# Improving Efficiency in Outpatient Clinic Scheduling

Master's Thesis Business Analytics

**Author:** Goos Neefjes (2640149)
1st supervisor: Dr. René Bekker
Company supervisor: Phillip Kersten
2nd reader: Dr. Kevin Luck

2025-11-26

## Abstract

Currently, the scheduling process at the outpatient clinic of the VU Medical Centre is handled manually, making it not only tedious and error-prone but also inefficient. This thesis presents an optimization algorithm designed to improve efficiency in patient scheduling, which will be applied at the outpatient clinic of the Amstdam UMC, Location VUmc. By integrating a predictive no-show model with a mathematical scheduling framework, the research addresses the challenge of incorporating patient no-show behavior into scheduling decisions. Using the open-source Kaggle medical appointment dataset, a LightGBM model was trained to estimate individual no-show probabilities, achieving an accuracy of 69 %. The best-performing scheduling approach, the Horizontal and Vertical Batch Algorithm, generates a six-week schedule within 20 minutes and within 2.6 % of the calculated lower bound. Applying the best double-booking strategy—the cost-based double-booking approach— results in an increase of 13.9% patients scheduled in the same time window then without applying a double booking strategy, a doctor idle time reduction of 22.34% and a cost reduction of 17%.

# Acknowledgements

I would like to acknowledge several individuals and the organization who have made this thesis possible. First of all, I would like to thank my company supervisor, Phillip Kersten, for his expertise and guidance. I am thankful to my academic supervisor, Prof. Dr. René Dekker, for his inspiration, feedback, and academic guidance. Your input during our meetings have helped me to get more structure in this thesis. I also appreciate the time taken by the second academic reader, Prof. Dr. Kevin Luck. Finally an appreciation for the VU Medical Centre for sharing their current scheduling practice and helping to shape the scheduling algorithm. Thank you for being part of this contribution.

# Contents

# Chapter 1

# Introduction

The Dutch healthcare system, long recognized as one of the most expensive in Europe, is currently under significant pressure [33]. Rising demand for care, persistent staff shortages, and the increasing burden on healthcare workers pose serious challenges to maintaining high-quality services. One key area under pressure is the outpatient clinic, which plays a central role in managing chronic diseases, coordinating follow-up care, and alleviating pressure on inpatient services. Despite its importance, access to outpatient care is becoming increasingly difficult due to long waiting times [26]. These delays can lead to postponed diagnoses, treatment, and follow-up care—ultimately resulting in poorer health outcomes, increased patient anxiety, and reduced trust in the healthcare system [14].

While new patients face long waiting times and staff shortages persist, the available workforce is often inefficiently utilized. Missed appointments and late cancellations contribute to wasted capacity and financial losses. One study found that, within a single year, 31% of appointments were either canceled or missed, causing an estimated 3–14% loss of total annual clinic income [25].

Demographic developments further intensify the pressure on outpatient capacity. The proportion of older adults—who typically require more frequent and specialized care—is steadily increasing. In the Netherlands, the population aged 65 and above is projected to account for 45.4% of the 20–65 age group by 2060, up from 35.5% in 2025 [10]. Many of these individuals live with chronic conditions such as diabetes, cardiovascular disease, or cancer, which require continuous monitoring and intervention.

The COVID-19 pandemic highlighted the fragility of scheduling systems: staff infections forced mass appointment cancellations, creating extensive waiting lists and significant workloads for schedulers. A study in Scotland showed that pre-pandemic elective care capacity had not been fully restored even by the end of 2023 [30]. Although the pandemic represented an extreme case, rescheduling and recovery remain necessary on a day-to-day basis due to staff sickness and other absences.

Therefore, the healthcare system urgently calls for **sustainable, collaborative solutions** [38]. One promising approach lies in improving **patient scheduling**, a critical yet **complex process**. Each patient must be assigned to the correct doctor and room. The complexity arises from the large number of possible combinations in real-world settings. A typical example of a hospital scheduling instance might include: 600 patients to schedule, 40 doctors and 14 rooms.

Without considering any additional constraints, the total number of possible combinations for assigning patients to doctors and rooms is: 600*40*14 = 336,000. Adding realistic constraints further increases complexity, see here a list of constraints used in the scheduling process of the VU Medical Centre which will be applied in our scheduling algorithm:

1. Patients have a sequence of appointments that must occur in the correct order (first appointment A, then appointment B).
2. Some follow-up appointments require a recovery time (appointment B can be scheduled one week after appointment A).
3. Some appointments must occur at a fixed interval (appointment B must be scheduled one hour after appointment A).
4. Each doctor has specific working shifts (morning, afternoon, or full day).
5. Each doctor has a defined specialty and cannot treat every patient.
6. Some rooms can accommodate only a certain care type (MRI Scan in MRI Room).
7. Some appointment types have a limited number of spots available each day.
8. Some appointments can only be scheduled during specific parts of the day.

These constraints make the patient scheduling problem even more complex, making it an difficult task to find an close to optimal solution within a reasonable computation time. Smart scheduling techniques are needed that can reach close to an optimal solution efficiently.

Another promising solution is **double booking**. By accurately predicting the no-show probability of each patient and double-booking those with a high no-show likelihood, the wasted capacity and financial losses will be reduced caused by missed appointments. Additionally, double bookings help reduce the long waiting time for new patients for scheduling an appointment.

## 1.1 Research Objective

This project aims to develop a mathematical model that generates close to optimal outpatient clinic schedules using an offline planning approach, where patients are scheduled from a predefined list rather than in real time (the online planning approach). An online approach is also possible, in this approach patients are scheduled on arrival. A hybrid method could also be considered: in the first phase (offline), a schedule is generated from existing requests; in the second phase (online), remaining time slots are allocated to new patient requests. However, this study focuses primarily on the **offline scheduling approach**.

The central research question is:

> *Which modeling approach can be employed for scheduling, and how can it be integrated with no-show information to increase staff efficiency?*

The proposed model will assign various outpatient types to appropriate physicians and consultation rooms while accounting for key constraints such as staff availability, scheduled breaks, and patient no-show probabilities. By integrating **predictive analytics** into the scheduling framework, the model aims to dynamically optimize appointment allocations, mitigate no-shows, and enhance overall clinic efficiency. **The developed mathematical model will be applied at the outpatient clinic of the VU Medical Centre**.

## 1.2 Host Organization

This research is conducted in collaboration with **PersonalAize**, an innovative IT company based in the Netherlands that focuses on improving healthcare quality and efficiency through advanced machine learning and artificial intelligence solutions. PersonalAize collaborates closely with medical professionals, researchers, and healthcare institutions to bridge the gap between technological potential and clinical application.

Insights of the **Amstdam UMC, Location VUmc** are used. Based on interviews, the required constraints, care types and specialties are collected and used in this research. Due to privacy restrictions, it is not possible to use real outpatient clinic data. Instead, dummy data closely related to the hospital's data are used.

## 1.3 Thesis Structure

This thesis is organized into seven chapters. First, all related literature is discussed, followed by an exploration of the dataset used in this research. Chapters 4 and 5 describe the methodology and scheduling algorithms, and the thesis concludes with the results, discussion, and conclusion. For a more detailed structure, see:

**Chapter 2: Related Work** Reviews the literature on no-show prediction approaches, mathematical scheduling models, and metaheuristic methods for scalable scheduling optimization.

**Chapter 3: Data** Explores the dataset used in this study.

**Chapter 4: Methodology** Describes the feature engineering, feature selection, and prediction model selection. Also presents the scheduling model (mathematical formulation, objectives, constraints, and extensions) with the double book techniques.

**Chapter 5: Scheduling Algorithms** Details the Vertical Batch Algorithm, Horizontal Batch Algorithm, their combined variant, and the Fix-and-Optimize approach, explaining design choices and computational properties.

**Chapter 6: Results** Outlines predictions on no-show behavior of patients and the experimental results of different scheduling algorithms. In addition, different double book strategies are analyzed.

**Chapter 7: Conclusion** summarize the results with additional comments.

**Chapter 8: Discussion** Discusses limitations, provides practical implications for hospital scheduling, and suggests directions for future research.

# Chapter 2

# Related Work

This research is focused on two main topics: a no-show prediction model and a mathematical scheduling model. Therefore Section 2.1 describes literature on no-show prediction models and Section 2.2 describes literature on the scheduling models. Since meta-heuristics are widely applied in order to solve realistic problem instances close to optimality, section 2.3 describes literature around meta heurstic techniques and Section 2.4 describes all methods used in this research and new contributions in the field.

## 2.1 No-show prediction model

This section starts with common features used in predicting no-show behavior. After the features, models used in literature will be described and the performance including a limitation. Ending this section with double booking strategies.

### 2.1.1 Features

Prediction models for no-show in healthcare typically employ a combination of administrative, scheduling, and electronic health record (EHR) data. For example, the Mayo Clinic developed a model using ten years of data from a pediatrics clinic—including 7,988 distinct patients and over 104,000 visits—based on appointment features (visit type, prior no-show count, appointment time), demographics (distance to clinic, race, number of household), and insurance information (insurance holder, total insurance carriers) [16]. Prior no-show count is a strong predictor for no-shows. Another large-scale study in a rural healthcare network analyzed over 1.2 million appointments and identified lead time (greater than 60 days) and age (21–30 years) as strong predictors of no-shows [6]. Common predictive variables used in studies include [2, 3, 4, 6, 8, 13, 16, 19, 24] :

- **Patient-level history:** previous attendance.
- **Appointment characteristics:** lead time and time of day.
- **Demographics:** age, gender, insurance status, and socioeconomic factors.

In many studies, a patient's history of missed appointments has proven to be the single most influential variable in predicting future no-shows [2, 13, 19, 16, 24].

### 2.1.2 Modeling Approaches

Various statistical and machine learning techniques have been explored for predicting no-shows, ranging from traditional regression models to more advanced deep learning frameworks. **Gradient Boosting Machines (GBM)** are an extremely popular machine learning algorithm that have proven successful across many domains and is one of the leading methods for winning Kaggle competitions [18]. In a study in corporation with a Spanish clinic, GBM achieved an average error of 29.08% when predicting no-show probabilities [24]. Huang et al used a logistic regression model to predict the no show behaviour in a clinic in the city of Livonia (US), logistic regression achieved a training error rate of 10.6% and a validation error of 13.9% when predicting no-show probabilities [16]. More recent studies apply **Neural Networks (NN)**. A study compared the performance of a logistic regression with a NN, despite the high accuracy of the NN predictions (72%), it is not possible to directly know the predictive features responsible for the result [6]. New research explores hybrid and interpretable deep learning models. For instance, a 2025 study introduced a **Multi-Head Attention Soft Random Forest (MHASRF)** model for no-show prediction, achieving an accuracy above 93% and providing interpretable insights into patient behavior patterns [3]. Another 2024 study used symbolic regression and instance hardness thresholding to address data imbalance and enhance model generalization [13]. Model performance varies widely depending on dataset size and features. Most studies report accuracy values of around 0.7 [24, 13]. However, there is one important limitation:

- **Data imbalance:** No-shows are relatively rare (e.g., 6.65–19.03% of appointments), which can bias predictions unless resampling or reweighting methods are applied [13].

### 2.1.3 Double booking

An application where no-show information are applied is the double booking strategy. Lotfi and Torres [22] schedule patients in time slots, if the no show probability of the patient is below a certain threshold extra patients will be scheduled in the same time slot until the expected number of patients in the slot has reached 1. Double booking has some positive effects. For example, a study by Huang and Hanauer [16] compared two overbooking strategies, with and without the use of no-show predictions. The authors reported a reduction of 27% in employees' idle time hours, and 3% in total costs when using double book strategy. At the same time it negatively influenced patient waiting times as more patients are scheduled during the day.

## 2.2 Scheduling Models

Scheduling techniques and mathematical models for outpatient clinics have been extensively studied. Ahmadi-Javid, Jalali, and Klassen [1] provide a comprehensive review of mathematical scheduling models in outpatient appointment systems. In the literature review, the techniques can be divided into two main modeling techniques: **Mixed Integer Linear Programming (MILP)** and **Stochastic Programming**. Beside the model techniques, **meta-heuristics** are used in these model techniques that can produce near-optimal solutions within practical computation times helping to reduce the problem complexity.

### 2.2.1 Mixed Integer Linear Programming

Burdett et al. [9] present three MILP models of varying complexity that assign patients to hospital resources that incorporate numerous real-world technical conditions.

- The first model assigns patients to hospital areas.
- The second assigns patients to specific rooms within those areas.
- The third integrates multiple healthcare resources such as physicians and nurses.

Each model determines either the maximum number of patients that can be treated within a given period or the time required to serve a defined patient cohort. They introduce *Patient Care Plans (PCP)*, which define for each patient type the required activities, responsible units, duration, and necessary resources. These models account for outpatients, inpatients, and emergency patients.

Similarly, Hooshangi-Tabrizi et al. [15] proposed an integer programming model that maximizes the number of scheduled patients while considering patient preferences and patient–specialist continuity. A secondary integer program was introduced to reschedule existing appointments with constraints on allowable time shifts and the proportion of appointments affected.

In another contribution, Schimmelpfeng, Helber, and Kasper [28] developed a **hierarchical three-stage model** that enables solving medium- to large-scale instances on standard computers.

- Stage 1 schedules patients to specific days.
- Stage 2 assigns patients to time slots within each day.
- Stage 3 allocates resources and groups treatments.

This solving structure allows independent rescheduling of specific days if disruptions occur, such as staff illness or patient cancellations.

### 2.2.2 Stochastic Programming Techniques

Stochastic programming is frequently used to capture uncertainty in appointment durations, no-shows, or patient routing. Leeftink, Vliegen, and Hans [20] applied a **two-stage stochastic integer programming** model for multidisciplinary outpatient clinics. The first stage determines "here-and-now" decisions, while the second stage incorporates scenario-based uncertainty related to patient routing. Since the number of scenarios grows exponentially, they employ the Sample Average Approximation (SAA) method to reduce computational burden. Shuang, Chen, and Cai [31] introduced a **two-stage stochastic model** combining inter-day and intra-day scheduling. The first stage optimizes doctor availability, while the second assigns patients to time slots. Using the SAA method, they successfully balanced flexibility and computational efficiency.

Anvaryazdi, Venkatachalam, and Chinnam [5] proposed a two-stage stochastic mixed-integer linear programming (SMILP) model integrated with simulation. Their approach captures uncertainty in no-shows, waiting times, and fairness in provider assignments. The results significantly outperformed standard scheduling practices focused solely on waiting time minimization. Berg et al. [7] developed a similar two-stage stochastic MILP to optimize sequencing decisions, minimizing waiting times while maximizing resource utilization. Their results demonstrated that data-driven, flexible scheduling consistently outperformed static, fixed schedules.

strategies have also been studied using stochastic models. Samorani and LaGanga [27] showed that overbooking can reduce productivity losses due to no-shows, though computational complexity increases rapidly with scenario count.

## 2.3 Meta-Heuristics

Although mathematical programming techniques offer optimality guarantee, they often require excessive computation time. To address this, many researchers use **meta-heuristics**—high-level optimization strategies that guide problem-specific heuristics toward near-optimal solutions. The common meta-heuristics used in hospital scheduling algorithms are:

#### Adaptive Large Neighborhood Search (ALNS) and Fix-and-Optimize

Masson, Lehuédé, and Péton [23] adapted an Adaptive Large Neighborhood Search (ALNS) algorithm for hospital scheduling. ALNS iteratively destroys and repairs large portions of the current solution via multiple operators, typically augmented with a meta-heuristic such as simulated annealing or a weight-based scheme to balance exploration and exploitation. This class of methods is also often referred as **Fix-and-Optimize (F&O)**. The basic structure of these meta-heuristcs are:

1. Start from an initial feasible solution.

2. In each iteration, "fix" most decision variables to current values, the other decision are free (the "neighborhood").

3. Solve or improve the subproblem defined by the free variables.
4. Accept or reject the improvement (sometimes using simulated annealing or threshold acceptance) and proceed to the next iteration.

F&O approaches have shown strong performance. For example, Wickert et al. [35] demonstrated that an F&O-based matheuristic could generate near-optimal schedules for instances with up to 150 physicians within 30 minutes, outperforming commercial MILP solvers (e.g. Gurobi) especially on larger instances. In other domains, Turhan and Bilgen [32] applied a F&O and simulated annealing in a nurse rostering setting, showing improved solution quality over other meta-heuristics. Liu, Wang, and Hao [21] propose a two-stage optimization framework with model reduction and decomposition for admission scheduling, reflecting the same spirit of F&O methods. Different warm start procedures are compared and their influence on the solution quality. Some of these procedures resulted in the optimal solution, indicating the importance of a good initial feasible solution.

Because F&O allows incremental improvement while managing complexity, it is widely used in large-scale scheduling problems in hospital settings (e.g. resource allocation, staff rostering, admission scheduling). Its flexibility to integrate with exact solvers and metaheuristics makes it a powerful tool in bridging the gap between pure heuristics and monolithic MILP models.

**Hybrid and Decomposition Methods.**

Chouksey, Agrawal, and Tanksale [12] combined a **Benders Decomposition Algorithm (BDA)** with several metaheuristics such as rolling horizon, parallel processing, and a Fix-and-Optimize (F&O) hybridized with Simulated Annealing (SA). The rolling-horizon variant delivered near-optimal solutions for small and medium instances and consistently produced feasible solutions for large instances within practical time limits; although the pure F&O approach was faster, the Benders + rolling-horizon scheme achieved a smaller optimality gap. A practical enhancement frequently used alongside these decomposition schemes is **subgrouping** (also called grouping or partitioning). Subgrouping partitions the full scheduling problem into smaller subproblems. For example, grouping doctors with the same specialty or clustering patients based on patient characteristic blocks. Then each subgroup is solved independently.

The literature shows several successful applications of subgrouping in scheduling. For example, Schneider et al. scheduled **surgery groups** (clusters of surgery types with similar characteristics) within OR blocks and demonstrated that grouping lowers downstream bed-usage variability and reduces the risk of overtime while preserving high OR utilization [29]. Wang et al. partitioned elective surgeries into more-predictable and less-predictable cohorts and used simulation to show that such partitioning can reduce elective waiting times and simplify operational decision-making, provided reassignment between groups is allowed when needed [34].

The typical architecture of decomposition methods is two-level: (i) define subgroups and solve each subgroup with a solver, and (ii) combining all indepedent solution in a final global solution. The additional benefit of subgrouping is that it enables parallel computing which can be useful in extreme large problem sizes.

### Particle Swarm Optimization (PSO)

Chen et al. [11] applied an improved **Particle Swarm Optimization (PSO)** algorithm to the hospital staff scheduling problem, where each particle represented a complete monthly schedule. The algorithm incorporated a repair mechanism to restore feasibility whenever hard constraints were violated, while soft constraint violations were penalized in the fitness evaluation to guide the search toward high-quality solutions. By dynamically adjusting the weight and learning coefficients, the approach balanced global exploration with local exploitation. The results showed that PSO effectively improved scheduling efficiency and reduced manual errors compared to traditional rule-based methods. Wu, Shen, and Zhang [36] applied a bi-layer discrete Particle Swarm Optimization (PSO) algorithm to optimize operating room scheduling. The method jointly planned and sequenced surgeries with objectives of maximizing patient satisfaction and minimizing overtime and resource costs, using specialized repair and crossover operators to maintain feasibility and improve efficiency. Although the PSO effectiveness, it is not widely applied in hospital scheduling settings.

## 2.4 Applied literature and novel contributions

Based on the literature, a MILP model will be developed that integrates key operational constraints derived from the outpatient scheduling process at the VU Medical Centre. A MILP models have a great performance and are less complicated than stochastic programming models. Because the real-life problem involves large instances with many patients, physicians, and rooms, solving the full MILP to optimality within a reasonable time is computationally infeasible. Therefore, a set of **meta-heuristic techniques** will be applied to obtain high-quality, near-optimal solutions in a tractable time frame. These methods from the literature will be compared to evaluate their respective advantages and limitations in this specific hospital scheduling context.

- **Rolling Horizon.** Schedulers at the VUmc typically plan appointments over a six-week horizon. This long planning period substantially increases problem complexity. The rolling horizon approach mitigates this by dividing the overall time horizon into smaller, subperiods that are solved sequentially.
- **Subgrouping.** The scheduling model must assign a large number of patients to multiple doctors with varying specialties, which significantly increases the problem scale. To address this, a subgrouping strategy will be applied, where patients or doctors are partitioned into smaller, more homogeneous subsets. Each subgroup is then scheduled independently, either sequentially or in parallel, to improve computation time. The results from all subgroups are later merged into a global schedule.
- **Fix-and-Optimize (F&O).** The Fix-and-Optimize approach is a well-established meta-heuristic widely used in hospital and workforce scheduling. Including F&O in this study provides a benchmark for comparison with the other meta-heuristics.

In addition to evaluating each meta-heuristic individually, this research will also develop and test a **hybrid meta-heuristic** that combines the rolling horizon and subgrouping strategy. This hybrid method has not yet been extensively studied in the literature, particularly within outpatient clinic scheduling. By integrating temporal decomposition (rolling horizon) with structural decomposition (subgrouping), the hybrid algorithm aims to exploit the strengths of both techniques: maintaining manageable time windows while solving multiple smaller subproblems concurrently. The approach is expected to yield scalable performance, improved computational efficiency, and robust schedule quality across large, realistic hospital instances.

The best performing meta-heuristic will be used and the double book strategy of Lotfi et al will be implemented. This strategy will be compared with the well known Bailey-Welch booking strategy and a booking strategy which will minimize the cost using a Monte Carlo simulations.

# Chapter 3

# Data

The dataset used in this research is open-source and originates from a Kaggle competition predicting patient no-shows [17]. It contains 110,527 medical appointments and 14 associated variables (features). The target variable is the show up variable. The remaining variables can be grouped into two categories: **time-related variables**, such as the appointment creation and appointment dates, and **patient characteristics**, such as gender, age, and scholarship status. An overview of all variables is provided in Table 3.1, and sample rows of the dataset are shown in Table 3.2.

| # | Feature | Description |
|---|---|---|
| 0 | Patient ID | Unique identification number for each patient. |
| 1 | Appointment ID | Unique identification number for each appointment. |
| 2 | Gender | Gender of the patient ({Male, Female}). |
| 3 | Scheduled Day | Date on which the appointment was scheduled. |
| 4 | Appointment Day | Actual date of the appointment. |
| 5 | Age | Age of the patient in years. |
| 6 | Neighbourhood | Neighbourhood in which the patient resides. |
| 7 | Scholarship | Whether the patient is enrolled in a scholarship program ({T, F}). |
| 8 | Hypertension | Whether the patient has hypertension ({T, F}). |
| 9 | Diabetes | Whether the patient has diabetes ({T, F}). |
| 10 | Alcoholism | Whether the patient has alcoholism ({T, F}). |
| 11 | Handicap | Whether the patient has a handicap ({T, F}). |
| 12 | SMS Received | Whether the patient received an SMS reminder ({T, F}). |
| 13 | Showed_Up | Whether the patient showed up ({T, F}). |
| 14 | Date Difference | Difference (in days) between the scheduled and appointment dates. |

Table 3.1: Overview of dataset features.

| Patient ID | AppointmentID | Gender | ScheduledDay | AppointmentDay | Age | Neighbourhood | Scholarship | Hypertension | ... | SMS_Received | Showed_Up | DateDiff |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5642903 | F | 2016-04-29 | 2016-04-29 | 62 | JARDIM DA PENHA | F | T | ... | F | T | 0 |
| 1 | 5642503 | M | 2016-04-29 | 2016-04-29 | 56 | JARDIM DA PENHA | F | F | ... | F | T | 0 |
| 2 | 5642549 | F | 2016-04-29 | 2016-04-29 | 62 | MATA DA PRAIA | F | F | ... | F | T | 0 |

Table 3.2: Sample records from the medical appointment dataset (partial view).

The target variable represents whether a patient shows up for their appointment. Figure 3.1 presents the distribution of this variable. Approximately 21% of appointments result in a no-show. At the VUmc outpatient clinic, the no-show rate is around 4% and 15% when cancellations are included. It is unclear whether cancellations are counted as no-shows in this dataset.

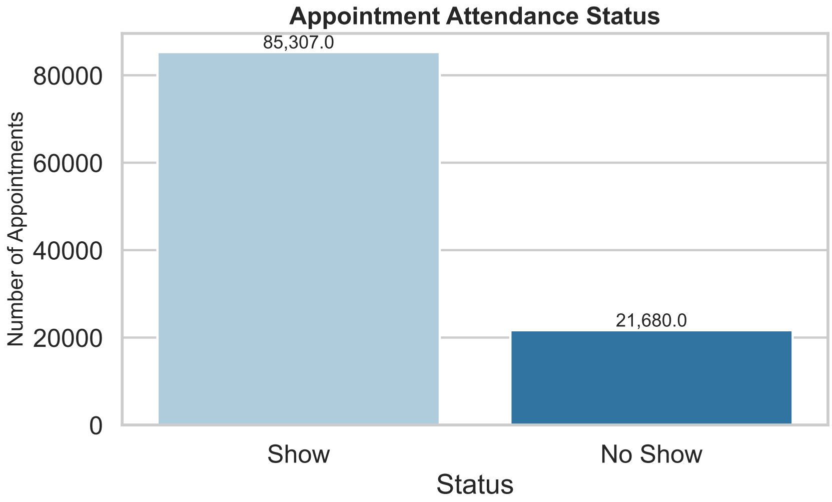


Figure 3.1: Show-up distribution of appointments.

Figure 3.2 presents the age distribution of patients, ranging from 1 to 115 years. The distribution remains relatively stable until around age 58, after which a gradual decline is observed.

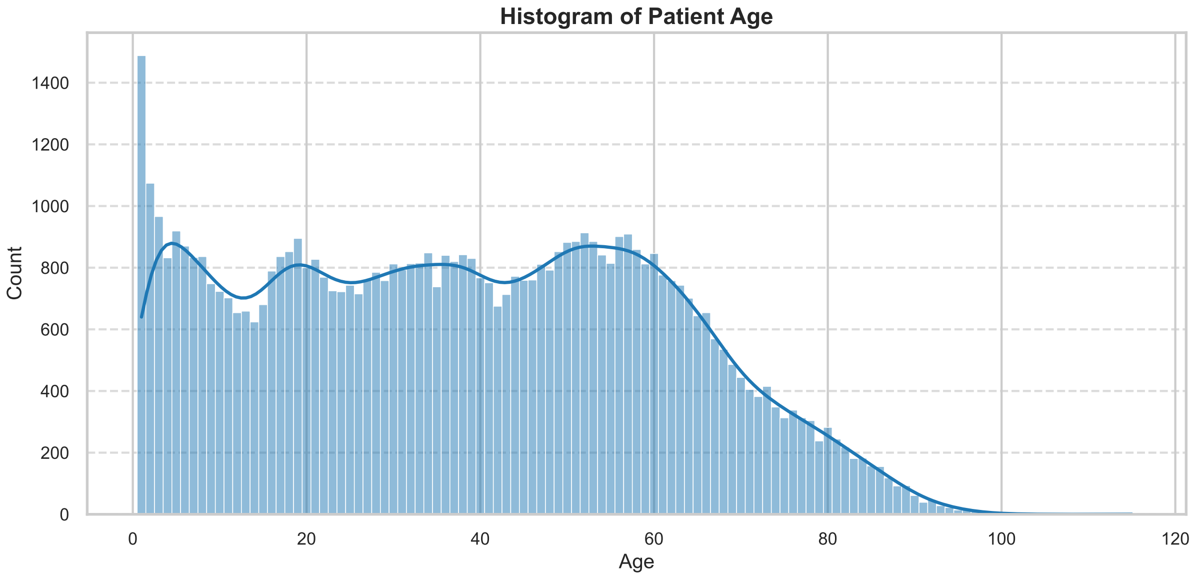


Figure 3.2: Distribution of patient age (in years).

Other patient-related features are represented as Boolean variables. An overview of these values are shown in Figure 3.3. All features, except *SMS_Received*, exhibit a strong imbalance, with fewer than 20% of true values compared to false values.

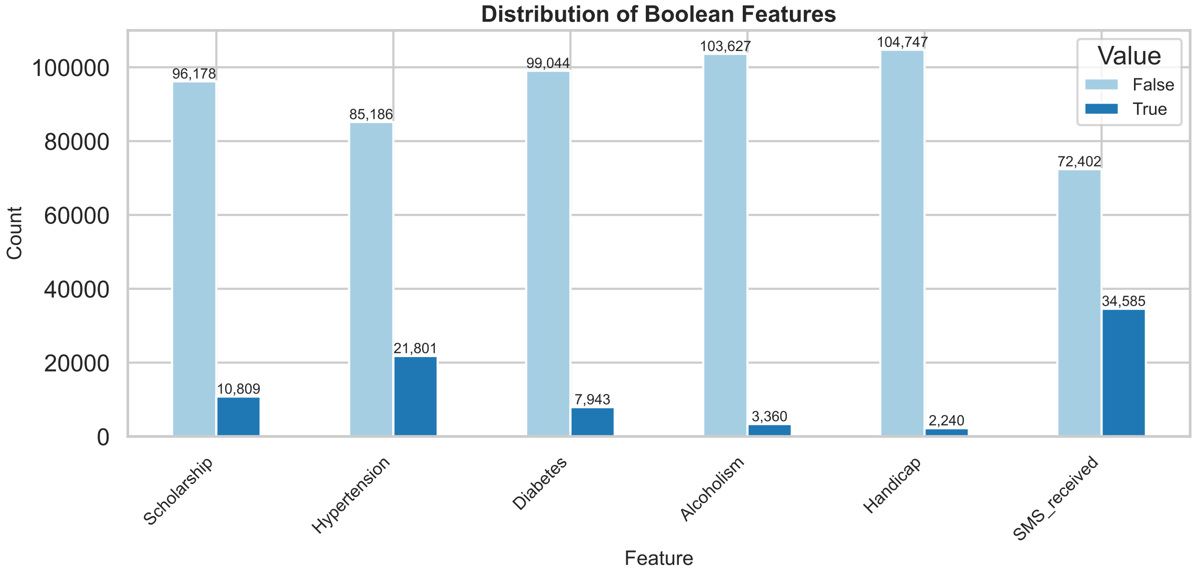


Figure 3.3: Overview of Boolean patient features.

The difference in days between the scheduled date and the appointment date are shown in Figure 3.4. It can be seen that a lot of appointments are scheduled on the same day or a couple days later while a small proportion of the appointment is scheduled 25 days later.

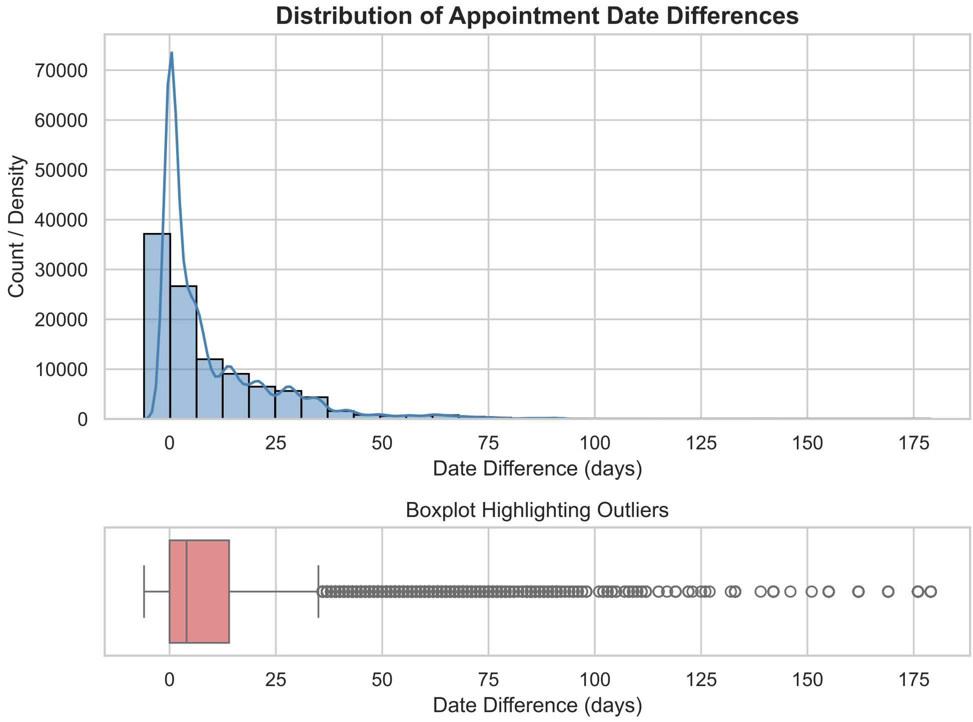


Figure 3.4: Difference in days between the scheduled and appointment dates.

The number of times a patient has visited the outpatient clinic, are shown in 3.5. Around 28.4% of patients visit the clinic more than once. These patients have a prior attendance history that can be compared to their current show-up behavior. For example, if a patient has five visits, the first four can be used to calculate their average historical show-up rate, which can then be compared to the fifth (most recent) appointment. Figure 3.6 illustrates this comparison.

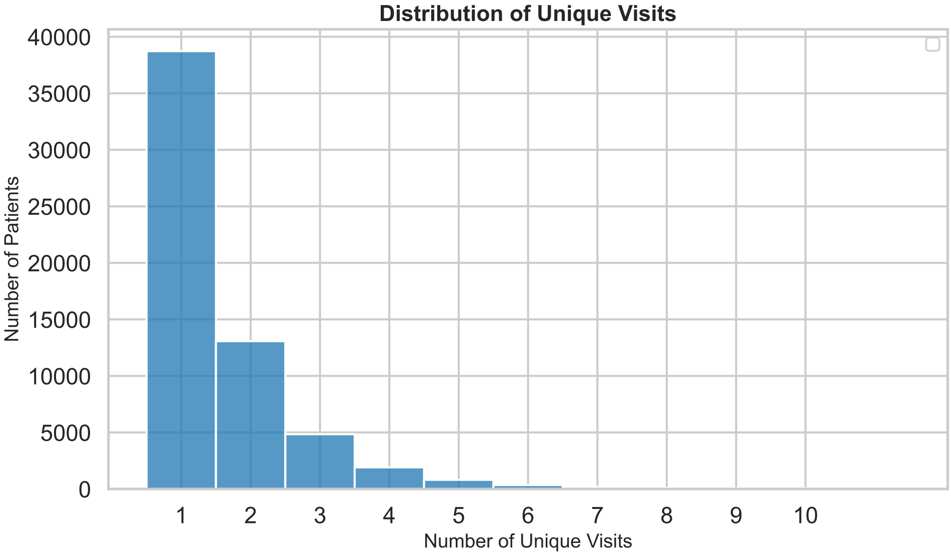


Figure 3.5: Distribution of the number of unique outpatient clinic visits per patient.

Patients with a previous show-up rate close to 1 are more likely to attend future appointments. However, for patients with a historical show-up rate between 0 and 0.9, this relationship weakens, and their likelihood of showing up becomes slightly lower than that of missing the appointment.

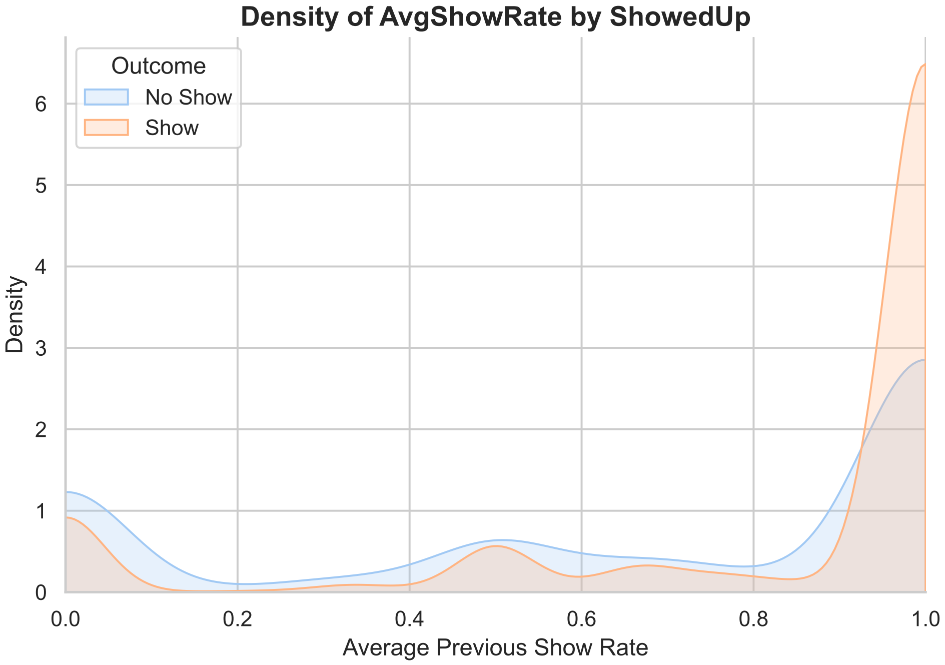


Figure 3.6: Comparison of previous and current show-up behavior.

To investigate the influence of these features on the no-show behaviour of the patients the correlation coefficient is calculated for these variables with the variable *Showed_Up*. See Table 3.3.

| Feature | Correlation with Showed_up |
|---|---|
| Age | 0.07 |
| Hypertesion | 0.04 |
| Diabetes | 0.02 |
| Handicap | 0.01 |
| Gender | 0.00 |
| Alcoholism | 0.00 |
| Scholarship | -0.03 |
| SMS_received | -0.13 |
| Date difference | -0.19 |

Table 3.3: Correlation coefficients

As it can be seen in Table 3.3, the features have no high correlation with the target variable *Showed_up*. The only features that have a absolute correlation of larger than 0.05 are *Age*, *SMS_received* and *Date difference*. These features will be further explored for interesting findings.

First, the feature *Date_difference*, which is the difference in days between scheduled day and appointment day, is compared with the no-show percentage in Figure 3.7. The values are grouped into bins, where the bar height indicates the no-show percentage and the number in each bin the number of appointments. **Appointments in which this day difference is equal to zero are left out since these appointments have a show-up rate of 0.953, see Figure 3.8**.

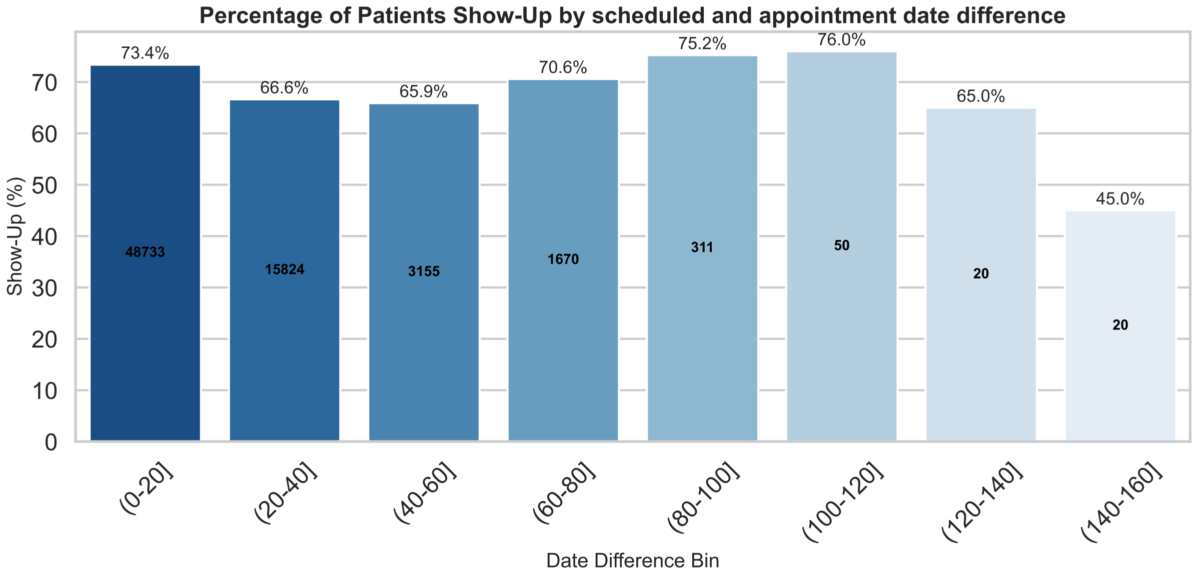


Figure 3.7: No-show compared for varying difference scheduled date and appointment date

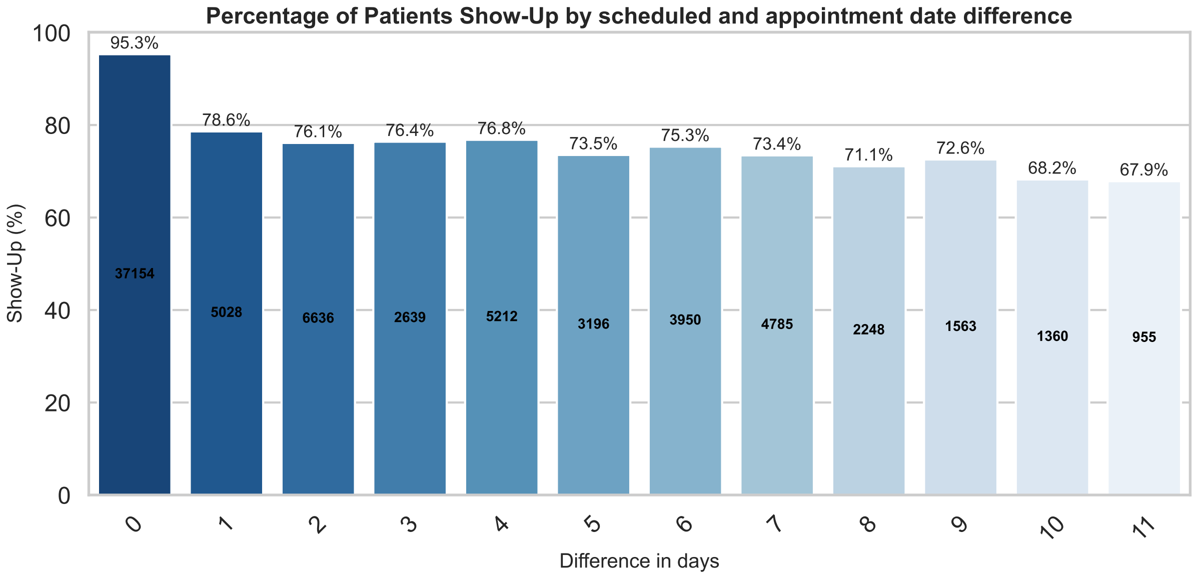


Figure 3.8: No-show compared to difference scheduled date and appointment date

Figure 3.8 displays the small *Date_difference*, if the appointment day and the scheduled day are the same, then there is a high chance that the patient will show up. Otherwise, there is a chance higher than 20% that a patient don't show up.

When comparing the no-show percentage with patient age, a lower show-up rate is observed among younger individuals (approximately 5–30 years old) compared to older patients (Figure 3.9). After about age 24, the show-up rate increases steadily until roughly 60 years of age. As shown in Figure 3.2, the number of available records declines sharply beyond this point, and results for patients over 90 years of age should therefore be interpreted with caution.

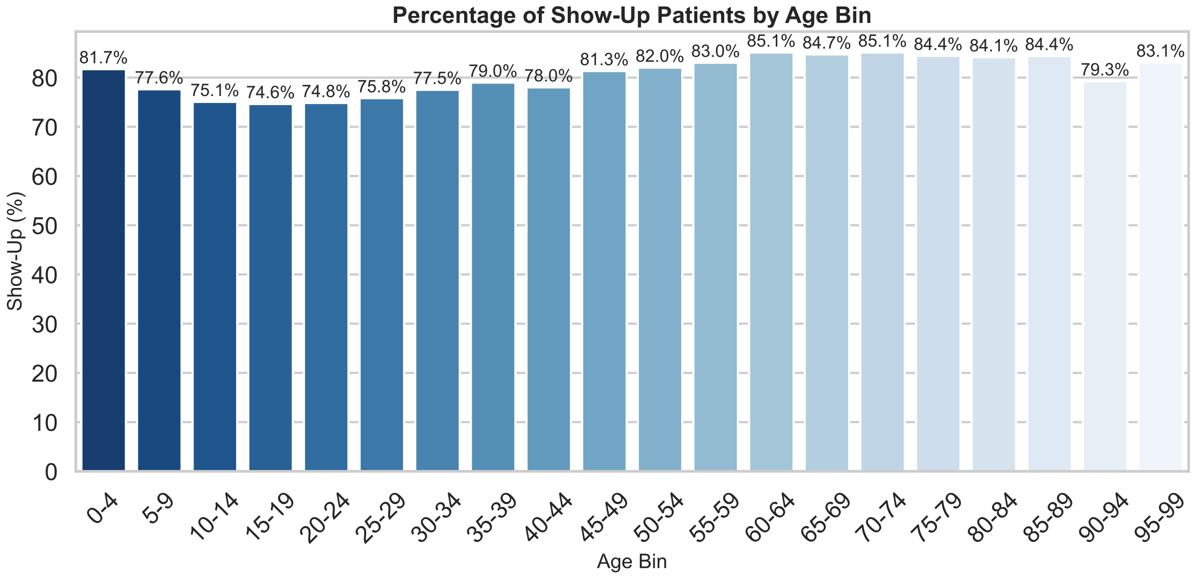
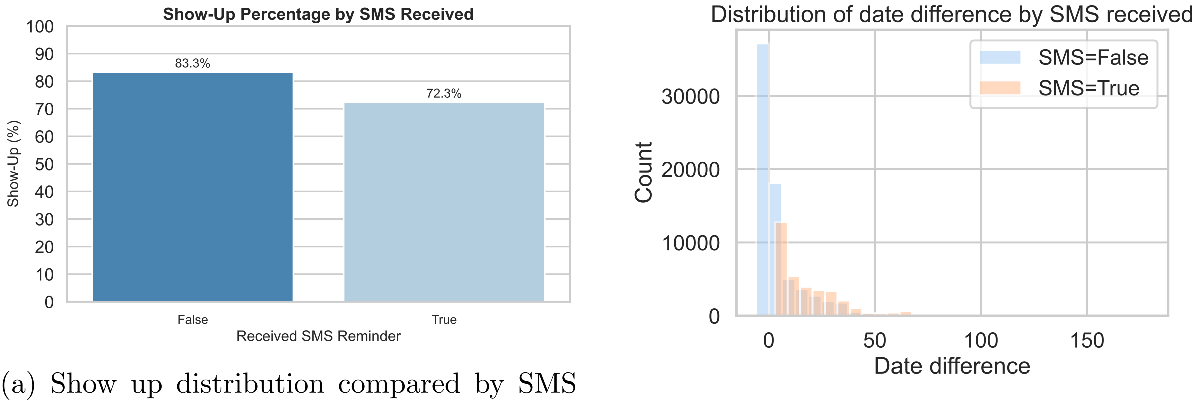


Figure 3.9: Show-up percentage by age group

To investigate the correlation between *Showed_up* and *SMS_received*, see Figure 3.10a. The figure displays the negative correlation with show up rate. This negative correlation means that patients receiving an SMS (a reminder about their appointment) are less likely to show up than patients without a reminder. This unnatural conclusion comes from the fact that patients that have a date difference larger than 1 day will receive an SMS while patients with a date difference of 0 or 1 day will not receive an SMS, see Figure 3.10b. It is know that the highest show up rate is among patients with a date difference of 0 or 1 day, see Figure 3.8. If the correlation between *Showed_up* and *SMS_received* is investigated for patients only with a date difference larger than 1 day, then patients with an SMS reminder are more likely to show up than patients without an SMS reminder, see Figure 3.11.



(a) Show up distribution compared by SMS received

(b) SMS received compared to date difference

Figure 3.10: Comparison SMS received

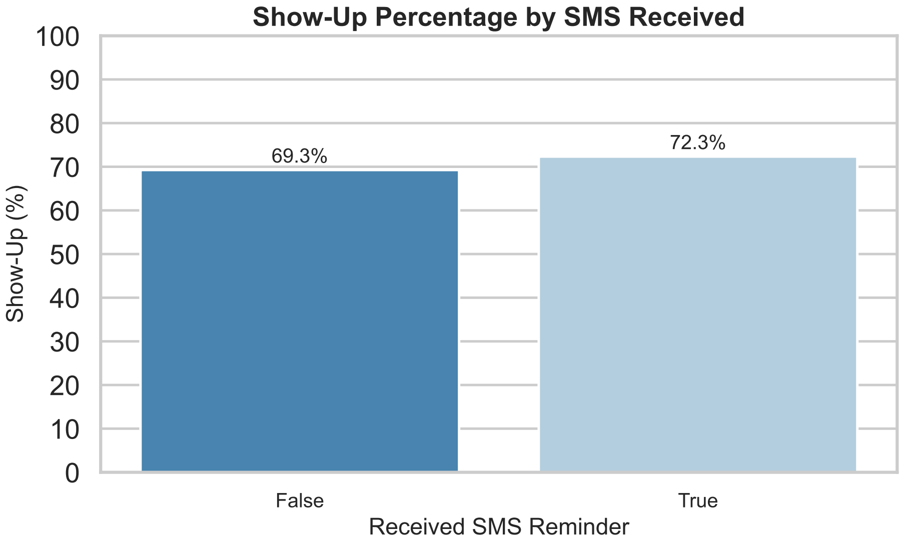


Figure 3.11: Show up distribution compared by SMS received excluding appointments with *Day_difference* of 0 and 1

# Chapter 4

# Methodology

In Section 4.1, we describe the process of feature engineering and selection, as well as the predictive model developed to estimate patient no-shows. Section 4.2 details the methodology used for the scheduling algorithms and Section  4.3 provides the mechanics of different double book strategies.

## 4.1 No-show prediction model

### 4.1.1 Feature Engineering

As displayed in Table 3.3. the existing features have a low correlation with the target variable. Therefore, feature engineering is applied to enhance the predictive power of the data. No data cleaning was needed, as the dataset is complete and all values are correctly labeled as integers, strings, or datetime objects. Furthermore, based on the data exploration, no outlier detection or removal was necessary. With the exception of the feature *Date.diff*, there were no indications of potential outliers. The feature *Date.diff* contains some relatively large values, but none are extreme or isolated from the rest of the data.

Prior to feature engineering, the dataset was split into multiple subsets, each containing all appointments of a unique patient. Consequently, the number of subsets corresponds to the number of unique patients in the dataset. The following features were created:

- **Avg_prev_show_rate**: The average show-up rate of a patient based on their previous completed appointments. The most recent appointment is excluded from this calculation and labeled as the *current appointment*, while all other appointments are labeled as *historical appointments* and used to calculate the average previous show-up rate. If there are no previous appointments than this feature is set equal to 0.5 to indicate a random show up rate.
- **Num_visits**: The total number of times a patient has visited the outpatient clinic.
- **Day_of_week**: The day of the week for the appointment, labeled as “Monday”, “Tuesday”, . . . , “Sunday”.

Once these features were created for each subset, only the data entries labeled as *current appointment* were retained for the final dataset. This resulting dataset contains all current appointments of each patient. The number of entries in this dataset is therefore equal to the number of unique patients in the original dataset.

The dataset exhibits a substantial class imbalance, with only 21% of appointments labeled as no-shows. The final dataset, containing the most recent appointments for all patients, includes approximately 60,000 records. Given the dataset's size, undersampling without replacement was applied to the majority class (show) to achieve a more balanced ratio of 65% show to 35% no-show. After this adjustment, the resulting dataset comprises 34,108 entries.

### 4.1.2 Feature Selection

After feature engineering, the coefficient of correlation is calculated again to evaluate their correlation with the target variable. Among the features, only *Avg_prev_show_rate* exhibited a notable correlation with the target, with a correlation coefficient of 0.54. Based on the correlation coefficients and graphical analyses, the following features were selected for predicting the patient show-up rate:

| **Feature Name** | **Type** | **Description** |
|---|---|---|
| Age | Int | Age of the patient in years |
| Hypertension | Boolean | Indicates whether the patient has hypertension (Yes = True) |
| SMS_received | Boolean | Indicates whether the patient received an SMS reminder (Yes = True) |
| Date.diff | Int | Difference in days between the scheduled date and the appointment date |
| Avg_prev_show_rate | Float | Average show-up rate based on previous appointments (0.5 if no previous appointments exist) |

Table 4.1: Overview of selected features for predicting the target variable *Showed_up*

### 4.1.3 Model Selection

For the model selection, a python package is used for quick evaluation. The package contains all classifier models and, if input data is provided, compares the prediction performances of these models based on minimal tuning. The package is called `LazyClassifier`. Based on the preliminary results, two best performing models were selected for further analysis: **LightGBM** and **Nearest Centroid**.

#### LightGBM

LightGBM is a gradient boosting framework that constructs an ensemble of decision trees in a sequential manner. Unlike traditional boosting methods, LightGBM grows trees leaf-wise rather than level-wise, allowing it to reduce loss more efficiently and often achieve better predictive performance. It is well suited for large datasets, can naturally handle missing values, and effectively captures non-linear relationships between features. In this work, LightGBM is used to model complex patterns in patient behavior that contribute to no-shows.

The mechanics of LightGBM rely on two core principles. First, **gradient boosting** iteratively improves the model by minimizing a differentiable loss function, enabling gradual refinement of predictive accuracy. Second, LightGBM employs **leaf-wise tree growth**, expanding the leaf that yields the greatest reduction in loss, which accelerates convergence and enhances model precision. LightGBM provides feature importance estimates, offering insight into which patient or appointment characteristics most strongly influence no-show predictions.

**Nearest Centroid**

The Nearest Centroid classifier is a distance-based algorithm that assigns each sample to the class whose centroid lies closest in the feature space. It is computationally simple and highly interpretable, making it a useful baseline model. Although it cannot capture complex non-linear interactions in the same way as LightGBM, it offers a clear reference point for understanding the underlying data structure and for assessing the added value of more sophisticated models.

The mechanics of the Nearest Centroid classifier are straightforward. First, it computes a **centroid** for each class by taking the mean feature vector of all samples belonging to that class. Next, during prediction, it measures the **distance**—typically Euclidean—between a new sample and each centroid, assigning the sample to the class with the smallest distance. The method offers strong interpretability, as feature contributions can be examined by comparing the relative distances to the class centroids.

### 4.1.4 Model Training and testing

For training and testing, the dataset with 34.108 data entries is used with the target variable and features described in Table 4.1. In this dataset the class imbalance is corrected. A 70/30 train-test split was applied, ensuring that both the training and test sets maintained a balanced class distribution.

For **LightGBM**, a gradient boosting decision tree model, hyperparameter tuning was performed using a grid search with 3-fold cross-validation on a subset of 5,000 samples. Key parameters such as `num_leaves`, `learning_rate`, `n_estimators`, `subsample`, and `colsample_bytree` were optimized to maximize accuracy. The best model was then evaluated on the test set using accuracy. See Table 4.2 for the parameter grid.

The best model achieved an **accuracy of 0.7** with the following parameters (bold):

| **Parameter** | **Values** |
|---|---|
| num_leaves | 20, **30**, 40 |
| max_depth | **-1** (no limit) |
| learning_rate | **0.05**, 0.10, 0.15 |
| n_estimators | 50, **100**, 150 |
| subsample | **0.6**, 0.7, 0.8 |
| colsample_bytree | 0.6, **0.7**, 0.8 |

Table 4.2: Hyperparameter Grid for LGBMClassifier

For the **Nearest Centroid** classifier, a distance-based model, hyperparameter tuning was carried out using a grid search with 3-fold cross-validation on a subset of 5,000 samples. The search space included multiple distance metrics (`euclidean, manhattan, chebyshev, minkowski`) as well as a range of shrinkage thresholds. Shrinkage thresholds adjust the class centroids by pulling them toward the global mean and setting small class–mean differences to zero, thereby reducing the influence of noisy or weakly informative features. The best-performing configuration was subsequently evaluated on the test set using accuracy as the primary performance metric.

| **Parameter** | **Values** |
|---|---|
| metric | **euclidean**, manhattan, chebyshev, minkowski |
| shrink_threshold | None, 0.001, 0.002, **0.005**, 0.01, 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.5, 0.7, 1.0 |

Table 4.3: Parameter grid for Nearest Centroid hyperparameter tuning.

Eventually the model with the distance metric `euclidean` and shrink_threshold of 0.005 is the best performing model with an **accuracy of 0.66**. Since the tuned lightGBM model is the best performing model, this model will be used in predicting the no-show behaviour of patients in the scheduling model.

### 4.1.5 Patient Selection

The final LightGBM model will be trained on the cleaned dataset, which contains 34,108 entries, using the features described in Section 4.1.2. However, not all entries are used for training: 5,000 patient records are set aside for testing the scheduling algorithm and using their no-show probability for analyzing the performance of different double book strategies. The remaining 29,108 entries are randomly sampled without replacement to form to train the prediction model. Once trained, the LightGBM model can predict the no-show probabilities for the 5,000 reserved patients, providing the basis for evaluating how different scheduling or double-booking strategies may affect appointment attendance.

# 4.2 Scheduling model description

## 4.2.1 Introduction notation

The input of the model consists of different sets such as time, demand and resources. For time we have a set of slots $T$ in which $H$ is the last slot in the set (make span). The demand and resources are discussed in more details below.

### Demand

For the demand the model needs a set of patients $P$ and set of care types $C$, in which each care type $c \in C$ has a care duration $l_c \in L$ (set of care durations) and a recovery time $b_c \in B$ (set of care recovery times). Every patient $p \in P$ has a list of care types for which the patient needs help ($C_p$) consisting of just one or more care types, in the correct order of scheduling. Therefore the first care type in $C_p$ needs to be scheduled before the second care type in the $C_p$, the second care type in $C_p$ before the third care type in $C_p$, etc. An example input with 3 patients and 3 care types is given below:

$$T = \{1, 2, 3, ..., H\}$$
$$P = \{P1, P2, P3\}$$
$$C = \{\text{Consult, MRI, Blood test}\}$$
$$L = \{\text{Consult = 1 slot, MRI = 2 slots, Blood test = 1 slot}\}$$
$$B = \{\text{Consult = 0 slots, MRI = 1 slots, Blood test = 0 slots}\}$$

$$C_1 = [\text{Consult}]$$
$$C_2 = [\text{Blood test, Consult}]$$
$$C_3 = [\text{Blood test, MRI, Consult}]$$

The care sequence of patient $P3$ is $C_3$. In this example, first a *Blood test* needs to be scheduled with a duration of 1 slot and no recovery time, then an *MRI* must be scheduled after the time slot of the *Blood test* with a duration of 2 slots and recovery time of 1 slot and the last care type *Consult* must be scheduled after the *MRI* with a care duration of 1 slot and no recovery time. See Table 4.4 for a simplified schedule for P3.

| slot 1 | slot 2 | slot 3 | slot 4 | slot 5 |
|---|---|---|---|---|
| Blood test | MRI | MRI | - (MRI recovery) | Consult |

Table 4.4: A simplified schedule for P3

### Resources

For the resources the model needs a set of doctors $D$ and a set of rooms $R$. Each doctor is specialized in certain care types, meaning the doctor can provide help for these care types. The same holds for a room, a room has certain capabilities, meaning that a room may facility certain care types. Additionally, a doctor has a certain shift $s \in S = \{\text{morning, afternoon, full day}\}$. This specifies which part of the day the doctor is available.

$$D = \{\text{D1 (Clinician), D2 (Echografist)}\}$$
$$R = \{\text{R1 (MRI Room), R2 (Doctor Office)}\}$$

A Clinician is specialized in a *Consult* and a *Blood test*, which can be facilitated in *Doctor Office*. An Echografist is specialized in an *MRI*, which can be facilitated in an *MRI Room*.

$$S = \{\text{morning (slot 1-3), afternoon (slot 4-6), full day (slot 1-6)}\}$$

$D1$ is available a full day and $D2$ is available in the morning.

A patient needs to be scheduled at the correct doctor and room, making the scheduling problem more complex. The simplified schedule of P3 in Table 4.4 may look like the schedule in Table 4.5 if doctors and rooms are incorporated.

| Doctor / Room | slot 1 | slot 2 | slot 3 | slot 4 | slot 5 |
|---|---|---|---|---|---|
| D1-R2 | Blood test | | | | Consult |
| D2-R1 | | MRI | MRI | - (unavailable) | - (unavailable) |

Table 4.5: A schedule for P3

The goal of the model is to schedule all patients while minimizing the make span ($H$), the number of slots needed for scheduling all patients. This will result in an optimal schedule, such as the schedule in Table 4.6 in which the min make span is equal to 5 slots.

| Doctor / Room | slot 1 | slot 2 | slot 3 | slot 4 | slot 5 |
|---|---|---|---|---|---|
| D1-R2 | Blood test (P3) | Consult (P1) | Blood test (P2) | Consult (P2) | Consult (P3) |
| D2-R1 | | MRI (P3) | MRI (P3) | X | X |

Table 4.6: Optimal schedule for P1, P2 and P3

### 4.2.2 Mathematical formulation: basic model

**Sets, parameters and decision variables**

| **Sets** | |
|---|---|
| **Patients and care** | |
| $P$ | Set of patients. |
| $C$ | Set of care types. |
| $C_p$ | Set of ordered list of care tasks required by patient $p \in P$. |
| $L$ | Set of care durations. |
| $B$ | Set of recovery times (break times). |
| **Resources** | |
| $S$ | Set of shifts, $S$ ={morning, afternoon, full day}. |
| $D$ | Set of doctors. |
| $R$ | Set of rooms. |
| $W_d \subset C$ | Set of specialities of doctor $d \in D$. |
| $F_r \subset C$ | Set of capabilities of room $r \in R$. |
| **Time** | |
| $T = \{t_1, t_2, \ldots, t_{H-1}\}$ | Discrete time-slots, $H$ is scheduling horizon. |
| **Parameters** | |
| $K$ | Number of slots per day, $K \in \mathbb{Z}^+$. In the example of **??** K = 6. |
| $M$ | The first number of slots of the day that are morning slots, $M \in \mathbb{Z}^+$ and $M < K$. In the example of 4.2.1 M = 3. Meaning that the first 3 slots of the day are morning slots. If K = 12, the morning slots are {1,2,3,7,8,9} |
| **Derived parameter** | |
| $a_{d,t}$ | $= 1$ if doctor $d \in D$ is available in time slot $t \in T$, else 0. The matrix A in which $a_{d,t}$ is an element of depends on the parameters $K$ and $M$. |
| **Decision Variables** | |
| $x_{p,c,d,r,t} \in \{0, 1\}$ | $= 1$ if patient $p$ starts care $c \in C_p$ with doctor $d \in D$ in room $r \in R$ at time $t \in T$, else 0. |
| $y_{p,c,d,r,t'} \in \{0, 1\}$ | $= 1$ if patient $p$ occupies slot $t'$ for care $c$ with doctor $d$ in room $r$, else 0. |
| $s_{p,c} \in \mathbb{Z}_{\geq 0}$ | Start time of patient $p$ with care task $c$. |
| $H \in \mathbb{Z}_{\geq 0}$ | Makespan (time horizon). |

Table 4.7: Sets, parameters, and decision variables

**Objective**

$$\min\ H$$

Minimize the make span that is the total number of slots needed to schedule all patients.

**Constraints**

$$\sum_{d\in D,\ r\in R,\ t\in T} a_{d,t} \cdot x_{p,c,d,r,t} = 1, \qquad \forall p \in P,\ \forall c \in C_p \tag{1}$$

1) Each patient $p \in P$ with care task $c \in C_p$ in the list of care sequence $C_p$ is assigned to a doctor $d \in D$ and a room $r \in R$ for exactly one time slot $t \in T$ if doctor $d$ is available at $t$.

$$s_{p,c} = \sum_{d\in D,\ r\in R,\ t\in T} t \cdot x_{p,c,d,r,t}, \qquad \forall p \in P,\ \forall c \in C_p \tag{2}$$

2) For a specific patient $p \in P$ with care task $c \in C_p$, which is assigned to a certain doctor $d \in D$ and room $r \in R$, the starting time $s_{p,c}$ equals the time slot $t$ where $x_{p,c,d,r,t} = 1$.

$$y_{p,c,d,r,t'} \geq x_{p,c,d,r,t}, \qquad \forall p \in P,\ \forall c \in C,\ \forall d \in D,\ \forall r \in R,\ \forall t,\ \forall t' \in \{t, \ldots, t + L_c - 1\} \tag{3}$$

3) If a specific patient $p \in P$ with care task $c \in C_p$ is assigned to a certain doctor $d \in D$ and room $r \in R$, then $y_{p,c,d,r,t'} = 1$ not only for the assigned slot $t$ but also for the following $L_c - 1$ slots.

$$\sum_{p\in P,\, c\in C_p,\, r\in R} y_{p,c,d,r,t'} \leq 1, \qquad \forall d \in D,\ \forall t' \in T \tag{4}$$

4) A doctor $d \in D$ can handle at most one patient per time slot $t' \in T$.

$$\sum_{p\in P,\, c\in C_p,\, d\in D} y_{p,c,d,r,t'} \leq 1, \qquad \forall r \in R,\ \forall t' \in T \tag{5}$$

5) Each room $r \in R$ can accommodate at most one patient at any time slot $t' \in T$.

$$s_{p,c_{i+1}} \geq s_{p,c_i} + L_{c_i} + B_{c_i}, \qquad \forall p \in P, \forall \text{ consecutive tasks } c_i, c_{i+1} \in C_p \text{ x } C_P \tag{6}$$

6) For a patient $p \in P$ with a care sequence larger than 1 ($len(C_p) > 1$), the next care task $c_{i+1}$ starts only after the previous one $c_i$ finishes and the required recovery (break) time in slots $B_{c_i}$ has passed.

$$H \geq s_{p,c} + L_c, \qquad \forall p \in P,\ \forall c \in C_p \tag{7}$$

7) The make span $H$ is the planning horizon and must be greater than or equal to the finish time of every patient $p$ and care task $c \in C_p$ with care duration (length) $L_c$.

(4.1)

**Doctor availability matrix**

Based on the constants $K$ (the number of slots per day) and $M$ (the number of slots of the day that are morning slots), the doctor availability matrix $a_{d,t}$ is created. Moreover, if a day consist of 20 time slots ($K = 20$) and a morning consist of 8 time slots ($M = 8$) then the afternoon consist of 12 time slots (20-8=12). If a doctor is available on Monday morning (6-10-2025) and Wednesday morning (8-10-2025) and the scheduling starting horizon is Monday moring (6-10-2025) then the doctor is available in the following slots:

$$a_{d,t} = 1 \text{ for } t \in \{1, 2, ..., 8, 41, 42, ..., 48\}$$

The constants also give context to the results of the model, if the minimum make span is 60 while $K = 20$ it is known that the model has found a min make span of 3 days and the doctor assignments can be translated to specific time slots.

The model described in Section 4.2.2 is a complex model which can be applied in practice but for most hospitals there are additional constraints such as max appointments per shift type, limitation on new patients or not only have a minimal recovery time but also a maximum care help date for necessary care needed for a specific time or discussing results. In order to integrate these constraints the model is extended with the following sets and constraints.

### 4.2.3 Mathematical formulation: extended model

As already known, the scheduling time window consists of $T = \{1, 2, 3, \ldots, H\}$. Each day consists of $K$ slots, where the first $M$ slots correspond to the *morning* shift and the remaining $K - M$ slots to the *afternoon* shift. The full-day shift covers all $K$ slots of the day. If, for example, a doctor works during a morning shift, the doctor is only available in the first $M$ slots of that day. A full-day doctor, however, is available throughout all $K$ slots.
Let $W = \{1, 2, \ldots, W_{\max}\}$ represent the set of days in the planning horizon, where $W_{\max} = \lceil |T|/K \rceil$. Each time slot $t \in T$ belongs to a specific day $w \in W$ and shift $s \in S$. The corresponding subset of time slots belonging to day $w$ and shift $s$ is denoted by $T_{w,s}$. Formally:

$$T_{w,s} = \{\, t \in T \mid \text{day}(t) = w,\ t \in T^s \,\},$$

where

$$\text{day}(t) = \left\lceil \frac{t}{K} \right\rceil, \qquad \text{slot\_in\_day}(t) = \big((t-1) \bmod K\big) + 1.$$

This decomposition allows the model to apply constraints on a per-shift, per-day basis and with:

$$T^s = \begin{cases} \{1, 2, \ldots, M,\ K, K+1, \ldots, K+M,\ 2K, 2K+, \ldots\}, & s = \text{mornimg}, \\ T/T^{morning}, & s = \text{afternoon}, \\ T, & s = \text{full}. \end{cases}$$

**Example of the extended setup** Consider again a small example:

$$T = \{1, 2, \dots, 12\} \qquad K = 6 \qquad M = 3$$
$$\Rightarrow 2 \text{ days } (W = \{1, 2\}), \text{ each with 6 slots.}$$
$$T^{\text{morning}} = \{1, 2, 3, 7, 8, 9\}, \quad T^{\text{afternoon}} = \{4, 5, 6, 10, 11, 12\}, \quad T^{\text{full}} = \{1, \dots, 12\}.$$

Then, for day $w = 1$:

$$T_{1,\text{morning}} = \{1, 2, 3\}, \quad T_{1,\text{afternoon}} = \{4, 5, 6\}, \quad T_{1,\text{full}} = \{1, \dots, 6\}.$$

and for day $w = 2$:

$$T_{2,\text{morning}} = \{7, 8, 9\}, \quad T_{2,\text{afternoon}} = \{10, 11, 12\}, \quad T_{2,\text{full}} = \{7, \dots, 12\}.$$

Additionally, patients can be classified as regular patients and new patients. Regular patients are patients that already have visited the hospital clinic. New patients are patients that visited the hospital clinic for the first time. These subsets, $T_{w,s}$, will be used in the new constraints to maximize the number of new patients per subset or maximize the number of new patients per doctor. It is possible that a regular or new patient needs to see the same doctor for their entire care sequence $C_p$, if this is not the case the patient can be scheduled at an other doctor for their follow up care.

### Example schedule for the extended model

This example illustrates how patients, doctors, rooms, and shifts are assigned according to the extended model constraints. The same information is used of Example 4.2.1 extended with:

- New patients: $P^{new} = \{P1, P2, P3\}$.
- Same-doctor patients: $P^{same} = \{P2\}$.
- Maximum appointments for shift $s \in S$ and care type $c \in C$: $A_{c,s} = 2$.
- Maximum number of new patients per doctor per shift: $N_{D1} = 2$, $N_{D2} = 1$.

| Doctor / Room | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 |
|---|---|---|---|---|---|---|---|---|---|
| D1-R2 | Blood test (P3) | Consult (P1) | - | - | - | - | Blood test (P2) | Consult (P2) | Consult (P3) |
| D2-R1 | - | MRI (P3) | MRI (P3) | x | x | x | - | - | - |

Table 4.8: Optimal schedule for P1, P2 and P3 based on extended model

**Explanation of Table 4.8:**

- Doctor availability is respected: D1 is available in slot $t \in \{1, 2, ..., 9\}$ and D2 in slot $t \in \{1, 2, 3, 7, 8, 9\}$.
- Maximum appointments per shift ($A_{c,s}$) are respected.
- Maximum new patients per doctor per shift ($N_d$) are respected.
- P2, being a same-doctor patient ($P^{same}$), is assigned entirely to D1.

### Sets, derived parameters and constants

| **Sets** | |
|---|---|
| $P^{new} \subseteq P$ | Set of new patients. |
| $P^{same} \subseteq P$ | Set of patients that require the same doctor for their entire care sequence. |
| $W = \{1, \ldots, W_{\max}\}$ | Set of calendar days, where $W_{\max} = \lceil \lvert T \rvert / K \rceil$. |
| $T_{w,s}$ | Set of time slots belonging to day $w \in W$ and shift $s \in S$. |
| **Derived Parameter** | |
| $M_{c,d}$ | $= \begin{cases} 1, & \text{if care type } c \text{ can be performed by doctor } d, \\ 0, & \text{otherwise.} \end{cases}$<br>This matrix ensures that each care type is only assigned to compatible doctors, based on their specializations. |
| **Constants** | |
| $A_{c,s}$ | Maximum number of appointments of care type $c \in C$ allowed per shift type $s \in S$. |
| $N_d$ | Maximum number of new patient appointments allowed for doctor $d \in D$ per shift. |

Table 4.9: Extended sets, derived parameter and constants for the Extended Model

### Extended constraints

$$\sum_{p \in P} \sum_{d \in D} \sum_{r \in R} \sum_{t \in T_{w,s}} x_{p,c,d,r,t} \leq A_{c,s}, \quad \forall c \in C,\ \forall s \in S,\ \forall w \in W \tag{9}$$

9) For each day $w$ and shift $s$, the total number of appointments of care type $c$ within the slots of $(w, s)$ cannot exceed $A_{c,s}$.

$$\sum_{p \in P^{new}} \sum_{c \in C} \sum_{r \in R} \sum_{t \in T_{w,s}} x_{p,c,d,r,t} \leq N_d, \quad \forall d \in D,\ \forall s \in S,\ \forall w \in W \tag{10}$$

10) For each doctor $d$, day $w$, and shift $s$, the number of new patients assigned during that shift cannot exceed $N_d$.

$$\sum_{r,t} x_{p,c_1,d,r,t} = \sum_{r',t'} x_{p,c_2,d,r',t'}, \qquad \forall p \in P^{same},\ \forall d \in D,\ \forall \text{ consecutive tasks } c_i, c_{i+1} \in C_p \text{ x } C_P \tag{12}$$

12) If a patient $p \in P^{same}$ requires the same doctor for all consecutive tasks $c_1, c_2 \in C_p$ must be assigned to the same doctor $d \in D$.

(4.2)

## 4.3 Double Book strategies

To further improve physician utilization and reduce idle time caused by patient no-shows, this research implements several **double booking strategies**. Each strategy differs in the way appointments are scheduled to balance the expected number of patients, physician availability, and potential patient waiting times. There are three double booking strategies.

(1) **Double booking, standard.** In the standard double-booking strategy, additional patients are scheduled so that the expected number of patients showing up is close to 1 for each appointment for a specific doctor.

Let $v_i \in [0,1]$ denote the predicted show-up probability for patient $i$. The expected number of patients who will appear in slot $t$ is:

$$\mathbb{E}[V_t] = \sum_{i \in \mathcal{P}_t} v_i,$$

where $\mathcal{P}_t$ represents the set of patients assigned to slot $t$. The goal of double booking is to get $\mathbb{E}[V_t] \approx 1^-$. Most patients have high show-up probabilities, typically greater than 0.6. Therefore, the double-booking strategy does not consider each time slot independently but it evaluates the **cumulative expected number of patients** over a specific half-day. Starting with the first appointment and continuing with the last appointment for a specific doctor in this specific half-day. Each appointment, the cumulative expected number of patients is divided by the current appointment number. If this outcome is still less than 1 with the additional patient included, double booking is possible.

For illustration, consider the schedule in Table 4.8, generated using the extended model. The show-up probabilities for the patients are:

$$v_1 = 0.7, \quad v_2 = 0.75, \quad v_3 = 0.65.$$

The schedule is divided into four half-days: $T_{1,\text{morning}}, T_{1,\text{afternoon}}, T_{2,\text{morning}}, T_{2,\text{afternoon}}$. As mentioned, for each half-day the cumulative expected number of patients is evaluated, starting with $T_{1,\text{morning}}$. with the following appointments for doctor D1:

$$\text{S1: Blood test (P3, show-up probability 0.65)}$$
$$\text{S2: Consult (P1, show-up probability 0.7)}$$

- At the first appointment (S1), the **expected number of patients showing up** is 0.65.
- After the second appointment (S2), the **cumulative expected number of patients** is $0.65 + 0.7 = 1.35$.

Now, if a fourth patient $P_4$ (show-up probability $v_4 = 0.6$) requires a Blood test, this patient can be double booked at the second appointment because:

$$\frac{\text{cumulative expected number of patients}}{\text{number of appointments}} = \frac{0.65+0.7+0.6}{2} \leq 1$$

To generalize this idea, the following set will be added:

- $\mathcal{A}_{d,T}$ = set of patients scheduled with doctor $d$ in half-day $T$

For a given doctor $d$ and half-day $T = T_{w,s}$, the expected cumulative number of patients is:

$$\mathbb{E}[V_{d,T}] = \sum_{i \in \mathcal{A}_{d,T}} v_i.$$

If there are $|\mathcal{A}_{d,T}|$ patients scheduled in that half-day, the average expected number of patients showing up per appointment until appointment $m$ where $m < |\mathcal{A}_{d,T}|$ is:

$$\overline{V}_{d,T} = \frac{\sum_{i \in \mathcal{A}_{d,T}}^{m} v_i}{m}.$$

### Double-Booking condition

An additional patient $j$ can be scheduled in appointment $z$ of that half-day if:

$$\frac{(\sum_{i \in \mathcal{A}_{d,T}}^{z} v_i) + v_j}{z} \leq 1,$$

When a patient $j$ is double booked with patient $i$ in time slot $t$ then this time slot is full. For each time slot a maximum of 2 patients can be scheduled.

Patient $j$ is added to the set $A_{d,t}$ when a double booking occurs. This process continues until no additional patients can be added without violating the constraint. The patients selected for double booking are those who are currently unscheduled, and they are chosen in increasing order of their show-up probabilities.

(2) **Double booking, Bailey–Welch.** The Bailey–Welch strategy modifies the standard approach by introducing an initial double booking at the start of the scheduling horizon. Subsequent double bookings are then determined following the same rules as the standard strategy. The rationale for the initial double booking is to mitigate slow start periods and reduce early shift underutilization. However, because the expected number of patients at the beginning of the shift can exceed one (around two), this may increase waiting times for patients scheduled early. This strategy thus represents a trade-off between early shift efficiency and patient punctuality, which can be particularly important in settings with variable arrival patterns.

(3) **Double booking, cost optimization.** This strategy aims to schedule double appointments in a way that minimizes the overall operational cost, including factors such as physician idle time, patient waiting time, and potential overtime. The optimization is performed using a **Monte Carlo simulation**, which repeatedly samples patient show-up scenarios based on predicted probabilities and evaluates the associated costs. For each candidate schedule, the expected cost is estimated, and the double booking configuration that minimizes this expectation is selected. The pseudo-code for the Monte Carlo-based cost optimization procedure is provided in Algorithm 1. For the cost overview see Table 4.10:

| **Function** | **Cost (€/hour)** |
|---|---:|
| Clinician | 240 |
| Doctor Assistant | 160 |
| CR-Thorax | 650 |
| Echografist | 800 |
| Spirometrie | 450 |
| General | 40 |
| **Cost of Waiting (patient)** | 10 |

Table 4.10: Cost Overview (Cost per Hour)

**Algorithm 1** Monte Carlo Simulation Cost

```
Require: Schedule $S$, patient show probabilities $P$, idle cost $c_{idle}$, waiting cost $c_{wait}$,
    $minutes\_per\_slot$ and $care\_durations$, number of simulations $n_{sim}$
Ensure: Estimated total costs: waiting, idle, and overtime
Initialize lists: $total\_wait$, $total\_idle$, $total\_overtime$
for $i = 1$ to $n_{sim}$ do
    for doctor d in S do
        $current\_time \leftarrow 0$
        $waiting\_cost, idle\_cost, overtime\_cost \leftarrow 0$
        for all slots $s$ in sorted order $S$ do
            $slot\_start \leftarrow s \times minutes\_per\_slot$
            $slot\_end \leftarrow slot\_start + minutes\_per\_slot$
            for all patient $p$ in $S[d, s]$ do
                $r \leftarrow$ random()
                if $r < P[p]$ then
                    $service\_time \leftarrow care\_durations[p] \times minutes\_per\_slot$
                    $start\_service \leftarrow \max(current\_time, slot\_start)$
                    $wait\_time \leftarrow start\_service - slot\_start$
                    $waiting\_cost \leftarrow waiting\_cost + (c_{wait}/60) \times wait\_time$
                    $current\_time \leftarrow start\_service + service\_time$
                    Mark $p$ as processed
                    $show \leftarrow show + 1$
                end if
            end for
            if $current\_time < slot\_end$ then
                $idle\_cost \leftarrow idle\_cost + (c_{idle}/60) \times minutes\_per\_slot$
            end if
        end for
        $schedule\_end \leftarrow (\max(S) + 1) \times minutes\_per\_slot$
        if $current\_time > schedule\_end$ then
            $overtime\_cost \leftarrow (c_{idle}/60) \times 2.5 \times (current\_time - schedule\_end)$
        end if
        Append $waiting\_cost$, $idle\_cost$, $overtime\_cost$ to totals
    end for
end for
return Mean of $total\_wait$, $total\_idle$, $total\_overtime$
```

# Chapter 5

# Scheduling Algorithms

Five algorithms are evaluated in this study: the *Vertical Batches Algorithm*, the *Horizontal Batches Algorithm*, the *Horizontal and Vertical Batches Algorithm*, the *Optimal Algorithm*, and the *Fix-and-Optimize Algorithm.* Each algorithm solves the scheduling problem and produces a schedule that can be represented as a table (see Table 4.6). In this table, each row consist of a doctor–room combination with the scheduled patients for a certain time slot. In Section 5.1, each algorithm is explained briefly with the help of the example described in Section 4.2.1 with 3 patients, 2 doctors and 2 rooms. This example will be used in the explanation of the algorithms. Section 5.2, 5.3, 5.4 and 5.5 contains the mathematical details of the algorithms.

## 5.1 Overview Algorithms

### 5.1.1 The Optimal algorithm

The optimal algorithm takes as input the full set of resources of 4.2.2 and solves the MILP in a single iteration for all patients. Table 4.6 illustrates the output of the Optimal Algorithm based on the example.

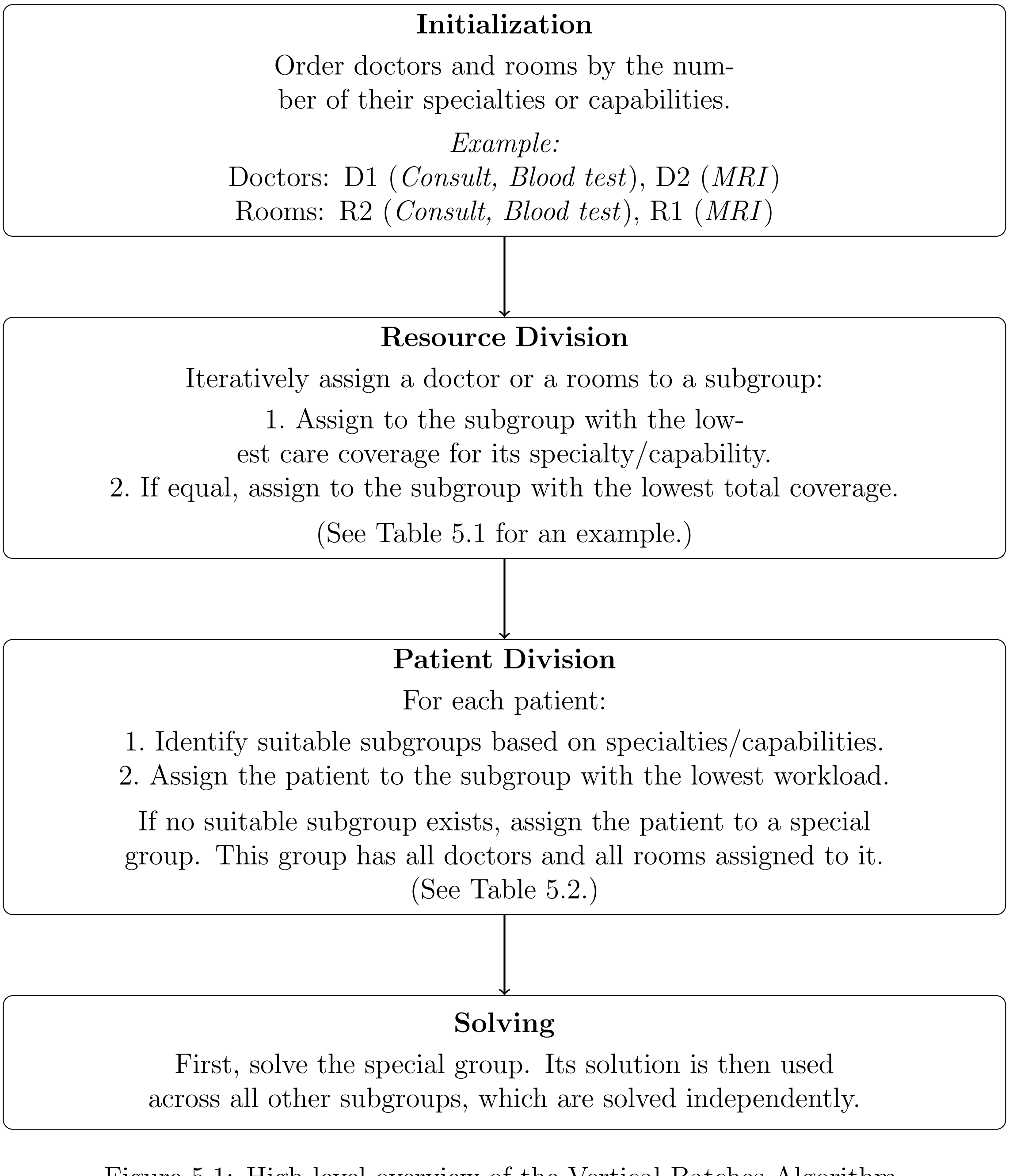
### 5.1.2 The Vertical Batches algorithm

The vertical batches algorithm partitions doctors and rooms into subgroups and solves the scheduling problem for each subgroup independently. This approach is inspired by Yüksektepe (2009)[37], who applied an MILP model for multi-class data classification by dividing the dataset into two subgroups using a similarity–dissimilarity function.

In our context, subgroups are formed based on the care coverage. Care coverage refers to the number of doctors or rooms available and qualified to provide a specific type of care. For example, Subgroup 1 may have a care coverage of 0 doctors and 0 rooms for Care A, while Subgroup 2 has 1 doctor and 1 room for the same care. Therefore, Subgroup 2 has a higher care coverage for Care A. Each doctor or room is assigned to the subgroup with the lowest care coverage. If there a multiple subgroups with the same low care coverage, a random subgroup is selected.

Patients are assigned to subgroups based on each subgroup's workload, defined as the number of patients requiring a specific type of care. For instance, Subgroup 1 may have a workload of 10 patients for Care B, while Subgroup 2 has a workload of 5 patients for Care B. In this case, Subgroup 1 has a higher workload for Care B than Subgroup 2. Patients are assigned to the subgroup with the lowest workload, if there are multiple subgroups with a low workload then a random subgroup is selected.

When this algorithm is applied on the example in 4.2.1, doctor D1 and room R2 form subgroup 1, while doctor D2 and room R1 form subgroup 2. The main steps of the algorithm are displayed in Figure 5.1 with additionally tables and figures to illustrate the process.



**Initialization**

Order doctors and rooms by the number of their specialties or capabilities.

*Example:*
Doctors: D1 (*Consult, Blood test*), D2 (*MRI*)
Rooms: R2 (*Consult, Blood test*), R1 (*MRI*)

**Resource Division**

Iteratively assign a doctor or a rooms to a subgroup:

1. Assign to the subgroup with the lowest care coverage for its specialty/capability.
2. If equal, assign to the subgroup with the lowest total coverage.

(See Table 5.1 for an example.)

**Patient Division**

For each patient:

1. Identify suitable subgroups based on specialties/capabilities.
2. Assign the patient to the subgroup with the lowest workload.

If no suitable subgroup exists, assign the patient to a special group. This group has all doctors and all rooms assigned to it. (See Table 5.2.)

**Solving**

First, solve the special group. Its solution is then used across all other subgroups, which are solved independently.

Figure 5.1: High-level overview of the Vertical Batches Algorithm.

In the initialization step, doctors and rooms are ordered according to the number of their specialties or capabilities from high to low. Starting with the highest, each doctor or room is assigned to a specific subgroup. For example, Doctor D1 has two specialties (*Consult* and *Blood test*), while Doctor D2 has one (*MRI*). Therefore, D1 is assigned first, followed by D2. A doctor (or room) is assigned to the subgroup with the lowest care coverage.

In Table 5.1, Doctor D2 is assigned while Doctor D1 is already in Subgroup 1. First rule 1 is checked from Figure 5.1 in the resource division block, ”Assign to the subgroup with the lowest care coverage for its specialty/capability”. D2's specialty (*MRI*) is not yet covered in both subgroups (0). Therefore the algorithm checks rule 2, ”If equal, assign to the subgroup with the lowest total coverage”. In this case, subgroup 2 has lower total care coverage (0) then subgroup 1 (2). Therefore D2 is assigned to subgroup 2. The same procedure is applied to assign rooms to subgroups.

**First check rule 1**

| **Subgroup** | 1 | 2 |
|---|---|---|
| **Doctors (D)** | | |
| Consult | 1 | 0 |
| Blood test | 1 | 0 |
| **MRI** | **0** | **0** |
| **Sum** | 2 | 0 |

**No result, check rule 2**

| **Subgroup** | 1 | 2 |
|---|---|---|
| **Doctors (D)** | | |
| Consult | 1 | 0 |
| Blood test | 1 | 0 |
| MRI | 0 | 0 |
| **Sum** | **2** | **0** |

**D2 is assigned to subgroup 2**

| **Subgroup** | 1 | 2 |
|---|---|---|
| **Doctors (D)** | | |
| Consult | 1 | 0 |
| Blood test | 1 | 0 |
| MRI | 0 | 1 |
| **Sum** | 2 | 1 |

Table 5.1: Example assignment D2

The patient division works mainly on the subgroup workload, patients are assigned to the subgroup which has the lowest workload. There is no specific order of assigning patients to subgroups. From the example, patient P3 has a workload of 3.

$$C_3 = [\text{Blood test, MRI, Consult}]$$

Only the workload of subgroups are compared that have the specialities and capabilities that are in the list of care of the patient. The patient division procedure is visualized in Table 5.2

In this example, there is no case where a patient could be assigned to multiple subgroups. If a patient has multiple suitable subgroups—meaning all subgroups that have the required specialties and capabilities— the patient is assigned to the subgroup with the lowest workload. For example, if both subgroups 1 and 2 would be suitable for patient P2 in iteration 2, the algorithm assigns P2 to subgroup 2 because it has the lowest total workload.

| **Patient Care Sequence** | **Sub 1** | **Sub 2** | **None/Special** | **Notes / Workload Update** |
|---|---|---|---|---|
| **Iteration 1** | | | | |
| $C_1$ [Consult] | Assigned | – | – | Only suitable subgroup, workload to 1 |
| $C_2$ [Blood test, Consult] | – | – | – | Not yet assigned |
| $C_3$ [Blood test, MRI, Consult] | – | – | – | Not yet assigned |
| **Iteration 2** | | | | |
| $C_1$ [Consult] | Assigned | – | – | Already assigned |
| $C_2$ [Blood test, Consult] | Assigned | – | – | Only suitable subgroup, workload updated to 3 |
| $C_3$ [Blood test, MRI, Consult] | – | – | – | Not yet assigned |
| **Iteration 3** | | | | |
| $C_1$ [Consult] | Assigned | – | – | Already assigned |
| $C_2$ [Blood test, Consult] | Assigned | – | – | Already assigned |
| $C_3$ [Blood test, MRI, Consult] | – | – | Assigned | No suitable subgroup; assign to special group |

Table 5.2: Patient division procedure

The last step in the algorithm is scheduling the appointments for each subgroup. This is done by first solving the special group, which is solved using all doctors and rooms.

| **Doctor–Room (Special Group)** | **Slot 1** | **Slot 2** | **Slot 3** | **Slot 4** | **Slot 5** |
|---|---|---|---|---|---|
| D1–R2 | Blood test (P3) | – | – | – | Consult (P3) |
| D2–R1 | – | MRI (P3) | MRI (P3) | – | – |

Table 5.3: Resulting schedule for the special group.

Solving the special group follows the same procedure as the optimal algorithm but just for a small set of patients, namely the patients that couldn't fit in one subgroup. The solution of this special group is used in all remaining subgroups that are solved independently to ensure that the model accounts for doctor and room unavailability during specific time slots.

| **Doctor–Room (Subgroup 1)** | **Slot 1** | **Slot 2** | **Slot 3** | **Slot 4** | **Slot 5** |
|---|---|---|---|---|---|
| D1–R2 | Blood test (P3) | Blood test (P2) | Consult (P2) | Blood test (P1) | Consult (P3) |

Table 5.4: Resulting schedule for Subgroup 1.

| Doctor–Room (Subgroup 2) | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| D2–R1 | – | MRI (P3) | MRI (P3) | – | – |

Table 5.5: Resulting schedule for Subgroup 2.

The final schedule combines all sub-schedules. See Table 5.6 for the final schedule.

| Doctor–Room (Optimal) | Slot 1 | Slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| D1–R2 | Blood test (P3) | Blood test (P2) | Consult (P2) | Blood test (P1) | Consult (P3) |
| D2–R1 | – | MRI (P3) | MRI (P3) | – | – |

Table 5.6: Final optimal schedule combining all subgroups.

### 5.1.3 The Horizontal Batches algorithm

The horizontal batches algorithm solves the scheduling problem by dividing the time horizon into smaller parts. For instance, if the problem requires scheduling over two weeks, the Horizontal Batches Algorithm solves each day separately and then combines the results into a full schedule. Figure 5.2 provides a visual representation of this process.

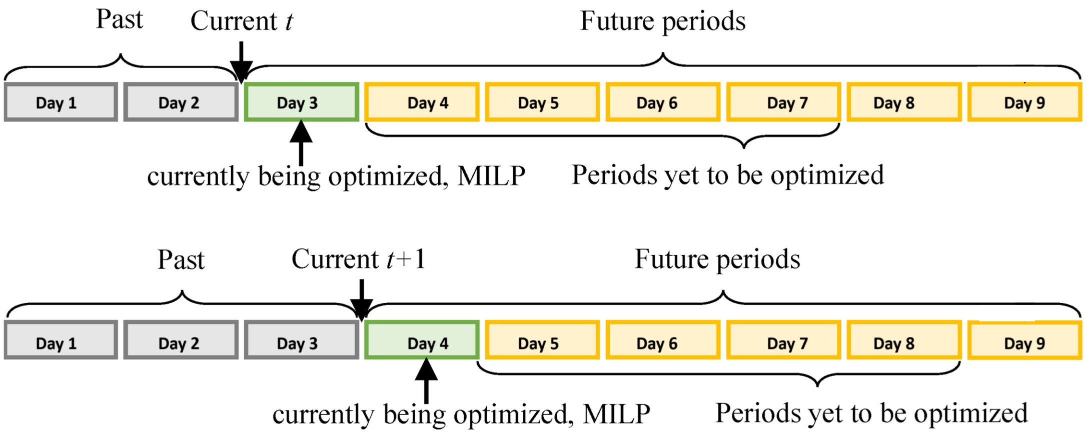


Figure 5.2: Visual representation of the Horizontal Batches Algorithm

In this algorithm, set $T$ is divided into the subsets $T_1, T_2, \ldots,$ where each subset represents a fraction of the total time horizon. Each fraction is solved sequentially, starting with the first time fraction $T_1$, then the second fraction $T_2$ until all patients are scheduled. In each iteration, all doctors and rooms are used in the model. Only uncompletely scheduled patients are incorporated into the model. The algorithm stops when all patients are completely scheduled.

In each iteration (time step) the algorithm attempts to schedule one additional care task from a patient's care sequence into the current time interval. The order in which patients are selected is as follows:

1. Uncomplete scheduled patients (patients of which not all care is scheduled)
2. Patients with the longest time span

- If scheduling the task is **infeasible** (for example due to timing conflicts or resource limits), the algorithm remembers the care *type* of that last attempted task and then:
  (a) excludes the current task from further consideration in this iteration, and
  (b) ignores any new patients that require the same care type for the remainder of the current scheduling process.
- If scheduling is **feasible**, the algorithm keeps the successfully added care task and allows that care type to be considered again in subsequent iterations (note: the partial solution is not reused only the care task information is retained).

The algorithm repeats these steps until either all patients' care sequences have been considered or every care type has been found infeasible for the current scheduling horizon. The final schedule comprises all care tasks that were successfully scheduled.

The same example will be used to explain the process. Only $C_3$ is changed to $C_3$=[Consult, MRI, Blood test] to better show the mechanism of this algorithm. For time division, the time will be divided into half-days (morning and afternoon). The algorithm starts solving the first time fraction, which in this case is slots 1, 2 and 3.
Patients time span is the sum of the care durations and care recovery times of the patients care sequence. The time span of patient 3 in the example is:

$$\text{P3} = 1 \text{ (Consult)} + 2 \text{ (MRI)} + 1 \text{ (MRI, recovery)} + 1 \text{ (Blood test)} = 5$$

The patient selection at the beginning of the example will be based on this time span, so the order will be:

1. P3 (5), 2. P2 (2), 3. P1 (1)

During the scheduling process of a certain time fraction every care task of the care sequences of these patients is checked, see Table 5.8. As already explained, every scheduling iteration there are three possibilities:

1. Feasible, reuse task
2. Infeasible, store task as infeasible and do not use this task in further scheduling iterations
3. Infeasible task, a task that already has been found infeasible. Continue to the next task.

| **Scheduling iterations** | **Solution** | **Infeasible tasks** | **Feasible tasks** |
|---|---|---|---|
| 1. P3 [Consult] | Feasible | – | P3 [Consult] |
| 2. P3 [Consult, MRI] | Feasible | – | P3 [Consult, MRI] |
| 3. P3 [Consult, MRI, Blood test] | Infeasible | [Blood test] | - |
| 4. P3 [Consult, MRI], P2 [Blood test] | Infeasible task | [Blood test] | P3 [Consult, MRI], P2 [Blood test] |
| 5. P3 [Consult, MRI], P2 [Consult] | Feasible | [Blood test] | P3 [Consult, MRI], P2 [Consult] |
| 6. P3 [Consult, MRI], P2 [Consult], P1 [Consult] | Feasible | [Blood test] | P3 [Consult, MRI], P2 [Consult], P1 [Consult] |

Table 5.7: Scheduling process of $T_1$

The result of the scheduling process in $T_1$ is Table 5.8, in $T_2$ the same scheduling process is applied and then all patients are completely scheduled. See Table 5.9 for the result of $T_2$. Together this result in a final schedule in Table 5.10.

| **Doctor–Room (Optimal)** | **Slot 1** | **Slot 2** | **Slot 3** |
|---|---|---|---|
| D1–R2 | Consult (P3) | Consult (P2) | Consult (P1) |
| D2–R1 | – | MRI (P3) | MRI (P3) |

Table 5.8: Solution schedule first time fraction Horizontal Batch Algorithm

| **Doctor–Room (Optimal)** | **Slot 4** | **Slot 5** | **Slot 6** |
|---|---|---|---|
| D1–R2 | Blood test (P2) | Blood test (P3) | - |
| D2–R1 | - | - | - |

Table 5.9: Solution schedule second time fraction Horizontal Batch Algorithm

| **Doctor–Room (Optimal)** | **Slot 1** | **Slot 2** | **Slot 3** | **Slot 4** | **Slot 5** |
|---|---|---|---|---|---|
| D1–R2 | Consult (P3) | Consult (P2) | Consult (P1) | Blood test (P2) | Blood test (P3) |
| D2–R1 | – | MRI (P3) | MRI (P3) | – | – |

Table 5.10: Final optimal schedule combining all subgroups.

### 5.1.4 The Horizontal and Vertical Batch algorithm

The horizontal and vertical batch algorithm combines the *Vertical Batch Algorithm* with the *Horizontal Batch Algorithm.* An algorithm that has not been found in literature. In the algorithm each subgroup that is solved as in *Vertical Batch Algorithm* but the time horizon is split up in time fractions and each time fraction is solved sequentially as in *Horizontal Batch Algorithms.* All sub schedules together result in the final schedule.

### 5.1.5 The Fix-And-Optimize algorithm

The Fix-and-Optimize Algorithm (F&O) approach is a popular method for solving complex problems. It has been successfully applied in several studies, such as Chouksey et al. (2024) and Wickert et al. (2020), among many others, where it demonstrated performance close to optimality with relatively low computational effort. The F&O approach always consists of two phases: a feasibility phase (fix), in which a feasible solution is constructed, and an optimization phase, in which this feasible solution is iteratively improved until no further progress can be made.

The feasible solution is obtained by sequentially scheduling patients one by one. For each patient, the solution is stored and then passed again to the solver together with the next patient that needs to be scheduled. In this way, a feasible solution is quickly found. See Figure 5.3 for this procedure.

**P1 (Consult) →**

| Doctor–Room | Slot 1 |
|---|---|
| D1–R1 | |
| D1–R2 | P1 (Consult) |
| D2–R1 | |
| D2–R2 | |

**P2 (Blood test, Consult) →**

| Doctor–Room | Slot 1 | Slot 2 | Slot 3 |
|---|---|---|---|
| D1–R1 | | | |
| D1–R2 | P1 (Consult) | P2 (Blood test) | P2 (Consult) |
| D2–R1 | | | |
| D2–R2 | | | |

**P3 (Blood test, MRI, Consult) →**

| Doctor–Room | Slot 1 | ... | Slot 4 | Slot 5 | Slot 6 | slot 7 | slot 8 |
|---|---|---|---|---|---|---|---|
| D1–R1 | | ... | | | | | |
| D1–R2 | P1 (Consult) | ... | P3 (Blood test) | | | | P3 (Consult) |
| D2–R1 | | ... | | P3 (MRI) | P3 (MRI) | | |
| D2–R2 | | ... | | | | | |

Figure 5.3: Example of the feasible phase of F&O Algorithm

After the feasible phase, the optimization phase begins. In this phase, a subset of patients is selected for rescheduling. The subsets are created based on the patients time spans and the parameter $g$, which determines the number of patients included in each subset.

The selection process follows a structured order: first the patient with the highest time span is chosen, then the patient with the lowest time span, followed by the second highest, the second lowest, the third highest, and so on. Below is an example based on $g = 2$.

**Step 1:** Ordered list of patients: [P3(5), P2(2), P1(1)]

**Step 2:** Subsets: [{P3, P1}, {P2}]

In each iteration, every subset is used to reschedule the patients. This is done by remove these patients from the current solution such that these slots are free. For each attempt, the current subset is extended by adding a group of patients who were scheduled last, giving the model the opportunity to adjust the makespan. The number of last-scheduled patients to be added is determined by the parameter $l$. The optimization procedure is illustrated in Figure 5.4. In this example, $l = 1$.

rescheduled patients:
{P3,P1}

| Doctor–Room | Slot 1 | slot 2 | Slot 3 | Slot 4 | Slot 5 | Slot 6 | slot 7 | slot 8 |
|---|---|---|---|---|---|---|---|---|
| D1–R1 | | | | | | | | |
| D1–R2 | P1 (Consult) | P2 (Blood test) | P2 (Consult) | P3 (Blood test) | | | | P3 (Consult) |
| D2–R1 | | | | | P3 (MRI) | P3 (MRI) | | |
| D2–R2 | | | | | | | | |

result rescheduling:
(Iteration 1)

| Doctor–Room | Slot 1 | slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| D1–R1 | | | | | |
| D1–R2 | P3 (Blood test) | P2 (Blood test) | P2 (Consult) | P1 (Consult) | P3 (Consult) |
| D2–R1 | | P3 (MRI) | P3 (MRI) | | |
| D2–R2 | | | | | |

rescheduled patients:
{P2,P3}

| Doctor–Room | Slot 1 | slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| D1–R1 | | | | | |
| D1–R2 | P3 (Blood test) | P2 (Blood test) | P2 (Consult) | P1 (Consult) | P3 (Consult) |
| D2–R1 | | P3 (MRI) | P3 (MRI) | | |
| D2–R2 | | | | | |

result rescheduling:
(Iteration 1)

| Doctor–Room | Slot 1 | slot 2 | Slot 3 | Slot 4 | Slot 5 |
|---|---|---|---|---|---|
| D1–R1 | | | | | |
| D1–R2 | P3 (Blood test) | P2 (Blood test) | P2 (Consult) | P1 (Consult) | P3 (Consult) |
| D2–R1 | | P3 (MRI) | P3 (MRI) | | |
| D2–R2 | | | | | |

Figure 5.4: Iteration 1, optimization phase of F&O Algorithm

If an improvement is found in an iteration, the algorithm proceeds to the next iteration, where both $g$ and $l$ are increased by 1. This provides the algorithm with greater flexibility to make local changes. However, it also increases the computational time. If a patient belongs to the selected subset and is also among the last $l$ scheduled patients, then the number of rescheduled patients will be smaller than $g + l$.

Rescheduling may sometimes lead to a solution that is worse than the current best. To handle this, a Simulated Annealing–inspired procedure is used, which allows worse solutions to be accepted with a probability that decreases as the number of iterations grows. This probability also depends on the absolute difference between the new solution and the current best: the larger the difference, the lower the probability of acceptance.

The algorithm terminates when no improvement is achieved in when a;l subsets are used for rescheduling, returning the best solution found.

## 5.2 Vertical Batch algorithm

As described in Section 5.1, the Vertical Batch Algorithm begins by partitioning the resources (doctors and rooms) using the procedure outlined in Algorithm 3. The procedure follows three main steps:

- **Initialization**: Define the number of subgroups. Each subgroup is initialized with empty sets of doctors and rooms. The *care coverage* is defined as the number of times a care type is covered by doctors or rooms within the subgroup.
- **Score calculation**: Each unassigned doctor or room is assigned to the subgroup with the highest score. The scoring is done separately for doctors and rooms. The scoring function for a subgroup $g$ is defined as $s^d$ for doctors and $s^r$ for rooms:

  $$s^d(g) = \sum_{d \in g} \sum_{c \in W_d} \frac{1}{1 + f(g,c)} \ , \ s^r(g) = \sum_{r \in g} \sum_{c \in F_r} \frac{1}{1 + f(g,c)} \tag{5.1}$$

  $$f(g,c) = \sum_{p \in P} \mathbf{1}\{\, v(p) = g \ \wedge \ c \in C_p \,\} \tag{5.2}$$

  - $v(p)$ is a function that returns a subgroup number for each patient $p$ that is assigned to a specific subgroup $g$. Returns 0 if a patient is not yet assigned.

  It sums over the care types in a doctor's specialties or a room's capabilities. A high score indicates a low care coverage of the subgroup of the care types which can be provided by the current unassigned doctor or room. The choice of scoring function is motivated by several desirable mathematical properties of the part within the sum. First, the function is strictly decreasing in $f(g,c)$ for a fixed $g$, ensuring that higher care coverage correspond to lower scores. Second, its hyperbolic shape yields high sensitivity for smaller care coverage while lower sensitivity for larger care coverage. Trying to ensure that each subgroup is able to cover all care, although this requirement becomes less critical when overall coverage is already high.
- **Assignment of doctors and rooms**: Each doctor and room is assigned to the subgroup with the highest score, determined by the scoring function. This ensures a balanced distribution of resources across subgroups. The assignment is done with Algorithm 1. This assignment is done iteratively until all doctors or rooms are assigned to a subgroup.

**Algorithm 2** Assign Doctor/Room (Entity) to the best fitting subgroup

```
function AssignEntity(entity_key, entity_cares, entity_type, subgroups)
    Initialize empty list of scores
    for each subgroup in subgroups do
        score ← $\sum_{c \in entity\_cares} \frac{1}{1+subgroup.care\_coverage[c]}$
        Append (score, subgroup_id) to scores
    end for
    Sort scores in descending order
    chosen_subgroup ← subgroup with highest score
    Add entity_key to chosen_subgroup[entity_type]
    for each care in entity_cares do
        Increment chosen_subgroup.care_coverage[entity_type][care] by 1
    end for
end function
```

**Algorithm 3** Partition Doctors and Rooms Evenly Across subgroups

```
procedure PartitionDoctorsAndRooms(doctor_info, room_info, all_specialities, num_subgroups)
    Initialize subgroups $\{1, .., num\_subgroups\}$ with empty doctors, rooms, and care_coverage

    Distribute Doctors:
    Group doctors by shift
    for Each shift group do
        Sort doctors by number of specialties, descending
        for Each doctor in group do
            AssignEntity(doctor, doctor_specialties, 'doctors', subgroups)
        end for
    end for

    Distribute Rooms:
    Sort rooms by number of capabilities, descending
    for Each room do
        AssignEntity(room, room_capabilities, 'rooms', subgroups)
    end for

    return subgroups
end procedure
```

The Vertical Batch Algorithm partitions the model 4.2.2 in $n$ (the number of subgroups) smaller parts and solves each part independent after the special group is solved. In mathematical terms, the sets $D, R$ are divided in $n$ subsets and $P$ in $n+1$ subsets:

$$P_1 \cup P_2 \cup \cdots \cup P_{n+1} = P, \quad P_i \cap P_j = \emptyset \text{ for } i \neq j$$
$$\text{(patient partitioning)}$$

$$D_1 \cup D_2 \cup \cdots \cup D_n = D, \quad D_i \cap D_j = \emptyset \text{ for } i \neq j$$
$$\text{(doctor partitioning)}$$

$$R_1 \cup R_2 \cup \cdots \cup R_n = R, \quad R_i \cap R_j = \emptyset \text{ for } i \neq j$$
$$\text{(room partitioning)}$$

$$T_i = \{t_1, t_2, ..., t_{H_i - 1}\}$$

The partition of the doctors and rooms follow the structure of Algorithm 3. The patients are partitioned illustrated in Table 5.2. After solving the special group $n+1$, for each subgroup $i$ in $\{1, 2, ..., n\}$ solve the following mathematical problem which has the same structure as **??** but now for each subset separately:

$$\text{Objective:} \qquad \min \; H_i$$

$$\sum_{d\in D_i,\, r\in R_i,\, t\in T_i} a_{d,t} \cdot x_{p,c,d,r,t} = 1, \qquad \forall p \in P_i,\ \forall c \in C_p \tag{13}$$

13) Each patient $p \in P_i$ with care task $c \in C_p$ in the care sequence $C_p$ is assigned to a doctor $d \in D_i$ and a room $r \in R_i$ for a specific time slot $t \in T_i$. This assignment defines the start time of the scheduling for subgroup $i$.

$$s_{p,c} = \sum_{d\in D_i,\, r\in R_i,\, t\in T_i} t \cdot x_{p,c,d,r,t}, \qquad \forall p \in P_i,\ \forall c \in C_p \tag{14}$$

14) The start time $s_{p,c}$ for patient $p \in P_i$ and care $c \in C_p$ equals the time slot $t$ where $x_{p,c,d,r,t} = 1$.

$$y_{p,c,d,r,t'} \geq x_{p,c,d,r,t}, \qquad \begin{array}{l}\forall p \in P_i,\ \forall c \in C_p,\ \forall d \in D_i,\ \forall r \in R_i,\\ \forall t \in T_i,\ \forall t' \in \{t, \dots, t + L_c - 1\}\end{array} \tag{15}$$

15) If a care task $c \in C_p$ is assigned to doctor $d \in D_i$ and room $r \in R_i$ at time $t$, then $y_{p,c,d,r,t'} = 1$ for all subsequent slots $t'$ within the duration $L_c$ of that care.

$$\sum_{p\in P_i,\, c\in C_p,\, r\in R_i} y_{p,c,d,r,t'} \leq 1, \qquad \forall d \in D_i,\ \forall t' \in T_i \tag{16}$$

16) Each doctor $d \in D_i$ can handle at most one patient at any time slot $t' \in T_i$.

$$\sum_{p\in P_i,\, c\in C_p,\, d\in D_i} y_{p,c,d,r,t'} \leq 1, \qquad \forall r \in R_i,\ \forall t' \in T_i \tag{17}$$

17) Each room $r \in R_i$ can host at most one patient at any time slot $t' \in T_i$.

$$s_{p,c_{i+1}} \geq s_{p,c_i} + L_{c_i} + B_{c_i}, \qquad \forall p \in P_i,\ (c_i, c_{i+1}) \in C_p \tag{18}$$

18) For each patient $p \in P_i$ with sequential care tasks, the next care $c_{i+1}$ can only start after the previous care $c_i$ has finished and the required break time $B_{c_i}$ has passed.

$$H_i \geq s_{p,c} + L_c, \qquad \forall p \in P_i,\ \forall c \in C_p \tag{19}$$

19) The makespan $H_i$ for subgroup $i$ must be at least as large as the finish time of all care tasks in $P_i$.

(5.3)

Then eventually the min make span $H$ is decided by the maximum of the make spans of these subgroups.

$$H = max\{H_1, H_2, ..., H_n\}$$

Optimizing $H$ means lowering the make span of the subgroup with the highest makespan. Therefore adjustments are made in order to optimize H. Patients from the subgroup with the largest makespan are reassigned to the subgroup with the smallest if the patient can be covered (at least 1 doctor and room that can help the patient). The maximum number of patients transferred is determined using equation 5.4. In which $sum(L)$ is the sum of all care durations and $len(C)$ the number of care types.

$$\text{Maximum number of moved patients} = \frac{max\{H_1, H_2, ..., H_n\} - min\{H_1, H_2, ..., H_n\}}{sum(L)/len(C)} \tag{5.4}$$

Patients are ordered (ascending) on their time span and patients with a low time span are first selected. Running the Vertical Batch Algorithm again for the new subsets of patients and continue this process until there is no improvement of $H$.

## 5.3 Horizontal Batch algorithm

In this research, the time horizon is divided into half-day and full-day intervals. With half-day intervals, the algorithm solves the scheduling problem per shift (morning and afternoon). Unlike the Vertical Batch Algorithm, where subgroups are solved independently, here they are solved sequentially: start with solving the first time interval (subgroup), and then continue to the next time interval where the solution of the previous time interval influences the current time interval.

The patient set is divided into three subsets: unscheduled patients $P^u$, scheduled patients $P^s$, and prioritized patients $P^p$ (patients whose care has started but whose full care sequence is not completely scheduled). At the start of the Horizontal Batch Algorithm, the subsets are initialized as follows:

$$P^s = \{\}, \quad P^p = \{\}, \quad P^u = P$$

The following relation always holds:

$$P^s \cup P^u \cup P^p = P$$

Solving the scheduling problem for each interval requires the following steps:

- **Initialization**: Select the doctors available in the current interval, as well as the rooms which can be used by the corresponding doctors. Patients are ordered in descending order of their time span. If a patient is completely scheduled it is assigned to $P^s$, if it is scheduled but not completely scheduled to $P^p$ and otherwise to $P^u$.

- **Scheduling**: In each iteration, a specific care–patient combination is used in the basic or extended scheduling model with a specific time interval. If the scheduling model returns an optimal solution, the combination is scheduled in the current interval. If it returns an infeasible solution, the care type is saved and skipped for later iterations. See Algorithm 4 for the pseudo-code of the scheduling step. A crucial aspect of this step is the order in which care–patient combinations are considered. The following order is taken into account:

  1. Priority patients with long time spans
  2. Priority patients with short time spans
  3. Unscheduled patients with long time spans
  4. Unscheduled patients with short time spans

  Since partially scheduled patients (priority patients) are considered first, the average waiting time (i.e., the additional time a patient must wait beyond their recovery time) is taken into consideration to reduce it. Additionally, if a patient has a long time span, the current and later intervals may become empty or sparse, since this patient is likely to have care with long recovery time. This unnecessarily increases the makespan. To minimize the overall makespan, it is essential to schedule long-span patients with long recovery times first, and short-span patients last.

- **Update**: Each iteration solves one time interval. After solving the current interval, time values are updated to ensure correct input for the next iteration. Priority patients are those already scheduled but not yet fully completed. When a patient transitions from unscheduled to priority, the ending time of their last scheduled care is remembered and used in the recovery constraint of the next iteration, to make sure the recovery time is correctly taken into account.

As described earlier, the algorithm uses the basic or extended scheduling model with a subset of P which is incrementally filled first with patients from $P^p$ and then with patients from $P^u$. If the model returns a solution $H$ outside the current time interval $T^i$, the solution is considered infeasible. If all care types have returned an infeasible solution once and there are still uncomplete scheduled patients, do the same procedure for interval $T^{i+1}$. If in interval $T^{i+1}$ all patients are completely scheduled, then $H^{i+1}$ determines the final makespan.

**Algorithm 4** Pseudocode Scheduling Step, Horizontal Batch Algorithm

```
Require: $schedule, P^u, P^s, P^p, T'$ (current time interval)
Ensure: $schedule, P^u, P^s, P^p$
 1: $P' = \{\}$
 2: infeasible_care = $\{\}$
 3: for each $p$ in ordered($P^p$) do
 4:     $C'_p = \{\}$
 5:     for each $c$ in $C_p$ do
 6:         if [$p$,$c$] $\notin$ schedule and $c \notin$ infeasible_care then
 7:             $P' = P' \cup \{p\}$ and $C'_p = C'_p \cup c$
 8:             status, objective = Model 4.2.2 or 4.2.3 (with $P = P'$, $C_p = C'_p$ and $T = T'$)
 9:             if status $\in$ {optimal,feasible} then
10:                 [$p$,$c$] $\longrightarrow$ schedule
11:             else
12:                 $P' = P'/\{p\}$
13:                 infeasible_care.add($c$)
14:             end if
15:         end if
16:     end for
17:     if $p$ is completely scheduled then
18:         $P^s = P^s \cup \{p\}$
19:         $P^p = P^p/\{p\}$
20:     end if
21: end for
22: for each $p$ in ordered($P^u$) do
23:     Same procedure as line 3-21
24:     if $p$ is completely scheduled then
25:         $P^s = P^s \cup \{p\}$
26:         $P^u = P^u/\{p\}$
27:     else
28:         $P^p = P^p \cup \{p\}$
29:         $P^u = P^u/\{p\}$
30:     end if
31: end for
32: return $schedule, P^u, P^s, P^p$
```

## 5.4 Horizontal and Vertical Batch algorithm

The Horizontal and Vertical Batch Algorithm combines both the Horizontal Batch Algorithm and the Vertical Batch Algorithm. Technical descriptions of these two algorithms are provided in Sections 5.2 and 5.3. The combined approach first applies the Vertical Batch Algorithm, in which subgroups are formed and solved independently. Each subgroup is then solved using the Horizontal Batch Algorithm, where the time horizon is divided into multiple parts that are solved sequentially. The min make span of this algorithm is:

$$H = \max\{H_1, H_2, \ldots, H_n\} \tag{5.5}$$

$$H_i = \text{Horizontal Batch Algorithm}(P_i, D_i, R_i), \quad i = 1, \ldots, n \tag{5.6}$$

## 5.5 Fix and Optimize algorithm

The Fix-and-Optimize (F&O) Algorithm consists of two phases: a construction phase and an optimization phase. See Algorithm 6 for the pseudocode.

**Construction phase.** The model (Section 4.2.2) is solved iteratively for one patient $p_i \in P$, while reusing the solution of previously scheduled patients $p_1, p_2, \ldots, p_{i-1}$. This process continues until all patients are scheduled (line 1 of Algorithm 6). No ordering is applied in selecting patients.

**Optimization phase.** Once a feasible solution is obtained, the optimization phase begins. Each iteration $i$ (lines 4–24 of Algorithm 6) proceeds as follows:

(1) **Creating subgroups.** Subgroups are formed based on patient time spans (see Algorithm 5). Given a predefined subgroup size, the algorithm pairs patients with complementary time spans—specifically, the patient with the longest time span is grouped with the patient with the shortest one. Patients with longer time spans typically require multiple care types with extended recovery periods, making them more difficult to reschedule than patients with shorter time spans, who usually need only a single care type and no recovery time. Once a subgroup reaches its capacity, a new subgroup is created.

(2) **Reschedule subgroup.** Each subgroup is used for rescheduling, along with the last $x$ patients based on ending time of the current solution. So in rescheduling the solution is reused and only the patients selected for rescheduling are removed from the solution and free to schedule on available spots. So the model (Section 4.2.2) is solved with:

$$P_i = P_k \text{ (patients in subgroup k)} \cup P_l \text{ (last } x \text{ patients scheduled)}$$

If the model returns a feasible or optimal solution, the algorithm checks whether the min make span is lower than the current min make span.

**True:** Update the current make span. If the new make span is also lower than the optimal make span, update the optimal make span as well.

**False:** A Simulated Annealing-inspired approach determines whether a worse make span is accepted. A random number $r \in [0, 1]$ is generated; if $r$ is below a threshold value, the worse make span is accepted and the current make span is updated.

If infeasibility occurs during rescheduling, the threshold for accepting a worse make span is given by:

$$\rho(i, H) = \frac{1}{i} - \frac{e(H)}{40} \tag{5.7}$$

where

$$e(H) = |H - current\_H|$$

At the start (iteration 1), nearly all worse solutions are accepted. In later iterations, only a small percentage are accepted. For $i \in \{1, 2, 3\}$, the threshold upper bounds are $\{1, 0.5, 0.33\}$, respectively. Larger deviations between the cost and the current cost decrease the probability of accepting the worse solution.

If in a certain iteration (iteration 1 or 2) no improvement is found, then the algorithm will stop and returns the best cost and best solution. This best cost is the min make span of the Fix and Optimize Algorithm.

An important parameter of the algorithm is the `group_size`, which determines the number of patients to be rescheduled simultaneously. Setting $group\_size = |P|$ transforms the Fix-and-Optimize algorithm into the Optimal Algorithm, guaranteeing the best solution but at the cost of high computation time. Conversely, setting $group\_size = 2$ yields low computation time but offers little opportunity for improvement. Independent of how the groups are formed. Thus, a trade-off must be made between solution quality and computation time. Experimental results indicate that group sizes between 3 and 6 provide the best balance between optimality and efficiency.

**Algorithm 5** Make Time Span based Subgroups

```
Require: A list of patients, group_size
Ensure: A list of subgroups
Sorted_patients_list ⟵ list of patients ordered by time span in descending order
Initialize $subgroups \leftarrow [\ ]$
while Sorted_patients_list not empty do
    Initialize empty set $group$
    while $|group| < group\_size$ do
        if $|group|$ is even then                ▷ Take patient with highest time span
            $group.add(Sorted\_patients\_list[0])$
        else                                     ▷ Take patient with lowest time span
            $group.add(Sorted\_patients\_list[-1])$
        end if
    end while
    if $|group| \neq 0$ then
        Append $group$ to $subgroups$
    end if
end while
return $subgroups$
```

**Algorithm 6** Fix-and-Optimize Algorithm (corrected)

```
Require: unscheduled_patients
Ensure: best_cost, best_solution
(best_cost, best_solution) ← ReturnFeasibleSchedule(unscheduled_patients)
current_cost ← best_cost
current_solution ← best_solution
for $i$ ← 1 to 3 do
    improvement ← False
    groups ← MakeSubgroups(unscheduled_patients, group_size = $2 + i$)
    for all group ∈ groups do
        last_patient_group ← ReturnLastPatients(current_solution, group_size = $2 + i$)
        (status, cost, solution, deviation) ← SolveSchedule(group ∪ last_patient_group, current_solution)
        if status ∈ {optimal, feasible} then
            if cost < current_cost then
                current_cost ← cost
                current_solution ← solution
                improvement ← True
                if cost < best_cost then
                    best_cost ← cost
                    best_solution ← solution
                end if
            else
                if Random(0, 1) < $\frac{1}{i} - \frac{cost - current\_cost}{40}$ then
                    current_cost ← cost
                    current_solution ← solution
                end if
            end if
        end if
    end for
    if improvement = False then
        break
    end if
end for
return (best_cost, best_solution)
```

# Chapter 6

# Results

This chapter covers the predictions obtained from the prediction model in section 6.1) and the results obtained from the experiments described in section 6.2. The experiments are divided into two main parts: the *baseline model scenario* and the *extended model scenario*. In section 6.3 different performance criterion's are explained. The algorithms are tested on a MD Ryzen 7 7730U with Radeon Graphics machine with 16GB ram, the result are presented in section 6.4. Additionally, different double book strategies are analyzed in section 6.5.

## 6.1 No-show predictions

The best LightGBM model from section 4.1.4 is used. It is trained on 29,108 patients. The model predicted the no-show probability for 5000 patients, which will be used in the scheduling. The model reached an accuracy of 0.69. The predictions can be summarized in Figure 6.1. In this figure the patients are grouped based on their predicted show-up probability. From this group the average show-up rate is calculated and used to compare this against the average predicted show-up probability. In this graph it can be seen that if the model predicts a show-up probability of 0.5 or higher, that this is closely aligns with the actual average show up rate. If the predicted show-up probability is lower than 0.5, this average show-up rate is higher than the predicted show-up rate.

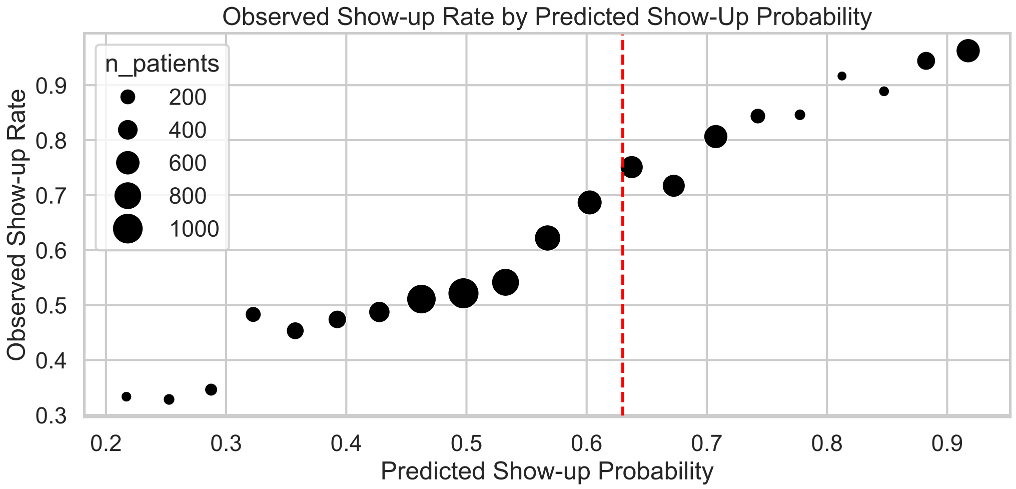


Figure 6.1: Predicted show-up compared to actual show-up

## 6.2 Experimental Setup

### 6.2.1 Baseline model scenario

To evaluate the performance of the different strategies/algorithms, a set of test instances is generated. These instances contain information for the baseline model and not yet the information needed in the extended model. The reason is that these instances contain key elements to compare the strengths and weaknesses of each strategy/algorithm within a reasonable computation time. Based on the results obtained from these instances, the best-performing strategy or algorithm will be tested on the extended model to assess its effectiveness under realistic conditions.

The instances are divided into three groups: A,B en C. Each group corresponds to a specific combination of doctors, rooms, and care types, see Table 6.1. A group can be interpreted as a hospital scenario with its own specific personnel working their shifts, with their specialties and with their rooms.

| **Group** | **#Doctors** | **#Rooms** | **#Care Types** |
|---|---|---|---|
| A | 4 | 3 | 3 |
| B | 8 | 6 | 6 |
| C | 12 | 8 | 8 |

Table 6.1: Overview of test instance groups

Each group is tested with different number of patients, these number of patients range from 25 until 200 in steps of 25.

$$\text{set of patients} = \{25, 50, 75, 100, 125, 150, 175, 200\}$$

In all groups, a day consists of 24 slots of 20 minutes each, representing an 8-hour working day. The first 4 hours (12 slots) are considered as morning slots, and the remaining 4 hours (12 slots) are the afternoon slots. A full day consist of the morning and afternoon slots. In the testing phase, there are no constraints regarding free working days; thus, doctors are assumed to work every day.

A schedule is considered complete when all patients have been assigned appointments. The primary objective is to minimize the number of days required to schedule all patients.

From the model point of view, in each group (A,B,C) the following sets are fixed: care types ($C$), care durations ($L$), recovery times ($B$), shifts ($S$), doctors ($D$) and rooms ($R$). Only the set of patients ($P$) is different ($|P| \in$\{25,50,...,200\}).

| **ID** | **Shift** | **Specialties** |
|---|---|---|
| D0 | Morning | C1, C2 |
| D1 | Afternoon | C1, C2 |
| D2 | Morning | C2, C3 |
| D3 | Afternoon | C1, C2, C3 |

Table 6.2: Doctors Group A

| **ID** | **Capabilities** |
|---|---|
| R0 | C1, C2, C3 |
| R1 | C2, C3 |
| R2 | C1, C2 |

Table 6.3: Rooms group A

| **Name** | **Duration (Slots)** | **Recovery (Slots)** |
|---|---|---|
| C1 | 1 | 4 |
| C2 | 2 | 2 |
| C3 | 2 | 6 |

Table 6.4: Care types group A

Between the groups (A,B,C) these previously mentioned fixed sets can be different. Different doctors with different specialties, different care types and so on. See Tables 6.2, 6.3 and 6.4 for detailed information about the resources and care information in group A. R0 in group A has different capabilities than R0 in group B, see Table 6.6. For the detailed doctor and care information for group B see Tables 6.5 and 6.7. For detailed information of group C, see Tables 6.8, 6.9 and 6.10

| **ID** | **Shift** | **Specialties** |
|---|---|---|
| D0 | Morning | C1, C2, C3, C4, C5 |
| D1 | Afternoon | C1, C2, C3, C4, C5, C6 |
| D2 | Morning | C2, C3, C4, C6 |
| D3 | Afternoon | C1, C2, C3, C4, C5, C6 |
| D4 | Morning | C1, C2, C3, C4, C5, C6 |
| D5 | Afternoon | C1, C2, C3, C4, C5, C6 |
| D6 | Full Day | C3, C4, C5 |
| D7 | Full Day | C1, C2, C4, C5, C6 |

Table 6.5: Doctors Group B

| **ID** | **Capabilities** |
|---|---|
| R0 | C1, C6 |
| R1 | C1, C2, C3, C4, C5, C6 |
| R2 | C1, C2, C3, C4, C5, C6 |
| R3 | C1, C4 |
| R4 | C2, C3, C4, C5, C6 |
| R5 | C4, C6 |

Table 6.6: Rooms group B

| **Name** | **Duration (Slots)** | **Recovery (Slots)** |
|---|---|---|
| C1 | 1 | 4 |
| C2 | 2 | 2 |
| C3 | 2 | 6 |
| C4 | 2 | 3 |
| C5 | 3 | 2 |
| C6 | 4 | 6 |

Table 6.7: Care types group B

| ID | Shift | Specialties |
|---|---|---|
| D0 | Morning | C7, C8 |
| D1 | Afternoon | C1, C3, C8 |
| D2 | Morning | C1, C2, C3, C5, C6, C7, C8 |
| D3 | Afternoon | C1, C2, C3, C4, C5 |
| D4 | Morning | C1, C5, C6, C7, C8 |
| D5 | Afternoon | C1, C2, C3, C4, C5, C7, C8 |
| D6 | Full Day | C2, C4, C8 |
| D7 | Full Day | C1, C3, C4, C6 |
| D8 | Morning | C1, C3, C4, C5, C6, C8 |
| D9 | Afternoon | C1, C2, C3, C5, C7, C8 |
| D10 | Full Day | C1, C2, C3, C4, C6, C7, C8 |
| D11 | Full Day | C2, C5 |

Table 6.8: Doctors group C

| ID | Capabilities |
|---|---|
| R0 | C1, C2, C3, C4, C5, C6, C7, C8 |
| R1 | C1, C2, C3, C4, C5, C7 |
| R2 | C1, C3, C4, C5, C6, C8 |
| R3 | C1, C2, C3, C4, C5, C6 |
| R4 | C1, C2, C4, C7 |
| R5 | C5, C8 |
| R6 | C1, C2 |
| R7 | C5 |

Table 6.9: Rooms group C

| Name | Duration (Slots) | Recovery (Slots) |
|---|---|---|
| C1 | 1 | 0 |
| C2 | 2 | 0 |
| C3 | 2 | 6 |
| C4 | 2 | 4 |
| C5 | 2 | 4 |
| C6 | 4 | 8 |
| C7 | 1 | 24 |
| C8 | 2 | 24 |

Table 6.10: Care types group C

### 6.2.2 Extended model scenario

Based on the results of the baseline model scenario, the two best performing algorithms will be tested on the extended model scenario. The extended model is inspired by the VU medical centre. The provided objective is to create an optimal schedule for 6 weeks. See Table 6.11 for all care types with their duration and recovery times. For the full detailed doctor and room information, see Tables A.1 and A.2. Summarized, these are the different types of doctors:

**Functions**: Clinician, Doctor Assistant, Echografist, CR-Thorax Specialist, Spirometrie specialist and Student

| **Care Types** | **Duration (Minutes)** | **Recovery** |
|---|---|---|
| C0: B.P.C. | 30 | - |
| C1: CPAP | 60 | - |
| C2: CR-Thorax | 20 | 1 Hour |
| C3: CT Scan | 30 | 2 Weeks |
| C4: Doctor Appointment (person) | 20 | - |
| C5: Doctor Appointment (virtual) | 15 | - |
| C6: Dynamap | 30 | - |
| C7: ECG (Blood Pressure) | 20 | - |
| C8: EKG | 10 | - |
| C9: Holter | 15 | 2 Weeks |
| C10: MRI | 30 | 4 Weeks |
| C11: Pacemaker | 20 | - |
| C12: PM/ICD | 20 | - |
| C13: Spirometrie | 20 | 1 Hour |
| C14: Syncope | 90 | - |
| C15: TTE | 45 | 1 Hour |
| C16: XECG (Fysical Test) | 30 | - |

Table 6.11: Care types VUmc

The following information is incorporated into the model:

- The working day is from 09:00 to 18:00 with 30 minutes break (13:00-13:30), this is considered a full day shift. Morning shifts last from 09:00 to 13:00, and afternoon shifts from 13:00 to 18:00 (or until 17:00, depending on the function). During the final hour (17:00–18:00), only digital doctor appointments may be scheduled.
- On a half day, a doctor may see up to 2 new patients and a total of 15 appointments. On a full day, the limits are 4 new patients and 25 appointments.
- At most 5 Holter and 5 XECG appointments are scheduled per day. For pacemaker appointments, the daily maximum are 10 in the morning and 7 in the afternoon.

The number of patients used to test the two best performing algorithms are:

$$\text{patients sizes} = \{250,500,750,1000\}$$

## 6.3 Evaluation Metrics

The primary key performance indicator (KPI) is:

- **Makespan** — the number of days required to schedule a given number of patients. When the makespan is fixed at six weeks, this KPI instead represents the number of patients that can be fully scheduled within that time horizon.

The makespan serves as the primary objective of the baseline model and is consistently used across all algorithms. Although this is the main KPI, two additional performance measures are also important: the time required to obtain a solution (*solving time*) and the waiting time experienced by patients.

**Solving time** refers to the computational time required for an algorithm or solver to produce a feasible or optimal schedule. Short solving times are crucial for practical applications, especially in settings where schedules must be updated frequently, such as during rescheduling or when new patient information becomes available.

**Patient waiting time** is defined as the delay between the completion of recovery and the start of follow-up care. Minimizing waiting time is essential for maintaining quality of care, ensuring timely follow-up, and reducing delays in the patient treatment pathway.

Based on these considerations, two secondary KPIs are defined:

- **Solving Time (CPU Time)**
- **Patient Waiting Time**

Patient waiting time was not included as a secondary objective in the baseline model to preserve its simplicity. Adding additional variables, constraints, or objectives increases the model's complexity and, consequently, the solving time. Since the model already struggles to find optimal solutions for small instances using only makespan as the objective, we chose to retain this single objective.

A lower bound is computed which is the minimal make span based on the capacity ignoring all the constraints. A scheduling solution has a make span that is higher or equal to the optimal make span which is higher or equal to this lower bound. Since the optimal make span is not always known this lower bound is used to have an indication how close the solution is to the optimal solution.

**Lower bound**
To determine the lower bound, we first calculate the total number of resource-slots required:

$$\#\text{ of required slots} = P1 \rightarrow 1 + P2 \rightarrow 2 + P3 \rightarrow 3 = 6.$$

Each time slot can accommodate a limited number of patients, determined by the most restrictive resource:

$$\min\{2\text{ doctors, }2\text{ rooms}\} = 2\text{ patients per slot.}$$

We now subtract this capacity from the remaining demand slot by slot:

$$\begin{aligned} \text{After slot 1:} &\quad 6-2=4, \\ \text{After slot 2:} &\quad 4-2=2, \\ \text{After slot 3:} &\quad 2-2=0. \end{aligned}$$

Once the remaining demand reaches zero, we have found the minimum number of required slots.

$$\boxed{\text{Lower bound} = 3 \text{ slots}}$$

## 6.4 Algorithm Performance

**Baseline model scenario**

The following algorithms from the methodology are evaluated: the *Vertical Batch Algorithm*, the *Horizontal Batch Algorithm*, the *Horizontal and Vertical Batch Algorithm*, the *Optimal Algorithm*, and the *Fix-and-Optimize Algorithm.*

The Optimal Algorithm is only tested on Group A, since it requires a lot of computation time and memory for larger instances. Even for the small instances of Group A, optimality cannot always be reached. Therefore, a one-hour time limit is added for every 25 patients scheduled. If the time limit is reached, the best solution returned by the Optimal Algorithm is considered “optimal”.
For each group (A,B,C), the metrics defined in section 6.3 are displayed. Additionally, 2 plots based on the loading time and the percentage of optimal solution is displayed.

- **Loading Time**: The time it takes to load the model and find appropriate subgroups instead of time spend to find a solution.
- **Fraction of optimal solutions**: The model returns optimal or feasible solutions. Feasible solution indicates that improvement is possible and is returned if the time limit is reached.

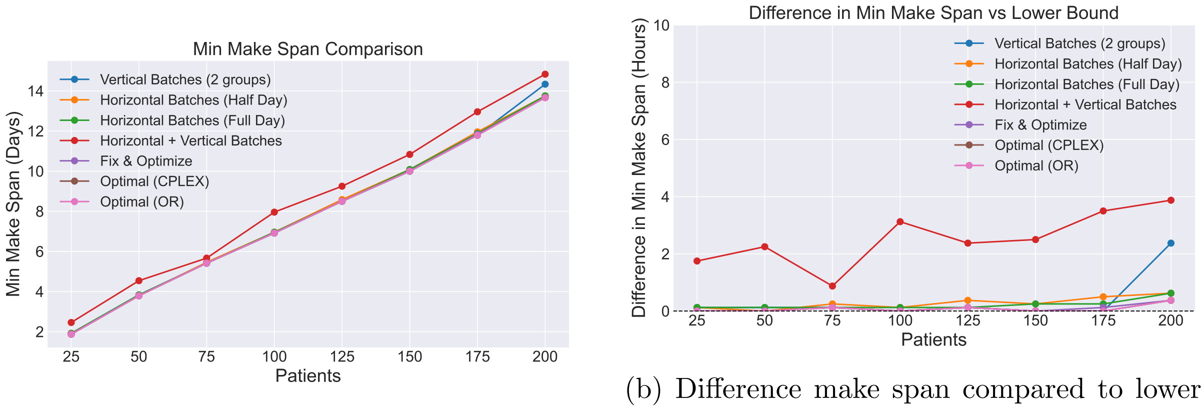
In total there are 8 different models used from the 5 algorithms which are mentioned. Namely:

**Vertical Batch Algorithm**: a model with 2 subgroups and a model with 3 subgroups

**Horizontal Batch Algorithm**: a model with full day intervals and a model with half day intervals

**Optimal Algorithm**:the complete model solved with CPLEX solver and Google OR solver. All models are solved with Google OR solver. Only the optimal algorithm is once solved with CPLEX solver to test the quality of an expensive commercial solver compared to a free open source solver.

### 6.4.1 Baseline model scenario: group A

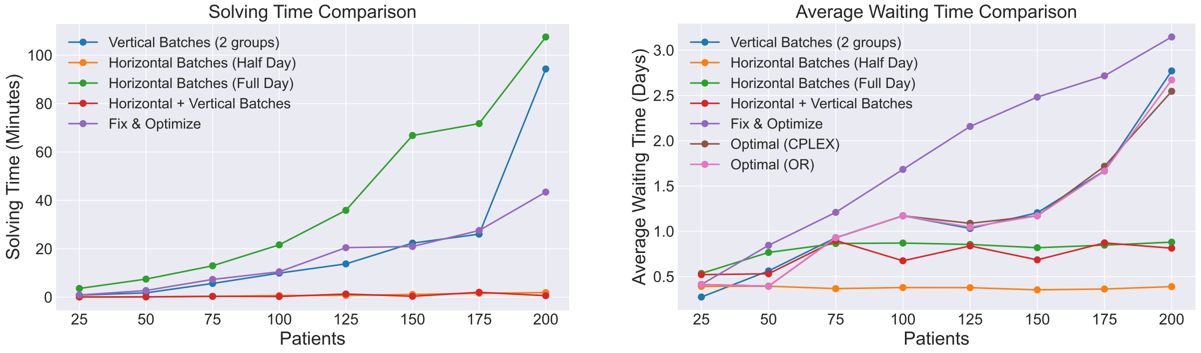


(a) Min make span

(b) Difference make span compared to lower bound

Figure 6.2: Group A, comparison make span

The minimum makespans of all strategies are closely aligned. As shown in Figure 6.2a, the *Horizontal and Vertical Batches* algorithm consistently returns the highest makespan across all patient sets. Figure 6.2b illustrates deviations relative to the lower bound: all strategies except the *Horizontal and Vertical Batches* and the *Vertical Batch with 2 groups* remain within 1 hour. With 200 patients, the makespan is about 14 days, resulting in differences of less than 0.5% with the best and worst solution.

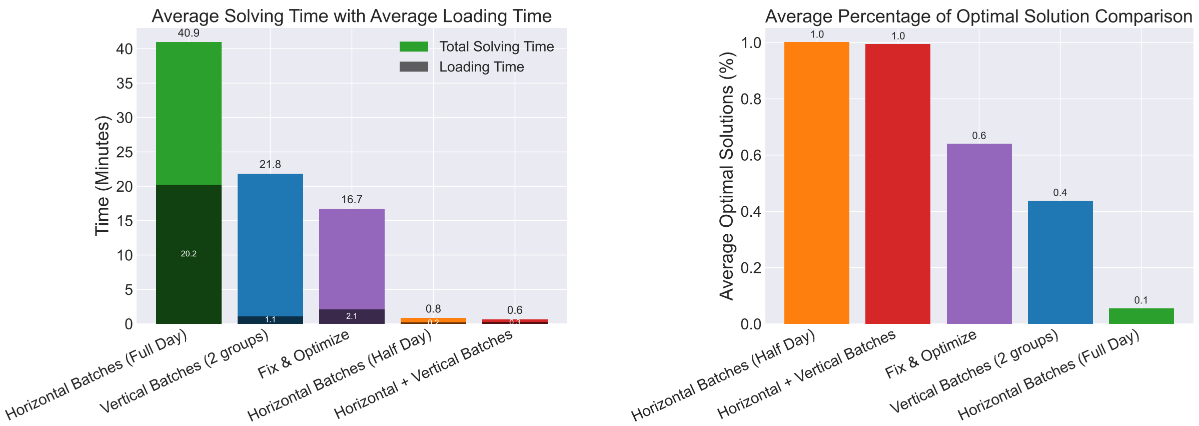


(a) Solving Time

(b) Average Waiting Time

Figure 6.3: Group A, Performance Metrics 1

Figure 6.3 summarizes the second part of the algorithmic performance. Solving times (Figure 6.3a) for *Horizontal Batches (Half Day)* and *Horizontal and Vertical Batches* increase much slower with patient count than other algorithms. The gap between the fastest and slowest run is 107 minutes, with only 0.5 minutes for *Horizontal and Vertical Batches* at 200 patients. Figure 6.3b shows the average waiting time, defined as the additional delay (excluding recovery) between consecutive treatments. This grows under *Fix and Optimize* and *Vertical Batches (2 groups)* but remains stable for other methods.



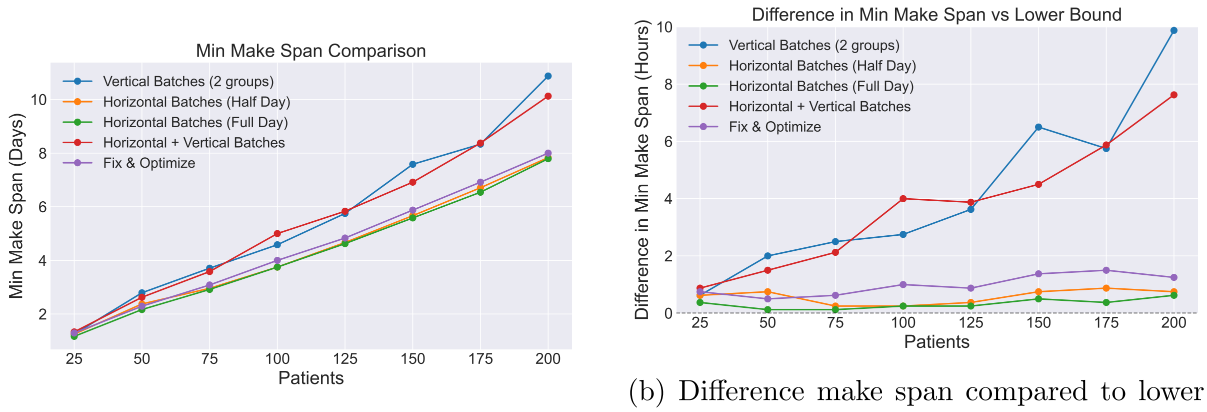
(a) Solving Time and Loading Time

(b) Fraction Optimal Solutions

Figure 6.4: Group A, Performance Metrics 2

Figure 6.4 summarizes the last part of the algorithmic performance averaged over the instances. Figure 6.4a compares solving time and loading time, i.e., data initialization and subgroup identification. The part of the loading time which include subgroup identification is time which could be improved. *Horizontal Batches (Full Day)* has the highest percentage of loading time, consuming a lot of time to find the appriopriate patients fitting in the 1 day time window. Finally, Figure 6.4b displays the fraction of optimal sub solutions. Both *Horizontal Batches (Half Day)* and *Horizontal and Vertical Batches* always returns the optimal solution, while *Horizontal Batches (Full Day)* most often returns feasible solutions, indicating a better solution could be possible.

### 6.4.2 Baseline model scenario: group B

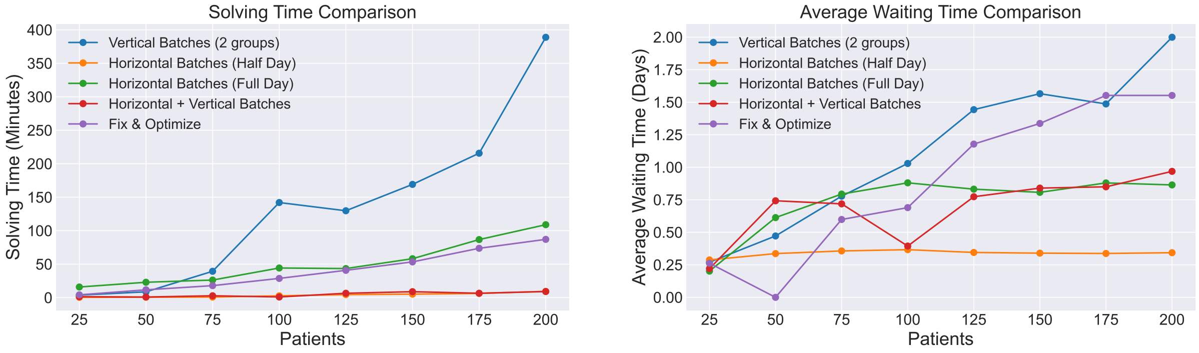


(a) Min Make Span

(b) Difference make span compared to lower bound

Figure 6.5: Group B, comparison make span

The minimum makespans of all strategies are again closely aligned. As shown in Figure 6.5a, only *Horizontal and Vertical Batches* and *Vertical Batches (2 groups)* consistently return the highest makespan across all patient sets. Figure 6.5b illustrates deviations relative to the lower bound. The difference between *Horizontal and Vertical algorithm* and *Vertical Batch algorithm* grow compared to other algorithms, as the number of patients grows.

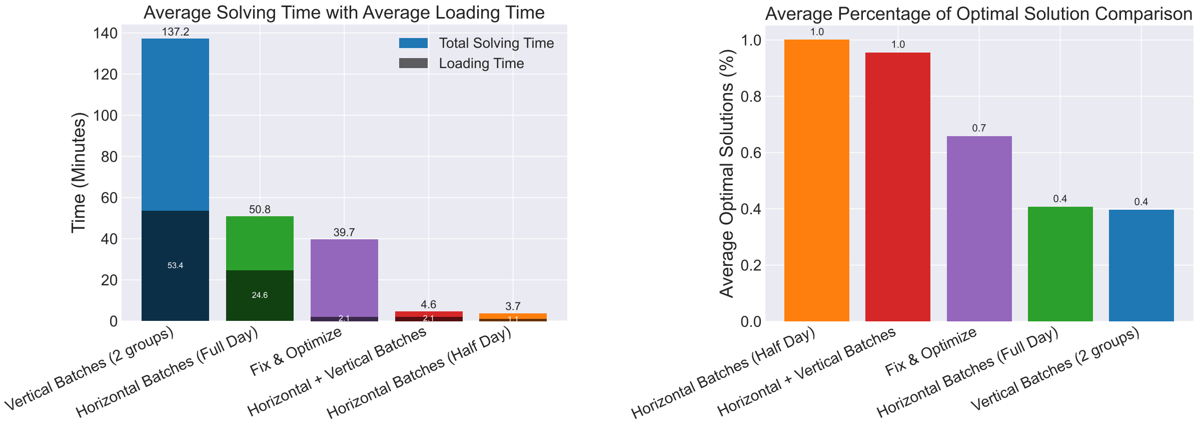


(a) Solving Time

(b) Average Waiting Time

Figure 6.6: Group B, Performance Metrics 1

Figure 6.6 summarizes the second part of the algorithmic performance of group B. The solving times (Figure 6.6a) follow trends similar to group A. Figure 6.6b also reflects trends consistent with group A.



(a) Solving Time and Loading Time

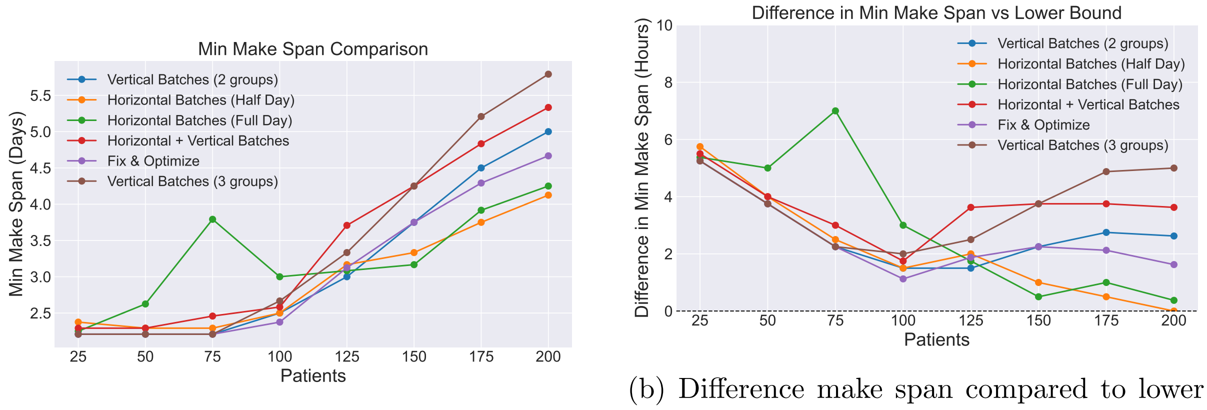
(b) Percentage Optimal Solutions

Figure 6.7: Group B, Performance Metrics 2

Figure 6.7 summarizes the last part of the algorithmic performance of group B. Figure 6.7a compares solving time with loading time, i.e., data initialization and subgroup identification. While *Horizontal Batches (Full Day)* again has a high percentage of loading time, *Vertical Batches (2 groups)* also exhibits a substantial loading time, with its average loading time equal to the average solving time of *Horizontal Batches (Full Day)*.

Finally, Figure 6.7b shows the fraction of optimal sub solutions returned by the model. The patterns mirror those of group A, but in this case *Horizontal Batches (Full Day)* produces a greater number of optimal solutions. Based on the results of group B, the parameters of the *Vertical Batches* algorithm were adjusted. Since the algorithm requires more time to find solutions, the time limit was increased with an additional hour for each set of patients. This prevents the algorithm from prematurely restarting the solving process when no solution is found.

### 6.4.3 Baseline model scenario: group C

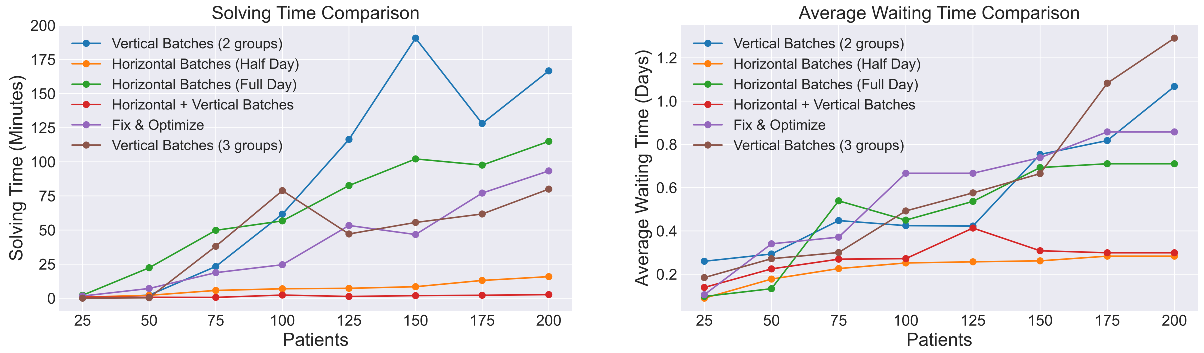


(a) Min make span

(b) Difference make span compared to lower bound

Figure 6.8: Group C, comparison make span

The minimum makespans of all strategies differ as the number of patients increases, as shown in Figure 6.8a. Figure 6.8b illustrates the deviations relative to the lower bound. *The Horizontal Batch algorithm* approaches the lower bound as the number of patients increases.



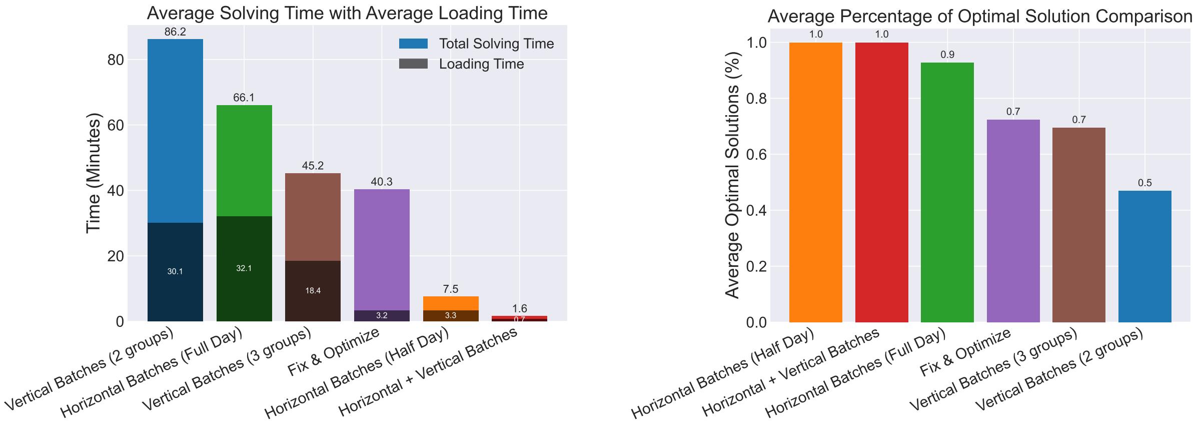
(a) Solving Time

(b) Average Waiting Time

Figure 6.9: Group C, Performance Metrics 1

Figure 6.9 presents the first part of the algorithmic performance results for group C. The trends resemble those observed in group B.

Figure 6.10 shows the second part of the performance results for group C. Again, the trends are consistent with those in group B. Due to the solving time adjustment in the *Vertical Batch Algorithm*, the average solving time decreased. Similarly, the *Horizontal Batch (Full Day)* model achieved a higher fraction of optimal solutions, also as a result of solving time adjustments.



(a) Solving Time and Loading Time

(b) Percentage Optimal Solutions

Figure 6.10: Group C, Performance Metrics 2

### 6.4.4 Extended model scenario

The following algorithms are used to test the extended model scenario: the *Horizontal Batch Algorithm (Half Day)* and the *Horizontal and Vertical Batch Algorithm (Half Day & Individual Subgroups)*. In the *Vertical Batch Algorithm*, subgroups are defined based on doctor characteristics. The first subgroup includes all unique doctors together with one representative of each non-unique doctor type, while each remaining duplicate doctor forms its own subgroup. To make this clear, see the example:

Set of doctors = {Clinician (Monday, Morning), Clinician (Monday, Morning), Echografist (Monday, Full Day), Echografist (Monday, Full Day), Student (Monday, Morning)}

**Subgroup 1:** {Clinician (Monday, Morning), Echografist (Monday, Full Day), Student (Monday, Morning)}
**Subgroup 2:** {Clinician (Monday, Morning)}
**Subgroup 3:** {Echografist (Monday, Full Day)}

In this example, only the Student is unique on Monday morning. Two Clinicians work the Monday morning shift, and two Echografists work the full-day shift on Monday. Therefore, the first subgroup consists of the Student, one Clinician, and one Echografist. Each of the remaining duplicate doctors is then assigned to its own subgroup. All subgroups contain the same rooms; the split is applied only to doctors.

Since in the extended model scenario the goal is to make a schedule for 6 weeks, the main KPI is changed to schedule as many complete patients as possible in this time window.

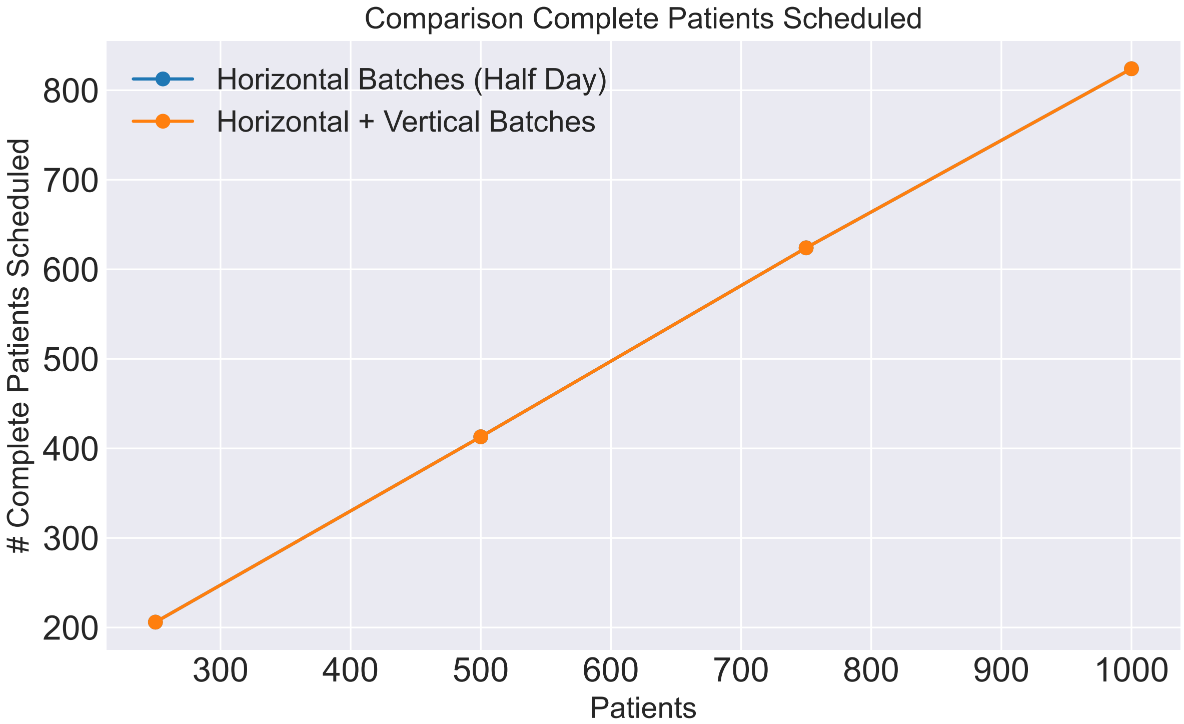
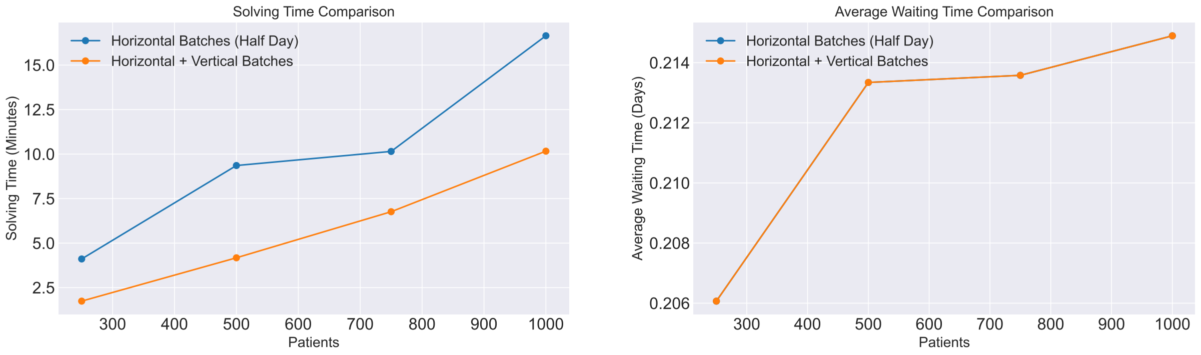


Figure 6.11: Extend model, comparison complete scheduled patients



(a) Solving Time

(b) Average Waiting Time

Figure 6.12: Extend model, performance metrics

Figure 6.11 presents the total number of patients scheduled over a six–week horizon for different sizes of patient sets. Both algorithms achieve the same number of complete scheduled patients. A similar observation holds for the average waiting time (see Figure 6.12b). The only difference between the *Horizontal Batches (Half Days)* and the *Horizontal and Vertical Batches* algorithm is the computation time. The *Horizontal and Vertical Batches* method consistently outperforms the former in terms of computation time, with reductions ranging from 35% to 60%. Therefore, the *Horizontal and Vertical Batches* algorithm is applied to solve the scheduling problem with a six–week scheduling horizon. The corresponding performance results are summarized in Table 6.12. **The Horizontal and Vertical Batches algorithm generates a six–week schedule within 20 minutes and achieves a solution with 2.6% difference compared to the lower bound.**

| **Metric** | **Value** |
|---|---|
| Solving Time (minutes) | 19.03 |
| Loading Time (minutes) | 16.73 |
| Percentage Working Time | 54.06% |
| Patients Completely Scheduled | 1544 |
| Average Waiting Time (days) | 0.04 |
| Lower Bound Gap | 2.6% |
| Average Doctor Utilization | 94.00% |

Table 6.12: Performance metrics of the scheduling algorithm.

## 6.5 Double Booking

The double booking strategy is analyzed based on the following performance metrics:

- **Average waiting time for appointment.** This is not the average waiting time used to compare different algorithms. With the comparison of the double booking strategy the average waiting time for being helped on arrival is used which is in minutes (instead of days).

- **Percentage of appointments covered by double booking.** The main principle of double booking is to ensure patient attendance and minimize the impact of no-shows. The percentage of appointments covered by double bookings represents the proportion of no-show appointments that are still attended by another patient. In an ideal scenario, this percentage would be 100%, indicating that every appointment affected by a no-show is effectively used by another patient due to double booking. A lower percentage indicates that some appointments remain unused due to no-shows, resulting in idle time and reduced scheduling efficiency.

- **Percentage doctor idle time.** Double bookings have positive effect on the doctor idle time, since the goal of double bookings is to lower the doctor idle times. This doctors idle time is only based on working hours, so excluding the break.

- **Doctor's overtime.** The double-booking strategy leads to more patients waiting throughout the day, including at the end of the doctor's shift. This often enforces the doctor to work overtime to help those still waiting. Since overtime is costly, it is important to minimize or avoid it.

- **Cost.** Double booking has a positive effect on doctors idle time but a negative effect on patients waiting time, both incure a cost. See Table 4.10 for the cost overview. Specialist making use of expensive equipments are more expensive than clinicians.

The results are shown in Table 6.13. For some metrics the best and worst solution is indicated with green and red, respectively. The no double booking strategy is the worst strategy based on most metrics. It has a low number of complete scheduled patients, high doctor idle time and high cost. The best strategy based on most metrics is the Bailey-Welch double book strategy. It has a high number of complete scheduled patients, low doctor idle time and high double book coverage. To see the complete distribution of the waiting times of each strategy, see Figure A.2. For the distribution of the doctor's overtime for each strategy see Figure A.2. For the distribution of idle time see Figure A.2. For the simulation results of the cost based double booking strategy see Figure A.1, in the results the average cost is taken of each metric which results in €65.718.

| **Metric (avg = average)** | **No DB** | **DB (SD)** | **DB (BW)** | **DB (cost)** |
|---|---|---|---|---|
| # No-shows | 529 | 587 | 699 | 591 |
| No-show percentage (%) | 17.88 | 18.35 | 19.16 | 18.31 |
| # Double bookings | 0 | 296 | 589 | 193 |
| # Complete scheduled patients | 1544 | 1856 | 2084 | 1759 |
| Doctor idle time (%, avg) | 22.24 | 17.89 | 15.21 | 17.27 |
| Doctor idle time (Hours, avg) | 402.75 | 343.75 | 284 | 324.75 |
| Doctor over time (Minutes, avg) | 0 | 14.63 | 33.75 | 5.25 |
| Covered by double bookings (%) | 0 | 37.3 | 65.4 | 36.8 |
| Average waiting time (Minutes) | 0 | 2.55 | 8.98 | 3.64 |
| Cost (€) | 79,184 | 74,933 | 72,254 | 65.718 |

Table 6.13: Performance Metrics Different Double Book Strategy

Based on the results, the average patient waiting time increases when double booking is implemented. Specifically, the Bailey–Welch strategy results in an average waiting time more than three times higher than that of the standard double booking strategy. This increase can be explained by a larger number of patients scheduled within the six-week time horizon. The benefits of double booking can be found in the reduction of doctor idle time. Without double booking, the doctors' idle time is approximately 22.24%, whereas under the Bailey–Welch strategy it decreases to around 15.21%. Across all strategies—except for the cost-based double booking strategy—a clear trend emerges: as the number of double bookings increases, doctor idle time and overall cost decrease, while patient waiting time increases. Notably, the cost-based double booking strategy achieves the lowest total cost, despite having relatively few double bookings and maintaining a low average patient waiting time.

As mentioned earlier, patient waiting times increase more under the Bailey–Welch strategy compared to the standard double booking approach. Figure 6.13 and Figure 6.14 present the cumulative distribution functions of waiting times for both strategies, providing a more detailed comparison. Under the standard double booking strategy, 90% of patients are seen within 10 minutes, whereas under the Bailey–Welch strategy, this is within 25 minutes

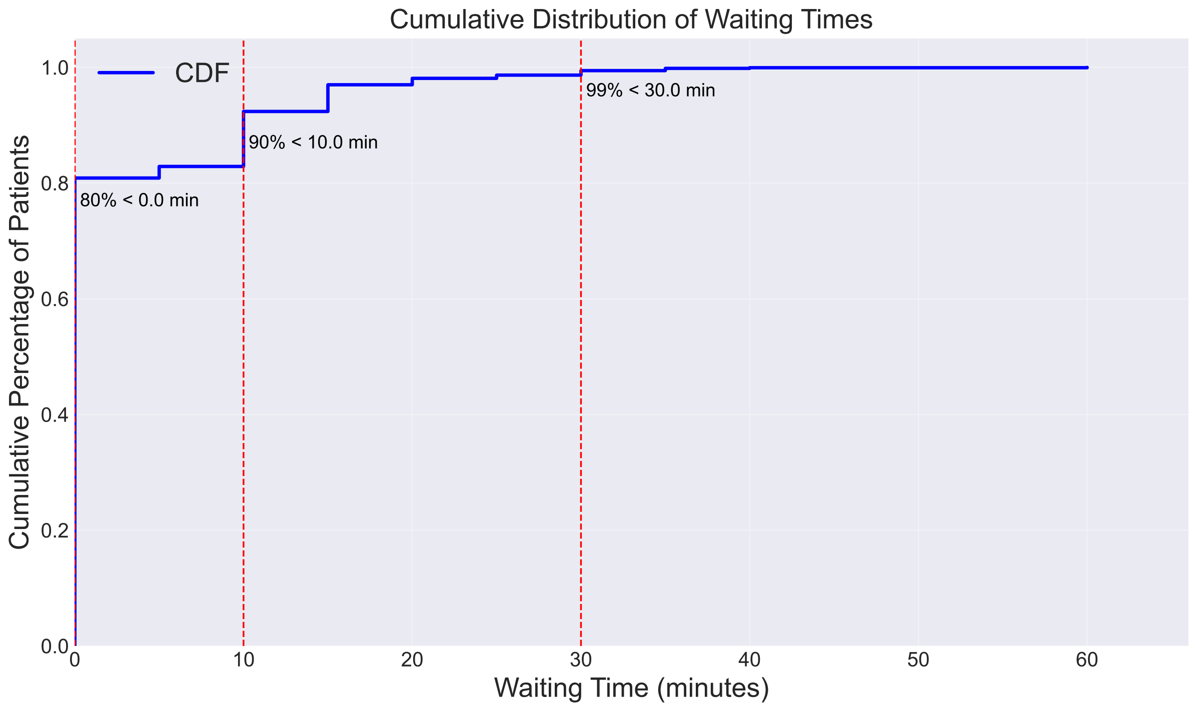


Figure 6.13: Cumulative distribution function under the Standard double book strategy

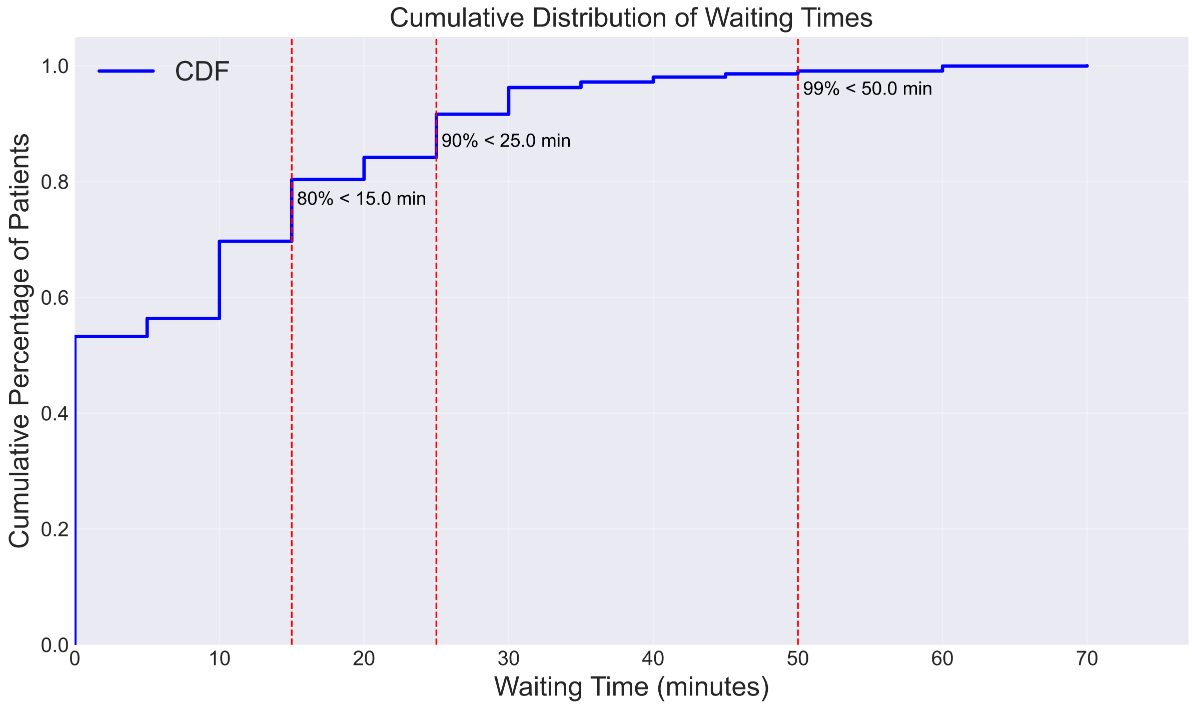


Figure 6.14: Cumulative distribution function under the Bailey-Welch double book strategy

In extreme cases, it can be stated that it is almost certain that a patient will be helpd within 30 minutes under the standard double booking strategy, and within 50 minutes under the Bailey-Welch strategy.

On average a patient requires two appointments in order to be completely scheduled. In the case in which no double book strategy is applied, 1544 patients are completely scheduled, that is approximately a total of 3088 appointments. All scheduled appointments can be classified as show or no-show. No-show appointments are appointments in which a patient did not showed up and show appointments are appointments where all patient did show up. In order to get an overview of the appointments affectected by no show or by the double booking see Figure 6.15 and Figure 6.16 for the appointment distribution. In this figure, an overview of show and no-show appointments are presented. From these no-show appointments, the double booked patient will visit a percentage of these appointments if double booking is applied (orange).

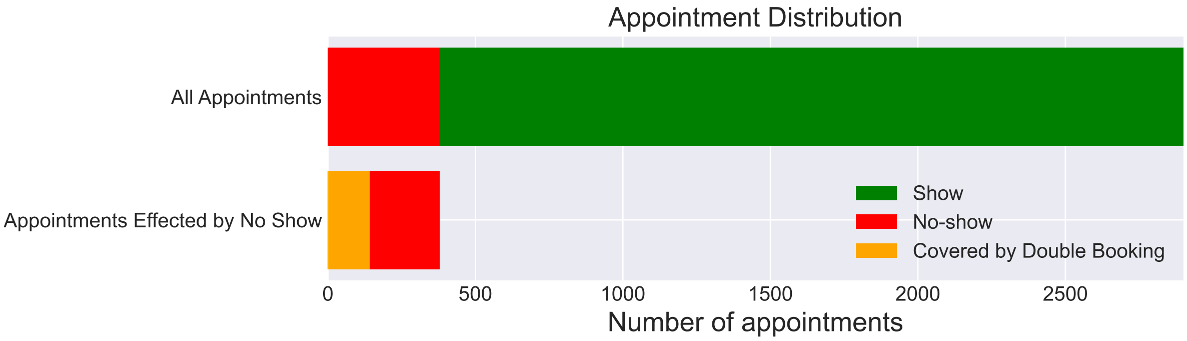


Figure 6.15: Appointment Distribution {Standard DB}}

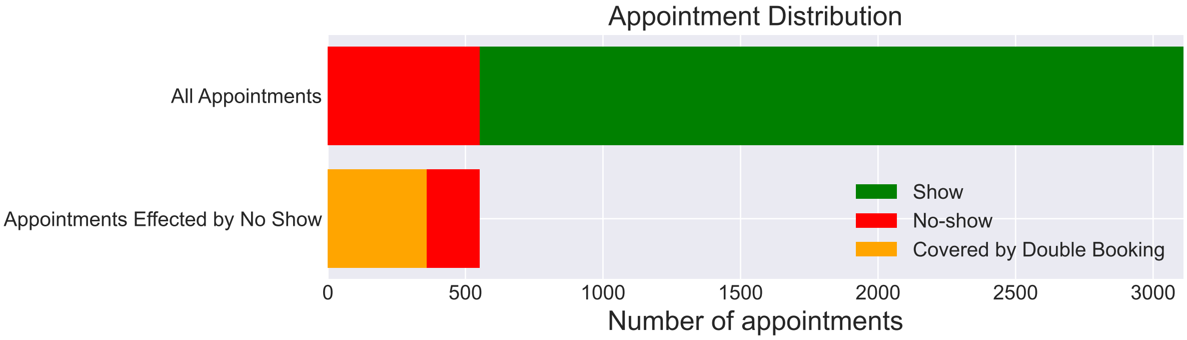


Figure 6.16: Appointment Distribution {Bailey-Welch DB}}

In the appointment distribution the coverage by double bookings can be seen. In the case of the Bailey-Welch strategy this coverage is higher than with the standard strategy.

To get an overview of the important results, a summary is displayed in Table 6.14. In this overview the main impact of the strategies are displayed on the main KPI.

| Metric | No DB | DB (standard) | DB (BW) | DB (cost) |
|---|---|---|---|---|
| Doctor idle time (%) | 22.24 | 17.89 | 15.21 | 17.27 |
| Average waiting time (Minutes) | 0 | 2.55 | 8.98 | 3.64 |
| Covered by double bookings (%) | 0 | 37.3 | 65.4 | 36.8 |
| Cost (€) | 79,184 | 74,933 | 72,254 | 65,718 |

Table 6.14: Performance Metrics Different Double Book Strategy

# Chapter 7

# Conclusion

This thesis addressed the complex problem of constructing efficient outpatient clinic schedules in the presence of patient no-shows and multiple operational constraints. The underlying scheduling problem requires assigning a set of patients each with a sequence of care types characterized by care durations and recovery times to a multi-resource environment consisting of doctors and rooms, each with specific capabilities and availabilities. The objective is to create a feasible schedule over a discrete time horizon while minimizing the makespan. Additional hospital requirements were incorporated, such as maximum appointments per shift and care type, limits on new patients per doctor or per shift, and the possibility that some patients must remain with the same doctor throughout their care sequence.

To support more effective scheduling, a no-show prediction model was first developed to estimate the probability that patients miss their appointments. A LightGBM classifier using patient demographics, temporal features, reminder information, and historical attendance achieved an accuracy of approximately 0.69. Although moderate, the model predicted low no-show probabilities more reliably than high ones, enabling meaningful differentiation between more and less reliable patients.

Subsequently, several scheduling algorithms were designed and compared, including the Optimal algorithm, Vertical Batch, Horizontal Batch, combined Horizontal and Vertical Batch, and the Fix-and-Optimize heuristic. These algorithms were evaluated first on a basic model and later extended to incorporate realistic constraints such as daily shift structures, per-shift appointment limits, and new-patient restrictions. In the basic model, the Horizontal Batch algorithm yielded the best makespan, while the combined Horizontal and Vertical Batch algorithm achieved the lowest solving time. When scaled to larger patient sets in the extended model—and with an adapted subgroup-formation method—both algorithms achieved comparable makespan performance. However, the combined Horizontal and Vertical Batch algorithm remained significantly faster, reducing solving times by 35%–60%. Based on these results, this algorithm is identified as the most effective and scalable scheduling approach.

In addition, several double-booking strategies were examined: the standard, Bailey-Welch, and cost-based approaches. While double booking increases patient waiting times, 90% of patients were still served within 10–25 minutes, and doctor idle time was markedly reduced. Without double booking, doctors were idle 22% of the time on average; this dropped to 17% with the standard and cost-based approaches and to 15% under Bailey-

Welch. The cost-based approach produced the best overall performance, achieving the lowest total cost of €65,718 compared with €72,254 for Bailey-Welch and €74,933 for the standard strategy.

In conclusion, this research demonstrates that integrating predictive no-show information into a mathematical scheduling framework can substantially enhance the efficiency of outpatient clinic operations. The combination of the mixed-integer linear programming model with the Horizontal and Vertical Batch algorithm offers high-quality schedules with strong computational performance, and when paired with a cost-based double-booking strategy, it effectively balances patient waiting times and resource utilization. This approach will be implemented at the outpatient clinic of Amsterdam UMC (location VUmc), supporting the development of more flexible, data-driven, and operationally efficient appointment schedules.

# Chapter 8

# Discussion

The findings of this study demonstrate that incorporating predictive analytics into appointment scheduling with a double book strategy can significantly improve outpatient clinic efficiency. The LightGBM model proved to be a moderate predictor on the synthetic data. The absence of real patient data limited the model accuracy for predicting no show. In the synthetic dataset the patient history, the no-show of previous appointments, proved to be a strong predictor for no-show behavior although this patient history was not present or limited to mostly one,two or three appointments. An improvement in the predictions could be made if real data are used that capture a longer patient history.

This improvement in no-show prediction could result in an improvement in double book efficiency, the so called "Covered by double bookings". Higher coverage means higher resource utilization which has a positive effect on the cost.

The Horizontal And Vertical Batch Algorithm, achieved near-optimal schedules with substantially lower computation times than other meta-heuristic. This balance between accuracy and scalability is crucial for applying the algorithm in practice where dynamic updates are frequent. For further improvement in computation time, the Horizontal And Vertical Batch algorithm could be adapted for parallel computing. Each subgroup is solved independent and could be send to a different machine to be solved.

Additionally, the scheduling model assumes deterministic service times and constant room availability, while in practice, these service times are variable and rooms can be unavailable due to maintenance and introduce further complexity. Future work could incorporate variable service times and room unavailability.

This study focused on double bookings as a way to deal with no-show behavior. It could be interesting to investigate the effect of scheduling more than 2 patients in certain time slots and compare this with the standard double booking strategy.

Overall, this research highlights the potential of data-driven scheduling to mitigate the impact of no-shows, reduce waiting lists, and improve resource utilization. The proposed framework provides a foundation for hospitals seeking to modernize their scheduling processes and transition toward intelligent, automated planning systems that increase operation efficiency.

# Appendix A

# Appendix

## A.1 Tables

| **ID** | **Shift** | **Specialty** | **Working Days** |
|---|---|---|---|
| D0 | Morning | Clinician | Monday |
| D1 | Morning | Clinician | Tuesday |
| D2 | Morning | Clinician | Wednesday |
| D3 | Morning | Clinician | Thursday |
| D4 | Morning | Clinician | Friday |
| D5 | Morning | Clinician | Saturday |
| D6 | Afternoon | Clinician | Monday |
| D7 | Afternoon | Clinician | Tuesday |
| D8 | Afternoon | Clinician | Wednesday |
| D9 | Afternoon | Clinician | Thursday |
| D10 | Afternoon | Clinician | Friday |
| D11 | Afternoon | Clinician | Saturday |
| D12–D17 | Full day | Clinician | Mon–Sat |
| D18–D23 | Full day | Clinician | Mon–Sat |
| D24–D29 | Morning | Doctor Assistant | Mon–Sat |
| D30–D35 | Afternoon | Doctor Assistant | Mon–Sat |
| D36 | Full day | CR-Thorax | All |
| D37 | Full day | Spirometrie | All |
| D38 | Full day | Echografist | All |
| D39 | Full day | General | Mon–Fri |
| D40 | Full day | General | Mon–Fri |

Table A.1: Doctors

| ID | Capability |
|---|---|
| R0 | CR-Thorax |
| R1 | CT Scan |
| R2 | Doctor Appointment |
| R3 | Doctor Appointment |
| R4 | Doctor Appointment |
| R5 | Doctor Appointment |
| R6 | Doctor Appointment |
| R7 | Dynamap |
| R8 | ECG (Blood Pressure) |
| R9 | EKG |
| R10 | Holter |
| R11 | MRI |
| R12 | Spirometrie |
| R13 | TTE |
| R14 | XECG |

Table A.2: Rooms

| **Rule** | **Entity** | **Condition / Description** |
|---|---|---|
| R0 | Patient | The following procedures have immediate results: XECG, ECG (BP), and PM/ICD. These procedures require a doctor appointment directly afterwards. |
| R0 | Patient | Procedures with results available within one hour include TTE, CR-Thorax, and Spirometry. These procedures require a doctor appointment after 1 hour. |
| R0 | Patient | The same doctor must be assigned for ECG, Holter, PM/ICD, TTE, and XECG procedures. |
| R1 | Doctor | Clinicians, Doctor Assistants, and General doctors can perform: B.P.C., CPAP, D.A., Dynamap, ECG, EKG, Holter, Pacemaker, PM/ICD, Syncope, TTE, and XECG. |
| R1 | Doctor | Echografists are responsible for CT Scans and MRI procedures. |
| R3 | Scheduler | Working hours: 09:00–18:00; Morning session ends at 13:00. |
| R3 | Scheduler | Appointment slot length: 5 minutes; Minimum workload per doctor: 44 half-days per year. |
| R3 | Scheduler | Maximum appointments per day: Half-day = 15, Full-day = 25, Holter = 5, XECG = 5, Pacemaker  10.7. |
| R3 | Scheduler | Maximum new patients per day: Half-day = 2, Full-day = 4. |
| R3 | Scheduler | Between 17:00 and 18:00, only virtual Doctor appointments are allowed. |

Table A.3: Detailed Scheduling Rules

## A.2 Graphs

### Simulation Results

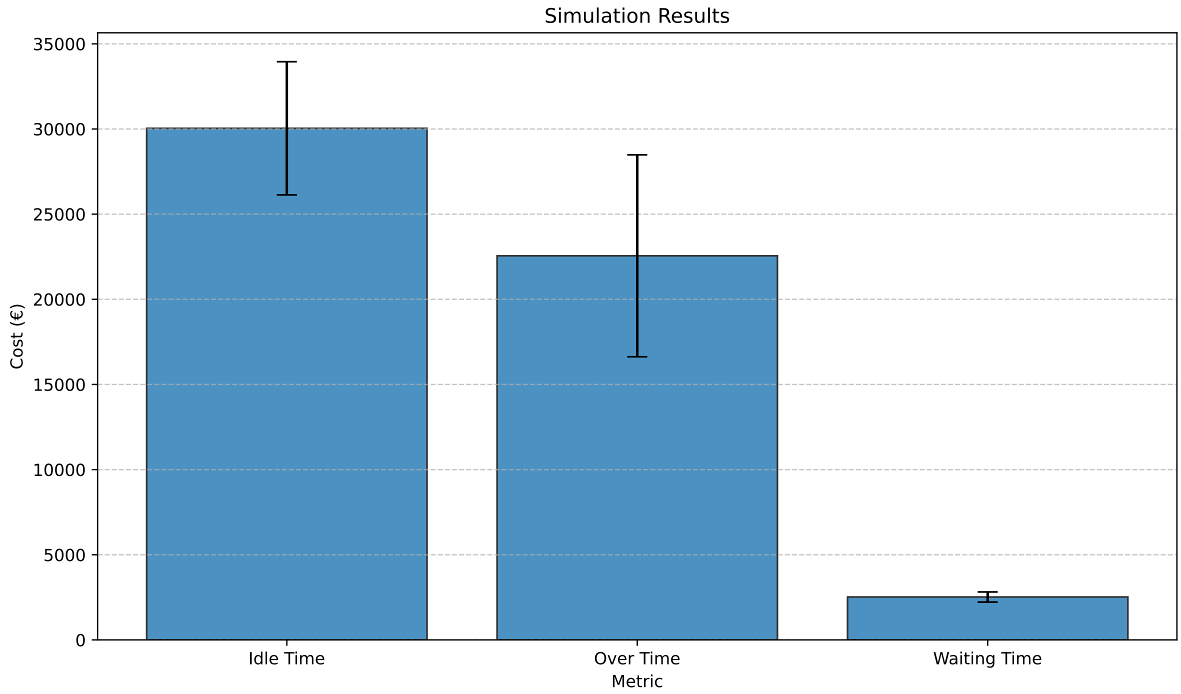


Figure A.1: Monte Carlo Cost Simulation Results

### Double Booking, Patient Waiting Time Distributions

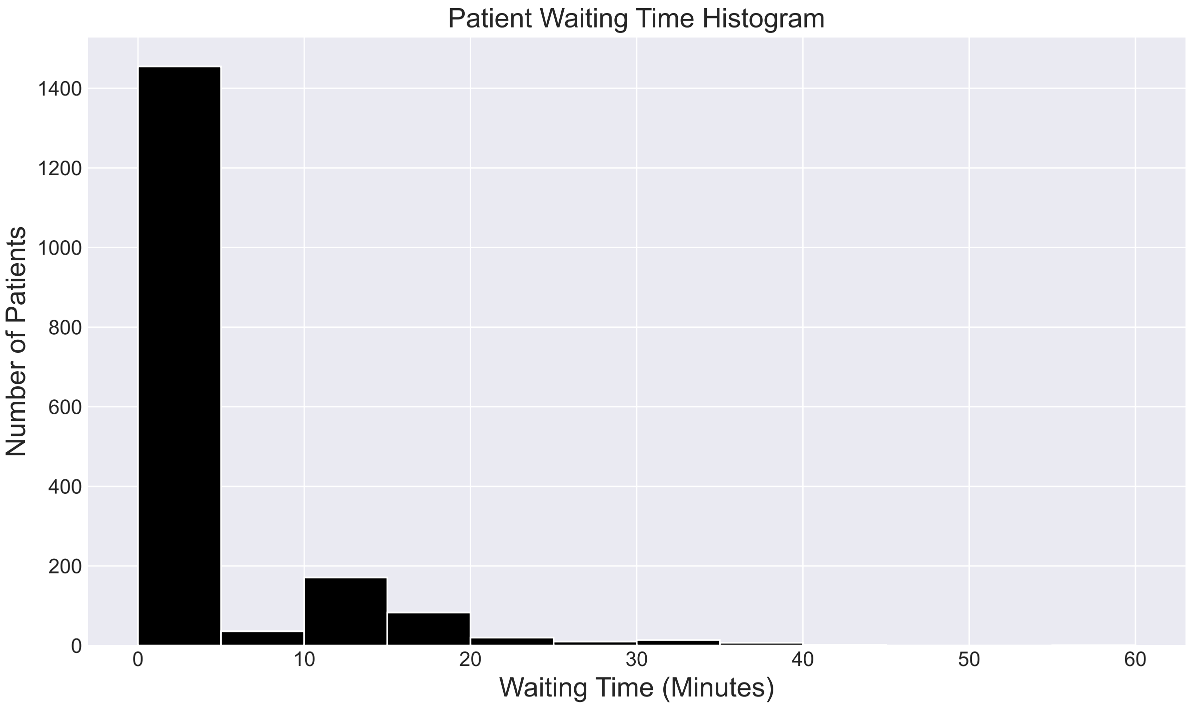


Figure A.2: Patient Waiting Time Distribution {Standard Double Book Strategy}

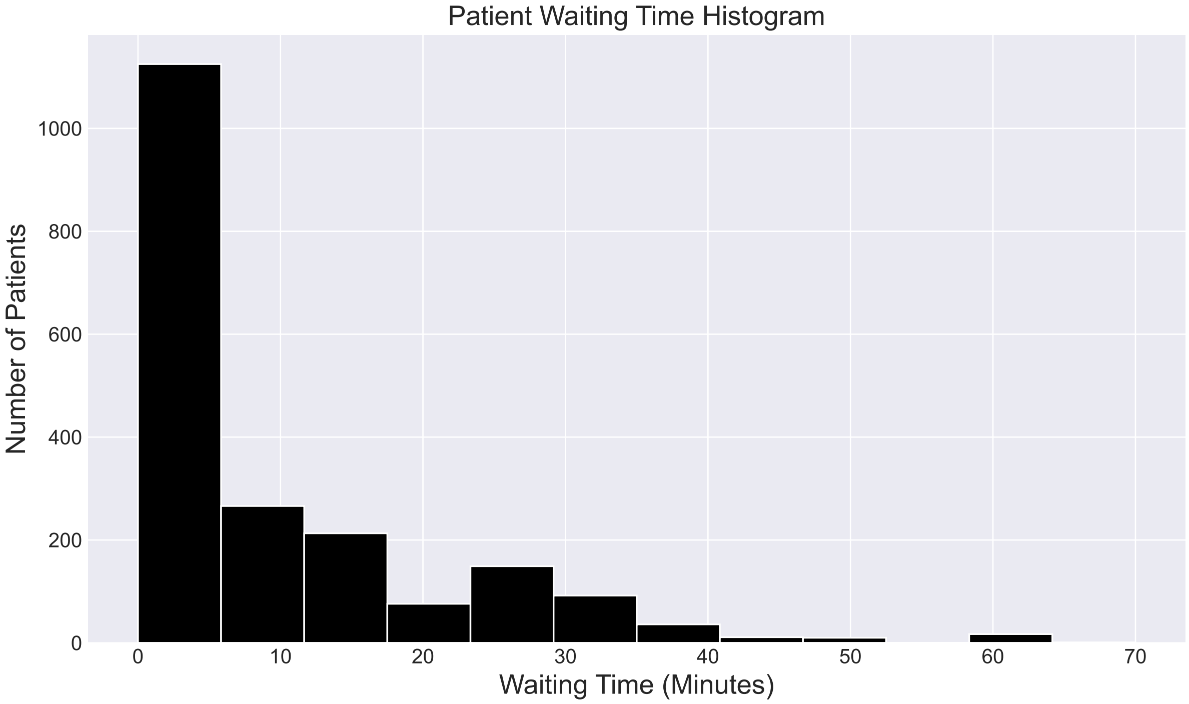


Figure A.3: Patient Waiting Time Distribution {Bailey-Welch Double Book Strategy}

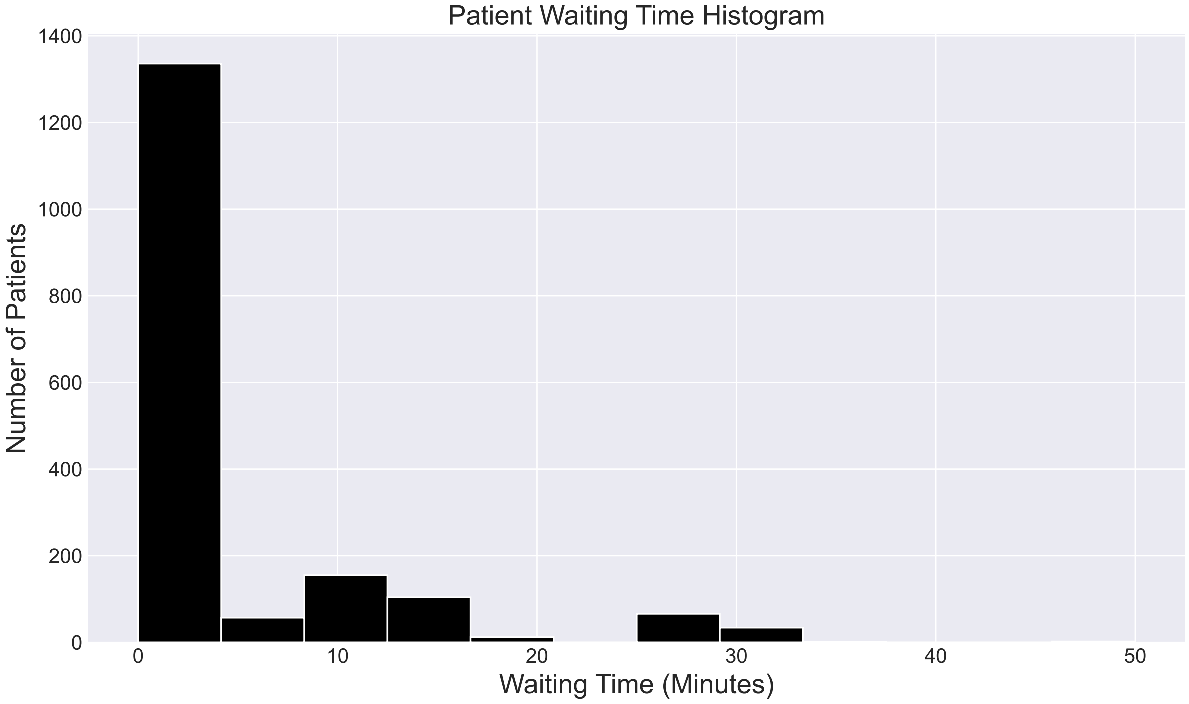


Figure A.4: Patient Waiting Time Distribution {Cost Double Book Strategy}

### Double Booking, Doctor's Idle Time Distributions

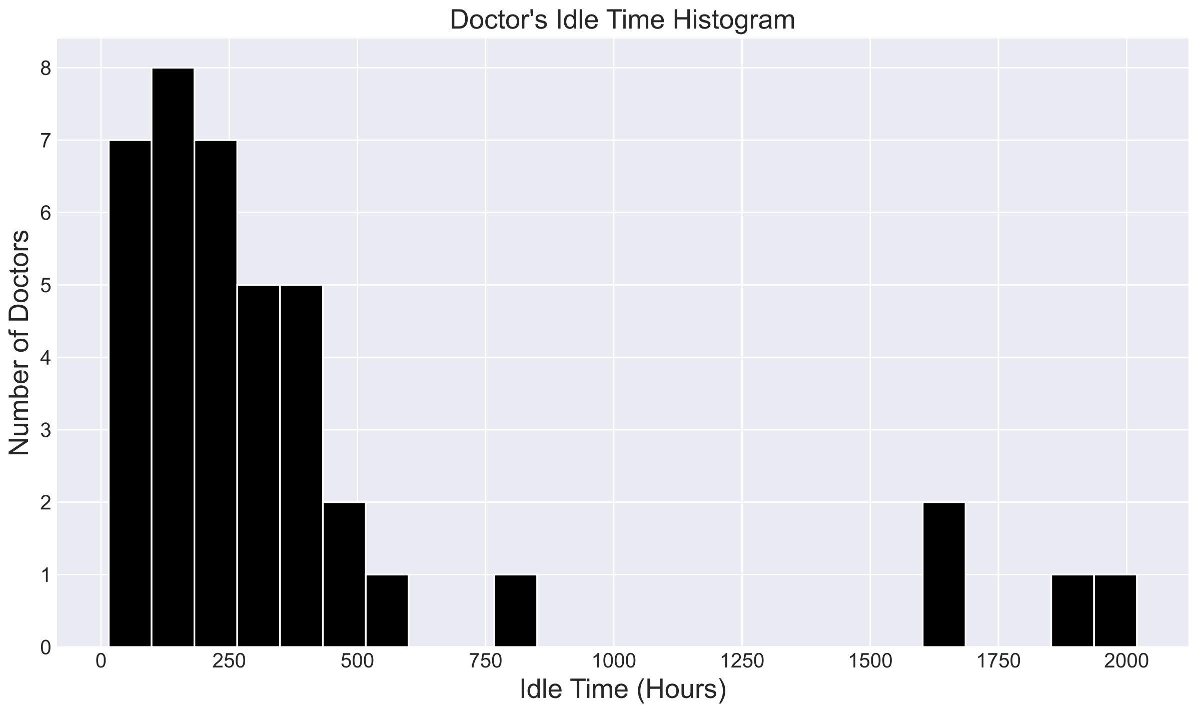


Figure A.5: Doctor's Idle Time Distribution {No Double Booking}.

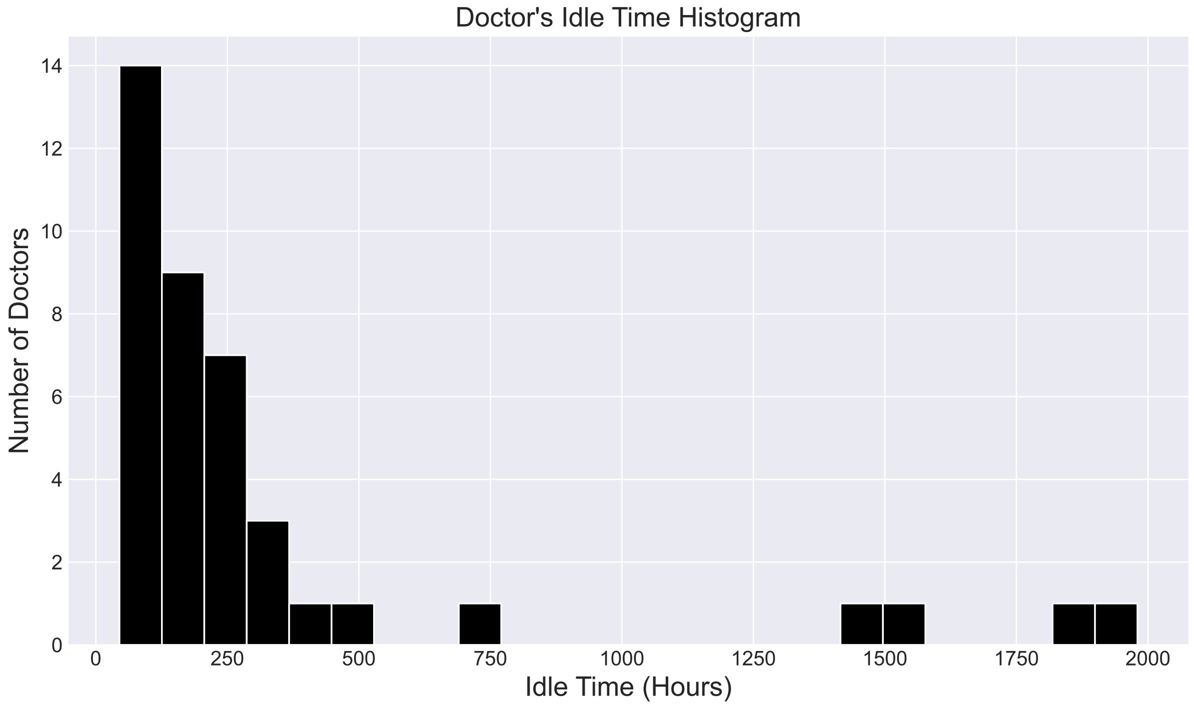


Figure A.6: Doctor's Idle Time Distribution {Double Booking Standard}.

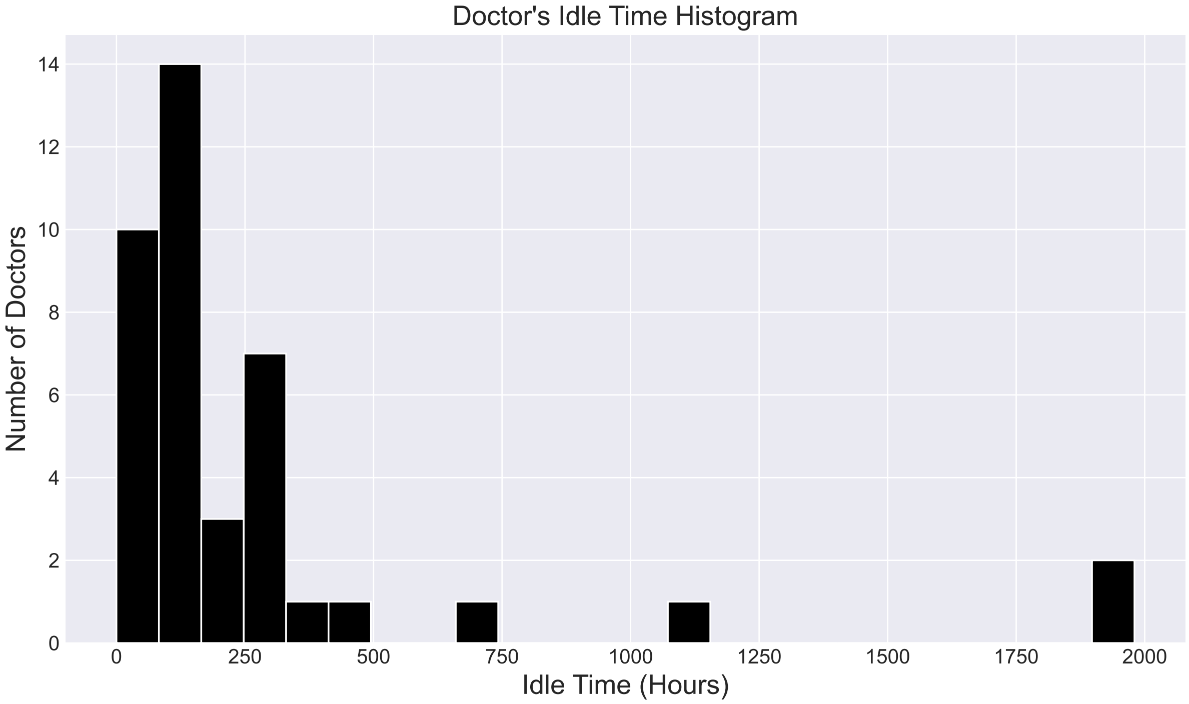


Figure A.7: Doctor’s Idle Time Distribution {Double Booking Bailey-Welch}.

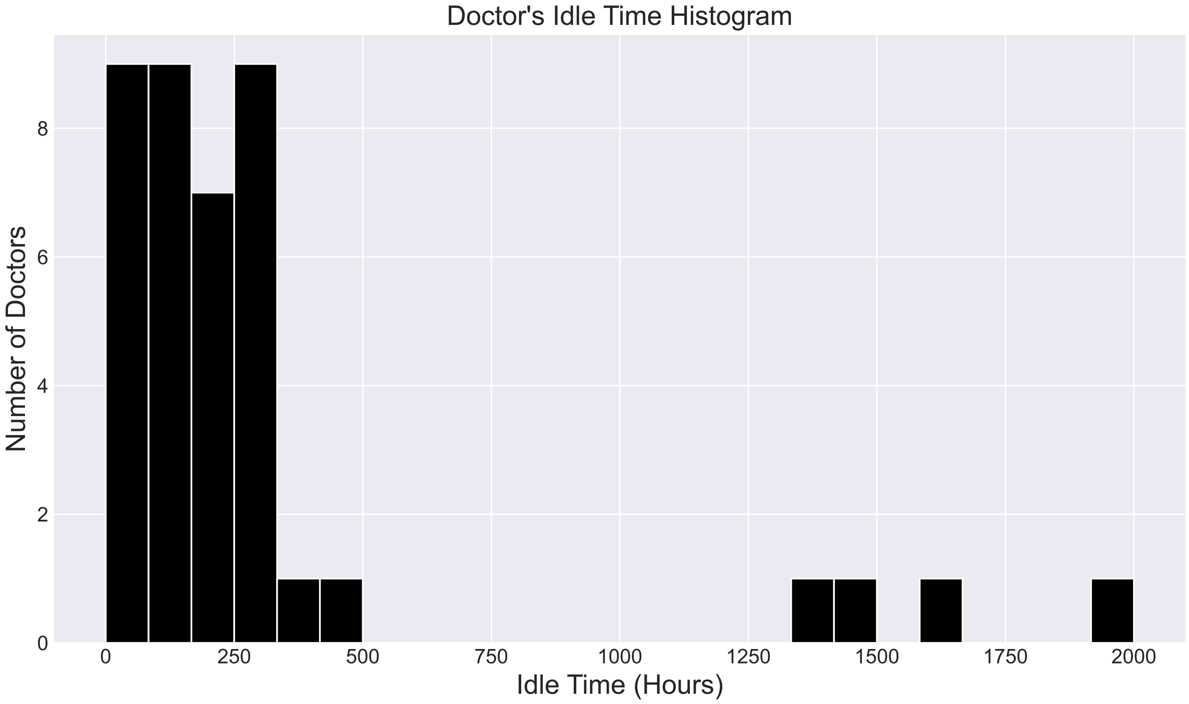


Figure A.8: Doctor’s Idle Time Distribution {Double Booking Cost}.

### Double Booking, Doctor's Over Time Distributions

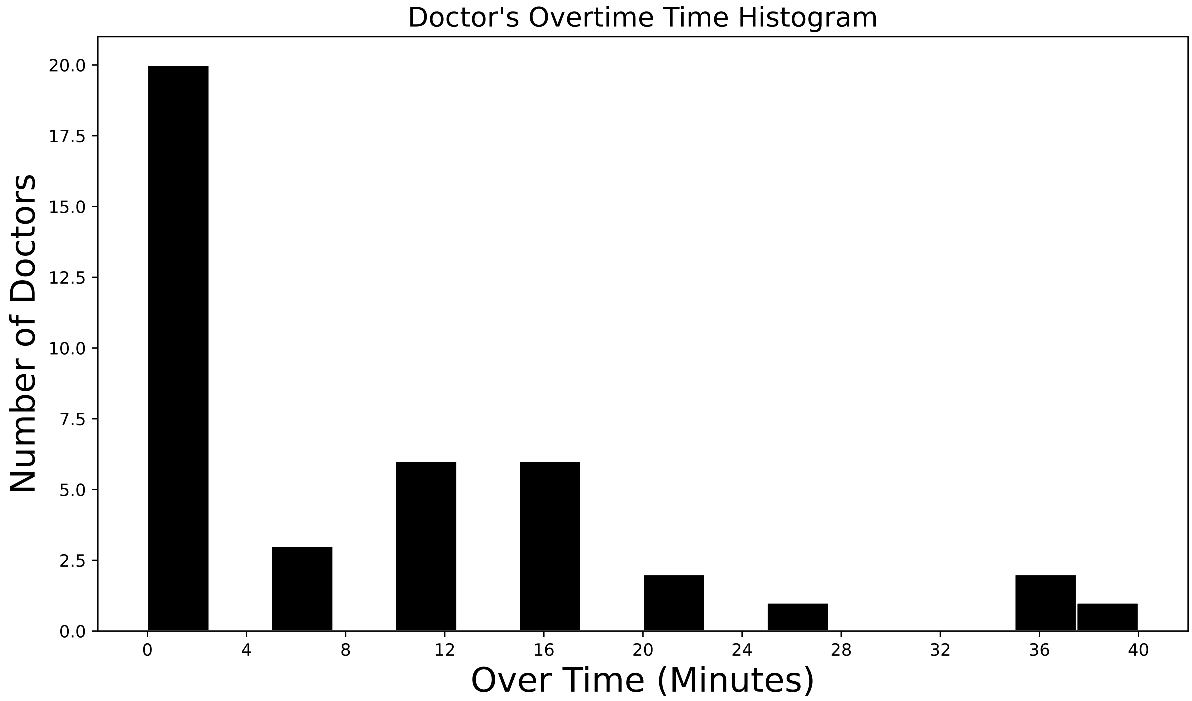


Figure A.9: Doctor's Over Time Distribution {Double Booking Standard}.

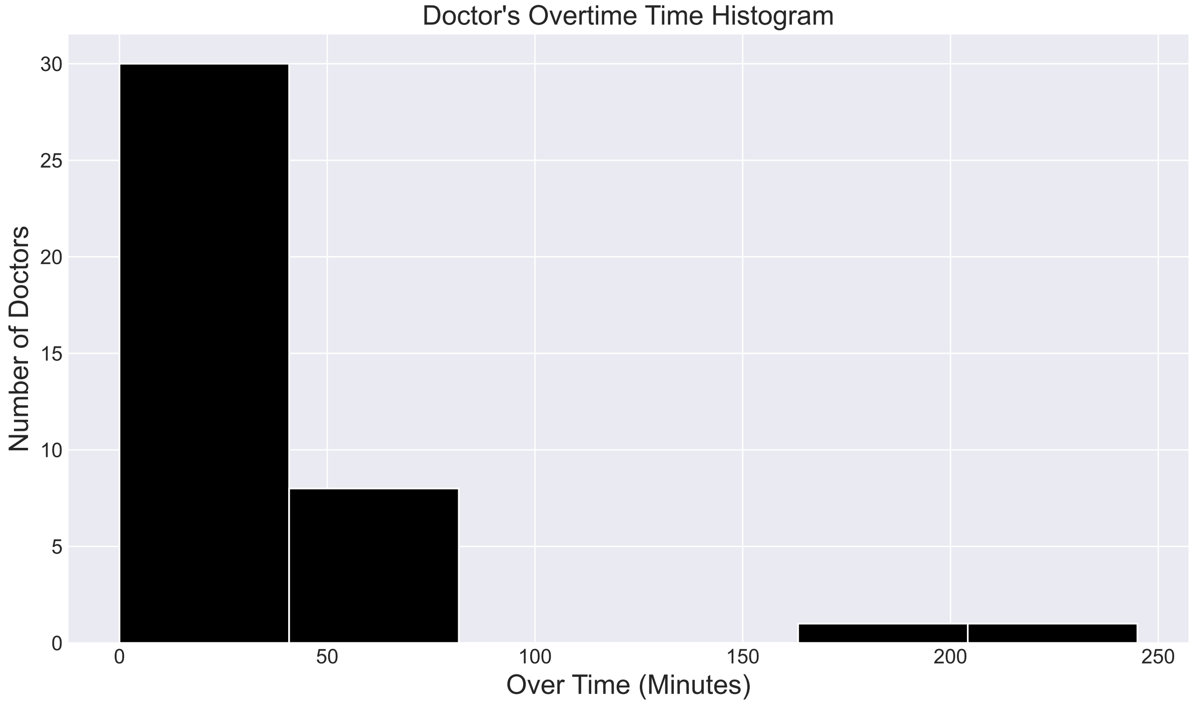


Figure A.10: Doctor's Over Time Distribution {Double Booking Bailey-Welch}.

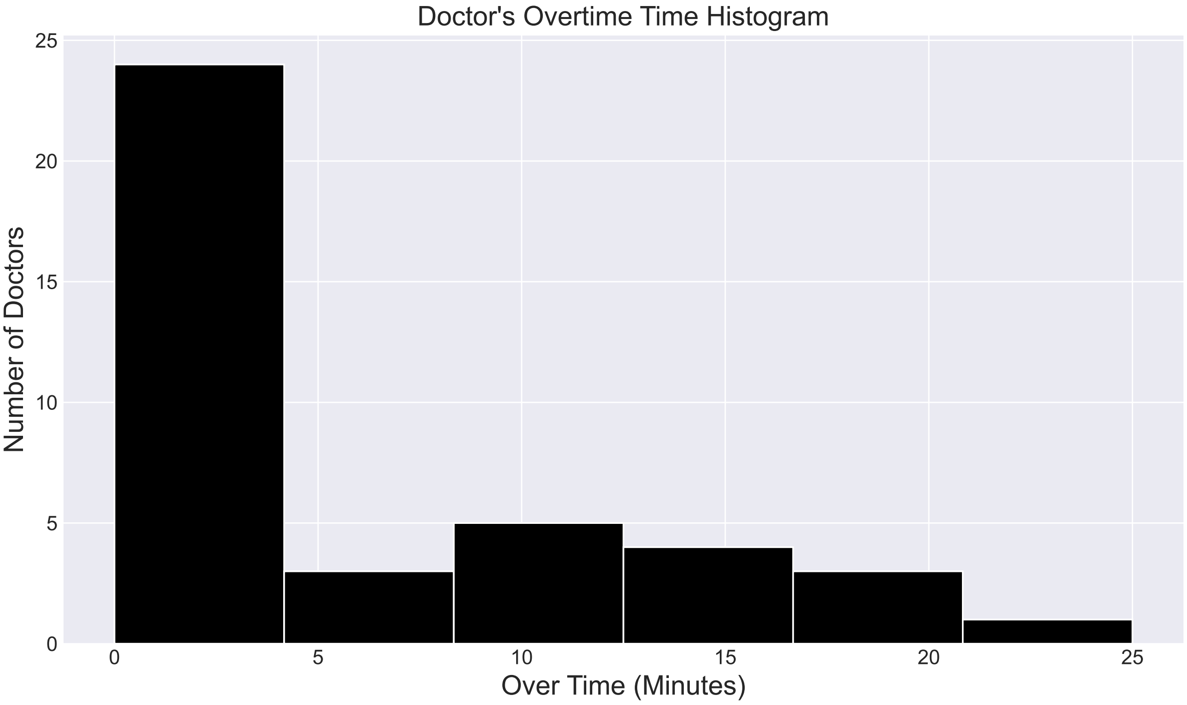


Figure A.11: Doctor's Over Time Distribution {Double Booking Cost}.

# Bibliography

[1] A. Ahmadi-Javid, Z. Jalali, and K. J. Klassen. "Outpatient appointment systems in healthcare: A review of optimization studies". In: *European Journal of Operational Research* 258.1 (Apr. 2017), pp. 3–34. DOI: 10.1016/J.EJOR.2016.06.064.

[2] A. Alaeddini et al. "A probabilistic model for predicting the probability of no-show in hospital appointments". In: *Health Care Management Science* 14.2 (2011), pp. 146–157. DOI: 10.1007/s10729-011-9148-9.

[3] N. N. Amalina, K. B. Ofori-Amanfo, and A. Heungjo. "A Multi-Head Attention Soft Random Forest for Interpretable Patient No-Show Prediction". In: *arXiv preprint* (2025). DOI: 10.48550/arXiv.2505.17344.

[4] Anonymous. "Development of an evidence-based model for predicting patient, provider, and appointment factors that influence no-shows in a rural healthcare system". In: *BMC Health Services Research* 23 (2023), p. 115. DOI: 10.1186/s12913-023-09115-6.

[5] Samira Fazel Anvaryazdi, Saravanan Venkatachalam, and Ratna Babu Chinnam. "Appointment scheduling at outpatient clinics using two-stage stochastic programming approach". In: *IEEE Access* 8 (2020), pp. 175297–175305. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3025997.

[6] D. Barrera Ferro et al. "Improving healthcare access management by predicting patient no-show behaviour". In: *arXiv preprint* (2020). DOI: 10.1016/j.dss.2020.113398.

[7] Bjorn P. Berg et al. "Optimal booking and scheduling in outpatient procedure centers". In: *Computers & Operations Research* 50 (Oct. 2014), pp. 24–37. ISSN: 0305-0548. DOI: 10.1016/J.COR.2014.04.007.

[8] S. Boughorbel, F. Jarray, and A. Kadri. "Fairness in TabNet Model by Disentangled Representation for the Prediction of Hospital No-Show". In: *arXiv preprint* (2021). arXiv:2107.XXXX [cs.LG]. URL: https://arxiv.org/abs/2107.XXXX.

[9] Robert L. Burdett et al. "A mixed integer linear programing approach to perform hospital capacity assessments". In: *Expert Systems with Applications* 77 (July 2017), pp. 170–188. ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2017.01.050.

[10] Centraal Bureau voor de Statistiek. *Hoe vergrijsd is Nederland?* 2022. URL: https://www.cbs.nl/nl-nl/visualisaties/dashboard-bevolking/leeftijd/ouderen.

[11] P.-S. Chen et al. "Applying heuristic algorithms to solve inter-hospital hierarchical allocation and scheduling problems of medical staff". In: *International Journal of Computational Intelligence Systems* 13.1 (2020), pp. 318–331. DOI: 10.2991/ijcis.d.200310.004.

[12] A. Chouksey, A. K. Agrawal, and A. N. Tanksale. "Accelerated bender's decomposition algorithm and hybrid heuristics for multi-period planning of maternal healthcare facilities in India". In: *Journal of the Operational Research Society* (2024), pp. 1–20. DOI: 10.1016/j.ejor.2024.04.013.

[13] C. Deina et al. "Decision analysis framework for predicting no-shows to appointments using machine learning algorithms". In: *BMC Health Services Research* 24 (2024), p. 37. DOI: 10.1186/s12913-023-10418-6.

[14] Diwakar Gupta and Brian Denton. "Appointment scheduling in health care: Challenges and opportunities". In: *IIE Transactions* 40.9 (2008), pp. 800–819. DOI: 10.1080/07408170802165880.

[15] Pedram Hooshangi-Tabrizi et al. "Improving patient-care services at an oncology clinic using a flexible and adaptive scheduling procedure". In: *Expert Systems with Applications* 150 (July 2020), p. 113267. ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2020.113267.

[16] Y.-C. Huang and D. A. Hanauer. "Patient no-show predictive model development using multiple data sources for an effective overbooking approach". In: *Applied Clinical Informatics* 5.3 (2014), pp. 836–860. DOI: 10.4338/ACI-2014-04-RA-0026.

[17] Joniarroba. *Medical Appointment No Shows*. Kaggle dataset. Accessed: 24 Sept 2025. Kaggle. 2017. URL: https://www.kaggle.com/datasets/joniarroba/noshowappointments.

[18] Kaggle Community. *What machine learning approaches have won most Kaggle competitions?* Accessed: 23 Oct 2025. 2021. URL: https://www.kaggle.com/discussions/general/248068.

[19] P. Kheirkhah et al. "Prevalence, predictors and economic consequences of no-shows". In: *BMC Health Services Research* 16 (2016), p. 13. DOI: 10.1186/s12913-015-1243-z.

[20] A. G. Leeftink, I. M.H. Vliegen, and E. W. Hans. "Stochastic integer programming for multi-disciplinary outpatient clinic planning". In: *Health care management science* 22 (1 Mar. 2019), pp. 53–67. ISSN: 1572-9389. DOI: 10.1007/S10729-017-9422-6. URL: https://pubmed.ncbi.nlm.nih.gov/29124483/.

[21] Haichao Liu, Yang Wang, and Jin-Kao Hao. "Solving the patient admission scheduling problem using constraint aggregation". In: *European Journal of Operational Research* 316.1 (2024), pp. 85–99. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2024.02.009. URL: https://www.sciencedirect.com/science/article/pii/S0377221724001012.

[22] V. Lotfi and E. Torres. "Improving an outpatient clinic utilization using decision analysis-based patient scheduling". In: *Socio-Economic Planning Sciences* 48.2 (June 2014), pp. 115–126. DOI: 10.1016/j.seps.2013.12.002.

[23] R. Masson, F. Lehuédé, and O. Péton. "An adaptive large neighborhood search for the pickup and delivery problem with transfers". In: *Transportation Science* 47.3 (2013), pp. 344–355. DOI: https://doi.org/10.1287/trsc.1120.0432.

[24] F. Mochón et al. "Machine-Learning-Based No Show Prediction in Outpatient Visits". In: *International Journal of Interactive Multimedia and Artificial Intelligence* 4.7 (2018), pp. 29–35. DOI: 10.9781/ijimai.2017.03.004.

[25] C. G. Moore, P. Wilson-Witherspoon, and J. C. Probst. “Time and money: effects of no-shows at a family practice residency clinic”. In: *Family Medicine-Kansas City-* 33.7 (2001), pp. 522–527.

[26] Nederlandse Zorgautoriteit. *Stand van de zorg 2024*. Tech. rep. Rapport uitgebracht op 8 oktober 2024. Nederlandse Zorgautoriteit, Oct. 2024. URL: `https://www.nza.nl/publicaties/stand-van-de-zorg`.

[27] Michela Samorani and Larry R. LaGanga. “Outpatient appointment scheduling given individual patient preferences and probabilistic service times”. In: *European Journal of Operational Research* 245.2 (2015), pp. 548–560. DOI: `10.1016/j.ejor.2015.03.013`.

[28] Katja Schimmelpfeng, Stefan Helber, and Steffen Kasper. “Decision support for rehabilitation hospital scheduling”. In: *OR Spectrum* 34 (2 Apr. 2012), pp. 461–489. ISSN: 01716468. DOI: `10.1007/S00291-011-0273-0/METRICS`. URL: `https://link.springer.com/article/10.1007/s00291-011-0273-0`.

[29] A. J. T. Schneider, J. Theresia van Essen, B. Carlier, et al. “Scheduling surgery groups considering multiple downstream resources”. In: *European Journal of Operational Research* 282 (2020), pp. 741–752. DOI: `10.1016/j.ejor.2019.09.029`.

[30] S. A. Shah et al. “Impact of COVID-19 pandemic on elective care backlog trends, recovery efforts, and capacity needs to address backlogs in Scotland (2013–2023): a descriptive analysis and modelling study”. In: *The Lancet Regional Health–Europe* (2025). DOI: `https://doi.org/10.1016/j.lanepe.2024.101188`.

[31] M. Shuang, S. Chen, and X. Cai. “A Two-stage Stochastic Programming Model for Outpatient Appointment Scheduling”. In: *IEEE International Conference on Industrial Engineering and Engineering Management* (Dec. 2019), pp. 79–83. DOI: `10.1109/IEEM44572.2019.8978589`.

[32] Aykut Melih Turhan and Bilge Bilgen. “A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem”. In: *Computers  Industrial Engineering* 145 (2020), p. 106531. ISSN: 0360-8352. DOI: `https://doi.org/10.1016/j.cie.2020.106531`. URL: `https://www.sciencedirect.com/science/article/pii/S0360835220302655`.

[33] P. Van der Maas. “Health care in the Netherlands”. In: *Volksgezondheid en Gezondheidszorg* (1999), pp. 102–105.

[34] Lien Wang et al. “On the use of partitioning for scheduling of surgeries in the inpatient surgical department”. In: *Health Care Management Science* 25.4 (2022), pp. 526–550. DOI: `10.1007/s10729-022-09598-0`.

[35] T. I. Wickert et al. “An integer programming approach for the physician rostering problem”. In: *Annals of Operations Research* 302.2 (2021), pp. 363–390. DOI: `10.1007/s10479-020-03552-5`.

[36] Xiuli Wu, Xianli Shen, and Linjuan Zhang. “Solving the planning and scheduling problem simultaneously in a hospital with a bi-layer discrete particle swarm optimization”. In: *Mathematical Biosciences and Engineering* 16.2 (2019), pp. 831–861. DOI: `10.3934/mbe.2019039`.

[37] F. Ü. Yüksektepe. “MILP Based Hyper-Box Enclosure Approach to Multi-Class Data Classification”. PhD thesis. Koç University, 2009. DOI: `https://doi.org/10.1142/9789812772954_0002?urlappend=%3Futm_source%3Dresearchgate`.

[38] Zorginstituut Nederland. *Jaarverslag 2024 Zorginstituut Nederland.* Tech. rep. Definitief verslag, uitgebracht aan de minister van Volksgezondheid, Welzijn en Sport op 12 maart 2025. Zorginstituut Nederland, 2025. URL: `https://www.zorginstituutnederland.nl/publicaties/jaarverslag/2025/03/12/jaarverslag-2024-zorginstituut-nederland`.