, along with the relevant data used. All models and methods presented throughout this text have been implemented in Julia version 1.8.5 [14], using the JuMP package [15], and solved with the Gurobi Optimiser version 10.0.1 [16], on a system with Intel(R) Core(TM) i7-4770 CPU (3.40GHz) processor with 4 cores and 16GB RAM.

4.1 Data

This section presents all the relevant data used for the tests in this chapter, all of which have been provided by Telia. To start, for this particular scheduling problem, there are a total of 45 available agents, all of which have at least one of five competencies (see Table 4.1). Figure 4.1 illustrates the available agents and what competencies they possess in a coloured grid.

Table 4.1: Parameters, with their corresponding data values, used in performing the tests and acquiring the results.

Parameter	Value in the tests
Number of agents $\|\mathcal{I}\|$	45
Number of competencies $\|\mathcal{K}\|$	5
Number of days $\|\mathcal{L}\|$	20
Number of time steps per day	32
Length of time steps	15min

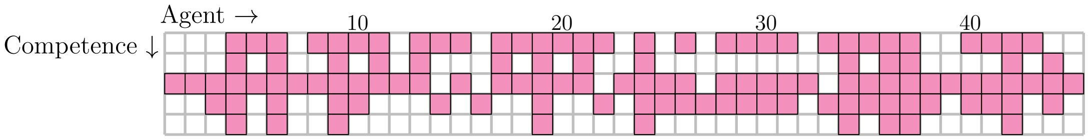


Figure 4.1: Grid representing the set $\mathcal{K}_i$, $i \in \mathcal{I}$ i.e. the set of competencies possessed by agent i. A coloured cell means they posses that competence. The transpose of the grid gives us $\mathcal{I}_k$, $k \in \mathcal{K}$, which denotes the set of agents with competence k. The horizontal numbering denotes every 10'th agent and is for visual clarity.

The parameter T_{hk}, first introduced in Section 3.1, denotes the average handling time (AHT) of calls per time step $h \in \mathcal{H}$, for every competence $k \in \mathcal{K}$. To simplify the implementation somewhat, we let T_{hk} be constant for all h so that it only depends on the given competence k, Table 4.2 shows the rounded average T_k value over the whole year of 2022. The quality of the given data is a bit questionable in the sense that there is no differentiation between after call-work and breaks, meetings etc., which means that the values in Table 4.2 are higher than reality. Not accounting for meetings and other activities outside of regular work hours, the vast majority of agents are full time employees and thus have at least a total of one hour of breaks. Assuming the average total break time is one hour per day, this would equate to 3200 extra seconds total per day for every agent. Since agents can have more than one competence, the 3200 seconds may be spread out over several competencies and thus it is practically impossible to figure out how much extra time every T_k - value actually has, given the data. For a comparison, Table 4.3 shows the average talk time for every competence (T_{hk} is the talk time added with after call-work). The goal for the maximal after call-work at the company is 90 seconds, in reality it averages closer to about 180 seconds (generally for all competencies, not just the ones considered for this problem), adding this value to Table 4.3 gives us Table 4.4. One can see that the values for competence 2 and 5 in Table 4.2 are much closer to the corresponding values in 4.4 than other competencies. This is because the demand is much lower for these two competencies specifically, compared to the others (see Figure 4.2b and 4.2e, as well as Figure 4.6 in Section 4.3), thus the sample size is smaller and there is a significantly lower probability that an agent takes a break or similar after handling a matter requiring one of those competencies, which also means that the after call values for competencies 2 and 5 in the data are closer to reality. It should also be mentioned that competencies 2 and 5 are chat errands only, which generally results in longer handling times compared to the rest which are phone calls only.

Table 4.2: Average T_k values, $k \in \{1, \ldots, 5\}$, over the year of 2022.

T_k	Value (s)
T_1	801
T_2	944
T_3	789
T_4	775
T_5	882

Table 4.3: Average talk time per competence $k \in \{1, \ldots, 5\}$, over the year of 2022.

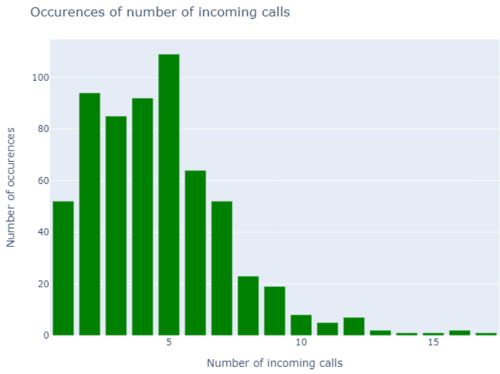
Competence	Talk time (s)
1	384
2	705
3	394
4	360
5	648

Table 4.4: Average talk time per competence $k \in \{1, \ldots, 5\}$, over the year of 2022, plus the average time spent doing after call-work.

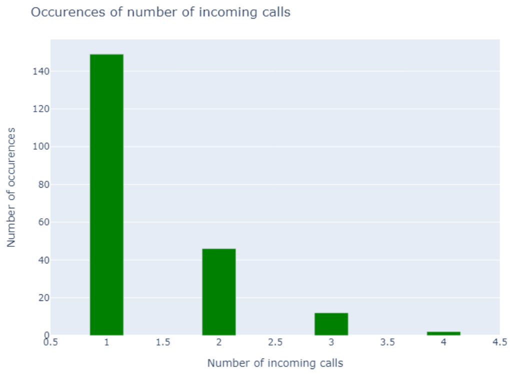
Competence	Talk time + after call-work (s)
1	564
2	885
3	574
4	540
5	828

Further, the preferred handling time, F_{hk} (see Section 3.1) is normally set to 900 seconds, which is why this value is used here as well. Recall from Section 3.2 that $F_{hk} \geq T_{hk}$ should hold, so for competence 2 in Table 4.2 specifically, the preferred handling time should be set to 944 seconds if those table values are used.

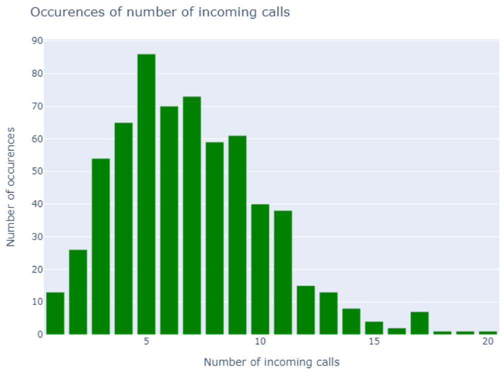
Finally, the number of incoming calls, $d_{\ell hk}$ (defined in Section 3.1), are presented in Figures 4.2a–4.2e as the number of occurrences of a certain amount of incoming calls during any time step, for the first 20 days in January 2022. The $d_{\ell hk}$ - values are used together with T_{hk} and F_{hk} in (3.1) (see Section 3.2) to calculate the demand, $D_{\ell hk}$, used in (3.2)–(3.5) (see Sections 3.2 and 3.2.1). Due to the format of the given data, we will assume that all calls arrive at the same time every time step. All tests in this project have been done over the same time period i.e. the first 20 days in January 2022.



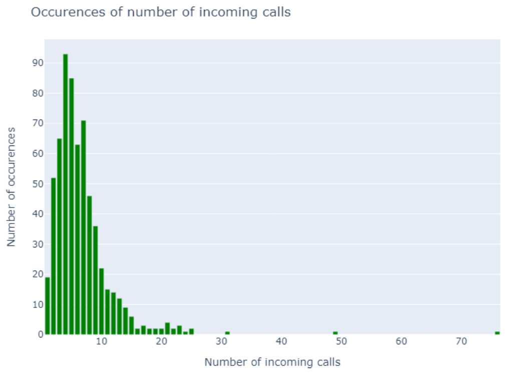
(a) Competence 1.



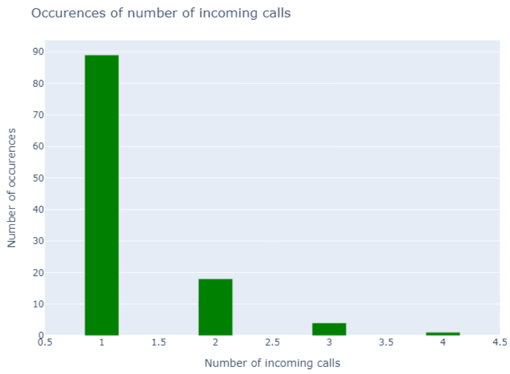
(b) Competence 2.



(c) Competence 3.



(d) Competence 4.



(e) Competence 5.

Figure 4.2: Bar chart depicting the number of occurrences of a specific $d_{\ell h k}$ (number of incoming calls) - value during any time step $h \in \mathcal{H}$, over the first 20 days in Januray 2022.

4.2 Multiobjective properties

In Section 3.2.1 we presented the model (3.4), which was to find an optimal set of shifts such that the total cost of hiring w agents for each shift j summed with the cost for being understaffed is minimised. One important detail that was left out about this particular problem is that it has two conflicting objective functions; one is to minimise understaffing, the other is to minimise the number of agents needed. The objectives are *conflicting* because it is not possible to minimise one objective without increasing the other, as will be illustrated shortly. Recall the theory regarding multiobjective optimisation from Section 2.3, the model in (3.3) is precisely a multiobjective linear problem with two conflicting objectives. Thus we know from Theorem 5 that we can find all Pareto optimal solutions to (3.3) by solving it for different values of the cost parameters c_1, c_2 and in fact guarantee Pareto optimality as long as the costs are > 0, and (3.3) is nondegenerate. Note however that there is no way of verifying nondegeneracy of (3.3) - it is actually quite likely that there are degenerate solutions since many different shift combinations could probably yield the same optimal value. Nevertheless, the optimal objective *values* are still the same, regardless if the solutions are degenerate or not - they all lie on the same level curve. Hence the *Pareto front* (see Definition 9 in Section 2.3) is still of great interest to study since it illustrates the contradicting nature of the two objective functions. Because there are two objectives in the model (3.3) specifically, any value of c_1 and c_2 respectively can simply be factored out as a constant (which will only move the Pareto front, not change its shape), leaving us with weights corresponding to the ratio of c_1 and c_2. Hence, in order to acquire all Pareto optimal values to (3.3), it is sufficient to study only the convex combination of the two objectives, thus we let $c_1 := \lambda$ and $c_2 := (1 - \lambda)$, $\lambda \in [0, 1]$. To find the correct Pareto optimal values it is necessary for the problem to be solved to optimality, which in our case currently is not possible without using Branch and Price, as mentioned in Section 5.1. Therefore, the LP-solution to (3.4) will have to suffice and, with c_1 and c_2 reformulated, we will solve the following master problem

$$\text{min.} \quad \lambda \sum_{k\in\mathcal{K}} \sum_{h\in\mathcal{H}} y^{\mathrm{u}}_{hk} + (1-\lambda) \sum_{k\in\mathcal{K}} \sum_{q\in\{s,m,l\}} \sum_{j\in\mathcal{J}^q} \left(\sum_{h\in\mathcal{H}} a_{hj} + B^q_{tot} \right) w_{jk}, \tag{4.1a}$$

$$\text{s.t} \quad \sum_{j\in\mathcal{J}} a_{hj} w_{jk} + y^{\mathrm{u}}_{hk} \geq D_{hk}, \qquad \forall h \in \mathcal{H}, k \in \mathcal{K}, \tag{4.1b}$$

$$\sum_{j\in\mathcal{J}} w_{jk} \leq |\mathcal{I}_k|, \qquad \forall k \in \mathcal{K}, \tag{4.1c}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} w_{jk} \leq |\mathcal{I}|, \tag{4.1d}$$

$$y^{\mathrm{u}}_{hk}, w_{jk} \geq 0, \qquad \forall h \in \mathcal{H}, j \in \mathcal{J}, k \in \mathcal{K}. \tag{4.1e}$$

to optimality, using column generation, in accordance with the discussions in Sections 3.2.1–3.2.2, for different values of λ, and one randomly chosen day (in this case day 1, see Figure 4.3). Note also that the subproblems (3.6)–(3.8) in Section 3.2.2 will have c_2 replaced with $(1-\lambda)$, since it is the same cost as c_2 in (3.4). To calculate D_{hk} in (4.1), we shall use the T_{hk}-values from Table 4.4. The initial feasible solution was chosen randomly to be one long shift.

Although we cannot guarantee optimality when solving (4.1) with integer requirements on w_{jk}, using the columns from the LP-solution, it can still be interesting to see what the Pareto front looks like for those integer solutions as well, since they form an upper bound on the *optimal* integer solutions, while the LP-solutions form a lower bound. Hence the 'true' Pareto front of (4.1) with w_{jk} integer will lie somewhere in-between the LP-Pareto front of (4.1) and the w_{jk}-integer Pareto front obtained using the same columns as for the LP-Pareto front. Figure 4.3 shows the Pareto optimal values to (4.1) in the two dimensional objective space, together with the corresponding values with integer requirements enforced on w_{jk}, where we call the λ-term 'Total understaffing', and the $(1-\lambda)$-term 'Total availability + paid breaks'.

Note that the objective value 0 for the 'Total availability + paid breaks'-objective has already been reached when $\lambda = 0.5$, both for the LP-Pareto front and the integer Pareto front. Similarly for the 'Total understaffing'-objective when $\lambda = 0.95$ for the LP-pareto front, hence it would be redundant to include points with $\lambda < 0.5$, > 0.95 in Figure 4.3 (this is not true for the integer Pareto front, since the 'Total understaffing'-objective is > 0 when $\lambda = 0.95$, but the points for when $\lambda > 0.95$ have still been omitted to allow for a clearer comparison).

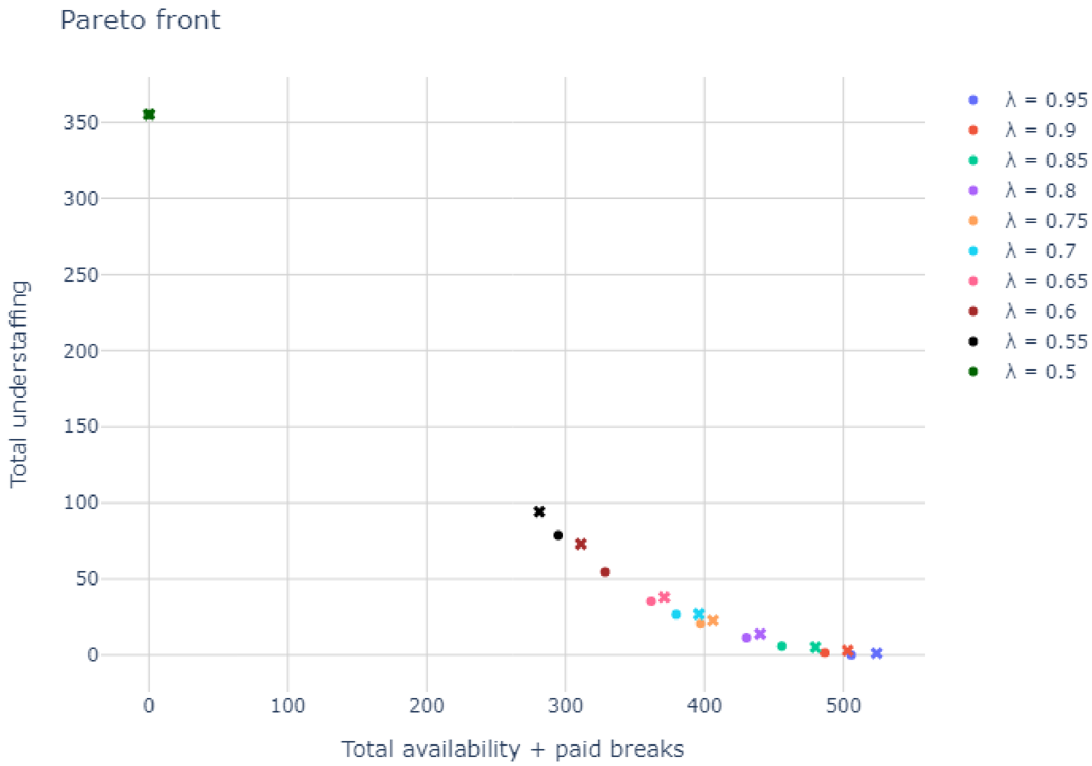


Figure 4.3: Illustration of Pareto optimal values in the two dimensional objective space for day 1, January 2022. All values come in pairs of the same colour, where the circle shapes belong to the Pareto front obtained from the LP-solutions to (4.1), and the cross shapes to the solutions to (4.1) with integer requirements on w_{jk} enforced, using the same columns as in the LP-solution. The integer Pareto points form an upper bound to the *optimal* integer Pareto points, while the LP-solutions form a lower bound. Note that when $\lambda = 0.5$, they both have the same values.

As we can see, Figure 4.3 clearly illustrates the conflicting nature of the two objectives (4.1) - we can see that it is not possible to minimise both simultaneously. The reason why the objectives have this property is due to the the summation of the availability with amount of understaffing per time step being bounded from below by D_{hk} in (4.1b). It is therefore of great interest that (4.1b) is studied closer to see how much of D_{hk} is covered by availability, rather than understaffing, which we will do in Section 4.3.

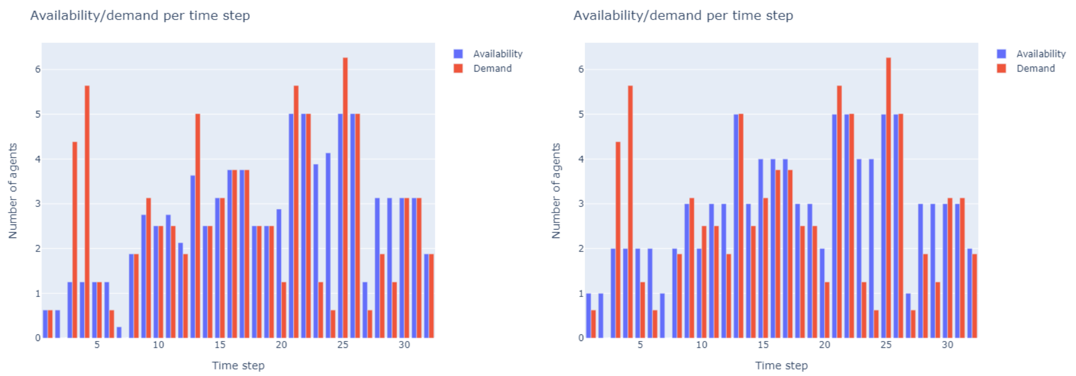
4.3 Demand vs. availablity

This section studies closer the constraint (4.1a) by studying how close the agent availability reaches up to the demand. The bar charts in Figure 4.4 illustrate $\sum_{h\in\mathcal{H}} D_{hk}$ and $\sum_{h\in\mathcal{H}}\sum_{j\in\mathcal{J}'} a_{hj}w_{jk}$ for all $\ell \in \mathcal{L}$ and one $k \in \mathcal{K}$ i.e. the total availability in relation to the total demand over 20 days for each competence.

(a) LP solutions. (b) Integer solutions.

Figure 4.4: Bar chart depicting total availability compared to total demand per day for competence 1 over the first 20 days in January 2022, with $\lambda = 0.7$. The left figure is the result obtained from the LP solution to (3.4) and the right figure is from the integer solution to (3.4) with the columns from the LP solution.

Because we sum over $h \in \mathcal{H}$, Figure 4.4 does not really say anything about how the availability versus demand varies between time steps within a certain day, which is important to have in mind since just because two bars have similar height it does not necessarily mean that the need for agents is even throughout the day. It is possible for example that some times of the day are severely understaffed while other times are quite overstaffed, but in the end they still sum up to equal the total demand. Thus, Figure 4.4 does not necessarily make any useful statements regarding agent availability on a day to day basis. However, Figure 4.4 does provide a sufficient overview of the overall demand and availability over a large time frame, as well as understaffing; by subtracting the bar values. An example of bar-pairs being of similar height in Figure 4.4, but having a quite uneven agent availability during certain parts of the day, is illustrated in Figure 4.5, where D_{hk} and $\sum_{j\in\mathcal{J}'} a_{hj}w_{jk}$ is plotted over h for day 1. Note, however, that calls that are not taken during a specific time step appear to 'vanish' in Figure 4.5, while in reality they are moved to the next time step, and then the next and so on until they get answered. For example, this means that having understaffing during one time step in Figure 4.5, followed by overstaffing in future time steps does not mean that those future time steps will actually be overstaffed in practice. Hence, the model in (3.4) is not entirely correct since it does not account for demand that is unable to be handled during time step h. But as we saw in Figure 4.4, most of the calls will be handled before the day is over, so it remains to see in practice if there are times during the day that have an excessive shortage of staff and how many calls are lost due to customer abandonment rates etc. (which also depend on how great the shortage of staff is).



(a) LP solution.

(b) Integer solution.

Figure 4.5: Bar chart depicting availability compared to demand per time step for competence 1, day 1, January 2022, with $\lambda = 0.7$. The left figure is the result obtained from the LP solution to (3.4) and the right figure is from the integer solution to (3.4) with the columns from the LP solution.

The optimal solution to the LP-relaxation of (3.4) tells how many agents with competence $k \in \mathcal{K}$ are needed for shift $j \in \mathcal{J}'$, and how much understaffing it then yields, for a certain value of λ. Summing w_{jk} over $j \in \mathcal{J}'$ we get the total amount of agents needed for competence $k \in \mathcal{K}$ and so in order to get an overview of how many agents are required per competence, and day, over a larger time-frame we illustrate this sum in Figure 4.6. We can see in Figure 4.6 that for competence 2 and 5 there is a quite significant drop in number of agents, due to the demand being lower (see Figures 4.2b and 4.2e in Section 4.1, as well as Figures A.2a and A.5a in Appendix A).

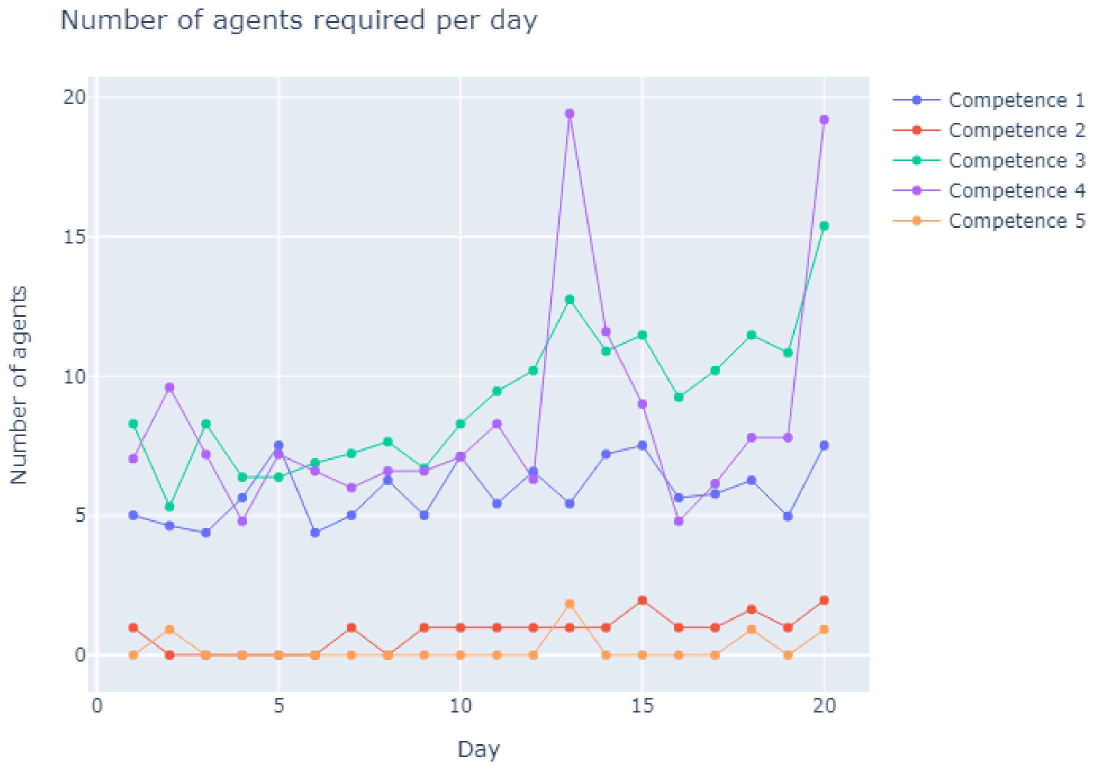


Figure 4.6: Plot of the number of agents required per competence over the first 20 days in January 2022, with $\lambda = 0.7$ (LP-solution).

4.4 Distribution of shifts among agents

Having obtained the LP solution to (3.4), we procure the integer solution by reinstating integer constraints and solving (3.4) with the same set of shifts used to get the LP solution. Recall from Section 3.2.3 that we use the integer solution, together with the generated shifts, as input to (3.9) in order to distribute the shifts among the available agents. The $x_{ij\ell}$-variable in (3.9) gives information about what shift $j \in \mathcal{J}$, agent $i \in \mathcal{I}$ is given on day $\ell \in \mathcal{L}$, and so, together with the set of shifts $\mathcal{J}_\ell$, one can create a schedule for every agent, as is illustrated in Figures 4.7 and 4.8.

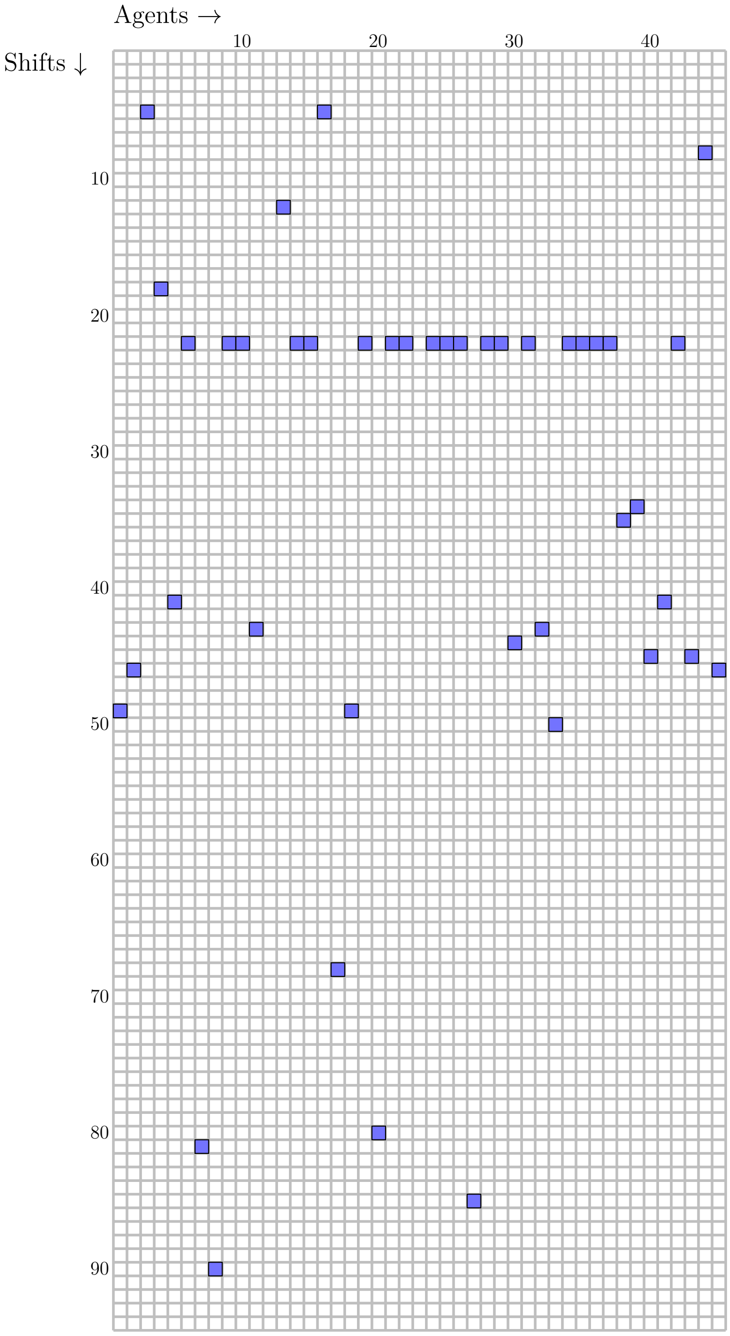


Figure 4.7: Grid depicting what agent works what shift (blue means they work) during day 1 in January 2022, with $\lambda = 0.7$. In total, 93 shifts were generated in addition to the starting shift (first row), adding up to 94 shifts total. The numberings denote every 10'th shift and agent respectively.

We can see from Figure 4.7 that 43 out of 45 agents work this particular day and in fact, the same 43 agents work the entire planning period $\mathcal{L}$, being the first 20 days in January 2022. This was deduced from the fact that all 20 days required 43 agents and the objective value of (3.9) also being 43, implying that the same 43 agents work during the entire planning period. Table 4.5 shows, for every shift in Figure 4.8, which agents (corresponding to Figures 4.7 and 4.1) work that shift during day 1, and what competencies they have.

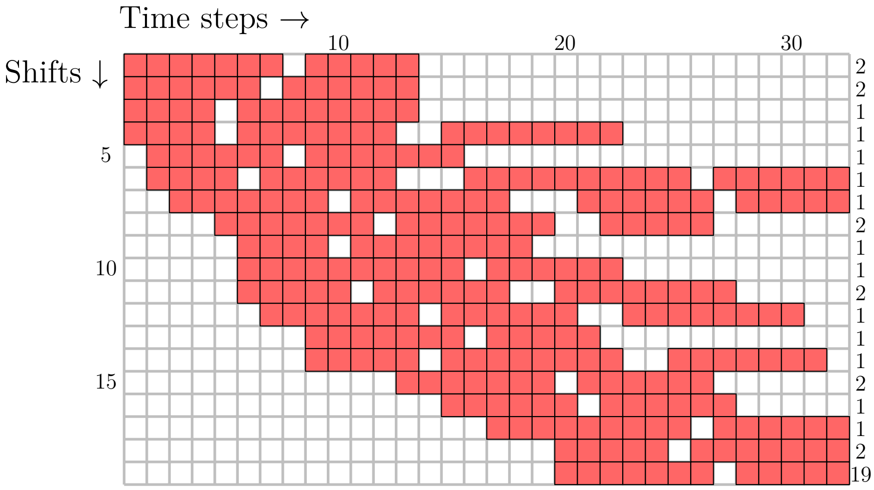


Figure 4.8: Grid depicting unique shifts from Figure 4.7, ordered by start time, during day 1 in January 2022, with $\lambda = 0.7$. The vertical numbers on the far left denote every 5'th shift, while those on the far right represent the amount of occurrences of the specific shift on every row. The horizontal numbers denote every 10'th time step. There are a total of 19 unique shifts, out of 43 total.

Table 4.5: Table of all the unique shifts used in the solution to the model in (3.9), which agents they were assigned to, as well as what competencies those agents have. The table values are for day 1 in January 2022, with $\lambda = 0.7$.

Shift	Agents	Competencies
1	[1], [18]	[3], $[1, 3]$
2	[5], [41]	$[1, 3]$, $[1, 3, 4]$
3	[27]	[4]
4	[39]	[3]
5	[7]	[3]
6	[38]	$[3, 4]$
7	[20]	$[1, 3]$
8	[3], [16]	$[3, 4]$, [4]
9	[13]	$[1, 2, 3]$
10	[8]	$[1, 3]$
11	[11], [32]	$[1, 2, 3]$, [3]
12	[33]	$[1, 4]$
13	[44]	$[2, 3, 4]$
14	[4]	$[1, 3, 4, 5]$
15	[40], [43]	$[1, 3, 4]$, $[1, 3]$
16	[17]	$[1, 2, 3]$
17	[30]	$[1, 3, 4]$
18	[2], [45]	[3], [3]
19	[6], [9], [10], [14], [15], [19], [21], [22], [24], [25], [26], [28], [29], [31], [34], [35], [36], [37], [42]	$[1, 2, 3, 4, 5]$, $[1, 2, 3, 4, 5]$, $[1, 3, 4]$, $[1, 4]$, $[1, 3]$, $[1, 2, 3, 4, 5]$, $[1, 2, 3]$, $[1, 4]$, $[1, 2, 3, 4, 5]$, $[3, 4]$, $[1, 3, 4]$, $[1, 3, 4]$, $[1, 3, 4]$, $[1, 3, 4]$, $[1, 2, 3, 4, 5]$, $[1, 3, 4]$, $[1, 2, 3, 4, 5]$, $[1, 2, 3, 4, 5]$, $[1, 2, 3, 4, 5]$

There is clearly a bias towards shorter shifts, which partly has to do with the model prioritising shorter over longer shifts if they have the same reduced costs. The reason for this particular prioritisation is simply because it is cheaper to assign a short shift to an agent compared to a long shift.

4.5 Model comparison to the in-use system

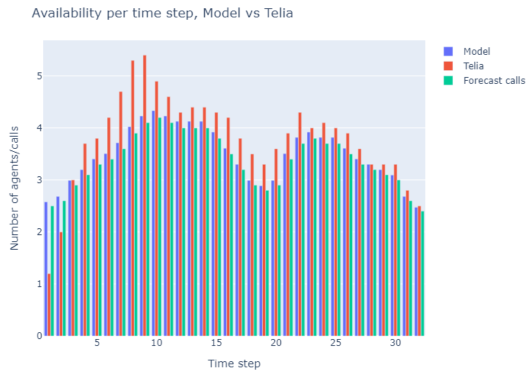
Based on a a forecast number of incoming calls per time step provided by Telia, we will compare the availability given by the model with the availability derived at the company using the same values, all forecast values that were provided are illustrated in Figures 4.9 and A.7a–A.10a. Note that because Telia does not use integer constraints on the availability, we use the LP solution for a fairer comparison. Keep in mind that the green bars in Figure 4.9 represent the prediction of the amount of calls that will come in per time step - we use these values to calculate the demand with (3.1), while the company uses Erlang A. We make sure to allow for minimal understaffing by letting $\lambda = 0.95$, and we can then conclude from Figure 4.9 that the company seems to generally derive a higher demand than the model. The value of λ makes an insignificant difference in this case, because the demand overall is low enough so that it can be reached at a minimum cost, regardless of the value of lambda, as long as it remains > 0.5. In general however, if the demand were higher, increasing λ would be a valid way of increasing availability in the model.

As mentioned in Section 1.1.1, the company takes into account several parameters when deciding demand, and so as a brief demonstration, Figure 4.9b illustrates the results from the model with the same forecast values and T_{hk}-values from Table 4.4 but also using a few of the more commonly used parameters at the company (see Figures A.7a–A.10b in Appendix A for competencies 2–5). Table 4.6 shows some commonly used parameters, together with their values, provided by Telia, which are added multiplicatively to the numerator of the demand calculation in (3.1).

Table 4.6: Table showing the values, provided by Telia, of the most common parameters used in deriving demand at the company. The values are added multiplicatively to the demand, thus resulting in a 64.64% increase.

Parameter	Definition	Value
Shrinkage	Agent absence (vacations, sick leaves, meetings, etc.)	12%
Service level	Percentage of calls answered over the entire day	75%
Occupancy	Percentage of time spent handling calls	88%
Efficiency	Percentage of agent adherence to their schedule	95%

(a) No parameters.



(b) With parameters.

Figure 4.9: Bar chart of availability per time step, day 1, competence 1, with $\lambda = 0.95$, compared with the derived availability at Telia for the same day and competence (LP solution). The values given by the model, and Telia, were both derived from the same data of forecast calls. In Figure 4.9a no parameters are included other than incoming calls ($d_{\ell hk}$), average handling time (T_{hk}) and preferred handling time (F_{hk}), while Figure 4.9b also includes 12% Shrinkage, 75% Service level, 88% Occupancy and 95% Efficiency, adding up to a 64.64% increase in demand when multiplying the percentages.

4.6 Benchmarks

This section includes benchmarks for finding the LP-solution to (4.1) with column generation, which is the part of the implementation responsible for almost all of the total computation time. The Gurobi solver, used in this project, will be compared to the free open source solver GLPK version 1.1.2 [17], in order to deduce whether one can achieve satisfactory results without having to put large amounts of money on using Gurobi. All benchmark values presented in Table 4.7 were obtained by using the "BenchmarkTools" package [18] in Julia.

Table 4.7: Table of benchmark values, and memory usage, for finding the LP-solution to (4.1) in Section 4.2, for day 1 in January 2022, with $\lambda = 0.7$. The sample size is 10 samples.

Solver	Mean ± std. (s)	Median (s)	Min – max (s)	Memory (GiB)
Gurobi	62.076 ± 2.103	62.193	57.560 – 65.355	7.86
GLPK	245.861 ± 11.173	244.812	232.316 – 266.516	8.88

Clearly, the GLPK solver is significantly slower than Gurobi. It takes over 4 minutes for GLPK to generate all the columns needed to find the LP-solution to (4.1), which is about 4 times longer than Gurobi, with GLPK also having a larger memory usage.

5

Conclusion

Throughout this text, we have studied the ability to schedule agents in a call centre using mathematical optimisation theory. In addition to deriving how many agents should work, we have also presented a way of simulating their schedules on a time step by time step basis, for any planning period, given sufficient data. Using simulations before actually implementing an idea into the real world can allow for great time- and cost efficiency. Therefore, this project could be a useful tool when making important decisions regarding staff allocation in any call centre.

It is quite likely, however, that the acquired results are optimistic due to the fact that demand is given time step per time step, without any regard of previous time steps. Assuming (realistically) that not all calls are handled within the expected handling time and/or all calls could not be answered immediately due to shortage of staff, some of the demand gets carried over to future time steps, which the model does not account for; we see a clear example of this in Figure 4.5. On the other hand, as also mentioned in Section 4.3, most of the total incoming calls seem to get answered eventually, so it remains to see how the model would work in practice before any final conclusions can be drawn.

As seen in Section 4.5, parameters play a large role in deriving accurate demand values, and when some of those are taken into account we get results that are quite close to the ones from the company. Implementing more adjustments to the demand calculation would likely give even more accurate results. We saw in Section 4.3 that only 43 out of 45 agents were needed. When parameters are included however, all 45 agents are required. Clearly the number of required agents also depends on how heavily understaffing is punished, and so simply increasing λ can also raise the amount of agents required. For higher demands, it might be of interest to derive how many agents are needed to satisfy demand at a certain level, which can be done by removing the constraints (3.4c) and (3.4d) from (3.4), and then extending the set of all agents $\mathcal{I}$ according to the solution so that (3.9) remains feasible.

In conclusion, the model seems to perform quite well as compared to the current system at the company, but it would require some further alterations to the implementation in order to include all the relevant parameters present in a real-world call centre, so that it can be compared to the current system under equal conditions. The results acquired this far, however, seem to imply that once the mentioned alterations have been made, the model will likely perform at least as good as the current system.

5.1 Complexity of the implementation

The overall run time of the implemented program is of great importance should it be used as a legitimate way to produce schedules over long periods of time at the company. One rather straight forward way of reducing computation times is to include more columns per iteration, rather than only one at a time. Adding the column with the smallest negative reduced cost equates to reducing the objective as much as possible per unit of increase from zero for the incoming column, this means that it is possible that the specific column does not increase very much and thus not contributing to the most reduction in the objective. Even if the incoming column does contribute the most to reducing the objective, there could be other columns that also have a negative reduced cost that also reduce the objective. In fact, adding *any* column, as long as its reduced cost is negative, will reduce the objective but possibly at bit less than the one with the smallest negative reduced cost. The addition of more columns might enable finding a solution faster because there will be less columns to 'reconsider' in future iterations, and is in fact one of the more widely used methods in reducing computation times [8].

The solution obtained from generating columns only solve the LP-version of the original problem - it is almost always the case that once we reinstate the integer requirements we no longer have enough columns to find the optimal solution. Usually, once the LP-solution is found by column generation the 'Branch and price' algorithm is then used to find an integer solution, which in turn has an exponential computational complexity. The Pareto fronts, illustrated in Figure 4.3, seem to imply that there is no huge difference between the upper bound integer solutions and the optimal integer solutions to (3.4). So it remains to see through further testing whether obtaining an optimal integer solution is necessary for producing satisfactory results and if it is worth the extra effort in implementing it, as well as the extra computational complexity and memory usage. There are also quite a few ways of reducing computation times for Branch and price that may or may not be applicable to this problem, some of which can be found in [19], [20] and [21], among many others. Specifically, [21], has many references to Branch and price methods and would be a good starting point for the interested reader.

Another way of improving computation times, also mentioned in [8], is by terminating the algorithm prematurely when no significant improvements to the objective have been made after a certain amount of iterations. This prevents the algorithm from getting 'stuck' trying to find the optimal solution when a sub-optimal solution would suffice.

As mentioned in Chapter 4, the Gurobi solver was used to procure all the results, which is one of the fastest solvers on the market and thus quite expensive. The alternative is to either use another method, e.g. create a list of all allowed shifts and then pick from that list every iteration based on the reduced costs, as mentioned in Section 3.2.2, or to use an open source solver, which might yield longer computation times, depending on the model. We presented some benchmarks in Section 4.6 where we compared the commercial solver Gurobi to the free open source solver GLPK, and we saw that clearly, for this problem, there was quite a significant difference in computation times, where Gurobi was far superior.

5.2 Feasible starting solution for column generation

The column generation algorithm requires a feasible solution in order to get started and so one may wonder what an adequate choice of starting solution might be and how it is found. Due to how the algorithm works, if there exists an optimal solution it will always be found no matter what solution it starts with [8]. As mentioned in Section 4.2, the starting set was chosen randomly to be one single long shift and there are surely better choices of starting sets that speed up the column generation convergence. However, for this problem in particular, there was no intuitive way of constructing a starting set that can be proven to be better than others, and nothing particularly helpful was found in the literature either. Due to how the model in (3.4) was defined, allowing understaffing ensures that a feasible solution can be found with practically any starting set, including the empty set. So the problem of finding a feasible starting solution is quite easy for this problem, but finding one that reduces convergence times the most remains nontrivial.

The easiest way of perhaps reducing the convergence time somewhat, but not necessarily optimally, is to simply start with a larger set and make sure that the union of the shifts in that set cover the opening hours. However, the trade off is unnecessary memory allocation, as well as potentially increased computation times due to the solver having to manage larger matrices - the entire point of using column generation is to reduce the size of the problem as much as possible so that it becomes easier to handle. For this reason, it might be beneficial to try and find somewhat of a middle ground in run time versus resource allocation, so that they remain within reasonable ranges. There are many methods and strategies that try to optimise the efficiency and convergence of column generation, many of them can be found in [8] and [21] and the references therein.

5.3 Further research

There are a few things that could be further worked upon in the future, should the interest arise, such as properly designing a user interface so that anyone not necessarily initiated with the theory beforehand is able to produce results suited to individual needs. Furthermore, one might want to look deeper into the demand of staff itself; rather than deriving it directly from a set of old data one might try to predict future demand using for example Markov chain Monte Carlo methods and perhaps obtain better values for the demand than those used in this project. One needs to make sure however that the model remains linear if such a method is to be implemented. As also mentioned in 5.1, further testing that compares the results obtained with- and without Branch and Price to the real world call centre needs to be conducted in order to properly state whether implementing Branch and price is necessary or not. Also, the implementation can always be more efficient in order to enable faster computation times.

It is possible to modify the model in (3.4) so that unanswered calls are taken into account, and one way of doing this could be for example to introduce an 'inventory' constraint: $q_{h+1} = (1 - f) \cdot (q_h + d_h - r_h)$, where q_h is the number of calls waiting for service at time step $h \in \mathcal{H}$, d_h is the number of incoming calls, r_h is the number of handled calls, and $0 < f \leq 1$ is the abandonment rate (percentage of calls abandoned by customers). The variable q_{h+1} should be added to the demand at time step $h + 1$ on the right hand side of the constraint (3.4b) - customer abandonment rate f can also be defined as a function of this sum, since customers are more likely to abandon the call the longer the wait time is.

Finally, if it is of interest that the form of employment is taken into account in the model formulations, since it is not uncommon for call centres to have a mixture of full- and part time employees. One can do so by perhaps constructing separate models to solve for every employment type, where the set of all agents $\mathcal{I}$ is split up so that agents with the same form of employment (or that work the same amount of hours) are in the same set. If one solves the model consisting of e.g. full time agents, one can then use the amount of understaffing (if there is any) from that solution as input to the model consisting of part time agents so that they can 'fill' the gaps, since their shifts are shorter and therefore more flexible. It would be beneficial to also take into account agents that e.g. can work only day time, or prefers working in the evening, particularly for call centres that have longer opening hours. There are many modifications that can be applied to the models in this text to allow for further generalisations.

Bibliography

1. Garnett, O., Mandelbaum, A. & Reiman, M. Designing a Call Center with Impatient Customers. *Manufacturing & Service Operations Management* **4(3),** 208–227 (July 2002).
2. Calabrio. *Erlang formula overview.* Calabrio Help Center. `https://help.calabrio.com/doc/Content/user-guides/schedules/about-erlang-formula.htm?tocpath=Schedulers%7COptimize%20schedule%7C_____3`.
3. Mandelbaum, A. & Zeltyn, S. in *Advances in Services Innovations* (eds Spath, D. & Fähnrich, K.-P.) 17–45 (Springer, Berlin, Heidelberg, 2007).
4. Cordone, R., Piselli, A., Ravizza, P. & Righini, G. Optimization of Multi-skill Call Centers Contracts and Work-shifts. *Service Science* **3(1),** 67–81 (Mar. 2011).
5. Bard, J. F. & Purnomo, H. W. Preference scheduling for nurses using column generation. *European Journal of Operational Research* **164(2),** 510–534 (July 16, 2005).
6. Xu, N., Zhao, M., Liu, Y. & Yang, F. A Column Generation Based Algorithm for Airline Cabin Crew Rostering Problem. *2018 IEEE 4th International Conference on Computer and Communications (ICCC), Computer and Communications (ICCC), 2018 IEEE 4th International Conference on,* 2511–2515 (Dec. 1, 2018).
7. Um-in, N. & Tharmmaphornphilas, W. *Agent scheduling of call center using decomposition technique* in *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (Dec. 2017), 80–84.
8. Desrosiers, J. & Lübbecke, M. E. in *Column Generation* (eds Desaulniers, G., Desrosiers, J. & Solomon, M. M.) 1–32 (Springer US, Boston, MA, 2005).
9. Gustavsson, E., Patriksson, M., Strömberg, A.-B., Wojciechowski, A. & Önnheim, M. Preventive maintenance scheduling of multi-component systems with interval costs. *Computers & Industrial Engineering* **76,** 390–400 (Oct. 1, 2014).

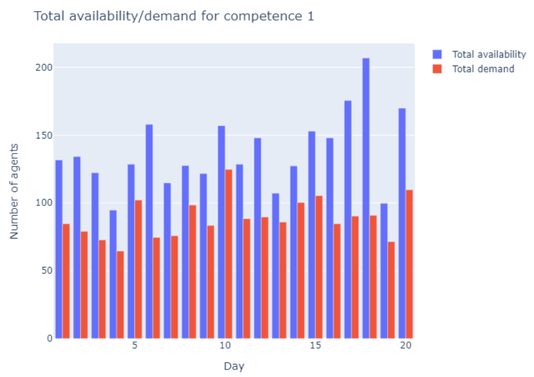
10. Strömberg, A.-B. *Lagrangean duals and subgradient methods for linear and mixed-binary linear optimization* compendium for the course TMA522 Large-scale Optimization, Chalmers. Oct. 29, 2022.
11. Andréasson, N., Evgrafov, A. & Patriksson, M. *An introduction to continuous optimization : foundations and fundamental algorithms* Third edition adjusted (Studentlitteratur, 2016).
12. Ehrgott, M. *Multicriteria Optimization* (Springer-Verlag, Berlin/Heidelberg, 2005).
13. *Graph theory | Problems & Applications | Britannica* May 24, 2023. `https://www.britannica.com/topic/graph-theory`.
14. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A Fresh Approach to Numerical Computing. *SIAM Review* **59(1),** 65–98 (Jan. 2017).
15. Dunning, I., Huchette, J. & Lubin, M. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review* **59(2),** 295–320 (Jan. 2017).
16. *Gurobi Optimizer Reference Manual* Gurobi Optimization. 2023. `https://www.gurobi.com`.
17. *GLPK - GNU Project - Free Software Foundation (FSF)* `https://www.gnu.org/software/glpk/glpk.html`.
18. *BenchmarkTools.jl* `https://docs.juliahub.com/BenchmarkTools/ZXPQo/1.3.2/`.
19. Desaulniers, G., Pecin, D. & Contardo, C. Selective pricing in branch-price-and-cut algorithms for vehicle routing. *EURO Journal on Transportation and Logistics. Special Issue: Advances in vehicle routing and logistics optimization: exact methods* **8(2),** 147–168 (June 1, 2019).
20. Desrosiers, J. & Lübbecke, M. E. in Cochran, J. J., Cox, L. A., Keskinocak, P., Kharoufeh, J. P. & Smith, J. C. *Wiley Encyclopedia of Operations Research and Management Science* eorms0118 (John Wiley & Sons, Inc., Hoboken, NJ, USA, Jan. 14, 2011).
21. Lübbecke, M. E. & Desrosiers, J. Selected Topics in Column Generation. *Operations Research* **53(6),** 1007–1023 (Dec. 2005).

A

Supplementary figures

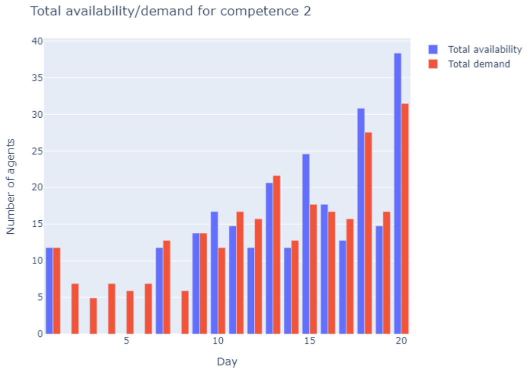
We include figures for the rest of the competencies when $\lambda = 0.7$, as well as when $\lambda = 0.95$ to see how the solution looks when we allow no understaffing. Further, the rest of the availability comparisons with Telia are also included for day 1, January 2023 with- and without parameters. All results, except for Figures A.7a-A.10b, are over, or within, the first 20 days in January 2022.

(a) $\lambda = 0.7$.

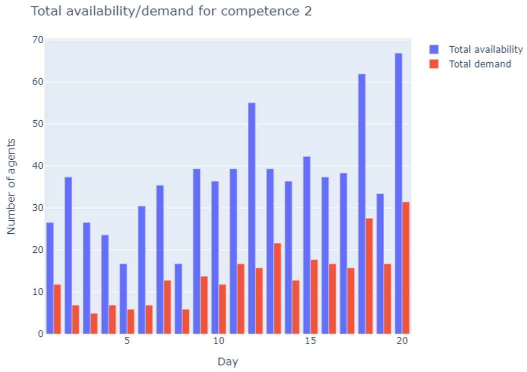


(b) $\lambda = 0.95$.

Figure A.1: Comparison of total availability/total demand per day for competence 1, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution). Figure A.1a also occurs as Figure 4.4.

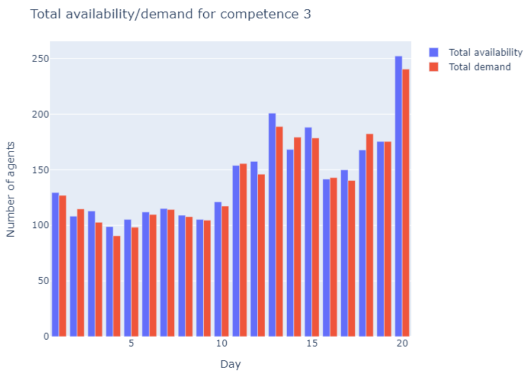


(a) $\lambda = 0.7$.



(b) $\lambda = 0.95$.

Figure A.2: Comparison of total availability/total demand per day for competence 2, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution).



(a) $\lambda = 0.7$.

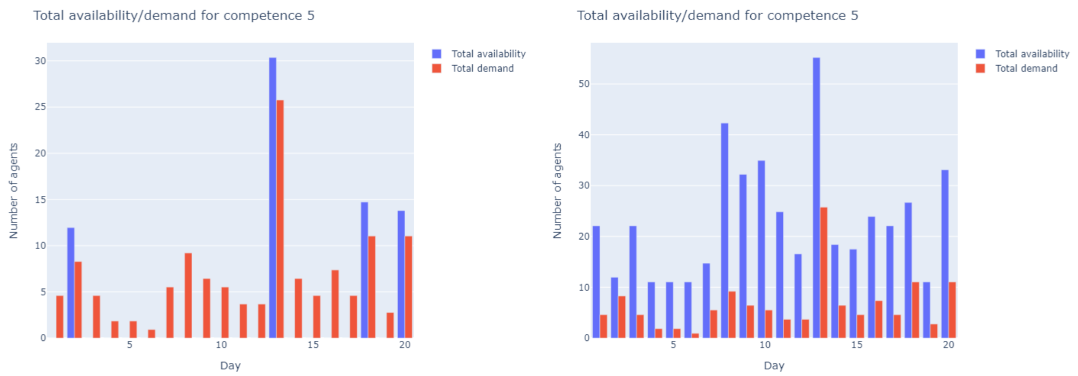
(b) $\lambda = 0.95$.

Figure A.3: Comparison of total availability/total demand per day for competence 3, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution).

(a) $\lambda = 0.7$.

(b) $\lambda = 0.95$.

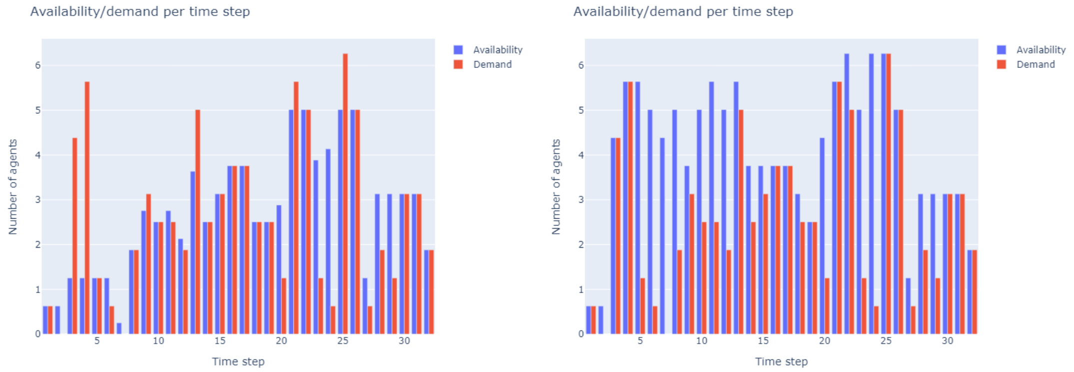
Figure A.4: Comparison of total availability/total demand per day for competence 4, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution).



(a) $\lambda = 0.7$.

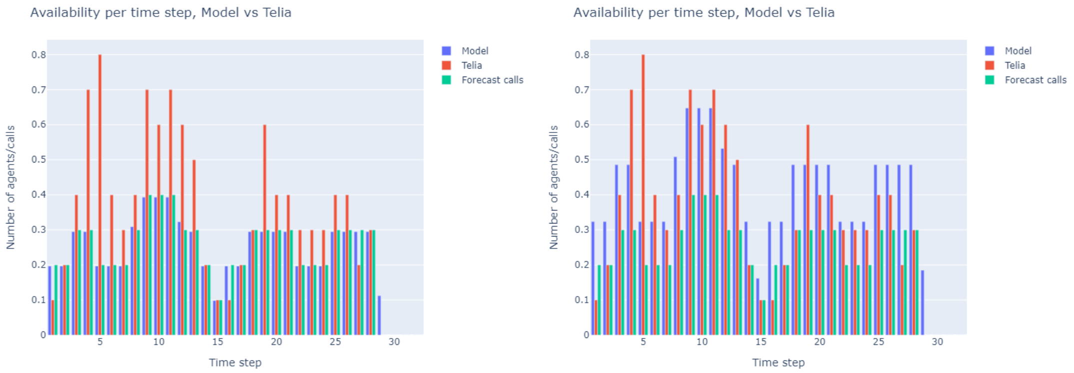
(b) $\lambda = 0.95$.

Figure A.5: Comparison of total availability/total demand per day for competence 5, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution).



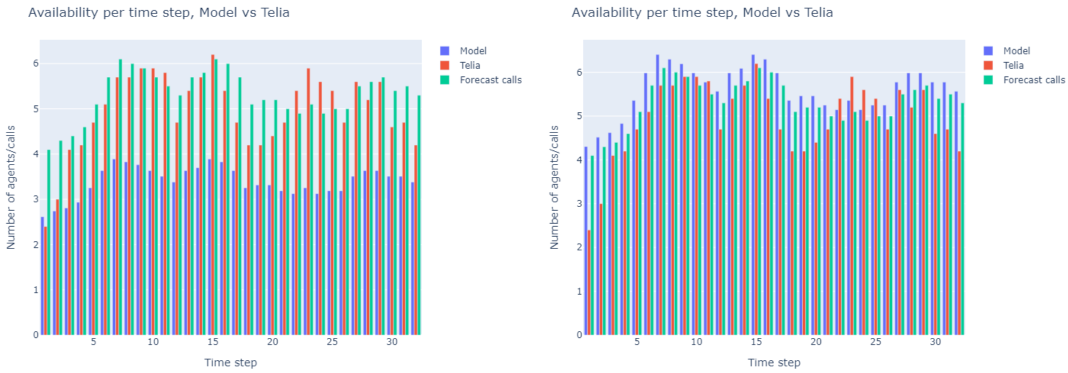
(a) $\lambda = 0.7$. (b) $\lambda = 0.95$.

Figure A.6: Comparison of availability per time step, day 1, competence 1, with $\lambda = 0.7$ and $\lambda = 0.95$ respectively (LP solution). Figure A.6a also occur as Figure 4.5a.



(a) No parameters. (b) With parameters.

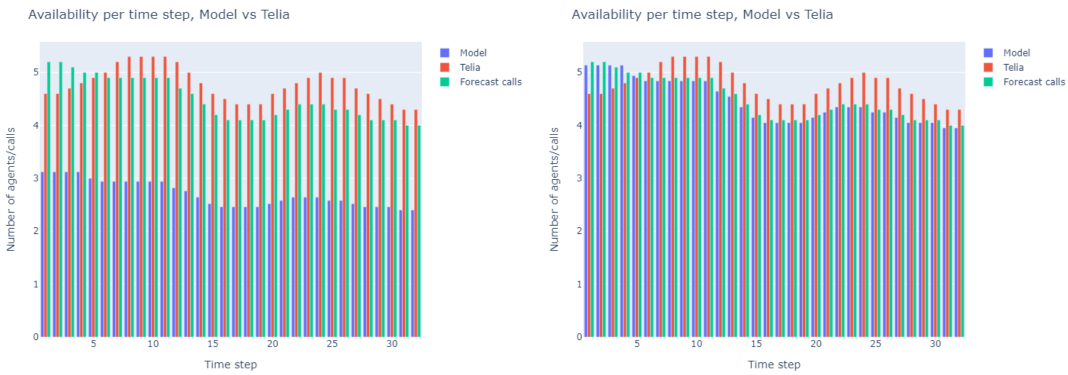
Figure A.7: Bar chart of availability per time step, day 1, competence 2, with $\lambda = 0.95$, compared with the derived availability at Telia for the same day and competence (LP solution). The values given by the model, and Telia, were both derived from the same data of forecast calls. In Figure A.7a no parameters are included other than incoming calls ($d_{\ell hk}$), average handling time (T_{hk}) and preferred handling time (F_{hk}), while Figure A.7b also includes 12% Shrinkage, 75% Service level, 88% Occupancy and 95% Efficiency, adding up to a 64.64% increase in demand when multiplying the percentages.



(a) No parameters.

(b) With parameters.

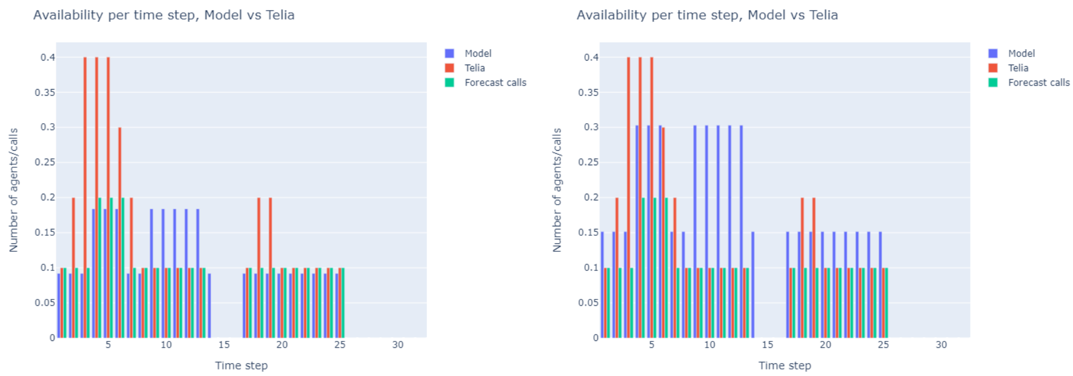
Figure A.8: Bar chart of availability per time step, day 1, competence 3, with $\lambda = 0.95$, compared with the derived availability at Telia for the same day and competence (LP solution). The values given by the model, and Telia, were both derived from the same data of forecast calls. In Figure A.8a no parameters are included other than incoming calls ($d_{\ell hk}$), average handling time (T_{hk}) and preferred handling time (F_{hk}), while Figure A.8b also includes 12% Shrinkage, 75% Service level, 88% Occupancy and 95% Efficiency, adding up to a 64.64% increase in demand when multiplying the percentages.



(a) No parameters.

(b) With parameters.

Figure A.9: Bar chart of availability per time step, day 1, competence 4, with $\lambda = 0.95$, compared with the derived availability at Telia for the same day and competence (LP solution). The values given by the model, and Telia, were both derived from the same data of forecast calls. In Figure A.9a no parameters are included other than incoming calls ($d_{\ell hk}$), average handling time (T_{hk}) and preferred handling time (F_{hk}), while Figure A.9b also includes 12% Shrinkage, 75% Service level, 88% Occupancy and 95% Efficiency, adding up to a 64.64% increase in demand when multiplying the percentages.



(a) No parameters.

(b) With parameters.

Figure A.10: Bar chart of availability per time step, day 1, competence 5, with $\lambda = 0.95$, compared with the derived availability at Telia for the same day and competence (LP solution). The values given by the model, and Telia, were both derived from the same data of forecast calls. In Figure A.10a no parameters are included other than incoming calls ($d_{\ell hk}$), average handling time (T_{hk}) and preferred handling time (F_{hk}), while Figure A.10b also includes 12% Shrinkage, 75% Service level, 88% Occupancy and 95% Efficiency, adding up to a 64.64% increase in demand when multiplying the percentages.

B

Code

The code written to produce the results is available at

`https://github.com/Arbeeer/MVEX03-Public.git`

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY