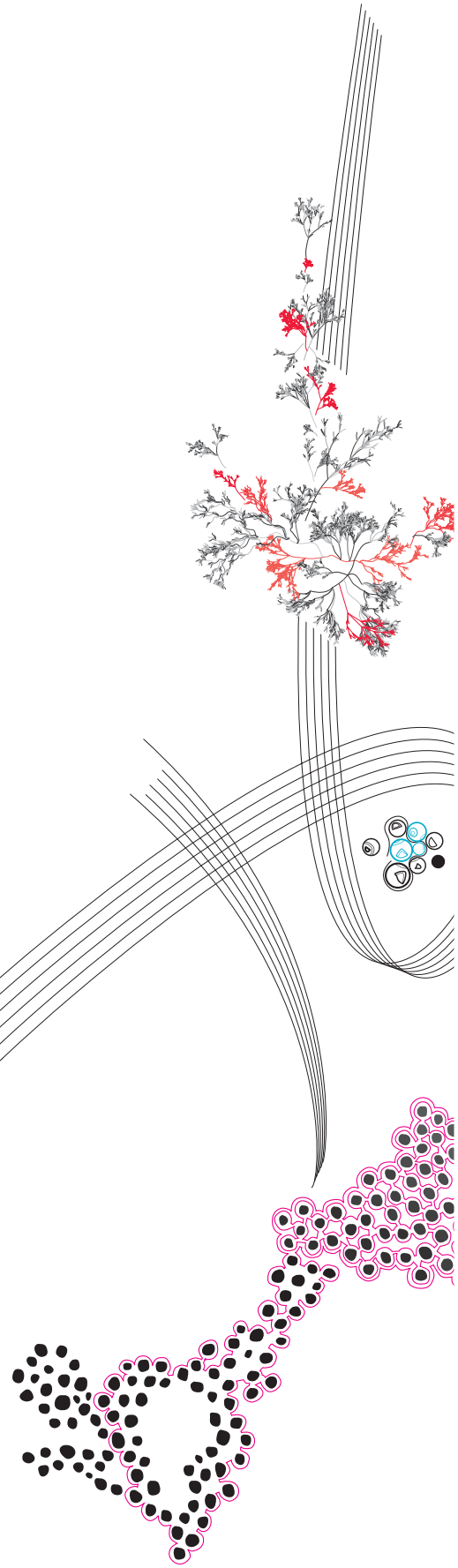
BSc Thesis Industrial Engineering and Management



# Optimizing Workforce Requirement Planning in Warehousing: Solving a Large-Scale MIP

Koray Tjark Durgut
S2816776

Supervisors: Dr. S. M. Meisel & Dr. D. Guericke

October 3, 2024

Department of Industrial Engineering and Management
Faculty of Behavioural Management and Social Sciences

UNIVERSITY OF TWENTE.

flaschenpost.de

# Management Summary

This research is conducted in cooperation with flaschenpost SE in Münster, Germany. Flaschenpost is an online supermarket. Having started its business with only beverages, the company has quickly extended its offering to include fresh and frozen food, products for personal care and hygiene, cleaning products, and even office and pet supplies.

## Problem Description

Scheduling workers in a warehouse or factory is one of the key decisions in operations since it greatly contributes to a company's overall costs and efficiency. Therefore, there is great interest in scheduling workers optimally. One of the most common methods to generate schedules is mixed integer linear programming (MIP). While MIP have multiple benefits, their formulations often become increasingly large and hard if we want to model a problem as close to reality as possible. We are given a large-scale MIP by flaschenpost and want to figure out how well we can solve it while not violating the given time constraints to generate shift schedules.

## Method

We decided to customize the logic of a Branch-and-Cut algorithm using a callback function in the code. The callback function is executed at different points in the Branch-and-Cut tree. It checks whether the current integer feasible solution violates at least one of the demand injection and propagation constraints. We do not add these constraints from the outset but add them iteratively only when needed. This way, we potentially significantly reduce the model's calculation time since we ideally end up with a smaller problem formulation.

## Results

We show that using our customized Branch-and-Cut logic can significantly reduce the calculation time for models using double time-indexed variables. Nonetheless, these improvements were not significant enough to reduce the calculation time of the fully specified workforce requirement planning MIP. The shift and area assignment have proven to contribute more to the model's complexity than expected and, therefore, need to be tackled in further research. We successfully simplified the model to a degree where the calculation time is fast enough to satisfy the company's requirements. These schedules only represent heuristic solutions and thus do not guarantee optimality. Nonetheless, the results are usable and indicate a first starting point for the shift scheduling problem. Additionally, this heuristic solution might become more relevant in further efforts to solve the original MIP as they serve as upper bounds for the objective value of the fully specified model.

Using the heuristic solution we managed to solve the problem within five minutes for all tested scenarios of demand and even with small a interval length $\Delta = 5$. This is a meaningful decrease in the required solution time. In comparison, the fully specified model with and without our BaC logic does not find an optimal solution for problem instances where $\Delta = 5$ and. Here, the optimality gap (i.e., the relative distance between the upper and lower bound) is approximately 50% on average after 8 hours of run time. Besides the performance increase in calculation time, the quality of the heuristic solution is much worse compared to the aforementioned upper and lower bounds. Using the bounds as benchmarks, the objective value of the heuristic solution is 25% to 135% worse. As already indicated, we conclude that further research is required to solve the problem satisfactorily. Given the apparent complexity of the model, we suggest pursuing a heuristic solution.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Flaschenpost SE, headquartered in Münster, Germany, is one of the country's leading delivery companies specializing in groceries and beverages. With over 30 warehouse locations in Germany, the company delivers to over 190 German cities (flaschenpost SE, 2024). Each year 20,000 employees work on fulfilling over 10,000,000 orders.

While the growth of revenue of online stores in Germany has slowed down compared to the years of the COVID-19 pandemic, the grocery sector has grown over-proportionally by 8% in 2022 (IFH Köln, 2023). Groceries and other quickly selling products such as hygiene and body-care products form the so-called Fast Moving Consumer Goods (FMCG). According to IFH Köln (2023), the FMCG sector is growing fast and already made up 12.1% of the total online revenue generated in Germany in 2022. Since the market in which flaschenpost operates is still relatively small but growing quickly, there exists a lot of competition, and companies have to work hard to gain a competitive advantage.

Most grocery delivery companies offer same-day delivery to their customers. In comparison to its competitors, however, flaschenpost promises to deliver orders within 120 minutes. The commitment to delivering within a few hours results in a delivery service called rush, instant, or real-time delivery (Wölck & Meisel, 2022). This type of delivery entails several challenges for its providers. Customers desire fast deliveries more frequently which means that schedules of processes and delivery routes need to be adjusted multiple times on the same day to fulfill demand. Processes and routes are determined by algorithms taking into account the ever-changing input such as incoming orders and available workforce or resources. The dynamism of the whole system creates the need for these algorithms that run in little time and still create sufficiently accurate and realistic results (e.g., shift schedules) ensuring that operations and delivery run efficiently.

The goal of this thesis is to contribute to the optimization of the operations in the company's warehouses. We aim to achieve this by solving the given MIP, which will output a workforce requirement plan for the warehouse. This plan will eventually be used to schedule the workers. By exploring solution different solution methods we aim to contribute to the company's efficiency and competitive advantage. The motivation for this thesis additionally stems from the increasing demand for instant delivery services and the subsequent need for companies to optimize all of their processes.

# 2. Problem Description

In this chapter, we outline the business problem lying at the core of this thesis and provide a comprehensive overview of its context. Section 2.1 introduces the business challenge, followed by a detailed identification of the core problem. Here we highlight the gap between the current situation and the desired state. The scope of research is defined afterward to establish boundaries for our research. In section 2.2 we present the specific assignment we received from the company, formulate the main research question, and line out our problem-solving approach. Finally, the chapter addresses the knowledge problems and we explain how we design our research to resolve them.

## 2.1 Business Problem

As explained in Chapter 1, Germany's online grocery delivery market is highly competitive. Flaschenpost is, therefore, continuously improving its operations to maximize profits and increase its market share. The overarching operational goals the company pursues are:

- accepting as many orders as possible
- fulfilling orders punctually (i.e., within 120 minutes)
- generating as few work hours as possible (e.g.., by not having too many employees working at once)

First, accepting as many orders as possible is crucial for increasing the revenue. By handling a high volume of orders, flaschenpost can increase its sales and expand its customer base, which is essential for growth. Second, fulfilling orders on time is essential to customer satisfaction and retention. The self-imposed time limit of 120 minutes manifests the value proposition of flaschenpost: it delivers orders much faster than its competitors. Lastly, generating as little work time as possible to achieve the previous goals is important for cost reduction. Operational efficiency and reducing work time do not only reduce labor costs but also optimize resource utilization. This can lead to higher profitability and allow the company to reinvest savings (monetary and workforce) into other business areas.

In the remainder of this section, we will identify and evaluate different areas of improvement in the operations of flaschenpost. From these, we will select a core problem that our research aims to solve and thus contribute to reaching the above-mentioned goals.

### 2.1.1 Core Problem Identification

We identify the problems of the operations department of flaschenpost in Figure 2.1. This figure illustrates the problems the company is facing concerning the scheduling of warehouse

workers as well as the causal relations between them. Finally, we show which of these problems is the core problem (i.e., root cause) and which are action problems (i.e., symptoms of the core problem). This allows us to describe the current situation of the warehouse workforce scheduling in a problem cluster. The aforementioned operational goals are now expressed as action problems as defined by Heerkens et al. (2017). We can do this since the company believes there is room for improvement for all of them. For this thesis, we only look at the operations of the warehouse and neglect other parts of the business.

Demand variability is not chosen as a core problem because it is not possible or hard to influence directly, especially when using common Industrial Engineering and Management methods. The demand forecast is also not considered further since another company department handles it. Evaluating the forecast quality and changing the procedure is, therefore, outside of the scope of our project. Nonetheless, we must rely on the forecast as input for the mathematical model. Thus, company employees and we need to pay close attention to its quality when working with the mathematical model.

Our research will focus on the remaining problem: the warehouse worker requirement planning (WRP) is done manually. This is a crucial problem to solve since it directly influences the extent to which the overarching company goals are achieved. A sub-optimal scheduling of the warehouse workforce leads to a capacity-demand mismatch. This mismatch manifests itself either through over- or understaffing in the warehouse. The first leads to higher than necessary labor costs and decreased productivity per worker, while the latter can result in too high employee utilization and backlogging of orders. If the workers work at a high utilization level or, in the worst case, even exceed their limit, high stress levels and dissatisfaction with their work will follow. Backlogging orders poses a problem because the company cannot deliver all orders on time or accept new orders since the system is already operating at full capacity.

Furthermore, we will be able to test the validity of our solution more easily if we choose workforce scheduling as a core problem than with the other options. Comparing the schedules made by the current planning method to our solution should prove to be a straightforward task. We expect that improving the requirement planning will lead to less capacity-demand mismatch and, therefore, improve the performance of flaschenpost in all three main action problems/operational goals. An overview of the potential core problems can also be found in Table A.1 of Appendix A.

### 2.1.2 Norm and Reality

According to Heerkens (2017), an "action problem is the discrepancy between norm and reality as perceived by the problem owner". The core problem we have selected for the bachelor thesis research can be formulated using this terminology as well.

Currently, the reality at flaschenpost is that the WRP for the warehouses is done manually using heuristics. In addition to this, the main problem is that there is no clear problem formulation. Consequently, it is also unclear what goals should be achieved on a tactical level and how these are defined and measured. Lastly, the company cannot optimally schedule the warehouse workforce since heuristics are used.

As a norm, the company wants to ensure that the requirement planning is done systematically for all warehouses in Germany. This includes a clearly defined problem formulation and a goal-driven method for the requirement planning that guarantees an optimal solution. Optimality, in this specific case, is defined as incurring as few tardiness penalties and shift assignment costs as possible while still fulfilling demand on time.

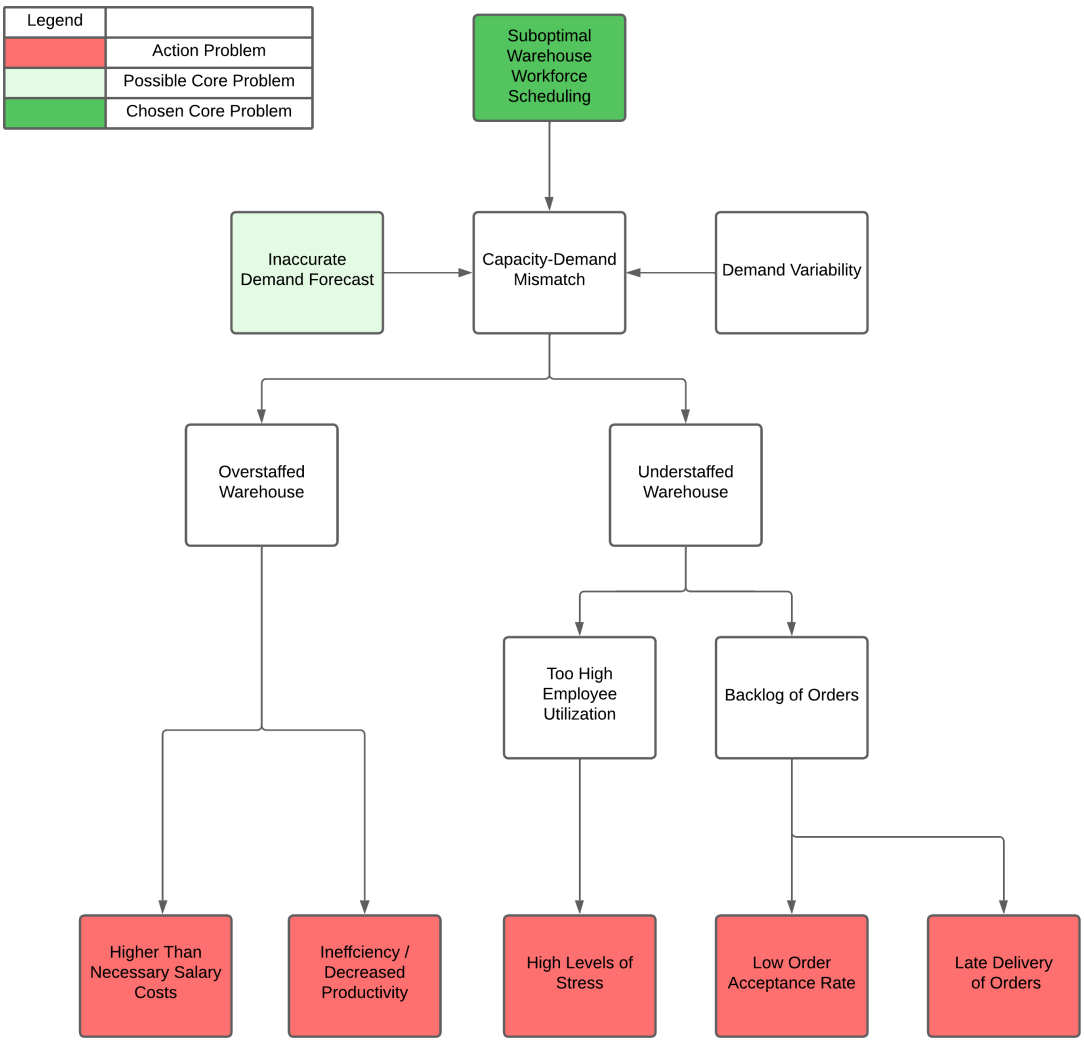



FIGURE 2.1: Problem Cluster

### 2.1.3 Main Research Question

We formulate the following main research question to solve the core problem of the problem cluster in Figure 2.1:

*"How can the MIP model for the warehouse worker requirement planning of flaschenpost SE be solved as well as possible while reducing computational time as much as possible?"*

We aim to solve the specific requirement planning problem for the company while reducing costs as much as possible. Additionally, the goal is to contribute to scientific knowledge. In particular, we give an example of how a MIP can be solved faster by using appropriate algorithms and formulations. The specific application to requirement planning and shift scheduling plays a secondary role since there exists a variety of problems of different domains using similar formulations. We will provide the answer to this research question in the form of a tool that satisfactorily solves the given MIP and a report explaining the reasoning, functionalities, and implications for the company.

### 2.1.4 Scope of Research

We already mentioned that we will only deal with the warehouses of the company. Next to that, we need to further define and specify the scope of our research. Due to the limited time available to finish the bachelor's thesis, we will focus on solving the already existing formulation of a MIP. Additionally, we will solve the WRP for the warehouse workers separately from the driver requirement planning. In the future (i.e., after the thesis), we can continue this project by implementing the driver requirement planning into the results of this thesis. The resulting optimization problem (for both warehouse workers and drivers) will provide a robust shift assignment over all scenarios.

For the evaluation, we will compare the requirement plan generated by the proposed MIP solution to the plans generated by HR. We limit this comparison to the warehouse in Münster but can assume that the results will be similar for other locations. We confidently make this assumption because most flaschenpost warehouses are configured similarly.

### 2.1.5 Problem-Solving Approach

We will use the Managerial Problem Solving Method (MPSM) as the framework to design our research and achieve the research aim described earlier. The MPSM was developed at the University of Twente and is commonly used to solve problems in Industrial Engineering and Management (Heerkens et al., 2017). In general, the MPSM can be seen as a guide to solving action problems. Since we intend to solve an action problem, the MPSM is our research's most relevant problem-solving approach.

The MPSM consists of the following seven phases:

1. Problem identification
2. Solution planning
3. Problem analysis
4. Solution generation
5. Solution choice
6. Solution implementation

7. Solution evaluation

This chapter of the report covers the first two phases of the MPSM. The first phase involves defining the problem to be solved. In the previous sections, we carried out the required activities (e.g., making a problem cluster and selecting a core problem). The rest of this chapter deals with formulating the problem-solving approach. The following section explains how we intend to address the remaining five phases by relating them to knowledge questions.

### 2.1.6 Knowledge Problems and Research Design

In the following section, we will explain the structure of the research design by connecting knowledge problems to the seven phases of the MPSM. This is necessary to have a well-structured approach to answering the main research question.

We define 9 sub-research questions and discuss whether further research is required as well as any arising concerns regarding validity and reliability. A summary of the research design can be found in Table A.2.

*1. How is the workforce requirement planning for the warehouse currently done at flaschenpost?*

This first question concerns the third phase of the MPSM. We will conduct informal interviews with our company supervisor and possibly other employees to answer this question. We can provide a detailed description of the current situation using the data from these conversations. We have already explored answers to this question in this chapter.

*2. How can the performance of the planning procedure be made measurable?*

This question also relates to phase three of the MPSM. From talking to the company supervisor, we already know that one of the main problems of the current situation is that there is no proper definition of goals. This, in turn, makes it impossible to measure how well the scheduling is done. Because of this, we first need to formulate specific goals and how to measure them since the quality of our solution cannot be evaluated otherwise. We will receive information about the desired KPIs and their operationalization from the company supervisor. This question is answered in chapter 5 by explaining the objective function of the MILP, which ultimately contains the KPIs we focus on.

*3. What algorithms and techniques for solving (mixed) integer linear programming problems exist?*

This question also relates to the fourth phase of the MPSM. The goal here is to determine which algorithms and techniques have been used to solve similar problems. We conduct a systematic literature review to determine appropriate ideas. We explore the realm of algorithms and solution techniques in chapters 3 and 4.

*4. Which solver has the best performance for the given model?*

This question concerns the fourth phase of the MPSM since it evaluates possible alternative solutions to the problem. We will evaluate which solver yields the lowest computational time by implementing the model in code and then using different solvers to solve it. After applying the different techniques to solve the MIP faster, we will repeat this step. During the experimental phase of our thesis we test different solvers. We therefore provide the respective

results in chapter 6.

*5. What adaptions can we make to the model to make it more realistic or - if needed - easier to solve?*

This question also concerns phase four of the MPSM. One change we need to make to the model is decreasing the time intervals. This will increase the accuracy of the model but yield higher computational times. Through experimentation, we will try to reduce the granularity as much as possible while staying within the time constraints. We discuss possible answers to this question in chapter 6.

*6. How does the new planning method influence the chosen KPIs?*

This question also relates to phase 5 of the MPSM. In this question, the different solutions created by the algorithm will be evaluated based on the KPIs defined in question 2. We will be able to determine how much we improved the situation by comparing the new values of the KPIs to the old ones.

*7. How can the new planning be implemented successfully?*

This question is also mostly related to the sixth phase of the MPSM. If our solution to the problem improves the current situation, the company is interested in implementing it. In cooperation with the responsible employees from flaschenpost, we will determine how this can best be done. While it might not be feasible to carry out this step fully due to the time constraints of the thesis, we will nonetheless give recommendations to the company to ensure that our work is used correctly. We conclude the paper by answering this question in chapter 8.

## 2.2 The Assignment

### 2.2.1 Warehouse layout and processes

To better understand the environment our assignment is located in, we introduce the layout of the warehouse as well as its basic processes. Figure 2.2 provides an overview of the layout of the warehouse. A typical flaschenpost warehouse has a picking area for groceries and FMCG, highly demanded beverages ("Schnelldreher"), and less-demanded beverages ("A"). In addition, there are also areas for handling incoming goods and deposits and a high-bay storage area in which crates of beverages are stored. The picking areas and the parking spaces ("P"), where the vans are loaded, are divided by a shelving unit. This "boxrange" is portrayed by the gray vertical rectangle in Figure 2.2.

In general, we distinguish between two types of workers in the warehouse. The pickers work on the left side of the boxrange (and the FMCG area). They pick products of incoming orders from the respective areas, load them into boxes, and eventually finish their picking route by depositing the completed order in the box range. The picking process, therefore, starts once demand enters the system and finishes once the picked order enters the boxrange.

The drivers - working on the right side of the boxrange - pick the orders placed on their delivery route from the boxrange and load them into their van. Once a driver has picked all required orders, they start their delivery route. The white arrows in Figure 2.2 show how the delivery vans enter and leave the warehouse.

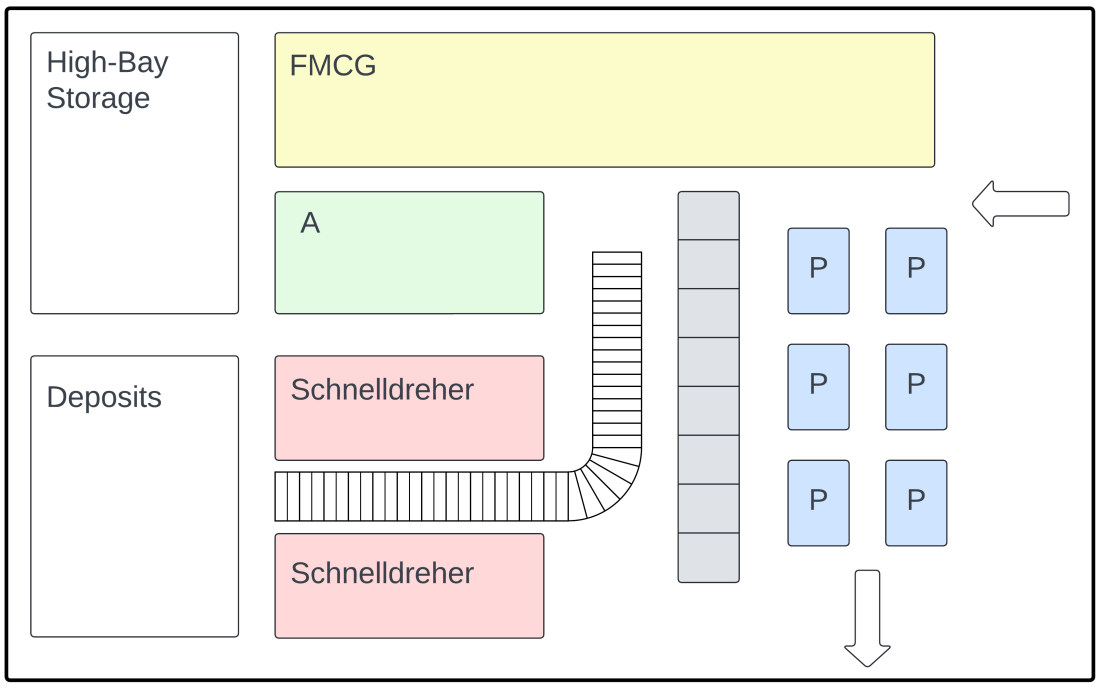



FIGURE 2.2: Exemplary warehouse layout at flaschenpost

The demand from the different working areas is split into instant and pre-order selling units, allowing for different tardiness thresholds. Customers place pre-orders in advance and have them delivered later (e.g., a customer places an order on Monday but wants the products delivered at 9:00 on Wednesday). Instant orders, in contrast, are placed by customers for immediate processing and delivery. This means an order is placed, and the customer expects delivery as soon as possible. To fulfill the goal of delivering to the customer within 120 minutes after placing their order, the picking process is not allowed to take more than a predefined duration. In our case, we have these hard deadlines and also soft deadlines after which a penalty is incurred because the delivery will arrive late at the customer.

### 2.2.2 The Workforce Requirement Planning Model

As mentioned in Section 2.1.4, our assignment focuses on the company's warehouses and deals with scheduling its workers. This means that we only schedule the pickers, so we neglect the drivers for now. In the remainder of this document, whenever we write warehouse workers, we therefore only refer to the pickers.

The company has already formulated a mixed integer linear program (MIP) for the WRP. Still, it has not been fully implemented in code and tried to solve it for realistic instances.

The mathematical model minimizes the overall cost of the generated schedules. The costs consist of the salary costs incurred for scheduling workers for shifts and the penalty costs which are generated whenever units are picked late. The optimal solution will presumably find a trade-off between these since they are inversely proportional: on one hand if enough workers are hired, units will never be picked late and therefore no penalty costs are incurred, on the other hand, it might be worth it to pick some units late if this means hiring one worker less. Based on a demand forecast the model dictates how many workers are hired for each shift and which orders they pick from each area (i.e., a virtual picking route is generated for

each order).

These decisions need to satisfy several constraints. For example, orders need to be picked after some amount of time to not violate the promise of delivering within 120 minutes too much. Additional constraints ensure that workers go to all required areas of the warehouse and accordingly spend the right amount of time on each of their picking routes. A worker also can only be hired for one shift each day and can only work actively during his shift which lasts a predefined amount of hours and includes breaks. Additionally, we add constraints that allow us to model different productivity curves to portray reality more realistically. For example, we would expect productivity to first increase at the beginning of a shift and then gradually decrease towards the end of the shift as a worker becomes more tired. We lastly enforce that all demand needs to be satisfied by the end of the day.

We aim to implement the model in code and solve it to optimality in a reasonable amount of calculation time if possible or find a way to generate a heuristic solution. This would mean sacrificing optimality for a fast calculation time.

To do this, we will tackle the problem on three different levels. First, we try to determine which solver works best for the given model. There exist different programs to solve MIP and there might be performance differences between them. Second, we will employ techniques and algorithms to solve an MIP faster, and lastly, we will consider reformulations to make the model more realistic if possible or reduce its complexity if necessary. Note that reformulations might also become necessary to apply the algorithms.

Since the model is relatively large, naively letting a solver solve it, is expected to yield long computational times with the initial formulation. Additionally, we want to make adaptations to the model to increase how well it reflects reality. For this, we are especially interested in reducing the time granularity of the model. Therefore, we will research and implement different algorithms to reduce the runtime while improving how realistically the model portrays the situation in an exemplary warehouse. In particular, we will use MIP solvers and customize their algorithms to solve the problem. While the performance of these solvers is generally good, we can use knowledge about the model to fit our needs better and achieve a faster calculation time.

### 2.2.3 Motivation of this thesis

Our goal for this thesis is to advance both scientific and technical knowledge.

From the scientific perspective, we try to contribute to existing knowledge about solving large-scale (M)IP problems. We apply this insight to a personnel scheduling problem. However, the algorithms and techniques we propose in this thesis can be used to solve various problems with different applications but similar formulations.

From a technical perspective, the outcome of this thesis could lead to practical implications for flaschenpost. Based on the proposed planning, costs can be minimized while keeping warehouse efficiency high. Our thesis can, therefore, significantly improve the operations at flaschenpost. These improvements eventually lead to increasing profit and customer satisfaction, thus contributing to the company's prosperity.

As mentioned earlier, flaschenpost wants to achieve certain overarching operational goals. To do so, the company examines a specific area due for improvement and then sufficiently accurately models the problem. Important questions must be answered to achieve this: What are the (decision) variables? What are the constraints that limit the variables? What is the

objective function? After analyzing the problem, it became evident that linear functions could describe these factors. The company, therefore, decided to solve the problem using a mixed integer linear programming (MIP) formulation.

# 3. Related Work

This chapter relates our assignment to existing literature. We give an overview of approaches to model and solve WRP. Additionally, since these models - like ours - tend to be large, we also discuss existing methods to solve large-scale MIP.

## 3.1 Worker Scheduling Problems

Worker scheduling problems — commonly referred to as Personnel Scheduling Problems (PSP) — are relevant to almost all industrial sectors. Examples include manufacturing processes, transportation, and emergency services.

The high cost of human resources and the need to make frequent scheduling decisions create the need for optimal or at least high-quality solutions (Guo, 2021). PSP have been studied extensively in the last decades due to economic motivation: "for many companies, labor is the major direct cost component" (Van Den Bergh et al., 2013).

PSP can be classified by characteristics like personnel type, shift definitions, and objectives (Van Den Bergh et al., 2013). Most PSP deal with scheduling full-time employees or a mix of full- and part-time employees. A heterogeneous workforce is modeled whenever tasks require a specific skill, and not all workers possess the skills to carry out all available tasks. Another differentiation between workers can be made based on their productivity. This criteria is very similar to grouping workers by their skills. Assigning less skilled or less productive workers to tasks leads to lower profits or a postponed due date.

In our case, we are dealing with a homogeneous workforce since the company does not have information about worker productivity that would allow for a distinction between workers regarding their productivity or skill level. However, these considerations should not be neglected because, in the future, the company might want to expand the model by introducing a heterogeneous workforce. In particular, the effect on postponed due dates and, thus, increased tardiness is of interest here since it is one of the main KPIs of the warehouse.

Since Dantzig's initial set-covering formulation (Dantzig, 1954), many studies have focused on enhancing workforce optimization by considering various factors. Due to the complexity of generalized set-covering formulations for PSP, MIP-based techniques are frequently used for modelling (Brusco, 1998). The literature suggests that MIP is indeed one of the most popular methods to formulate PSP (Van Den Bergh et al., 2013). MIP falls in the category of mathematical programming; other methods of this class include linear programming, dynamic programming, and integer (linear) programming. Problem formulations of this class allow the researcher to add constraints and objective functions that fit their particular requirements.

## 3.2 Large-Scale MIP

These problem formulations quickly result in MIP formulations that include numerous variables. These large-scale models are hard to solve — often to an extent where standard solvers cannot find optimal solutions in a reasonable amount of time.

Heuristics and exact algorithms exist to calculate a feasible/optimal solution to the MIP (Hong et al., 2019). While heuristics find good feasible solutions within a limited calculation time, they do not guarantee finding a globally optimal solution. The solution method chosen depends highly on the requirements of the problem owner.

In practice, researchers are accustomed to solving large-scale MIP by employing decomposition techniques (Ernst et al., 2004). The general idea is to "divide and conquer": the problem is split into sub-problems that are easier to solve (Van Den Bergh et al., 2013). While we do not discuss the details of these techniques further, we briefly mention Bender's decomposition (Benders, 1962), Dantzig-Wolfe decomposition (Dantzig & Wolfe, 1960), and Branch-and-Price algorithms.

Next to the above-mentioned decomposition techniques, we can think about adding strong valid inequalities (Felici & Gentile, 2004) and designing customized branching rules or tighter LP bounds (Heimerl & Kolisch, 2009).

In our MIP formulation, all decision variables except for one are time-indexed. Some variables even have two time indices: one for the time interval when demand enters the system and one for the interval during which demand is fulfilled. Decreasing the length of the time intervals (i.e., increasing the number of time intervals) leads to more decision variables and, thus, a harder-to-solve problem.

Multiple papers tackle this problem of fine time granularity time by combining individual intervals into time buckets. Dash et al. (2012) presents a time bucket formulation for the traveling salesman problem with time windows (TSPTW). TSPTW aims to find the least expensive route that visits a set of cities exactly once while ensuring each city is visited within a given time window. Dash et al. (2012) presents a reformulation of the classical time-indexed TSPTW that partitions the time windows into subwindows or "buckets".

Even though we do not deal with a TSPTW or a classical routing problem, the idea of this time discretization might still be useful to our case. Since demand in our model enters in a specific interval and needs to be handled before a given time, we can transfer the idea of time windows to it. The interval in which demand enters the system and the one where it needs to be handled at the latest can be seen as the release time $R$ and the deadline $D$, respectively. We can then start with this initial time window and, using cuts (as explained above), separate it into smaller subwindows.

# 4. Theoretical Background

This chapter serves as a brief introduction to linear and integer programming. We explain the basic ideas, what makes integer programming special, and methods to solve these types of models.

## 4.1 Linear Programming

As explained earlier, the company has decided to model the WRP as a MIP. In general, a programming problem is a problem that involves achieving an object while respecting certain constraints (Metei, 2019). When this objective and restrictions are translated into mathematical expressions such as equations or inequalities, it becomes a mathematical programming problem. If these expressions and equations are linear, then they are termed linear programming problems (LP). A linear programming problem essentially focuses on optimizing a linear objective function while satisfying a set of linear equality and/or inequality constraints. The general formulation of a linear programming problem is as follows:

| | |
|---|---|
| maximize | $c^T x$ |
| subject to | $Ax \leq b$ |
| | $x \geq 0$ |

TABLE 4.1: Standard LP formulation

The Simplex Algorithm is one of the standard algorithms used to solve LP. It usually solves even large problems efficiently but, in worst-case scenarios, does not always run in polynomial time. Other algorithms, however, have been proven to run in polynomial time, even in the worst case. Two examples of these are the Ellipsoid Algorithm (Khachiyan, 1979) and Karmarkar's algorithm (Karmarkar, 1984).

## 4.2 Integer Programming

Linear programming problems in which all of the variables are required to be non-negative integers are called integer linear programming problems - ILP for short (Winston, 1996). In our case, we are dealing with a mixed integer linear programming problem (MIP). This means that only some of the variables need to take integer values. In general, we define the standard integer (linear) programming problem as follows:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z} \end{array}$$

TABLE 4.2: Standard ILP formulation

Many important real-world problems of almost all management disciplines and many fields of engineering can be expressed using (mixed) integer linear programming formulations (Vanderbei, 2007) and usually deal with the efficient allocation of some resources (Gass & Fu, 2013, pp 771-783). A few typical examples of applications are:

- Scheduling Problems
- Network and Graph Problems
- Knapsack Problems
- Location, Routing and Scheduling Problems (e.g., Travelling Salesman Problem)
- Problems with Non-Linear Objective Functions

Multiple methods to solve MIP efficiently exist. The key concept these algorithms use to solve MIP is the concept of LP relaxation (Winston, 1996, ch. 9). The LP relaxation of a MIP is obtained by omitting all integrality constraints (i.e., the variables no longer need to take only integer or binary values). Therefore, the LP relaxation is a less constrained - or relaxed - version of the MIP formulation. Its feasible region contains the feasible region of the corresponding MIP (i.e., the feasible region of the MIP is a subset of the feasible region of its LP relaxation. Figure 4.1 depicts the feasible region of an IP and of its LP relaxation; these regions are also called the IP and LP hull, respectively. In the following, we explain how the Branch-and-Bound and Branch-and-Cut algorithms work.

MIP prove to be much harder to solve than LP problems and they are NP-hard problems in general (Conforti et al., 2014). This means no generic efficient (polynomial-time) algorithm exists for solving MIP. However, this does not mean that every MIP is NP-hard, and it is, therefore, possible to solve certain instances in polynomial time.

### 4.2.1 Cutting Planes

We consider the ILP problem (1):

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z} \end{array}$$

and its linear programming relaxation (2):

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array}$$

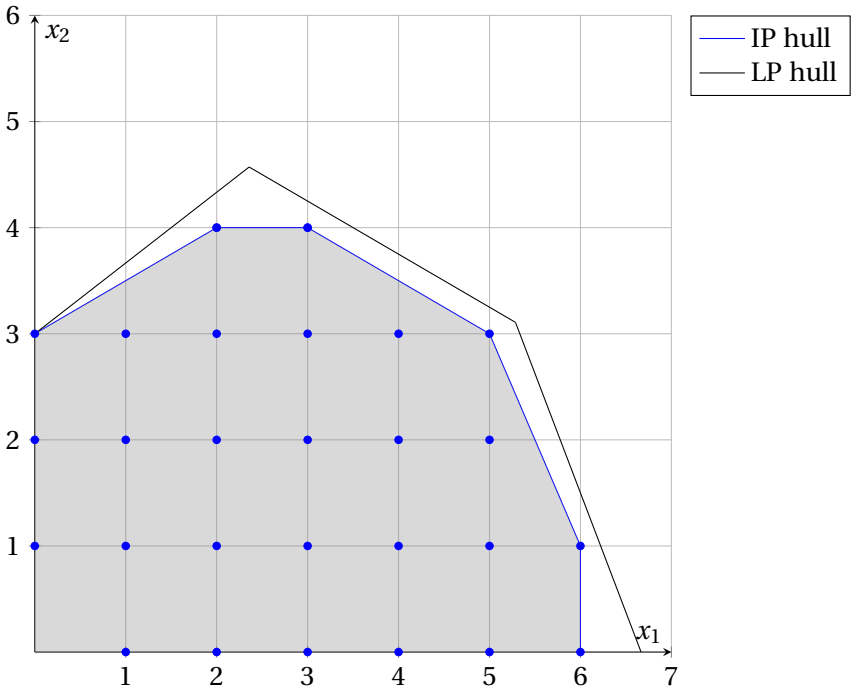



FIGURE 4.1: IP and LP Hull

Cutting plane methods solve the ILP by solving a sequence of linear programming problems (Bertsimas & Tsitsiklis, 1997). The method is described in the following:

**Algorithm 1** A generic cutting plane algorithm

1. Solve the linear programming relaxation (2). Let $x^*$ be an optimal solution.
2. If $x^*$ is integer stop; $x^*$ is an optimal solution to (1).
3. If $x^*$ is not integer, add a linear inequality constraint to (2) that all integer solutions to (1) satisfy, but $x^*$ does not; go to Step 1.

The performance of cutting plane methods depends on the choice of the inequality used to cut $x$. An example of a cutting plane algorithm are the so-called Gomory cuts.

### 4.2.2 Branch-and-Bound

Branch-and-bound explores feasible integer solutions. Instead of iterating over the entire feasible set, it uses bounds on the optimal solution to avoid exploring parts of the search tree that will not yield an optimal solution.

Let $F$ be the set of feasible solutions to the problem:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x \in F \end{array}$$

Now, we partition $F$ into subsets $F_1, F_2, .., F_k$ and solve separately each subproblem:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x \in F_i, \forall i \end{array}$$

Afterward, we compare the objective values of the subproblems and choose the best one. If a subproblem proves to be (almost) as hard to solve as the original problem, we split it into further subproblems. This is called branching and results in a tree of subproblems (see Figure 4.2).

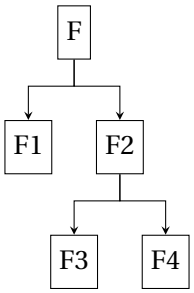



FIGURE 4.2: Branch and Bound Tree

We assume that there exists an efficient algorithm to compute the lower bound $b(F_i)$ to the optimal objective value for each subproblem:

$$b(F_i) \leq \min_{x \in F_i} c^T x$$

While the optimal cost to a subproblem might be difficult to obtain exactly, a lower bound might be much easier to calculate. One of the most popular methods to obtain this lower bound is the linear programming relaxation.

While carrying out the algorithm, we occasionally solve a subproblem to optimality (or evaluate the objective value of a feasible solution). This allows us to establish an upper bound $U$ on the optimal objective value associated with the best feasible solution encountered so far, which is called the incumbent.

**A Generic Branch-and-Bound Algorithm**

At any point, the algorithm keeps a set of active (yet to be solved) subproblems and the objective value U of the best feasible solution so far in memory. We can initialize U by setting it equal to $\infty$ or some feasible solution if available. A typical stage of the algorithm then looks as follows:

**Algorithm 2** A generic Branch-and-Bound algorithm

1. Select an active subproblem $F_i$.
2. If the subproblem is infeasible, delete it; otherwise compute $b(F_i)$ for the corresponding subproblem.
3. If $b(F_i) \geq U$, delete the subproblem.
4. If $b(F_i) > U$, either obtain an optimal solution to the subproblem or break it into further subproblems, which are added to the list of active subproblems.

Three parameters can be changed depending on the specific problem to improve the performance of the algorithm:

- There are different ways of choosing an active subproblem
- There are several ways to obtain a lower bound (e.g . LP relaxation)
- There are several branching rules (i.e., ways of breaking a problem into subproblems)

| Solver | License | Supports |
|---|---|---|
| Artelys Knitro | Commercial | **(MI)LP**, (MI)SOCP, (MI)NLP |
| Cbc | EPL | **(MI)LP** |
| CPLEX | Commercial | **(MI)LP**, (MI)SOCP |
| GLPK | GPL | **(MI)LP** |
| Gurobi | Commercial | **(MI)LP**, (MI)SOCP |
| HiGHS | MIT | **(MI)LP**, QP |
| SCIP | Apache | **(MI)LP**, (MI)NLP |

TABLE 4.3: MIP solvers supported by JuMP

### 4.2.3 Branch-and-Cut

This method combines the cutting planes algorithm and Branch and Bound. It utilizes cuts when solving the subproblems created by branching. In particular, the formulation of the subproblems is improved with additional cuts to improve the bounds obtained by the linear programming relaxations. This tightens the feasible region of the MIP and reduces the size of the search tree. The Branch and Cut framework is one of the most commonly used exact techniques for solving MIP problems (Zhang et al., 2023).

## 4.3 Software and Solvers

In practical applications, both LP and MIP include many variables and constraints that make them infeasible to solve by hand. Due to this, professionals in the field use specialized solvers in their work. The performance of these solvers has increased significantly in the last two decades (Koch et al., 2022). These improvements are not only caused by more powerful computers but especially by new calculation techniques and algorithms. Many problems that could not be solved 20 years ago can now be solved within seconds.

We will implement and solve the mathematical using the programming language Julia. In particular, we use JuMP to solve our optimization problem. JuMP is a domain-specific modeling language for mathematical optimization embedded in Julia (Lubin et al., 2023) and supports mixed integer linear programming. JuMP provides access to more than 40 commercial and open-source solvers. A few examples of solvers that support MIP are listed in Table 4.3.

The solvers use different algorithms to solve MIPs. While information about the specific algorithms used for the open-source solvers is easy to find, the commercial solvers keep their algorithms secret. We want to test the performance of different solvers for two main reasons. First, the solvers might have performance differences for our specific problem due to the differences in the solving procedure. The second reason is economic: Acquiring the licenses for open-source solvers is free, while one has to pay to use the commercial ones. Note that we obtained free academic licenses for our research for the commercial solver Gurobi, which the company cannot use since it is for non-commercial use only. We aim to determine the difference between the solvers' performances and evaluate whether or not upgrading to a commercial solver is worth the costs.

# 5. Problem Formulation and Solution Algorithm

## 5.1 Mathematical Model

This section introduces the mathematical model initially developed by our company supervisor, M. Wölck. All changes we made to the model have been agreed upon with the company supervisor. We outline the assumptions, variable notation, and decision model for the WRP. By doing so, we translate the real-world problem described earlier into a formal mathematical problem description. This model is the starting point for the subsequent analysis and solution generation.

In the remainder of the section, we describe an MIP for the WRP problem of flaschenpost. The model aims to minimize the total costs of the shift assignment and the tardiness penalties.

### 5.1.1 Assumptions

To model the complex business problem we need to make assumptions. These allow us to simplify reality and translate it into a MIP. The following outlines key assumptions we make, what they are based on, and how they might be relaxed later on.

1. Time over the day can be discretized into time intervals of predefined length without incurring intractable modeling errors.

The warehouse is operating from 6:00 until 24:00 if required. We start with an interval length of ten minutes to keep the model as small as possible. The granularity could further be reduced, leading to higher computational times. Decreasing the interval length should be done carefully as the model becomes more realistic but harder to solve. In practice, users of our solution should closely monitor how far the abstraction yielded by this assumption strays from the real-life situation as demand patterns might differ too much from real scenarios if the intervals are too big.

2. All random variables can be approximated by a single scenario without incurring intractable modeling errors.

We want to use a deterministic point forecast for the demand and for now we also model the other input parameters as deterministic. To create a stochastic model, we would need to create a stochastic demand forecast and implement a stochastic optimization model. We can relax this assumption in the future by using, for instance, a scenario decomposition approach or stochastic branch-and-bound. This assumption will not influence the quality of our output much. So far the company has used the given type of forecast and fared well with it.

3. Customer demand can be split into selling units coming from different working areas.

Incoming demand consists of individual selling units (i.e., products) that need to be picked from different working areas. Recall that a typical flaschenpost warehouse has an area for groceries and FMCG, highly-demanded beverages, and less-demanded beverages. This assumption is reasonable for most retail applications. However, remember that we assume that this splitting is deterministic. This assumption is realistic since the company already differentiates between these types of incoming demand. In the future, analysis of the share of each might reveal a distribution. The values can then be adjusted.

4. We can approximate worker productivity for the different working areas.

As mentioned above, we assume that productivity is deterministic. We can correct productivity by multiplying it by the worker's dropout probability (i.e., the probability that the worker does not appear to his shift). The company does not collect data on the productivity of individual workers and thus we need to make this assumption.

5. Worker productivity can either be implemented by modeling rates or by modeling routes of a fixed length.

When we approximate working via rates, then selling units are handled (e.g., picked) continuously. When we approximate picking via routes, selling units are handled in batches. In the latter approach, they are only fulfilled (e.g., put into the boxrange) when the route is finished. While routes are more realistic, they are more challenging to model and require more computational time18. Each working area may use either approach.

6. All demand has to be fulfilled at the end of the workday.

This assumption ensures that we do not have leftover demand at the end of the day. This includes leftover picking demand, incoming goods, and deposits. This assumption does not influence the quality of the solution. Instant demand should always be fulfilled the latest by the end of the day because of the promise to deliver within 120 minutes and pre-order demand can be injected into our system at a time that allows for it to be picked respecting the tardiness thresholds.

7. There is no unfulfilled demand at the beginning of the workday.

This assumption ensures that there is no unfulfilled demand at the beginning of the day. We can relax this assumption and start the day with outstanding demand.

8. We can fix a subset of the worker shifts so that the optimization cannot change them.

This assumption ensures that we can fix shifts that are already booked by workers. This is necessary since the company allows the workers to select shifts up to one month in advance using a rolling horizon approach.

### 5.1.2 Notation

| Symbol | Description |
|---|---|
| $\mathscr{W}$ | set of warehouse workers $i \in \mathscr{W}$ |
| $\mathscr{S}$ | set of shifts $t \in \mathscr{S}$ |
| $\mathscr{T}$ | set of time intervals $t \in \mathscr{T}$ of length $\Delta$, i.e., $\mathscr{T} = \{1, 2, .., T-1, T\}$ |
| $\mathscr{T}_\phi^b$ | set of time intervals $T_\phi^b = \mathscr{T} \setminus \{1, 2, .., (M_\phi - 3), (M_\phi - 2)\}$ |
| $\mathscr{T}_\phi^e$ | set of time intervals $T_\phi^e = \mathscr{T} \setminus \{T - (M_\phi + 1), T - (M_\phi + 1) + 1, ..T - 1, T\}$ |
| $\Phi$ | set of working areas $\phi \in \Phi$ |
| $\phi^q$ | working mode switching area |
| $\mathscr{D}$ | set of customer demand for every time interval |
| $d_{\phi t}^i$ | instant demand for area $\phi$ that enters the system at the beginning of time interval t |
| $d_{\phi t}^p$ | pre-order demand for area $\phi$ that enters the system at the beginning of time interval t |
| $s$ | shift s is active at $s = (s_0, s_1, .., s_T) \in \{0,1\}^{T+1}$ |
| $p_{si\phi}$ | worker productivity for shift s at $p = (p_0, p_1, ..., p_T) \in \mathbb{R}_+^{T+1}$ (measured in selling units picked up during the time interval) |
| $P_s$ | effective shift length, i.e., number of intervals the shift is active minus the number of intervals a worker has to take a break |
| $Q$ | mode switching duration, measured in number of intervals the shift is active minus the number of intervals a worker has to take a break |
| $c_s$ | cost $c_s \in \mathbb{R}_+$ of shift $s$ |
| $\theta_\phi$ | maximum allowed tardiness $\theta_\phi \in \mathbb{N}_+$ for selling units of area $\phi$ |
| $\Pi_\phi^i$ | tardiness occurs if instant selling units are not picked after $\Pi_\phi^i \in \mathbb{N}_+$ time intervals |
| $\Pi_\phi^p$ | tardiness occurs if pre-order selling units are not picked after $\Pi_\phi^i \in \mathbb{N}_+$ time intervals |
| $\pi$ | tardiness penalty $\pi \in \mathbb{R}_+$ per selling unit per time interval |
| $M_\phi$ | length of a picking route measured in time intervals $M_\phi \in \mathbb{N}_+$ |
| $\mathbb{P}(s,i)$ | probability $\mathbb{P}(s,i) \in [0,1]$ that worker $i$ does not appear to shift $s$ |

TABLE 5.1: Notation of parameters

| Decision | Description |
|---|---|
| $x_{si}^s$ | decision on worker shift assignment $x_{si}^s \in \{0,1\}$ |
| $x_{it}^w$ | decision on when the worker is actively working (which is only possible while on shift) $x_{it}^w \in \{0,1\}$ |
| $x_{i\phi t}^p$ | decision on worker picking area assignment $x_{i\phi t}^p \in \{0,1\}$ |
| $x_{i\phi t}^r$ | decision on when picking route is finished $x_{i\phi t}^r \in \{0,1\}$ |
| $x_{i\phi t}^v$ | decision on amount of selling units on a picking route $x_{i\phi t}^v \in \mathbb{N}_+$ |
| $x_{i\phi t}^{v,i}$ | decision on amount of instant order selling units on a picking route |
| $x_{i\phi t}^{v,p}$ | decision on amount of pre-order order selling units on a picking route |
| $x_{\phi t t'}^{v,i}$ | decision on how many instant order selling units from $\phi$ that entered at $t'$ are fulfilled during $t$ |
| $x_{\phi t t'}^{v,p}$ | decision on how many pre-order order selling units from $\phi$ that entered at $t'$ are fulfilled during $t$ |

| | |
|---|---|
| $x^{u,i}_{\phi t t'}$ | decision on how many instant selling units that entered the system at the beginning of $t'$ are still in the system at the end of $t$ |
| $x^{u,p}_{\phi t t'}$ | decision on how many pre-order selling units that entered the system at the beginning of $t'$ are still in the system at the end of $t$ |

TABLE 5.2: Notation of decision variables

### 5.1.3 Decision Model

In the warehouse $|\mathcal{W}|$ workers are assigned to shifts $s \in \mathcal{S}$ to fulfill a demand forecast $\mathcal{D}$. The currently available data allows us to model a homogeneous workforce with a performance curve for each shift that reflects that a worker's performance decreases over time (workers may have different performance curves for different working areas). Workers' performance is measured as the expected number of selling units a worker can handle during the respective time interval. The model decides how many workers are hired for a shift and thus during which intervals they work actively. Throughout the day workers are assigned to routes to fulfill demand in the different working areas. The model also decides how many selling units of each type (pre-order and instant) are picked on a single route.

To simplify the notation, we denote all decision variables by $x$ and highlight their meaning in the superscript. We differentiate between instant and pre-order selling units by writing $i$ or $p$ at the last position in the superscript. We index decision variables in the subscript. Time is indexed by $t$. The time index $t$ is always put at the end of the subscript. All sets are in italics. Penalties are denoted by $\pi$. We denote the indicator function by $\mathbb{1}$.

Demand always enters the system at the beginning of a time interval. Demand is always fulfilled during a time interval. Tardiness is measured at the end of a time interval.

$$\min_{x} \quad \sum_{s\in\mathcal{S}}\sum_{i\in\mathcal{W}} c_s x^s_{si} + \sum_{t\in T^\Delta}\sum_{\phi\in\Phi}\left(\sum_{t'=0}^{t-\Pi^i_\phi} \pi x^{u,i}_{\phi,t,t'} + \sum_{t'=0}^{t-\Pi^p_\phi} \pi x^{u,p}_{\phi,t,t'}\right)$$

s.t.

$$\sum_{s\in\mathcal{S}} x^s_{si} \le 1 \qquad \forall i \in \mathcal{W} \tag{1}$$

$$x^w_{it} \le \sum_{s\in\mathcal{S}} s_t x^s_{si} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T} \tag{2}$$

$$\sum_{t\in\mathcal{T}} x^w_{it} \le P_s + T(1 - x^s_{si}) \qquad \forall s \in \mathcal{S}, i \in \mathcal{W} \tag{3}$$

$$\sum_{\phi\in\Phi} x^p_{i\phi t} \le x^w_{it} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T} \tag{4}$$

$$x^p_{i,\phi,t-1} + \tfrac{1}{Q}\sum_{t'=t-Q}^{t-1} x^p_{i,\phi^q,t'} \ge x^p_{i,\phi,t} \qquad \forall i \in \mathcal{W}, \phi \in \Phi \setminus \{\phi^q\}, t \in \mathcal{T} \setminus \{1\} \tag{5}$$

$$x^r_{i\phi t} \le \tfrac{1}{M_\phi}\sum_{t'=t-M_\phi+1}^{t} x^p_{i\phi t'} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{6}$$

$$x^r_{i\phi t} \le 1 - \sum_{t'=t-M_\phi+1}^{t-1} x^r_{i\phi t'} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{7}$$

$$x^v_{i\phi t} \le M x^r_{i\phi t} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{8}$$

$$x^v_{i\phi t} \le \sum_{s\in\mathcal{S}}\sum_{t'=t-M_\phi+1}^{t} x^s_{si} p_{st'} \mathbb{P}(s,i) \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{9}$$

$$x^{v,i}_{i\phi t} + x^{v,p}_{i\phi t} \leq x^{v}_{i\phi t} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{10}$$

$$\sum_{t'=max(1,t-\theta_\phi)}^{t-M_\phi+1} x^{v,i}_{\phi t t'} \leq \sum_{i \in \mathcal{W}} x^{v,i}_{i\phi t} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{11}$$

$$\sum_{t'=max(1,t-\theta_\phi)}^{t-M_\phi+1} x^{v,p}_{\phi t t'} \leq \sum_{i \in \mathcal{W}} x^{v,p}_{i\phi t} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^b_\phi \tag{12}$$

$$x^{u,i}_{\phi t t'} = d^i_{\phi,t} - \mathbb{1}_{M_\phi=1} x^{v,i}_{\phi t t} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^e_\phi \tag{13}$$

$$x^{u,p}_{\phi t t'} = d^p_{\phi,t} - \mathbb{1}_{M_\phi=1} x^{v,p}_{\phi t t} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^e_\phi \tag{14}$$

$$x^{u,i}_{\phi t t'} = x^{u,i}_{\phi,t-1,t'} - x^{v,i}_{\phi,t,t'} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^b_\phi, t' \in \mathcal{T}^b_\phi : t'+M_\phi-1 \leq t \tag{15}$$

$$x^{u,p}_{\phi t t'} = x^{u,p}_{\phi,t-1,t'} - x^{v,p}_{\phi,t,t'} \qquad \forall \phi \in \Phi, t \in \mathcal{T}^b_\phi, t' \in \mathcal{T}^b_\phi : t'+M_\phi-1 \leq t \tag{16}$$

$$x^{u,i}_{\phi \min(T,t'+\theta_\phi) t'} = 0 \qquad \forall \phi \in \Phi, t' \in \mathcal{T}^e_\phi \tag{17}$$

$$x^{u,p}_{\phi \min(T,t'+\theta_\phi) t'} = 0 \qquad \forall \phi \in \Phi, t' \in \mathcal{T}^e_\phi \tag{18}$$

$$x^{s}_{si} \in \{0,1\} \qquad \forall s \in \mathcal{S}, i \in \mathcal{W}$$

$$x^{w}_{it} \in \{0,1\} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x^{p}_{i\phi t} \in \{0,1\} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}$$

$$x^{r}_{i\phi t} \in \{0,1\} \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}$$

$$x^{v}_{i\phi t} \geq 0 \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}$$

$$x^{v,i}_{i\phi t} \geq 0 \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}$$

$$x^{v,p}_{i\phi t} \geq 0 \qquad \forall i \in \mathcal{W}, \phi \in \Phi, t \in \mathcal{T}$$

$$x^{v,i}_{\phi t t'} \geq 0 \qquad \forall \phi \in \Phi, t, t' \in \mathcal{T} : t' \leq t - M_\phi$$

$$x^{v,p}_{\phi t t'} \geq 0 \qquad \forall \phi \in \Phi, t, t' \in \mathcal{T} : t' \leq t - M_\phi$$

$$x^{u,i}_{\phi t t'} \geq 0 \qquad \forall \phi \in \Phi, t, t' \in \mathcal{T} : t' \leq t - M_\phi$$

$$x^{u,p}_{\phi t t'} \geq 0 \qquad \forall \phi \in \Phi, t, t' \in \mathcal{T} : t' \leq t - M_\phi$$

The objective function computes the sum of shift costs and tardiness penalties for pre- and instant orders. The duration until tardiness arises may differ for pre- and instant orders.

Constraint (1) ensures that a worker is only assigned one shift at most. Variable $x^s_{si}$ equals one if worker $i$ is assigned shift $s$ and zero otherwise. Constraint (2) ensures that a worker only works during his shift. Variable $x^w_{it}$ equals one if worker $i$ is working at time interval $t$ and zero otherwise. Constraint (3) ensures that a worker assigned to shift $s$ can work at most $P_s$ time intervals.

Constraint (4) ensures that we assign a worker only to one area at a time. Variable $x^p_{i\phi t}$ equals one if worker $i$ is working in area $\phi$ at time interval $t$ and zero otherwise.

Constraint (5) ensures that switching to a different working area takes $Q$ time intervals during which the worker may not handle any selling units.

Constraint (6) ensures that routes for each working area $\phi$ take at least $M_\phi$ time intervals until they are finished (and that we assign the worker to the working area for the complete duration). Variable $x^r_{i\phi t}$ equals one if worker $i$ finishes a route in area $\phi$ at the end of time interval $t$. Constraint (7) ensures that we assign a worker to one route at a time. Note that worker $i$ can finish a route in area $\phi$ only every $M_\phi$ time intervals.

Constraint (8) ensures that selling units can only be fulfilled at the end of a picking route. Variable $x^v_{i\phi t}$ is the amount of selling units on the respective route. Constraint (9) ensures that the worker's productivity limit is not violated (i.e., the amount of selling units he can handle during each time interval).

Constraint (10) splits the selling units collected in a route into instant and pre-order selling units. Constraints (11) and (12) map the selling units collected on the respective routes to demand that entered the system at previous time intervals and ensure that the demand enters the system before the worker starts the route.

Constraints (13) and (14) inject instant and pre-order demand into the model, respectively.

Constraints (15) and (16) consider instant and pre-order selling units that entered the system at $t'$, respectively. Unhandled demand is either carried over to the next time interval or handled.

Constraints (17) and (18) ensure that all demand is fulfilled before the maximum allowed tardiness or by the end of the day.

The remaining constraints limit the domain of the decision variables.

### 5.1.4 Model Progress Overview

So far flaschenpost has only formulated the model and coded a starting version. We adjusted and expanded upon this initial code incorporating all constraints properly, fixing the indexing, and making the model work for values of $\Delta$ smaller than 10 minutes.

As can be seen above, the model uses a large number of constraints and decision variables. Especially the constraints dealing with the shift, area, and route assignment of the workers and deciding when selling units should be picked are prone to increase in number. These constraints are formulated for sets of time intervals $\mathcal{T}^b_\phi$ and $\mathcal{T}^e_\phi$ or combinations of them (using double-time indexed variables). Reducing the interval length $\Delta$ therefore leads to an even larger amount of constraints and thus decisions that need to be made. We expect most of the model's complexity to come from this.

## 5.2 Algorithm

### 5.2.1 Reformulations

We must reformulate parts of the mathematical model to use the time bucket formulation and solve the MIP faster. In the following, we describe in detail all of the changes made and the implementation of a customized Branch-and-Cut algorithm.

#### Symmetry of the Solution

In our MIP model, all workers have the same performance curve, and the associated costs of hiring workers are the same for each one. Therefore, the modeled workforce is homogeneous. This characteristic of the warehouse workforce creates significant symmetry in the context of our problem. Symmetry can lead to multiple equivalent solutions, which might significantly increase the computational complexity - especially when using Branch-and-Bound or Branch-and-Cut algorithms (Margot, 2009). This issue is especially pressing for us since we aim to solve the MIP using a Branch-and-Cut algorithm. The following example illustrates the problem with symmetry:

Let $x$ be a solution to the MIP in which we require three workers to fulfill demand and, for simplicity, assume that there is only one shift. The productivity of all workers $i \in \mathcal{W}$ is the same. Since the salary costs per shift do not differ from each other either, the following solutions are equivalent (i.e., they lead to the same objective function value):

1. $x^s_{1,1} = 1$, all other 0
2. $x^s_{1,2} = 1$, all other 0
3. $x^s_{1,3} = 1$, all other 0

Recall that $x^s_{s,i} = 1$, means that worker $i$ is assigned to shift $s$. The gravity of this problem increases with the number of required workers and shifts and is additionally amplified for time-indexed variables. We introduce the following symmetry-breaking constraints to the model to mitigate this negative effect:

$$\sum_{s \in \mathcal{S}} x^s_{s,i} \geq \sum_{s \in \mathcal{S}} x^s_{s,j} \qquad i \in \mathcal{W}, j \in \mathcal{W} : \underset{i<j}{} \tag{5.1}$$

These constraints prioritize workers with smaller indices by ensuring that the sum of the shift assignment variables for a worker with a smaller index is at least as large as that for any worker with a larger index. We effectively reduce the number of equivalent solutions that only differ by a permutation of the workers, improving the solving process's calculation time.

#### New Demand Injection and Propagation Constraints

To be able to use the Time Bucket Formulation (TBF), we rewrite the demand injection and propagation constraints for instant and pre-orders (13, 15 and 14, 16), respectively:

$$x^{u,i}_{\phi t t'} \geq d^i_{\phi t'} - \sum_{t''=t'}^{t} x^{v,i}_{\phi t'' t'} \qquad \forall \phi \in \Phi, t' \in \mathcal{T}, t \in \mathcal{T} \underset{t'+M_\phi-1 \leq t}{} \tag{5.2}$$

$$x^{u,p}_{\phi t t'} \geq d^{p}_{\phi t'} - \sum_{t''=t'}^{t} x^{v,p}_{\phi t'' t'} \qquad \forall \phi \in \Phi, t' \in \mathcal{T}, t \in \mathcal{T} \atop t'+M_\phi-1 \leq t \tag{5.3}$$

The new constraints ensure that the number of selling units left to pick at the end of period $t$, which entered the system at the beginning of period $t'$, is equal to the demand $d_{\phi t'}$ (i.e., demand from period t') minus all selling units that have been picked in the previous intervals. These new constraints now manage how demand is handled and carried over to the next period. By combining two constraints of the old formulation into one, we expect to be able to apply our customized cutting planes more effectively.

**Demand Matching Constraints for TBF**

Using the new demand injection and propagation constraints 5.2 and 5.3, the number of picked selling units only has a lower bound (with the old constraints, the amount of picked selling units had to equal the amount that entered the system exactly). The practical implication is that it becomes possible to pick more selling units than the total daily demand requires. This happens when the already hired workforce has the capacity to pick additional selling units without needing to hire additional workers, thereby incurring no extra costs. We provide an example to clarify the issue:

Assume we divide the day into three time intervals, the picking route length $M_\phi$ for area $\phi$ equals one, and instant order demand entering the system at the beginning of interval three equals one. Using the new demand injection and propagation constraints leads to the following constraint:

$$x^{u,i}_{\phi,3,3} \geq 1 - x^{v,i}_{\phi,3,3} \tag{5.4}$$

And since all demand must be satisfied by the end of the day, the terminal condition is:

$$x^{u,i}_{\phi,3,3} = 0 \tag{5.5}$$

Combining equations 5.4 and 5.5 yields the following expression:

$$x^{v,i}_{\phi,3,3} \geq 1 \tag{5.6}$$

From equation 5.6, it is now reasonably straightforward to see that cost-wise, it does not make a difference what value $x^{v,i}_{\phi,3,3}$ takes (i.e., how many selling units that entered the system at the beginning of $t = 3$ are picked during interval three) as long as it is between one and the productivity of the workforce already active during interval three.

To fix this issue, we introduce the following "demand matching" constraints, which ensure we pick the exact amount of selling units required.

$$\sum_{t=t'+M_\phi-1}^{\min\{t+\theta_\phi,T\}} x^{v,i}_{\phi t t'} = d^{i}_{\phi t'} \qquad \forall \phi \in \Phi, t' \in \mathcal{T} \tag{5.7}$$

$$\sum_{t=t'+M_\phi-1}^{\min\{t+\theta_\phi,T\}} x^{v,p}_{\phi t t'} = d^{p}_{\phi t'} \qquad \forall \phi \in \Phi, t' \in \mathcal{T} \tag{5.8}$$

### 5.2.2 Implementation Strategy

This section briefly explains our strategy of implementing customized cutting planes into the solver's solving process.

User cuts are added by the user, usually to tighten the LP relaxation or separate a search tree node into subproblems. This type of cut must not exclude any otherwise integer-feasible solutions from solution space because it is not part of the model's initial formulation. Adding a user cut reduces the LP solution space and thus the difference between the LP and IP hull (recall these two ideas from chapter 4). Therefore, we only add these cuts to help potentially solve the model faster, but they are never required for the model's logical correctness.

In our case, we have to generate many constraints to model the problem correctly. At the optimal solution, we expect many of them to be redundant or non-binding. We, therefore, remove them from the initial formulation and iteratively add these constraints to the model only when they are violated. It is necessary to do this until no violated constraint is left to be added to the model since these constraints are part of the original problem and, therefore, are required for the model to be correct. In contrast to user cuts, lazy constraints can cut off integer-feasible solutions.

## 5.3 Customized Branch-and-Cut Algorithm

In the following section, we explain the logic of our customized branch-and-cut algorithm. Note that we give examples only using the constraints and variables for instant order demand; the same applies to pre-order demand.

Recall the demand injection and propagation constraints we introduced in section 5.2.1:

$$x^{u,i}_{\phi t t'} \geq d^{i}_{\phi t'} - \sum_{t''=t'}^{t} x^{v,i}_{\phi t'' t'} \qquad \forall \phi \in \Phi, t \in \mathcal{T}, t' \in \mathcal{T} : \atop t'+M_\phi-1 \leq t \tag{5.9}$$

We initialize the model only with constraints for variables $x^{u,i}_{\phi t t'}$ on the left-hand side for which $t = \min\{t + \theta_\phi, T\}$. All other $x^{u,i}_{\phi t t'}$ do not have a lower bound (except being non-negative). This, combined with the fact that assigning a positive value to these variables might result in tardiness costs, leads the optimization to set all other of these variables equal to zero. This, however, is not a feasible solution because even if selling units are picked at a time such that tardiness occurs, this is not represented in the value of the variables $x^{u,i}_{\phi t t'}$ and thus distorts the optimal value of the objective function ultimately leading to a non-optimal shift schedule.

To correct this and maintain the advantage of not generating all constraints from the start, we add additional constraints in the form of equation 5.9 whenever required. A constraint becomes required when the current feasible solution becomes infeasible after adding that constraint.

The procedure of our customized algorithm is shown in Figure 5.1.

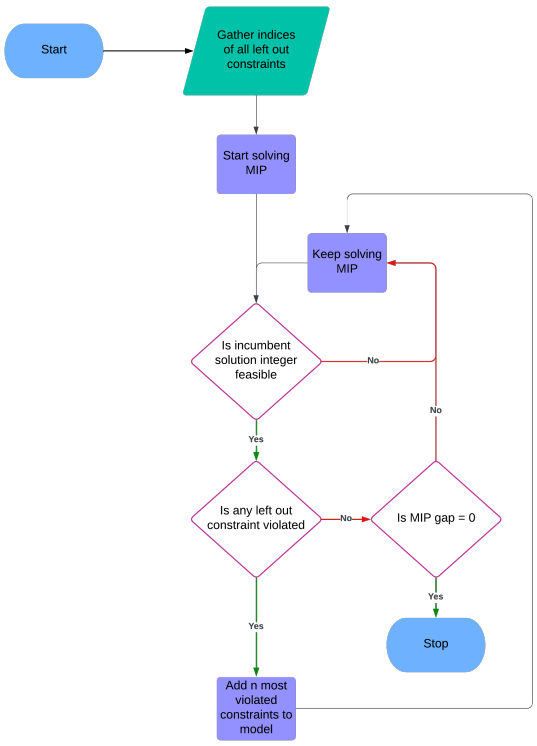



FIGURE 5.1: Flowchart of our Customized BaC Algorithm

# 6. Numerical Experiments

## 6.1 Problem Instances

For our experiments, we look at a typical flaschenpost warehouse. We need to introduce different problem instances to make our experiments realistic and at the same time test the validity of our solution approaches.

### 6.1.1 Warehouse Setup

For simplicity, we model a warehouse with two picking areas and an unlimited workforce. 66% of the total daily demand come from area one while the rest comes from the second area. These areas could for example be the areas for FMCG and highly demanded beverages as shown in Figure 2.2.

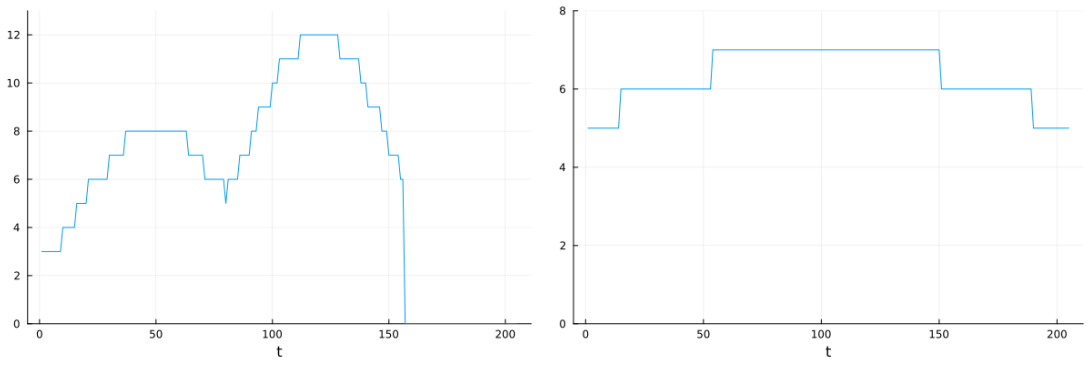
### 6.1.2 Demand

The website of flaschenpost accepts orders from Monday to Saturday from 8:00 until 20:00. This means that new orders instant orders can enter the system at any point in this period. Pre-orders are also placed through the website but are not meant for direct processing and can be picked later (up to two days). We model the warehouse to be open from 7:00 until 24:00. This way we ensure that (i) there is always enough time at the end of the day to pick all remaining demand, and (ii) that pre-orders placed on previous days can be picked before new instant orders enter the system. This means that pre-order demand can be in the system from 7:00 onwards while instant demand only enters after 8:00.

The company identified two demand patterns from previous research about incoming demand. On weekdays, the number of incoming orders peaks around 11:00 and 18:00 while on Saturdays the orders are mostly equally distributed over the day. The total daily demand for our experiments is 2000 selling units which are split equally among instant- and pre-orders.

Figure 6.1 shows these two patterns for $\Delta = 5$ and 66% (i.e., area 1) of total demand. The instant order demand curve looks the same for all cases except that all demand before 8:00 ($t = 13$ if $\Delta = 5$) is set to zero (see Figure B.1). The shares of selling units per area have been agreed upon and deemed realistic enough by our company supervisor. However, if this opinion changes based on data collected in the future, these values can easily be adjusted to generate a new solution.

Figures for less overall demand (33%) can be found in Figure B.2. Similarly we generate the demand curves for $\Delta = 10$.




(A) Demand on weekdays

(B) Demand on Saturdays

FIGURE 6.1: Pre-order demand patterns for $\Delta = 5$ and 66% of total demand

## 6.2 Experimental Setup

We solve the model both using our branch-and-cut algorithm and without. The algorithms are written in Julia 1.10.3 (Bezanson et al., 2017) while we construct our model with JuMP (Lubin et al., 2023) and solve it using Gurobi 11.0.1, SCIP, and GLPK. All computational results are obtained on a mobile workstation with an INTEL ® 2.60GHz processor running Windows 11 and 16.0 GB of RAM.

Carrying out the initial set of experiments, we quickly notice that the model's calculation time is too high. We therefore relax the model to solve it faster. We need to carefully balance the benefits of decreasing the calculation time and the drawbacks of further abstracting reality. In total, we conduct the following set of experiments:

- Initial performance testing: comparison of the fully specified model with and without customized BaC algorithm
- Performance for double time-indexed variable model: in this specification of the model we look at the part of the model that deals with the decision of when selling units are picked. Here we are mainly left with constraints using double time indices, for which we expect our customized BaC algorithm to perform best if we increase the time granularity (i.e., decrease $\Delta$).
- Decomposition approach: we remove the shifts from the problem and solve the WRP for each warehouse area individually. We additionally use a heuristic to generate schedules from this simplified version of the original model.

# 7. Empirical Results

## 7.1 Initial Performance Testing

We start our experiments by testing the fully specified model without cuts. The results are supposed to serve as a benchmark for further experimentation. We use Gurobi as the solver for now since we expect it to perform best. For the remainder of this chapter, we measure the solutions' performance by the calculation time (i.e., how many seconds the solver needs to find the optimal solution) and the MIP gap (i.e., the relative gap between the upper and lower bound of the objective value) in case the solver does not find the optimal solution after one hour. The MIP gap is defined as $MIPgap = (UB - LB)/UB$.

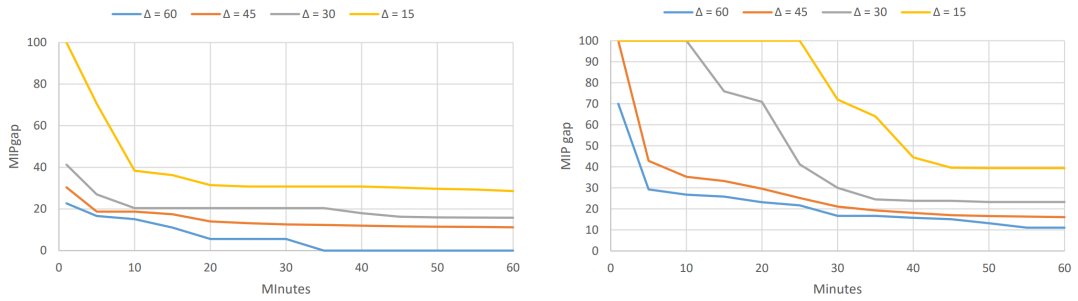
Throughout the initial tests, it quickly becomes evident that the fully specified model is hard for all the solver - even for bigger values of the interval length $\Delta$. We also notice that the problem formulation using our customized Branch-and-Cut logic yields a much worse MIP gap and calculation time. This is no surprise since - as mentioned above - we expect the cuts to work better for models with finer time granularity. At the same time, this also indicates that the shift and area assignment, even though not using double time-indexed time variables, contributes significantly more to the model's complexity than expected. The solver needs to evaluate these constraints again after we add a cut, possibly changing the picking schedule. Since we do this many times to save the generation of as many demand injection and propagation constraints as possible, the solution's performance using our customized cuts is much worse.

After letting the solver run for eight hours, the best bounds for the objective value of the problem instance where $\Delta = 5$ are 393 and 208 for the upper and lower bounds respectively. This means that we are still left with a MIP gap of approximately 47%. Clearly the problem using an interval length this small, is much harder to solve than we anticipated.

Table 7.1 shows the MIP gap achieved after 60 minutes (if the MIP gap is zero, an optimal solution has been found before the time limit was reached) for different problem instances, and both the original formulation and the one using lazy constraints. Figure 7.1 shows how the MIP gap progresses over time. Since the problem seems too hard for now, we decide to relax it by removing and adjusting some constraints. We do this step-wise until we find a problem specification that can be solved in a reasonable amount of time while ideally still portraying reality well enough. For the performance of this smaller model, the tradeoff between time spent in the callback function and solving a more complex model will be essential to the calculation time.

| | Original | Custom BaC |
|---|---|---|
| Δ | Relative Gap | Relative Gap |
| 60 | 0.0% | 11.0% |
| 45 | 12.5% | 16.1% |
| 30 | 15.8% | 23.3% |
| 15 | 28.6% | 39.4% |

TABLE 7.1: Computational Results of the Fully Specified Model after 60 Minutes




(A) Original Formulation

(B) Using Custom BaC

FIGURE 7.1: MIP Gap Progression

## 7.2 Performance for Time-Indexed Variable Model

For the cases described above, the model already only uses picking rates ($M_\phi = 1$), which we expected to be computationally less expensive than routes. Nonetheless, we already have noticed that the model is still too hard, so we need to make use of further simplifications.

First, notice that if $M_\phi = 1$, then the variable $x^r_{i\phi t}$ is not needed anymore since it will always have the same value as $x^p_{i\phi t}$. Thus, for these formulations, we can remove constraints (5) to (7) and reformulate constraint (8): $x^v_{i\phi t} \leq M x^p_{i\phi t}$. Next, we also reduce the mode switch time $Q$ to zero, which means that constraint (5) also becomes irrelevant. Unfortunately, none of these changes reduce the calculation time by much. This means that the route and shift assignment are more complex than expected—even for small instances and using picking rates.

Finally, the problem is easily solved if it is reduced to only the decision model, which decides when and where selling units are picked (i.e., removing constraints (2)—(8)). In the following section, we use this problem specification to focus on analyzing the performance of the cuts because the calculation time is reasonably short to conduct further experiments, and the cuts only affect the remaining constraints in the model.

### 7.2.1 Finding the Right Amount of Cuts

As explained above, the trade-off between saving the generation of constraints at the outset and the time spent in the callback function is crucial to the performance of our lazy constraints. We do not want to add too many constraints at once since this might drastically increase the complexity and, thus, the calculation time of the model. Therefore, we must find the right number of cuts to add to the problem simultaneously. Lazy constraints can be added to the problem whenever the solver finds a feasible fractional or integer solution. Our callback function is called and returns the $n$ most violated constraints, which are subsequently added

to the model.

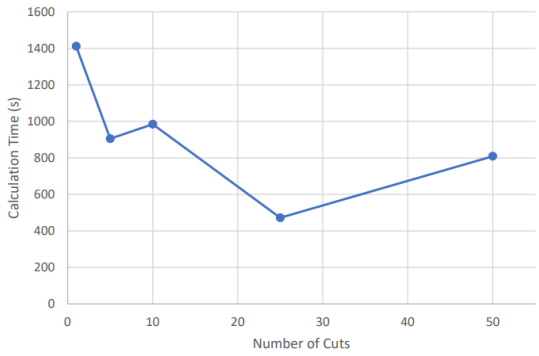



FIGURE 7.2: Average calculation time in for different numbers of cuts

As mentioned above, we use the completely relaxed model since we can focus on the effect of the cuts on the constraints using double time-indexed variables and reduce the interval time $\Delta$ to 5 minutes. We test the performance for adding up to 1, 5, 10, 25, and 50 constraints at a time. The results are presented in Figure 7.2. We can see that, on average, adding up to 25 cuts to the model is the most effective to solve the MIP fast.

The calculation time is also significantly different from the calculation time of the model without our customized Branch-and-Cut logic. While for $\Delta = 5$ and adding up to 25 cuts at once, the average calculation time is 472 seconds, it takes Gurobi an average of 1048.35 seconds to find the optimal solution without using our Branch-and-Cut logic. This means that for this instance the calculation time can be decreased by approximately 55%.

## 7.3 Removing the Shifts

As mentioned above and shown by the preceding experiments, shift scheduling seems to contribute significantly to the complexity of the model. We therefore remove all variables and constraints associated with the shift scheduling from the original model and change the objective function to the following:

$$\min_{x} \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{W}} x_{it}^{w} + \sum_{t \in T^{\Delta}} \left( \sum_{t'=0}^{t-\Pi^{i}} \pi x_{t,t'}^{u,i} + \sum_{t'=0}^{t-\Pi} \pi x_{t,t'}^{u,p} \right) \tag{7.1}$$

The objective function still minimizes the tardiness costs but now minimizes the sum of intervals in which workers actively work instead of the costs incurred for assigning shifts to workers.

### 7.3.1 Decomposition Approach

With this new formulation, workers will not switch between the areas since this would mean spending intervals on changing the area in which no selling units can be picked. As a result of this, the optimal solution for this new version of the problem will never have a worker switch areas. By omitting the shift scheduling and only hiring workers for the intervals they are needed, we can afford to hire as many workers as required since the ratio of active intervals and intervals they are hired for is always 1:1.

With the original version, a worker needs to be hired and paid for a whole shift and might switch areas during the shift. Switching areas takes time and thus reduces the ratio of active intervals and intervals the worker is hired for. Thus it is cheaper to have one worker pick selling units from different areas instead of hiring a new one (if their capacity allows for it) to maximize the active intervals during a shift (i.e., value/picked units per shift).

Finally, this means we can now decompose the problem into subproblems for the individual areas and solve them separately since there will no longer be any interactions between the areas and their assigned workers.

We expect to get faster and better results even for harder problem instances. However, we also expect to sacrifice optimality in dealing with this simplified model. The changed model can be found in Appendix B.

We use the output of this simplified version as input to an IP to map shifts to requirements per time interval $t$.

### 7.3.2 Shift Scheduling

The solution of the decomposed subproblems, that is how many workers are required in each interval, can be used to create a shift schedule. We model a problem in which shifts take either 6 or 9 hours. A worker can start a shift between 7:00 and the last hour which allows them to finish their shift before 24:00. This means that the latest a 6 (9) hour shift can start is at 18:00 (15:00). The following shows the IP we use to generate the heuristic shift schedule. We show the model for the formulation where $\Delta = 5$.

**Variables and Parameters**

| Variable | Description |
|---|---|
| $\mathscr{S}$ | set of shift lengths $s \in \mathscr{S} = \{6,9\}$ |
| $\mathscr{T}$ | set of all time intervals $t \in T = \{1,2,..,204,205\}$ |
| $\mathscr{T}_s$ | set of time intervals in which a shift of length $s$ can start; i.e., $t \in \mathscr{T} = \{1,2,..,204-12s,205-12s\}$ |
| $d_t$ | required amount of active workers $d$ in interval $t$ |
| $x_{ts}$ | decision on how many $s \in \mathscr{S}$ hour shifts start at the beginning of interval $t \in \{\mathbb{Z}_+ \mid 1 \leq t \leq 205\}$ |

TABLE 7.2: Notation of decision variables

**Decision Model**

$$\min_{x} \quad \sum_{s\in\mathscr{S}} \sum_{t\in\mathscr{T}_s} s x_{ts}$$

$$\text{s.t.} \quad \sum_{t'=\max(1,t-72)}^{t} x_{t,6} + \sum_{t'=\max(1,t-108)}^{t} x_{t,9} \geq d_t \qquad \forall t \in \mathscr{T} \qquad (1)$$

$$x_{ts} \in \mathbb{Z}_+ \qquad \forall t \in \mathscr{T}, s \in \mathscr{S} \qquad (2)$$

The objective function computes the sum of shift costs and tardiness penalties for pre- and instant orders. The duration until tardiness arises may differ for pre- and instant orders

Constraint (1) ensures that we fulfil the "demand" of workers in each interval by either hiring for a six or nine hour shift.

Figure 7.3 shows the solution strategy using the decomposition approach. Using Gurobi the individual MIP for weekdays and each area can all be solved in under 120 seconds and the IP described in 7.3.2 is solved instantaneously for all instances. The overall time to generate a schedule for a given day is thus much less than 5 minutes.

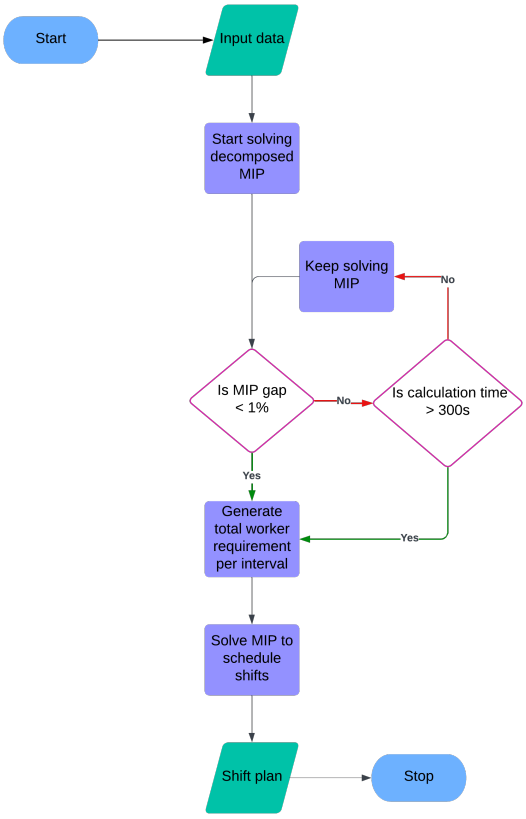



FIGURE 7.3: Solution strategy using decomposed model

We modified our code so it can tell us at which interval and how many shifts of each length need to start. Additionally, we generate a plot that shows the requirement per interval which is generated by the approach explained in Section 7.3.1 and the shift schedule (i.e., how many workers are working in the warehouse). This output is generated for weekdays and for Saturdays based on the demand patterns explained previously. Figures 7.4a and 7.4b show this for weekdays and Saturdays respectively. For our problem instances, we use seven six-hour shifts on the weekdays as well as two six-hour shifts and two nine-hour shifts each on Saturdays.

| shift length (h) | nr of shifts | start interval |
|---|---|---|
| 6 | 3 | 6 |
| | 1 | 70 |
| | 1 | 79 |
| | 2 | 86 |

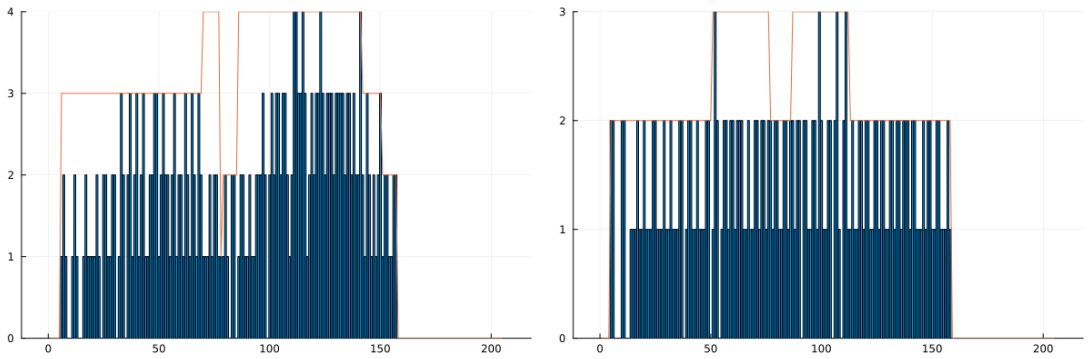
TABLE 7.4: Shifts on Weekdays

| shift length (h) | nr of shifts | start interval |
|---|---|---|
| 6 | 1 | 5 |
| | 1 | 87 |
| 9 | 1 | 5 |
| | 1 | 51 |

TABLE 7.5: Shifts on Saturdays

The scheduled shifts as well as their start times (in intervals) can be found in Tables 7.4 and 7.5 respectively.

In the case of weekdays, we schedule seven six-hour shifts. After subtracting the time for breaks from these, this yields an objective function value of 492. This means that in the worst case (taking the bounds obtained in section 7.1) the heuristic solution yields an objective value more than twice as high (approximately 135% worse) as with the original formulation. In the best case, our solution is still 25% worse than the upper bound of the optimal solution.




(A) Weekday Shifts

(B) Saturday Shifts

FIGURE 7.4: Worker requirement per interval and scheduled shifts. The bars show how many workers (y-axis) are required for each interval t from 1 to 205 (x-axis). The orange line shows how many workers are actually working according to the IP that schedules the shifts.

# 8. Conclusions and Recommendations

This chapter summarizes our research by giving recommendations and explaining opportunities for further research. Additionally, we answer our main and sub-research questions and discuss the results of the experiments. We examine the results in regards to their quality, reliability, and scientific and practical relevance.

## 8.1 Conclusions

We started our research with the research question: *"How can the MIP model for the warehouse worker requirement planning of flaschenpost SE be solved optimally while reducing computational time as much as possible?"* The goal was to find out whether and, if yes, to what extent we can solve the MIP model and how this could potentially influence the scheduling in the warehouses. To determine the answer to this question we will answer the sub-research question in this section.

**1: How is the requirement planning for the warehouse currently done at flaschenpost?**

In the introduction of this paper, we outline in detail how the operations in the warehouse work and how the planning is done accordingly. We model a homogeneous workforce and a working day from 7:00 until 24:00. The workforce is currently scheduled using a heuristic approach. This means that there is no method implemented yet that guarantees an optimal schedule.

**2: How can the performance of the panning procedure be made measurable?**

According to the formulation of the MIP, we use tardiness penalties and salary costs as the main KPIs. We are interested in how the new model, which, given the right assumptions and input data, yields an optimal solution, compares to the workforce schedules generated thus far. Next to these economic KPIs we also analyze the solution on a technical level. Here we especially examine the calculation time of MIP. For our given problem instances we wanted the calculation time to stay well under one hour. This is because we conducted our experiments for relatively small instances (so a larger instance would take even more time) and the MIP needs to be calculated daily in order to generate the tactical working schedules.

**3: What algorithms and techniques for solving (mixed) integer linear programming problems exist?**

We identified cutting planes and Branch-and-Bound algorithms as the most common and basic methods to solve MIP problems and their LP relaxations. All of the solvers we examined for this research combine these methods into a Branch-and-Cut framework. After hav-

ing identified this, we set out to customize the solver's branching or cutting rules to leverage information about the model that is not obvious to the solver.

**4: Which solver has the best performance for the given model?**

Unfortunately, none of the solvers could solve the fully specified model using the techniques we suggested. For the relaxed models, however, we identify Gurobi as the by far superior solver given the calculation time.

**5: What adaptions can we make to the model to make it more realistic or - if needed - easier to solve?**

We decrease the interval length $\Delta$ to make the model more realistic. This allows for a more accurate demand forecast as input. However, the experiments have shown that small values of $\Delta$ greatly increase the model's complexity. Additionally, even modeling the problem with rates instead of routes does not sufficiently decrease computation time. For further research, we need to either make additional adjustments to the model's formulation or implement more techniques to solve the current version faster.

For the time being, we suggest using the shift scheduling heuristic we describe in Section 7.3.1. Shift plans quickly be generated using this solution strategy and the company can compare the output to the currently generated schedules to decide whether or not the solution we propose provides sufficient advantages to pursue the ideas we explain in this thesis further.

**6: How can the new planning be implemented successfully?**

To implement the new planning method, further research on how the MIP can be solved faster is required. We answer this question as well as question 5 in more detail in section 8.2.

## 8.2 Further Research and Recommendations

At this point in the research, we cannot yet give recommendations with implications for the actual operations. We, therefore, focus on recommendations regarding the planning and modeling side of the problem.

While we have proven that the picking part of the MIP for the requirement scheduling can be solved faster by applying cuts to the demand constraints, our work opens up future research possibilities. This section briefly explains what can be done in the future to build on the work of this thesis.

### 8.2.1 Reflection of the Thesis

Before we give any recommendations, we briefly want to reflect on the work done. In conducting our research, several challenges and limitations became evident. The given MIP model proved to be harder to solve than expected and we made a mistake in implementing a constraint which required us to revise the work done up to that point. Although time seemed abundant in the beginning, it quickly became clear that in order to implement the full algorithm we decided on during the literature research required in-depth knowledge of linear (integer) programming and several other mathematical concepts. Due to this, we could not implement the time bucket formulation fully and therefore only touched upon its potential.

Nonetheless, we created a working code which generates a simplified solution for the given model and should provide a good starting point for further research.

### 8.2.2 Changes to the Model and Code

Since we were focusing on improving the model's performance by cuts, including the double time-indexed variables, we have not yet had time to implement other ideas for improvement. The following briefly outlines areas of interest.

#### More Cuts

We have shown that introducing our customized branch-and-cut logic can reduce the complexity of the model caused by double time-indexed variables. However, in the given model, this is not enough to reduce the calculation time to an acceptable amount.

Introducing similar cuts to other constraints of the model - especially the shift and area assignment - might lead to further improvements. However, these constraints must be designed differently since they are not suitable for time discretization in the same way as the demand injection and propagation constraints. If this proves unsuccessful at reducing the calculation time, a reformulation of the model might be worth considering.

#### Implementing a Proper Time Bucket Formulation

Dash et al. (2012) introduce a lot of techniques to solve time-indexed MIP. From all these ideas we have only used the idea of implementing time-windows for our given MIP. Throughout experimentation, it became evident that this idea already helps solve parts of our model faster.

Due to our experimental results, the results presented by Dash et al. (2012), and our company supervisor's experience with the techniques explained in that paper, the company should try to implement all of the ideas presented. Doing so will likely yield even greater results in decreasing the computational time. We have decided to not pursue this solution strategy further in this paper due to the time constraints.

#### Heuristics

One of the main issues stopping the model with cuts from being solved faster is that the solver takes a long time to find a feasible solution. The reason for this is, that our algorithm keeps adding cuts such that the solutions found so far become infeasible. The problem with this is that a feasible solution (incumbent) would deal as an upper bound and thus cut off a lot of nodes of the search tree with an objective value higher than that of the incumbent. If this problem persists even after implementing all of the ideas presented by Dash et al. (2012), introducing another customized heuristic might prove useful.

It might be worth including a heuristic at different points of the solving process—especially when no feasible solution has been found so far. Sticking to our Branch-and-Cut logic, this heuristic would need to retrieve the (fractional) values of the LP relaxation and use them to construct a feasible solution. The basic idea and sequence of this heuristic could look like this:

**Algorithm 3** Heuristic to Generate a Feasible Solution

1. Set the unpicked values $x^{u,i}_{\phi,t,t'}$ and $x^{u,i}_{\phi,t,t'}$ equal according to when they are picked.
2. Construct a feasible route for selling units from interval $t$ that are picked in interval $t'$ and area $\phi$.
3. Make sure to assign active ($x^{w}_{i,t}$) workers to the required areas ($x^{p}_{i\phi t}$).
4. Schedule shifts according to when workforce is required.

While this is a very simple idea to construct any feasible solution, it might already help the solver a lot if called early in the solving process. Of course, a more sophisticated heuristic can be implemented and provide tighter bounds and thus a shorter calculation time.

### 8.2.3 Data Analysis

This thesis's main focus is applying a customized Branch-and-Cut algorithm to solve a MILP, and we thus neglected to make the input data as realistic as possible. For our experiments, we created dummy data roughly representing a flaschenpost warehouse. To gain a more sophisticated and accurate result, we should analyze the relevant data to increase the realisticity of the model.

The demand forecast is important to a good solution since it directly dictates how much workforce is needed at a given time. Careful data analysis should be applied to guarantee an accurate forecast.

Another important factor is the productivity functions, which determine how many selling units can be picked in a certain time. This can be more accurately determined by analyzing the performance data of warehouse workers. Special attention should be paid to changes over time and differences per area and worker.

# References


Benders, J. F. (1962, 12). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252. Retrieved from https://doi.org/10.1007/bf01386316 doi: 10.1007/bf01386316

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear Optimization*.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017, 1). Julia: A fresh approach to Numerical Computing. *SIAM review*, *59*(1), 65–98. Retrieved from https://doi.org/10.1137/141000671 doi: 10.1137/141000671

Brusco, M. J. (1998, 9). Solving personnel tour scheduling problems using the dual all-integer cutting plane. *IIE transactions*, *30*(9), 835–844. Retrieved from https://doi.org/10.1023/a:1007552217588 doi: 10.1023/a:1007552217588

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Perfect formulations*. Retrieved from https://doi.org/10.1007/978-3-319-11008-0_4 doi: 10.1007/978-3-319-11008-0\{_}4

Dantzig, G. B. (1954, 8). Letter to the Editor—A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America*, *2*(3), 339–341. Retrieved from https://doi.org/10.1287/opre.2.3.339 doi: 10.1287/opre.2.3.339

Dantzig, G. B., & Wolfe, P. (1960, 2). Decomposition principle for linear programs. *Operations research*, *8*(1), 101–111. Retrieved from https://doi.org/10.1287/opre.8.1.101 doi: 10.1287/opre.8.1.101

Dash, S., Günlük, O., Lodi, A., & Tramontani, A. (2012, 2). A Time Bucket Formulation for the Traveling Salesman Problem with Time Windows. *Informs Journal on Computing*, *24*(1), 132–147. Retrieved from https://doi.org/10.1287/ijoc.1100.0432 doi: 10.1287/ijoc.1100.0432

Ernst, A., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004, 2). Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, *153*(1), 3–27. Retrieved from https://doi.org/10.1016/s0377-2217(03)00095-x doi: 10.1016/s0377-2217(03)00095-x

Felici, G., & Gentile, C. (2004). A polyhedral approach for the staff rostering problem on JSTOR. *www.jstor.org*. Retrieved from https://www.jstor.org/stable/30046074

flaschenpost SE. (2024). *Unternehmen | flaschenpost se*. https://www.flaschenpost.de/unternehmen. (Accessed: 01.04.2024)

Gass, S. I., & Fu, M. C. (2013). *Encyclopedia of Operations Research and Management Science*. Springer.

Guo, J. (2021, 1). Optimization approaches for solving Large-Scale personnel scheduling problems. *Deep Blue (University of Michigan)*. Retrieved from https://hdl.handle.net/2027.42/169758 doi: 10.7302/2803

Heerkens, H., van Winden, A., & Tjooitink, J.-W. (2017). *Solving managerial problems systematically* (First Edition ed.).

Heimerl, C., & Kolisch, R. (2009, 3). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR spectrum/OR-Spektrum*, *32*(2), 343–368. Retrieved from https://doi.org/10.1007/s00291-009-0169-4 doi: 10.1007/s00291-009-0169-4

Hong, Y.-C., Cohn, A., Gorga, S. M., O'Brien, E., Pozehl, W., & Zank, J. (2019, 5). Using optimization techniques and multidisciplinary collaboration to solve a challenging Real-World residency scheduling problem. *INFORMS journal on applied analytics*, *49*(3), 201–212. Retrieved from https://doi.org/10.1287/inte.2019.0995 doi: 10.1287/inte.2019.0995

IFH Köln. (2023). *Online monitor 2023*. https://einzelhandel.de/images/attachments/article/2876/HDE_Online_Monitor_2023.pdf. (Accessed: 08.04.2024)

Karmarkar, N. (1984, 12). A new polynomial-time algorithm for linear programming. *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 302–311. Retrieved from https://doi.org/10.1145/800057.808695 doi: 10.1145/800057.808695

Khachiyan, L. G. (1979). A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR 244 [Proceedings of the USSR Academy of Sciences]*, 1093 - 1096.

Koch, T., Berthold, T., Pedersen, J., & Vanaret, C. (2022, 1). Progress in mathematical programming solvers from 2001 to 2020. *EURO journal on computational optimization*, *10*, 100031. Retrieved from https://doi.org/10.1016/j.ejco.2022.100031 doi: 10.1016/j.ejco.2022.100031

Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*. doi: 10.1007/s12532-023-00239-3

Margot, F. (2009). *Symmetry in Integer Linear Programming*. Springer. Retrieved from https://doi.org/10.1007/978-3-540-68279-0-17 doi: 10.1007/978-3-540-68279-0\{_}17

Metei, A. J. (2019). *Optimization using linear programming*.

Van Den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013, 5). Personnel scheduling: A literature review. *European journal of operational research*, *226*(3), 367–385. Retrieved from https://doi.org/10.1016/j.ejor.2012.11.029 doi: 10.1016/j.ejor.2012.11.029

Vanderbei, R. J. (2007). *Linear Programming*. Springer Science & Business Media.

Winston, W. L. (1996). *Operations research*. Brooks/Cole.

Wölck, M., & Meisel, S. (2022, 7). Branch-and-Price Approaches for Real-Time Vehicle Routing with Picking, Loading, and Soft Time Windows. *INFORMS journal on computing*, *34*(4), 2192–2211. Retrieved from https://doi.org/10.1287/ijoc.2021.1151 doi: 10.1287/ijoc.2021.1151

Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., & Yan, J. (2023, 1). A survey for solving mixed integer programming via machine learning. *Neurocomputing*, *519*, 205–

217. Retrieved from https://doi.org/10.1016/j.neucom.2022.11.024 doi: 10.1016/j.neucom.2022.11.024

# A. Appendix A

## A.1 Problem Identification

| Problem | Influencable? | In the scope? | Comments |
|---|---|---|---|
| Manual shift planning using heuristics | Yes | Yes | We could focus the research on finding an improved scheduling method for warehouse workers and drivers |
| Inaccurate demand forecast | Yes | Yes | We could focus on improving the forecasting of demand to be better prepared to fulfil it |
| Demand variability | No | No | This is an extrinsic factor and therefore very hard if not impossible to control -> also the company does not want to artificially limit demand but rather try to fulfill all existing demand |

TABLE A.1: Potential Core Problems

## A.2 Research Design

| Knowledge Question | MPSM Phase | Research Type | Research Population | Data Gathering Method | Research Strategy | Presentation of Outcomes |
|---|---|---|---|---|---|---|
| How is the requirement planning for the warehouses currently done at flaschenpost? | 3 | Descriptive | flaschenpost employees (mainly supervisor) | Qualitative | Unstructured interviews | Summary of current planning and explanation why it is sub-optimal |
| How can the performance of the planning procedure be made measurable? | 4 | Descriptive | flaschenpost employees (mainly supervisor) | Qualitative | Unstructured interviews | List of KPIs |
| What data can we use as input for the model? | 4 | Descriptive | Company supervisor | Quantitative | Data analysis | Data formatted so it can be used as input to the model |
| Which solver has the best performance for the given model? | 5 | Evaluative | MILP solvers (software) | Quantitative | Experiments | Comparison of solvers |
| What algorithms for solving MILP exist? | 4 | Descriptive | Literature | Qualitative | Systematic Literature Review | List of algorithms including description and uses |
| Which algorithm is most suitable for the given problem? | 5 | Evaluative | Mathematical model | Quantitative | Experiments and evaluation of results | Decision about which algorithm yields the lowest computation time |

| | | | | | | |
|---|---|---|---|---|---|---|
| What adaptions can we make to the model to make it more realistic? | 4 | Descriptive / Explanatory | Mathematical model, literature | Quantitative | Experiments | Conclusion about how much the interval length can be reduced |
| How does the new planning method influence the chosen KPIs? | 5 | Explanatory | Output of the model, historical company data | Quantitative | Comparison of experiment results and current planning/schedule | Evaluation whether or not / by how much the shift scheduling can be improved with the results of the model |
| How can the new planning be implemented successfully? | 6, 7 | Descriptive | flaschenpost employees (mainly supervisor) | Qualitative | Unstructured interviews, literature review | Plan of action for the company |

TABLE A.2: Research Design

# B. Appendix B

## B.1 Decomposed Model

$$\min_{x} \quad \min_{x} \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{W}} x^{w}_{it} + \sum_{t\in T^{\Delta}} \left( \sum_{t'=0}^{t-\Pi^{i}} \pi x^{u,i}_{t,t'} + \sum_{t'=0}^{t-\Pi} \pi x^{u,p}_{t,t'} \right)$$

s.t.

$$x^{p}_{it} \le x^{w}_{it} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T} \qquad (1)$$

$$x^{r}_{it} \le \frac{1}{M} \sum_{t'=t-M+1}^{t} x^{p}_{it'} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}^{b} \qquad (2)$$

$$x^{r}_{it} \le 1 - \sum_{t'=t-M+1}^{t-1} x^{r}_{it'} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}^{b} \qquad (3)$$

$$x^{v}_{it} \le M x^{r}_{it} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}^{b} \qquad (4)$$

$$x^{v}_{it} \le \sum_{s\in\mathcal{S}} \sum_{t'=t-M+1}^{t} x^{s}_{si} p_{st'} \mathbb{P}(s,i) \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}^{b} \qquad (5)$$

$$x^{v,i}_{it} + x^{v,p}_{it} \le x^{v}_{it} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}^{b} \qquad (6)$$

$$\sum_{t'=max(1,t-\theta)}^{t-M+1} x^{v,i}_{tt'} \le \sum_{i\in\mathcal{W}} x^{v,i}_{it} \qquad \forall t \in \mathcal{T}^{b} \qquad (7)$$

$$\sum_{t'=max(1,t-\theta)}^{t-M+1} x^{v,p}_{tt'} \le \sum_{i\in\mathcal{W}} x^{v,p}_{it} \qquad \forall t \in \mathcal{T}^{b} \qquad (8)$$

$$x^{u,i}_{tt'} = d^{i}_{t} - \mathbb{1}_{M=1} x^{v,i}_{tt} \qquad \forall t \in \mathcal{T}^{e} \qquad (9)$$

$$x^{u,p}_{tt'} = d^{p}_{,t} - \mathbb{1}_{M=1} x^{v,p}_{tt} \qquad \forall t \in \mathcal{T}^{e} \qquad (10)$$

$$x^{u,i}_{tt'} = x^{u,i}_{t-1,t'} - x^{v,i}_{,t,t'} \qquad \forall\ t \in \mathcal{T}^{b}, t' \in \mathcal{T}^{b} : t'+M-1 \le t \qquad (11)$$

$$x^{u,p}_{tt'} = x^{u,p}_{t-1,t'} - x^{v,p}_{t,t'} \qquad \forall\ t \in \mathcal{T}^{b}, t' \in \mathcal{T}^{b} : t'+M-1 \le t \qquad (12)$$

$$x^{u,i}_{\min(T,t'+\theta)\,t'} = 0 \qquad \forall t' \in \mathcal{T}^{e} \qquad (13)$$

$$x^{u,p}_{\min(T,t'+\theta)\,t'} = 0 \qquad \forall t' \in \mathcal{T}^{e} \qquad (14)$$

$$x^{s}_{si} \in \{0,1\} \qquad \forall s \in \mathcal{S}, i \in \mathcal{W}$$

$$x^{w}_{it} \in \{0,1\} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x_{it}^{p} \in \{0,1\} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x_{it}^{r} \in \{0,1\} \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x_{it}^{v} \geq 0 \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x_{it}^{v,i} \geq 0 \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

$$x_{it}^{v,p} \geq 0 \qquad \forall i \in \mathcal{W}, t \in \mathcal{T}$$

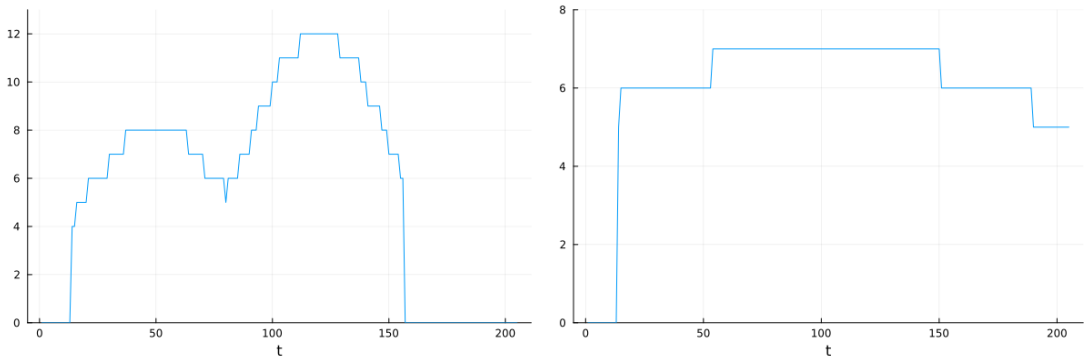
$$x_{tt'}^{v,i} \geq 0 \qquad \forall t, t' \in \mathcal{T} : t' \leq t - M$$

$$x_{t'\geq 0}^{v,p} \qquad \forall t, t' \in \mathcal{T} : t' \leq t - M$$

$$x_{tt'}^{u,i} \geq 0 \qquad \forall t, t' \in \mathcal{T} : t' \leq t - M$$

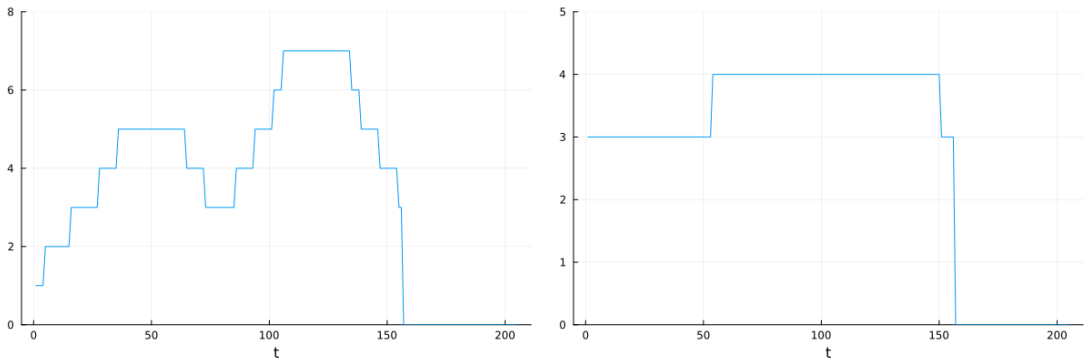
$$x_{tt'}^{u,p} \geq 0 \qquad \forall t, t' \in \mathcal{T} : t' \leq t - M$$

## B.2 Experiments




(A) Demand on weekdays (B) Demand on saturdays

FIGURE B.1: Instant-order demand patterns for $\Delta = 5$ and 66% of total demand




(A) Demand on weekdays (B) Demand on saturdays

FIGURE B.2: Pre-order demand patterns for $\Delta = 5$ and 33% of total demand