NTNU
Norwegian University of
Science and Technology

Master's thesis

2019

Sander Sebastian Henschien Coates, Vegard Pedersen

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Economics and Management
Department of Industrial Economics and Technology
Management

Sander Sebastian Henschien Coates
Vegard Pedersen

Solving the Healthcare Rostering Problem with Branch and Price

A New Decomposition and Speed-Up Techniques

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Solving the Healthcare Rostering Problem with Branch and Price

A New Decomposition and Speed-Up Techniques

Sander Sebastian Henschien Coates
Vegard Pedersen

Industrial Economics and Technology Management

Submission date:	June 2019
Supervisor:	Troels Martin Range
Co-supervisor:	Kjartan Kastet Klyve

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem Description

The purpose of this thesis is to improve optimization-based solution methods for the Healthcare Rostering Problem (HRP). The HRP is concerned with generating staff schedules to cover the demand for labor in healthcare institutions. Within rostering, healthcare rostering is particularly complex due to fluctuating demand, 24/7 staffing and an increasing focus on employee preferences. Solutions to the HRP are expected to be in compliance with all rules governing the healthcare rostering process and balance the two goals of minimal costs and maximal employee satisfaction.

Preface

This thesis concludes our Master of Science in Industrial Economics and Technology Management at the Norwegian University of Science and Technology (NTNU). It was written during the spring of 2019 as the final work of our specialization within Managerial Economics and Operations Research. The thesis is a continuation of our preparatory research project conducted during the fall of 2018.

We would like to express our sincere gratitude towards our supervisors Troels Martin Range and Kjartan Kastet Klyve for invaluable guidance and support during the work with the thesis. The willingness to engage in our work and insightful feedback provided has been essential in developing the final result. We would also like to extend our thanks to the Department of Industrial Economics and Technology Management staff for their shared enthusiasm in operations research courses and their guidance towards writing our thesis. Finally, we would like to share our appreciation for the guidance and contacts supplied by the PwC Scholarship working group.

Sander Sebastian Henschien Coates and Vegard Pedersen

Trondheim, June 2019

Summary

Rostering refers to the process of generating staff schedules, or *rosters*, and has received wide attention within operations research. The healthcare rostering problem (HRP) is addressed in this thesis and a new approach that contributes to closing the gap between operations research theory and real world practice is proposed. The HRP considered in the thesis is inspired by rostering in Nordic healthcare institutions.

Through a literature review, a need for improved solution methods in rostering is identified. Healthcare rostering is highlighted as an especially interesting domain due to the high complexity and large institutions considered. Some of the particulars in healthcare causing complexity are fluctuating demand, 24/7 staffing and an increasing focus on employee preferences. The discoveries from the literature review motivate the wide problem definition presented in the thesis and the work on developing improved solution methods.

The HRP defined in the thesis includes several rules that must be followed, and the inclusion of aspects like employee preferences for working patterns stretching over more days distinguish the thesis from problem definitions often seen in research. The HRP balances two goals: minimizing costs and maximizing employee satisfaction. A mathematical model of the HRP is formulated with Dantzig-Wolfe decomposition to be solved with a branch and price algorithm. The restricted master problem (RMP) used in column generation is formulated as a mixed integer program and the employee specific sub problems (SPs) as shortest path problems with resource constraints (SPPRCs). To model a wider variety of constraints than is common in the literature without making the SPs too complex, some of the employee specific constraints are handled in the RMP instead of in the SPs. This allocation of constraints differentiates the model in this thesis from the common modeling in literature.

The proposed model is solved with branch and price. The LP-relaxed RMP is solved with the simplex method while the SPPRC-formulated SPs are solved with dynamic programming using a labelling algorithm. Inspired by the success of speed-up techniques in research, and motivated by the complexity of the proposed model, several methods to improve computation time are proposed and evaluated. A modified set of commonly used benchmark test instances with variety in staff size are used to evaluate the performance of the model.

Without the implementation of speed-up techniques, the solution method suggested yields feasible rosters for instances with up to 30 employees within eight hours. However, the rosters are not optimal and solutions that are more than ten percent better in terms of objective value may exist for all instances. This motivates the implementation of speed-ups. Increasing the number of solutions retrieved from each SP in each iteration, limiting the number of extended labels from each node when applying dynamic programming, stopping column generation prior to reaching optimality, column elimination and handling more constraints in the SPs are examples of successful speed-ups tested. Less successful is the implementation of partial column generation and an aggressive branching strategy. After applying the best combination of speed-ups found, better rosters are produced for most instances, and optimality gaps are improved to less than ten percent in several cases. The model proposed scales well in the number of employees, and high quality rosters are obtained for instances with up to 30 employees.

Sammendrag

Timeplanlegging har mottatt mye oppmerksomhet i forskning innen operasjonsanalyse. *Rostering* refererer til generering av timeplaner, eller *rosters*. Denne masteroppgaven omhandler timeplanlegging i helsesektoren gjennom å ta for seg *The Healthcare Rostering Problem* (HRP). En ny tilnærming som bidrar til å lukke gapet mellom teori i operasjonsanalyse og reell praksis foreslås. HRPet som undersøkes i denne oppgaven er inspirert av timeplanlegging ved nordiske helseinstitusjoner.

Gjennom et litteraturstudie har behovet for forbedrede løsningsmetoder i timeplanlegging blitt identifisert. Timeplanlegging i helsesektoren er understreket som spesielt interessant grunnet høy kompleksitet og store institusjoner. Blant karakteristikkene som bidrar til kompleksitet i helsesektoren er varierende etterspørsel, 24/7 bemanning og et økende fokus på ansattes preferanser. Funnene fra litteraturstudiet motiverer den vide problembeskrivelsen som presenteres i oppgaven og arbeidet med å utvikle forbedrede løsningsmetoder.

HRPet som er definert i oppgaven inkluderer et flertall av regler som må følges, og inkluderingen av aspekter som ansattes preferanser for arbeidsmønster som strekker seg over flere dager skiller denne oppgaven fra problembeskrivelsene som typisk finnes i forskning. HRPet balanserer to mål: minimering av kostnader og maksimering av ansattes tilfredshet. En matematisk modell av HRPet er formulert med Dantzig-Wolfe dekomponering for å løses med *branch and price*. RMPet som brukes i kolonnegenerering er formulert som et MIP og de ansatte-spesifikke sub-problemene (SPene) som korteste veiproblemer med ressursbegrensninger (SPPRCer). For å modellere et større sett av begrensninger enn det som er vanlig i litteraturen uten å gjøre SPene for komplekse, er noen av de ansatte-spesifikke begrensningene håndtert i RMPet i stedet for i SPene. Denne allokeringen av begrensninger skiller modellen i denne oppgaven fra vanlig modellering i litteraturen.

Den foreslåtte modellen er løst med *branch and price*. Det LP-relakserte RMPet løses med simplex-metoden mens de SPPRC-formulerte SPene løses med dynamisk programmering ved bruk av en *labelling*-algoritme. Inspirert av suksessen til *speed-up*-teknikker i forskningen, og motivert av kompleksiteten i den foreslåtte modellen, er et flertall av metoder for å forbedre beregningstiden foreslått og evaluert. Et modifisert sett av vanlig brukte referansetestinstanser med varierende antall ansatte er brukt til å evaluere modellen.

Uten implementasjonen av *speed-up*-teknikker oppnår løsningsmetoden mulige timeplaner for instanser med opp til 30 ansatte innen åtte timer. Timeplanene er imidlertid ikke optimale og løsninger som er mer enn ti prosent bedre målt i objektivverdi kan muligens finnes for alle instanser. Dette motiverer implementasjonen av *speed-ups*. Økning av antall løsninger som hentes fra hvert SP i hver iterasjon, begrensning av antall forlengede *labels* fra hver node i dynamisk programmering, stopping av kolonnegenerering før optimal løsning er funnet, fjerning av kolonner og håndtering av flere begrensninger i SPene er eksempler på vellykkede *speed-ups* som er testet. Mindre vellykket er implementasjonen av delvis kolonnegenerering og en aggressiv *branching*-strategi. Ved å bruke den beste konfigurasjonen av *speed-ups* genereres bedre timeplaner for de fleste instansene, og optimalitetsgapene forbedres til mindre enn ti prosent i flere tilfeller. Den foreslåtte modellen er godt skalerbar med antall ansatte og timeplaner av høy kvalitet oppnås for instanser med opptil 30 ansatte.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

CG	Column Generation
CP	Constraint Programming
DWD	Dantzig-Wolfe Decomposition
HRP	Healthcare Rostering Problem
ILP	Integer Linear Programming
INRC	International Nurse Rostering Competition
LP	Linear Programming
MIP	Mixed Integer Programming
MP	Master Problem
NRP	Nurse Rostering Problem
REF	Resource Extension Function
RMP	Restricted Master Problem
SP	Sub Problem
SPP	Shortest Path Problem
SPPRC	Shortest Path Problem with Resource Constraints

Chapter 1

Introduction

Rostering refers to the process of generating staff schedules, or *rosters*, for a group of employees who collectively work to satisfy the demand for some goods or services (Ernst et al., 2004; Dohn and Mason, 2013). Rostering problems involve generating rosters that comply certain rules while minimizing costs and/or maximizing employee satisfaction. In this thesis, rostering within healthcare is approached from an operations research perspective to develop new methods for solving rostering instances of real-world complexity. A model is proposed to generate staff schedules in situations that resemble rostering instances at Nordic healthcare institutions. Typical healthcare rostering rules and employee preferences are considered. The model is solved using branch and price and the effect of several speed-up techniques are analyzed.

Rostering has been subject to operations research for decades. Ernst et al. (2004) trace the origin to Edie (1954) who touches into the field when working on traffic delays at toll booths. Since then, rostering methods have been researched for several applications, such as airlines, railways, healthcare systems and call centers (Ernst et al., 2004; Van den Bergh et al., 2013), and continues to be of interest in current research (Asta et al., 2016; Smet et al., 2016; Gérard et al., 2016; Rahimian et al., 2017; Václavík et al., 2018).

Van den Bergh et al. (2013) review 291 articles on personnel scheduling and staffing and find that *nurse* and *other healthcare scheduling* are the most common application areas, accounting for 85 of the articles reviewed[1]. The main explanation for the special attention drawn to healthcare and more specifically nurse rostering is the added complexity associated with fluctuating demand, 24/7 staffing and focus on employee preferences at large institutions. Cheang et al. (2003) substantiate this through mentioning of the time-consumption associated with manual scheduling, the need for staffing at all hours and the

[1]Van den Bergh et al. (2013) distinguish between application areas *general*, *nurse*, *other healthcare*, *protection/emergency*, *call center*, *other*, *airline*, *railway*, *bus*, *manufacturing*, *retail* and *military* in personnel scheduling research.

diversity in staff requests as the main reasons for special academic interest in the nurse rostering problem (NRP). Bard and Purnomo (2005) also refer to 24/7 staffing and employee preferences as complicating factors in nurse rostering.

Discussions with healthcare professionals and operations researchers with experience reveal that many healthcare institutions set up rosters manually and end up with sub-optimal rosters. A survey by Jensen et al. (2008) showed that Danish hospitals on average spent between 24 and 29 working minutes per week per employee on rostering. For a hospital like St. Olavs hospital in Trondheim, Norway, employing over 8 000 FTEs[2], this would represent over 95 FTEs just for rostering purposes each year. Using the average Norwegian cost per FTE from 2016[3], the total annual rostering cost would be more than NOK 72 million. This cost does not include the cost of sub-optimal rosters from the manual planning process that can lead to, e.g., increased wage costs and dissatisfied employees. The total cost saving potential from developing better rostering tools is substantial.

One of the major shortcomings of research on personnel scheduling in general is insufficient implementation (Van den Bergh et al., 2013), the restrictive problem setting of research being blamed. Ernst et al. (2004) also identify generalization of models and methods as an important area for future research. Considering the application area of nurse rostering, Burke et al. (2004) review the research of the last 40 years and make the claim that current research is too simplistic and not suitable for real world implementation. This is in line with the more recent conclusions drawn by Kellogg and Walczak (2007) and Van den Bergh et al. (2013). Only a few papers describe the implementation of the mathematical algorithms, and Burke et al. find just a hand-full of approaches that are applied at hospitals.

This thesis addresses the gap between operations research personnel scheduling theory and healthcare rostering practice. The model proposed takes into account typical rules faced in healthcare rostering as well as employee preferences regarding assignments of single shifts and shift sequences. The rules considered are inspired by current research, interviews with Norwegian healthcare professionals and the thesis supervisors with experience from Nordic healthcare rostering.

Several solution methods to solve healthcare rostering problems are suggested in operations research literature. Branch and bound, decomposition methods, artificial intelligence methods, heuristics and metaheuristics are common choices (Burke et al., 2004; Van den Bergh et al., 2013). A decomposed model able to handle a wide variety of constraints is proposed in this thesis. To solve the model, a branch and price method is suggested where the decomposed model is solved with column generation (CG) utilizing both mixed integer programming (MIP) and dynamic programming. Readers who find themselves unfamiliar with CG and/or branch and price are recommended to read Lübbecke and Desrosiers (2005) and/or Barnhart et al. (1998). A more extensive exploration of CG is provided by Desaulniers et al. (2006).

In the decomposed model, all overall rostering constraints and some employee specific constraints are handled in the restricted master problem (RMP), which is solved by MIP.

[2]full-time equivalents, https://stolav.no/om-oss/nokkeltall-for-st-olavs-hospital

[3]https://www.ssb.no/arbeid-og-lonn/statistikker/arbkost

All other employee specific constraints are handled in employee specific sub problems (SPs) solved as shortest path problems with resource constraints (SPPRCs) with a dynamic programming labelling algorithm. The allocation of constraints between the RMP and the SPs was the focus of our preparatory research project (Coates and Pedersen, 2018). The inclusion of employee specific constraints in the RMP differentiates the solution method in this thesis from the common decomposition applied in research, and enables efficient modeling of a wider specter of constraint types.

Central to the discussion in the thesis is the evaluation of several speed-up techniques proposed to reduce the computation time required to solve the suggested model. The effects of a CG stop criterion, returning several columns from each SP, limited label extension, partial CG, handling more constraints in the SPs, multiple variable aggressive branching and column elimination are evaluated on 20 constructed test instances of varying complexity, designed to reflect Nordic healthcare rostering conditions. The test instances are inspired by common benchmark instances found in the healthcare rostering literature and interviews with healthcare professionals.

Implementation represents a substantial part of the underlying work in the thesis, and the model was programmed using the open source programming language Python. All code is available online[4] for the reader to explore.

The thesis is structured into seven chapters. A review of relevant literature is presented along with the contributions of the thesis in Chapter 2. In Chapter 3, the healthcare rostering problem (HRP) is described in detail. Based on the problem description, the mathematical model is presented in Chapter 4. In Chapter 5, the solution methods used to solve the mathematical model are presented, including the speed-up techniques to be evaluated. Chapter 6 begins with a description of the implementation of the model, before the test instances are described. Thereafter, the computational results from solving the model and the effects of the suggested speed-up techniques are evaluated. The concluding remarks are presented in Chapter 7, and areas for future research are proposed.

[4]https://github.com/sandercoates/HRP

Chapter 2

Literature Review

In this chapter, a review of relevant literature to the thesis is presented. Research on staff rostering and healthcare optimization is discussed, and a theoretical context for the work on improving solution methods for the healthcare rostering problem is provided.

As discussed in Chapter 1, rostering problems have received wide attention within operations research. Because the thesis is written from an operations research perspective, the literature search that formed the basis for this chapter was focused on papers from this field of study. Results from searches on *Google Scholar* and *Oria* with relevant keywords[1] together with suggestions from the thesis supervisors with academic experience from healthcare rostering produced an initial collection of papers. Among these were two extensive literature reviews on personnel scheduling by Ernst et al. (2004) and Van den Bergh et al. (2013) that helped expand the literature search. During the process of reviewing the literature, the search was widened to also include papers that are cited by, and citing, the papers found.

The chapter is structured into three sections. First, the literature review is focused in Section 2.1 by identifying the domains of healthcare planning and personnel scheduling that are relevant to the thesis. The state of relevant research is then discussed. Second, a description of solution methods that have been applied to relevant problems within operations research is given in Section 2.2. Special attention is given to methods based on branch and price because of the popularity in research and particular relevance to the thesis. Third, in Section 2.3, an overview of the literature is given and the contributions of the thesis are discussed. Literature regarding general solution methods used in the thesis (e.g. dynamic programming) is not discussed in this chapter, but is rather presented along with the description of the relevant solution methods in subsequent chapters. However, examples of applications of solution methods, such as the early use of branch and price in nurse rostering by Jaumard et al. (1998), are included in this chapter.

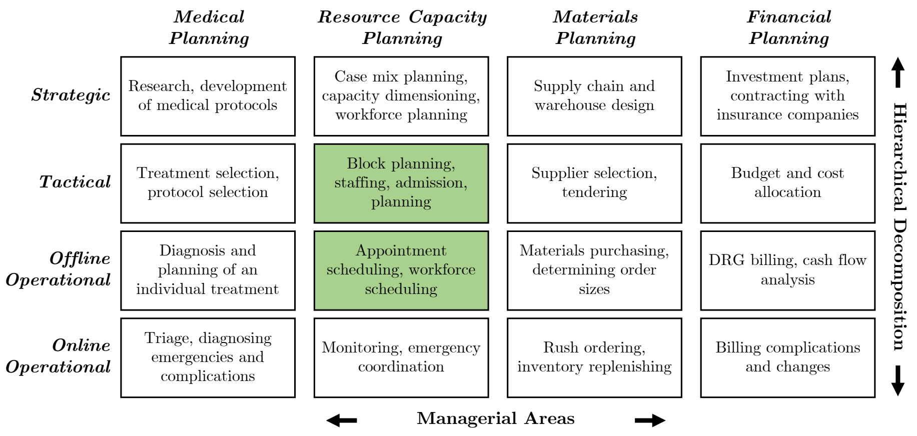
[1]E.g. *nurse rostering*, *branch and price for rostering* and *rostering heuristics*.

2.1 Focusing the Literature Review

Healthcare planning and rostering are large domains. In this section, the parts of the literature that are important to the thesis are identified. Healthcare planning in general is discussed in Section 2.1.1 to identify the parts of healthcare planning that are addressed in this thesis. In Section 2.1.2, the rostering process is described in more detail, and the areas of rostering considered in the thesis are presented. The state of the identified field of research is discussed, bridging a further discussion of solution methods in Section 2.2.

2.1.1 The Healthcare Planning Process

To understand the healthcare planning process, the framework developed by Hans et al. (2012) is useful. It divides healthcare operations into four managerial areas and four hierarchical levels of control, as shown in Figure 2.1. The four key managerial areas identified are *medical planning*, *resource capacity planning*, *materials planning* and *financial planning*. Rostering is part of the resource capacity planning. The first three levels (*strategic*,



	Medical Planning	*Resource Capacity Planning*	*Materials Planning*	*Financial Planning*
Strategic	Research, development of medical protocols	Case mix planning, capacity dimensioning, workforce planning	Supply chain and warehouse design	Investment plans, contracting with insurance companies
Tactical	Treatment selection, protocol selection	Block planning, staffing, admission, planning	Supplier selection, tendering	Budget and cost allocation
Offline Operational	Diagnosis and planning of an individual treatment	Appointment scheduling, workforce scheduling	Materials purchasing, determining order sizes	DRG billing, cash flow analysis
Online Operational	Triage, diagnosing emergencies and complications	Monitoring, emergency coordination	Rush ordering, inventory replenishing	Billing complications and changes

Figure 2.1: Classification framework for healthcare planning and control presented by Hans et al. (2012) The focus of this thesis is highlighted in green.

tactical and *offline operational planning*) of the four hierarchical levels address decision making before events occur. The fourth (*online operational planning*) involves reactive decisions during operation. Depending on the institution, the rostering process spans over some or all of the first three levels. According to Ernst et al. (2004), the first part of the rostering process involves determining the number of staff required to meet demand, and this is part of the strategic phase. This is however not considered in this thesis as the workforce size in the problem considered is assumed known. Following the strategic phase, individuals are allocated to shifts as to meet required demand at different times, which can be part of both the tactical and offline operational planning phase. The fourth level includes emergency coordination and differs largely from regular rostering. The focus of this thesis is on the middle two hierarchical levels highlighted with green in Figure 2.1. By adjusting the length of the planning period and the complexity of day to day schedules, the methods developed in this thesis can handle both levels of planning. Hans et al. (2012) argue that many organizations fail to consider how the different hierarchical levels influence each other, and thus there is a need for tools that improve planning across hierarchical levels. By improving methods for detailed planning over longer time horizons, one can better integrate the hierarchical levels. Developing better solution methods for rostering is an important part of this.

2.1.2 The Rostering Process

An early classification of personnel scheduling problems was proposed by Baker (1976), who argued that there are three main groups of problems: *shift scheduling*, *days off scheduling* and *tour scheduling*. Shift scheduling deals with the assignment of staff to different shifts on a day. Days off scheduling determines which employees work when on a higher level, and is necessary when the operating week of the workplace does not match the employees' working week. Hospitals require staffing seven days a week, making days off scheduling important in healthcare rostering. Tour scheduling is a combination of shift scheduling and days off scheduling, where employees are assigned to specific shift types on specific days, constituting a *tour*. In this thesis, the term *roster line* is used as an equivalent term to tour, describing a sequence of shifts over some planning period.

Ernst et al. (2004) present a more granular classification, suggesting that the rostering process consists of six modules: (i) *demand modeling*, (ii) *days off scheduling*, (iii) *shift scheduling*, (iv) *line of work construction*, (v) *task assignment* and (vi) *staff assignment*. Of these, *demand modeling* and *task assignment* are not discussed in this thesis. To narrow the scope, demand is assumed known and is input to the problem considered. Task assignment is typically done on a more detailed level than the shift assignment in the thesis. For cases where tasks span over entire shifts, task assignment can be handled through shift assignment by duplicate shifts that represent different tasks. This is, however, not considered further in this thesis. The other modules presented by Ernst et al. are highly relevant for the thesis. It should be noted that the *staff assignment* module consisting of assigning individual staff to lines of work is assumed integrated in the *days off scheduling* and *line of work construction*. This allows accommodating individual preferences during the construction of lines of work and is key to create high quality roster lines.

Another distinction between rostering problems is identified by Burke et al. (2004) as the two alternatives *cyclical* (also referred to as *fixed*) and *non-cyclical* (alternatively *flexible*) scheduling. In *cyclical* scheduling, each employee follows a repetitive sequence of assignments. Burke et al. find that cyclical schedules, although easier to generate, are only applicable in a few rare cases of nurse rostering, and are not flexible enough to take into account employee preferences. Furthermore, Burke et al. observe that employees seem to prefer "ad hoc" schedules since they address fluctuating hospital demands together with the flexibility needed to accommodate individual preferences. Vanden Berghe (2002) makes a similar claim. Although cyclical schedules present good solutions in many applications (e.g. in industrial situations with periodic morning-day-night cycles), a non-cyclical approach is chosen in this thesis to account for the complexity in healthcare rostering. Consequentially, the entire planning period is considered without cycle simplifications.

As discussed in Chapter 1, there is special attention drawn to rostering within healthcare due to high complexity and large institutions. This results in highly constrained problems that are hard to solve (Ernst et al., 2004). To get an overview, commonly occurring constraints based on Cheang et al. (2003) are listed in Table 2.1. Among the listed constraints, this thesis considers all but combinations of employees that must or must not work together at the same time (11).

Table 2.1: Commonly occurring constraints suggested by Cheang et al. (2003). The wording has been modified slightly to reflect other healthcare staff than just nurses. The type of constraint is indicated in parenthesis: min – minimum; max – maximum; consec – consecutive; req – requirement.

1.	Workload (min/max)	9.	Holidays and vacations (predictable)
2.	Consec days working same shift (min/max)	10.	Working in weekends
3.	Consec days working (min/max)	11.	Groups of employees that work together
4.	Employee skill level and categories	12.	Shift patterns
5.	Employee preferences or requirements	13.	Historical record (e.g. previous assignments)
6.	Employee days off (min/max/consec)	14.	Assignment (e.g. shift assigned once)
7.	Free time between working shifts (min)	15.	Staff demand for different types
8.	Shift type(s) assignments (max/req)	16.	Other requirements

Vanden Berghe (2002) too provides a comprehensive overview of the constraints typically considered in healthcare scheduling, presented from a Belgian point of view. This overview closely resembles the list of Cheang et al. So does the generalization of typical nurse rostering constraints put forth by Smet et al. (2014). Constraint types presented in the mentioned papers have inspired the rules modeled in this thesis, described further in Chapter 3.

Due to the complexity of the rostering problems faced in healthcare, there is a need for advanced solution methods to solve real world problems. As mentioned in Chapter 1, a restrictive problem setting not taking into account all constraints present in real world applications of rostering is often seen in research (Van den Bergh et al., 2013; Burke et al., 2004). This is substantiated through the papers reviewed in this thesis, and no model has been found to take into account the same set of constraints as the one in this thesis. However, there are several suggested approaches found to solve models of high complexity. A discussion of healthcare rostering solution methods follows in Section 2.2.

2.2 Solution Methods

Several optimization-based solution methods have been proposed in operations research to solve rostering problems. Although the focus of the thesis is on healthcare rostering, it is of interest to consider methods applied to adjacent domains of rostering such as airline crew as well. In this section, a broad selection of papers utilizing optimization-based solution methods are discussed.

In their review of personnel scheduling, Van den Bergh et al. (2013) find that set-covering mathematical formulations are most common and claim that these allow the researcher to customize constraints based on particular needs. Other common solution methods include constraint programming and metaheuristics. The set-covering problem can be solved using a standard optimization solver, usually utilizing a branch and bound algorithm combined with a selection of heuristics. However, set-covering formulations tend to grow large in terms of variables. To these problems, a heuristic and/or decomposition method, such as Dantzig-Wolfe decomposition (DWD) with column generation, is commonly applied in the literature. A selection of examples is discussed in the following. The branch and price algorithm, in which decomposed problems are solved using CG in a branch and bound tree, addresses the issue of many variables by only considering a restricted set of variables generated in an iterative process[2]. Therefore, papers utilizing branch and price are discussed first, before a wider set of solution methods are presented.

Branch and Price
An early application of branch and price to the nurse rostering problem is presented by Jaumard et al. (1998). In their paper, Jaumard et al. give an overview of the nurse rostering problem and propose a branch and bound framework where each node is solved with CG. The CG restricted master problem is solved as a mixed integer programming problem and the sub problems as shortest path problems with resource constraints. Jaumard et al. claim that their proposal is the first exact solution approach presented for a flexible realistic model of the NRP. The proposed approach is, however, only tested in a preliminary manner with a limited number of constraints considered.

Since Jaumard et al. (1998), branch and price has received substantial attention in rostering research. For example, Dohn and Mason (2013) propose a branch and price algorithm for the generalized staff rostering problem. Unlike Jaumard et al. (1998) who apply an inequality demand coverage constraint and only consider under-coverage, Dohn and Mason apply an equality demand coverage constraint, thereby restraining both under- and over-coverage. A similar modeling of demand coverage is applied in this thesis, reasoned by the wish to control over-coverage because it can lead to inefficiencies and patient discomfort, further discussed in Chapter 3. Dohn and Mason claim to deal with all constraints listed by Cheang et al. (2003), including individual preferences. There is, however, no mention of patterns (i.e. preferences on shift sequences over several days) which is a significant feature of the model proposed in this thesis.

The CG pricing problem is solved by Dohn and Mason (2013) as a three-stage, nested shortest path problem with resource constraints. The method is inspired by a two-stage

[2]A theoretical introduction to branch and price is provided in Chapter 5.

approach by Mason and Smith (1998). Dohn and Mason split the first stage of Mason and Smith into two stages. In the first and lowest level stage, Dohn and Mason combine shifts into on stretches, followed by the pairing of on stretches with off stretches to form work stretches in the second and higher level stage. In the third and highest level stage, roster lines are generated by lining up work stretches in sequence. The algorithm is tested on three nurse rostering instances, and short computation time is highlighted as the main benefit. High-quality solutions for the four-week long instances are produced in less than 15 minutes.

Maenhout and Vanhoucke (2010) solve the nurse rostering problem with branch and price in a similar fashion to Dohn and Mason (2013). They also model demand coverage as an equality constraint. After arguing that branch and price is popular as an exact solution method for the NRP, Maenhout and Vanhoucke underline the computationally expensive CG SP as a major limiting process of the branch and price algorithm. Where Dohn and Mason alleviate SP computation time with a three-stage nested SPPRC approach, Maenhout and Vanhoucke apply a selection of techniques such as partial CG, SP upper bound pruning and a two-phase SPPRC solution approach. The effects of different speed-up techniques, including the aforementioned SP speed-ups, are reported after testing on artificially generated instances. All speed-ups tested are found to reduce computation time.

In a more recent paper, Václavík et al. (2018) propose a machine learning approach to speed up the pricing problem in a general branch and price algorithm. They confirm the claim of Maenhout and Vanhoucke (2010) that the pricing problem is the most computationally expensive part of the branch and price algorithm. Václavík et al. consider the NRP, as well as scheduling of time-division multiplexing for multi-core platforms, and find that more than 90 percent of computation time is spent solving the pricing problem. Because solving the pricing problem of branch and price is often repetitive and involves minor changes, knowledge from previous solutions is used to predict tighter exact upper bounds in the current pricing problem. Václavík et al. find an average reduction of 40 percent computation time for the nurse rostering problem when testing on the benchmark instances presented by Burke and Curtois (2014).

In a branch and price framework, the choice of decomposition strategy to enable CG is a key decision. Most common is the decomposition on employees, i.e. solving individual employee SPs. Beliën and Demeulemeester (2006), on the other hand, apply an activity-based decomposition to schedule hospital trainees. The following year, a comparison between activities-based and employee-based decomposition was made (Beliën and Demeulemeester, 2007). Activity-based decomposition is found to be superior in terms of computational performance when evaluating performance on the hospital trainee test set from Beliën and Demeulemeester (2006). However, Beliën and Demeulemeester argue that employee-based decomposition may be used in a wider variety of problems because constraints are typically employee specific, rather than activity specific. They find that for their test set, activity schedules automatically satisfy the staff constraints, thus promoting activity-based decomposition's superiority. With a large degree of employee specific constraints in this thesis, the model proposed is decomposed based on staff.

Gamache et al. (1999) review rostering problems for aircrew that are somewhat different in nature from healthcare. However, they discuss interesting solution methods within a CG framework also relevant for healthcare problems. As for many of the previously discussed papers, the SPs are solved as SPPRCs. Like Maenhout and Vanhoucke (2010), Gamache et al. suggest the use of partial CG with promising results. They also stop each LP problem before optimality is reached to reduce computation time in each node, a method also tested in this thesis.

The air transportation industry is also the focus of Lusby et al. (2012), only this time, ground crew rostering rather than aircrew staff is considered. They suggest an interesting approach to deal with long planning horizons of six months. A CG algorithm is applied to smaller overlapping time blocks that joined together make up the full planning period. The rostering problems of the time blocks are solved sequentially, with consistency between time block solutions enforced by shift assignment fixing in overlapping regions. The SP is solved as an SPPRC, and resembles the SP solution approach chosen in this thesis. Lusby et al. find that the algorithm performs very well on artificial test instances imitating real life instances.

Returning to the domain of healthcare rostering, Bard and Purnomo (2005) propose a CG method where a double swapping heuristic is used to generate the columns, as opposed to exact methods seen in the previous examples. They present a model that combines the minimization of violation of employee preferences and cost of hiring external personnel, and are able to handle many types of employee-specific rules using both hard and soft constraints. However, their model is dependent on a quite good input schedule as basis for the swapping heuristic, and is thus in some cases not applicable. Testing on instances inspired by a large U.S. hospital yields computation times of only few minutes, and Bard and Purnomo claim full implementation at hospitals is feasible.

Like Maenhout and Vanhoucke (2010) and Václavík et al. (2018), Burke and Curtois (2011) confirm that solving the SPs is the most computationally expensive part of branch and price for the NRP. Burke and Curtois apply branch and price to a set of benchmark instances that are also used in this thesis. They solve the SPs with dynamic programming, and implement a version of limited label extension which is also tested in this thesis with good results. Lübbecke (2005) has a similar problem formulation and uses dual variable fathoming to speed up the SP solution time. During the solving of SPs, the optimality search is pruned if no possible negative reduced cost solution can be found.

The papers presented so far demonstrate the large interest in branch and price for solving rostering problems in healthcare. The success by many authors with branch and price motivates the choice of solution method in this thesis. Many interesting methods of reducing computation time have been presented. In the following, a broader selection of papers that utilize other methods than branch and price to solve optimization-based rostering is presented. These papers provide a more complete overview of the current available methods and give a thorough context to the thesis.

Other Exact Methods
Among other exact methods than branch and price, integer and MIP are popular choices (Van den Bergh et al., 2013). Rönnberg and Larsson (2010) study the potential for automatically generating feasible schedules based on individual schedules proposed by nurses,

and present an integer linear programming (ILP) model solved in CPLEX with branch and bound. They are able to create feasible schedules, but not to solve the model to optimality. Moz and Pato (2004) approach the problem of rescheduling nurses in the case of employee absence by ILP models and investigate methods to deal with the challenge of large models. After testing an initial multi-commodity flow formulation on a Portuguese public hospital, a second model with node aggregation is developed. Moz and Pato conclude that although the models show promising progress, the rescheduling problems have not yet been satisfactory solved. These examples corroborate the need for improved solution methods.

Valouxis et al. (2012) divide the nurse rostering problem into two smaller problems solved in succession. In the first problem, an ILP model sets the workload of nurses in a week without the allocation of shifts. In the second problem, the shifts are assigned. Local searches are added to consider combinations of employee schedules with good results. Valouxis et al. (2012) obtain the best results for 42 out of 60 International Nurse Rostering Competition instances presented by Haspeslagh et al. (2014).

Among other exact methods, Ovchinnikov and Milner (2008) assign medical residents at a US college to rotation schedules by using a spreadsheet model to solve relatively small scope problems. They argue that spreadsheet models are highly accessible and may ease the transition from manual scheduling to automated scheduling. De Grano et al. (2009) combine a first phase auction where employees bid for their preferences and a second phase MIP model that assigns shifts considering the bids while satisfying rules. The approach is tested at a US hospital with encouraging results. Auctioning preferences is an interesting alternative to the method in this thesis where every preference is weighted with a cost in a single objective optimization problem, rather than a two-phase auction and optimization process. Trilling et al. (2006) compare an ILP and a constraint programming (CP) model for the NRP. Testing at a French public hospital, the ILP model is found superior to the CP model in terms of computation time. However, Trilling et al. find that the solutions offered by the CP model distribute shifts more fairly than the ILP model.

Heuristics
In healthcare rostering it is in many cases sufficient to find good, rather than optimal, solutions. Burke et al. (2004), for instance, claim that optimal solutions may be unreachable and meaningless, and that heuristic approaches to quickly generate feasible schedules for administrators should be in focus. Interviews with professionals also indicate that good, but not necessarily optimal, solutions often suffice because aspects like employee preferences make an optimum challenging to define. A small optimality gap is therefore of low importance. Several recent papers show promising results without applying exact methods. These commonly make use of a collection of heuristics or metaheuristics and are discussed in the following.

Smet et al. (2014) introduce a generic model to handle several complex real world problem aspects and present a suite of hyper-heuristics to solve it. The hyper-heuristics apply methods from a heuristics set, and perform well compared to another benchmark metaheuristic. Smet et al. (2016) present three construction heuristics, based on decomposition methods, and a large-scale neighborhood search algorithm. The approaches are intended for multi-skilled task scheduling. Rahimian et al. (2017) also present a neighborhood search algorithm. They rely on a combination of heuristics and integer programming, much like some of the ideas behind the solution methods proposed in this thesis. Their model is also quite similar to the one developed in this thesis, but does not include, for example, pattern constraints and constraints on strict days off.

Van der Veen et al. (2015) present a heuristic to solve the weekend rostering problem. It is argued that shift preferences primarily focus on weekends, and that it therefore makes sense to first assign weekend shifts, followed by assignment of weekday shifts. Van der Veen et al. claim this resembles manual scheduling, and is an effective approach. The paper is believed to be the first addressing weekend rostering, and invites further research on the heuristic.

Several more examples of heuristic approaches to rostering can be found that make use of various approaches like genetic algorithms and simulated annealing (e.g. Aickelin and Dowsland (2000), Burke et al. (2002), Burke et al. (2003), De Causmaecker and Berghe (2004), Abobaker et al. (2011), Asta et al. (2016)). The diversity in methods found shows that there are many different approaches that might improve rostering solutions. An important conclusion to be drawn is that the success of different methods largely depends on the underlying assumptions regarding the instances to be solved. The thesis focuses on a comprehensive problem description including aspects such as workload regulations, strict days off and pattern constraints. Thus, elements from several of the discussed solution methods are relevant to consider. An overview of the literature presented here and a discussion of the position of this thesis in the literature is given in Section 2.3.

2.3 Literature Overview

Through this literature review, rostering problems have been found to receive wide attention in operations research and to represent a class of problems that are challenging to solve. The rostering process is an important part of healthcare planning, and the NRP is particularly well studied in literature. Several optimization-based solution methods have been proposed to solve rostering problems. Among exact methods, branch and price has been found to perform well when applied to large instances, but improvements are still required to solve large instances with real-world complexity. Heuristic solution methods often produce sufficiently good solutions within less computation time. A summary of the papers discussed in the review is given in Table 2.2. The papers are attributed according to the instances considered and solution methods applied. The thesis is added to the table for comparison.

An important observation from the papers studied is the large variation in problem definitions. This is exemplified by the statistics on skills and individual preferences in Table 2.2, but has also been seen on other problem attributes like the problem objective or constraints considered. A gap between research and practice within the field of rostering has been identified, much because of the low generality of models and lack of large scale implementation. These two features substantiate the need for improved methods that can handle a wide variety of rostering instances imitating problems of real world complexity.

Table 2.2: Overview of the papers reviewed. N.a. cells indicate missing information or non applicability for the respective paper. The classification of skills and individual preferences refers to whether the paper in question considers skills and individual preferences. Solution method abbreviations: B – Branch and Price; C – Column Generation (without branch and price); D – Dynamic Programming; H – Heuristic; I – Integer Linear Programming/Mixed Integer Programming; M – Machine Learning; S – Spreadsheet Model.

Paper	Application area	Planning period [weeks]	Employees	Shift types[1]	Skills	Individual preferences	Solution method
This thesis	**Healthcare**	**4 – 24**	**10 – 50**	**3 – 5**	**Yes**	**Yes**	**B, D**
Abobaker et al. (2011)	Nurse	2	11 – 73	3	Yes	Yes	H
Aickelin and Dowsland (2000)	Nurse	1	30	3	Yes	Yes	H
Asta et al. (2016)	Nurse	4	16	3	No	Yes	H
Bard and Purnomo (2005)	Nurse	4 – 6	20 – 100	5	No	Yes	C
Beliën and Demeulemeester (2006)	Healthcare	n.a.[2]	8	n.a.[2]	n.a.[2]	No	B
Beliën and Demeulemeester (2007)	Healthcare	n.a.[2]	8	n.a.[2]	n.a.[2]	No	B
Burke and Curtois (2011)	Nurse	1 – 7	10 – 54	1 – 12	Yes	No	B, D
Burke et al. (2002)	Nurse	4	9	4	Yes	Yes	H
Burke et al. (2003)	Nurse	2	20 – 30	2	Yes	Yes	H
De Causmaecker and Berghe (2004)	Nurse	n.a.	n.a.	3	Yes	Yes	H
De Grano et al. (2009)	Nurse	4	73	6	No	Yes	I
Dohn and Mason (2013)	Nurse	4	85	5	Yes	Yes	B, D
Gamache et al. (1999)	Aircrew	4	55 – 1151	n.a.	No	No	B, D
Jaumard et al. (1998)	Nurse	2	41	n.a.	Yes	Yes	B
Lusby et al. (2012)	Air Crew	26	139	11	No	No	B, D, H
Maenhout and Vanhoucke (2010)	Nurse	4	30	4	Yes	Yes	B, D
Moz and Pato (2004)	Nurse	4	19 – 32	3	No	No	I
Ovchinnikov and Milner (2008)	Medical Residents	52	15	n.a.	Yes	No	S
Rahimian et al. (2017)	Nurse	2 – 52	8 – 150	1 – 32	No	Yes	H
Rönnberg and Larsson (2010)	Nurse	8	31	3	Yes	Yes	I
Smet et al. (2014)	Nurse	4 – 13	19 – 32	9 – 27	Yes	Yes	H
Smet et al. (2016)	General	1 – 4	10 – 40	4	Yes	No	H
Trilling et al. (2006)	Nurse	1 – 2	10 – 20	4	No	No	I
Václavík et al. (2018)	Nurse	1 – 4	8 – 30	1 – 5	Yes	Yes	B, M
Valouxis et al. (2012)	Nurse	4	10 – 50	3 – 5	Yes	Yes	I
Van der Veen et al. (2015)	General	4 – 8	10 – 60	2 – 5	Yes	Yes	H
Vanden Berghe (2002)	Nurse	n.a.	20	6	Yes	Yes	H

[1]Excluding off shifts.
[2]Beliën and Demeulemeester consider trainees at hospitals. The trainees are to be scheduled to activities over several available periods, depending on their current progress in education.

As mentioned in Chapter 1, the thesis addresses this gap between theory and practice within rostering in healthcare. A model is proposed with both hard and soft constraints that account for many of the issues that must be considered in Nordic healthcare rostering. The inclusion of more than two days long employee-specific patterns as soft constraints is one of the included features that is not commonly seen in research.

The thesis contributes to the exploration of solution methods by suggesting an uncommon decomposed mathematical model formulation where several employee-specific constraints are handled in the RMP. The convention in the literature is to handle all such constraints in the SPs. The combination of the success of branch and price in many papers and the often encountered challenge of long computation time in the SPs motivate the unconventional approach chosen in this thesis. A branch and price algorithm is proposed to solve the suggested model, and a study of several speed-up techniques inspired by this review are implemented and tested.

A large part of the studied papers consider the NRP. In this thesis, a slightly more flexible version of the NRP, also applicable for other healthcare staff, such as physicians and surgeons, is discussed. The model proposed is tested on a set of benchmark instances of different size. By applying several speed-ups, the thesis aims at solving instances of high complexity compared to what has been seen in the literature. Both the number of shift types, employees and weeks in the planning period are varied to evaluate performance along these dimensions. A detailed description of the problem considered in the thesis follows in Chapter 3.

Chapter 3

Problem Description

In this chapter, the healtcare rostering problem is described. As discussed in Chapter 1, rostering in the healthcare domain is approached from an operations research perspective in this thesis. The different parts of the healthcare planning process and rostering process were described in Chapter 2, as well as the typical rules that should be followed in a healthcare rostering process. This provides a basis to describe the HRP in this chapter.

The chapter is structured into two sections. In Section 3.1, an overview of the HRP is presented, followed by a detailed discussion of the problem, relevant taxonomy and assumptions made. Subsequently, the rules that govern the rostering process are presented in Section 3.2.

3.1 The Healthcare Rostering Problem

As discussed in Chapter 2, rostering problems in healthcare are well researched in the literature. Many authors focus on the nurse rostering problem due to its complexity and importance at large hospitals. The HRP is intended to be a flexible version of the NRP, also applicable to hospital employees other than nurses, e.g. physicians and surgeons, that face similar staffing requirements. Problem characteristics are described such that rosters may, with minimal modification, be generated for different employee groups in healthcare. It should, however, be noted that the HRP resembles the NRP to a large extent.

The main task in the HRP is to allocate healthcare staff to shifts over a given planning period. Associated with every shift is a demand for labor that must be covered. The shifts are skill specific, meaning only employees with certain skills can satisfy the labor demand associated with the shift. As discussed in Chapter 2, the demand for labor is assumed given in the HRP considered. The internal workforce size is also assumed fixed and given, but a restricted number of external substitute employees may be hired and assigned at extra cost to cover demand.

A collection of rules apply to how shifts may be assigned to internal employees. In accordance with these rules, each employee is assigned a sequence of shifts over the entire planning horizon. Such a sequence of shifts is referred to as a *roster line*. The employee is not necessarily assigned to a shift every day, and may in some cases be assigned to multiple shifts on a day. The combination of all employee roster lines constitutes a *roster*. The relation between roster line and roster is illustrated in Figure 3.1.

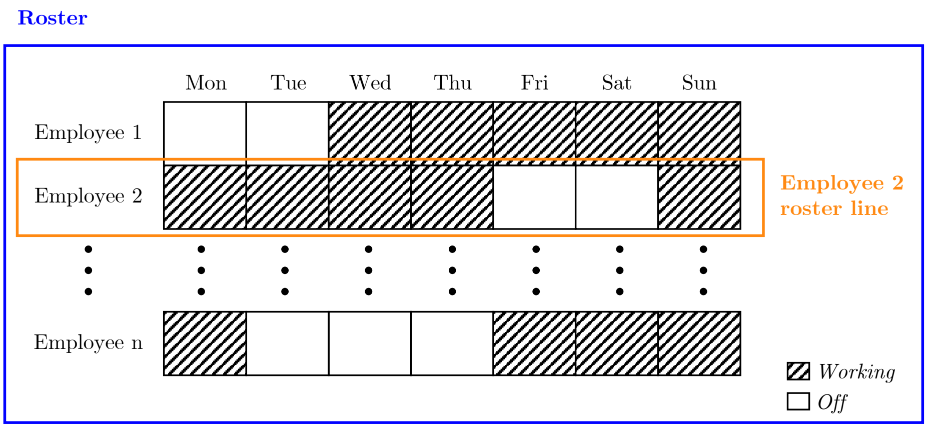


Figure 3.1: Illustration of the relation between roster lines and an overall roster.

Assuming that shifts do not vary between days, the term *shift type* is introduced to denote the day-independent characteristics of a shift, like the start time, end time and skill set required. Furthermore, shift types can be categorized into *shift groups* according to their start times. Examples of such shift groups are morning shifts, evening shifts and night shifts.

The quality of an HRP solution is evaluated according to how well the solution balances the two goals of minimal costs and maximal employee satisfaction. The monetary costs associated with the roster are mainly comprised of salary costs and costs for hiring external personnel and are easily quantified. Costs of assignment differ depending on the shift and the skill level of the employee. Additional costs may be incurred due to certain roster line rules (e.g. overtime). Employee satisfaction is, on the other hand, harder to quantify. It is assumed in the HRP that employee satisfaction is first and foremost dependent on the accommodation of preferences and compliance with rules. Other factors, such as the work environment, are not considered in this thesis. To balance the goals of minimal costs and maximal employee satisfaction, it is assumed that employee satisfaction can be quantified as costs and compared with the monetary rostering costs to make up a single cost minimization goal. Such a quantification is commonly found in the literature (Abobaker et al., 2011; Bard and Purnomo, 2005; Aickelin and Dowsland, 2000; Burke et al., 2003; Maenhout and Vanhoucke, 2010), although highlighted as a demanding exercise by healthcare professionals interviewed during the work with this thesis. A more detailed discussion of costs is provided in Chapter 6, when discussing the instances used to test the model proposed in the thesis.

3.2 The Healthcare Rostering Problem Rules

As mentioned in Section 3.1, the roster to be generated in the HRP must comply with a collection of rules. An overview of the rules that are typically considered in the NRP was given in Chapter 2, based on Cheang et al. (2003), Vanden Berghe (2002) and Smet et al. (2014). In this section, the rules considered in the HRP of this thesis are discussed. Published research, Norwegian labor regulations (Ministry of Labour and Social Affairs, 2005), interviews with professionals in Norwegian healthcare and the thesis supervisors with experience from Nordic healthcare rostering have inspired the set of rules. Interviews conducted during the work with this thesis included discussions with a doctor involved with rostering at a large Norwegian hospital and a nurse with several years of experience with work at a hospital who is currently employed at the Norwegian Directorate of eHealth. First, the rules that apply to the roster on an aggregate level are discussed. Because most rules apply to the construction of roster lines, Section 3.2.1 is dedicated to a detailed discussion of these.

In a roster, all employees must be assigned a roster line and the combination of roster lines must satisfy the demand for labor for all shifts in the planning period. Only a restricted amount of under- or over-coverage is permitted. Under-coverage is, as discussed in Section 3.1, handled by hiring external substitute staff and is limited to ensure continuation in patient treatment and internal competency. Such considerations have been highlighted from several of the professionals interviewed. Over-coverage is limited to prevent large fluctuations in the service capacity, a disadvantage also highlighted by experienced professionals. It is assumed that the disadvantage of such fluctuations can be represented by a cost, similar to the quantification of employee satisfaction. The remaining rules apply to the roster lines of individual employees and are discussed in Section 3.2.1.

3.2.1 Roster Line Rules

As pointed out by Cheang et al. (2003) and confirmed by Norwegian healthcare professionals, an employee is restricted to work at most a maximum number of consecutive days before having a day off. It is often also required that once the employee starts a sequence of days working, the sequence must continue for a minimum number of consecutive days. Dohn and Mason (2013) present the notion of a *work stretch*, consisting of an *on stretch* (period of consecutive days working) followed by an *off stretch* (period of consecutive days off). This is illustrated in Figure 3.2. The rules requiring a maximum and minimum number of consecutive days working are equivalent to requiring a maximum and minimum length of on stretches. Similarly, there might be restrictions on maximum and minimum consecutive days working specific shift groups. For instance, there might be a requirement that the employee works at most three consecutive night shifts.

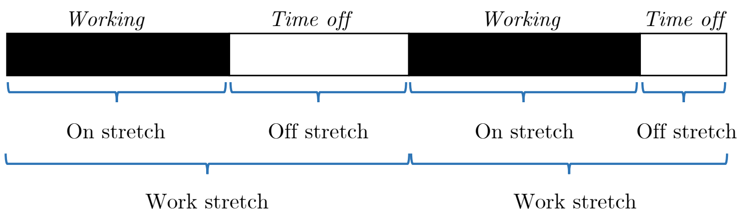


Figure 3.2: Illustration of on stretch, off stretch and work stretch. A work stretch is a combination of an on stretch and an off stretch.

Between working shifts, the employee is required to rest for a certain minimum time, as is commonly required in the Nordic countries. In addition to the absolute minimum requirement, a penalty is added to the roster line cost if rest time does not exceed a higher threshold to reflect the employee discomfort resulting from reduced rest. Combined with the penalty, it is required that the reduced rest is only taken a maximum number of times over some norm period. This norm period is stated as a rolling time horizon, meaning that over no period of given length may an employee exceed the maximum number of days with reduced rest. This prevents the employee from, for example, having many days of little rest in the end of one norm period and also the beginning of the next.

Whenever the employee works a shift on a weekend, it is commonly required that the employee works both weekend days. That is, the employee must either work both Saturday and Sunday or have the entire weekend off. Violation of this has proved to cause large dissatisfaction[1]. A late shift on Friday (i.e. ending after midnight) followed by a weekend off is also prohibited to ensure the weekend is indeed entirely without work. The number of weekends off for a given employee is restricted by a minimum during a period of a certain number of weeks. As for required rest, this rule is enforced over a rolling time horizon to prevent many weekends working towards the end of one period and at the beginning of the next.

It is common to distinguish between different types of time off. One common distinction is between a day off and a strict day off (Norsk Sykepleierforbund, 2019). A strict day off requires that the time off between two working shifts is greater than some threshold, and is considered better for the employee than a regular day off (only requiring that no shift is worked that day). Often, there is a required minimum number of strict days off during a given norm period, again considering a rolling time horizon.

The employee is contracted to work a number of hours over some period. This number of hours varies due to different types of contracts and part time employees. Both under- and overtime incur costs. There are also limits on how much under- and overtime is allowed.

[1]E.g. this article from the Norwegian Broadcasting Corporation: https://www.nrk.no/finnmark/sykepleiere-ma-jobbe-44-lordager-i-aret_-_-det-er-et-overtramp-1.13406864

In practice, hours counted in the workload calculation may differ from worked hours for some shift types. For example, Norwegian physicians may be assigned to a shift type where the doctor is allowed to be at home, but with ability to serve at the hospital within a given response time. For working such a shift type, the hours counted are only 1:4 of the hours worked (Den Norske Legeforening, 2017).

Professionals' experience indicates that working patterns are important to employee satisfaction. Typically, work stretches with little variation in the starting time of the shifts are preferred, while large fluctuations tend to cause discontent. For example, five consecutive morning shifts from Monday to Friday is often a preferred working week, while working a night shift followed by an evening shift and then a morning shift does not constitute a favorable roster line. Some working patterns are even prohibited because of the strong aversion associated with them.

The problem description from this chapter provides a basis for formulating the mathematical model in Chapter 4. Based on the problem assumptions made, a decomposed model is formulated that takes all rules presented into account.

Chapter 4

Mathematical Model

In this chapter, a model of the healthcare rostering problem described in Chapter 3 is presented. The model is formulated with Dantzig-Wolfe decomposition to be solved with column generation and branch and price. In Chapter 2, branch and price with a decomposition on employees was found to be one of the primary solution methods in rostering. In this thesis, the HRP is decomposed into a set partitioning restricted master problem and a sub problem formulated as a shortest path problem with resource constraints for each employee. The chapter is structured into four sections. Succeeding the following introduction to DWD and CG, the RMP and SPs are presented in Sections 4.1 and 4.2 respectively. This provides a foundation for discussing the handling of individual employee roster line constraints in Section 4.3. A compact formulation of the mathematical model in this chapter is appended in Appendix A.

Decomposition methods have been set forth to help solve large and complex problems. The idea is to decompose a problem into smaller and less complex problems that are systematically solved to ultimately yield a solution to the complete problem. The basic principles of DWD were first developed and published by Dantzig and Wolfe (1960). Readers not familiar with DWD are recommended to read the general introduction by Lundgren et al. (2010). In addition to a Dantzig-Wolfe reformulation, some method to solve the problem is needed for a full decomposition. One such method is column generation. A brief summary of the general principles of DWD and CG is provided in the following. Details on the CG solution method are provided in Chapter 5. Consider problem (4.1) and assume that the constraint $Ax \leq b$ is complicating, i.e. the problem is relatively easy to solve without this constraint. This is typically the case if the problem has a block angular structure as illustrated in Figure 4.1. For such a problem structure, the problem can be decomposed

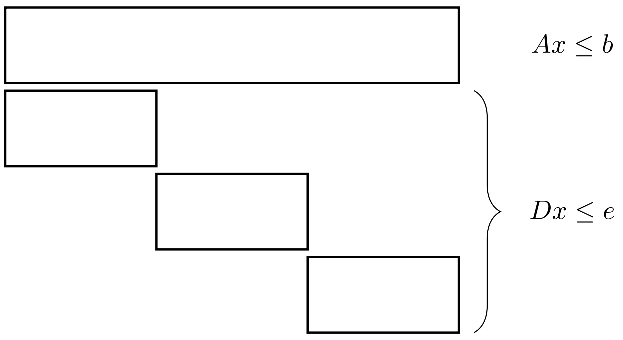


Figure 4.1: Illustration of block angular structure in a linear optimization problem. Note how the $Ax \leq b$ constraint complicates the problem by connecting the otherwise independent blocks of the $Dx \leq e$ constraint.

into several smaller problems without the complicating constraint.

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad Ax \leq & b \\ Dx \leq & e \\ x \in & \{0,1\} \end{aligned} \tag{4.1}$$

Assuming that the feasible region is convex, the feasible region without the complication constraint can be reformulated. The natural description $X_D = \{x \in \{0,1\} : Dx \leq e\}$ can be expressed with an interior representation as $X_D = \{x \in \{0,1\} : x = \sum_{k\in\mathcal{K}} \overline{x}_k\lambda_k, \sum_{k\in\mathcal{K}} \lambda_k = 1, \lambda_k \geq 0\}$, where $\overline{x}_k,\ k \in \mathcal{K}$ are the extreme points of the feasible region and λ_k are weighting variables for each extreme point. Problem (4.1) is then equivalent to problem (4.2), referred to as the master problem (MP).

$$\begin{aligned} \min \quad & \sum_{k\in\mathcal{K}} c^T \overline{x}_k \lambda_k \\ \text{s.t.} \quad \sum_{k\in\mathcal{K}} A\overline{x}_k\lambda_k \leq & b \quad | \quad \pi \leq 0 \\ \sum_{k\in\mathcal{K}} \lambda_k = & 1 \quad | \quad \omega \in \mathbb{R} \\ \sum_{k\in\mathcal{K}} \overline{x}_k\lambda_k \in & \{0,1\} \\ \lambda \geq & 0 \end{aligned} \tag{4.2}$$

Because the number of columns $k \in \mathcal{K}$ in problem (4.2) tend to be large, column generation is often used as a solution method. A restricted MP, or RMP, is introduced that is similar to (4.2), but only contains a subset $\mathcal{K}' \subseteq \mathcal{K}$ of the columns. In each iteration of CG, the integrality ($\sum_{k\in\mathcal{K}} \overline{x}_k\lambda_k \in \{0,1\}$) of the RMP is relaxed and the resulting LP problem is solved to optimality, yielding dual variables π and ω. Note that relaxing integrality is necessary to obtain the dual variables. The SP, often called pricing problem, takes these dual variables into account and generates new columns for the RMP with the lowest possible reduced cost. The SP is formulated in problem (4.3) where the objective function is the reduced cost expression from the RMP. Notice how the complicating constraint, $Ax \leq b$, does not appear in the SP. If the problem has a nice structure, like the block angular structure illustrated in Figure 4.1, the SP can then be solved as a sequence of smaller SPs.

$$\begin{aligned} \min \quad & \overline{c} = c^T x - \pi^T Ax - \omega \\ \text{s.t.} \quad & Dx \leq e \\ & x \in \{0,1\} \end{aligned} \tag{4.3}$$

An SP solution with negative reduced cost may be added as a column to the RMP and improves the LP solution if pivoted into the optimal basis. The LP relaxed RMP is then solved again and updated dual variables are obtained to be used in the SP of the next iteration. If no solution with negative reduced cost can be found, the current solution to the RMP is optimal, and the CG procedure terminates. Methods to find integer feasible solutions from the LP optimal RMP solution are discussed in Section 5.4.

When the SP is decomposed into smaller problems, these are typically referred to as the SPs, and a solution from one of the smaller SPs is often referred to as a column. As found in Chapter 2, rostering problems using DWD are usually decomposed on employees to create one SP for each employee. This is also the chosen approach in this thesis.

4.1 Restricted Master Problem

In this section, the RMP of the basic HRP is presented. It is based on the problem description in Chapter 3 and resembles many of the models found in literature, e.g. Dohn and Mason (2013). A set of employees $\mathcal{E}$, a set of days $\mathcal{D}$ in the planning period and a set of shift types $\mathcal{S}$ are inputs. The subset $\mathcal{S}^W \subset \mathcal{S}$ contains the shift types associated with a working shift. The combination of a day d and a shift type s gives a shift, with a demand for B_{ds} employees. The convention used in the thesis is that a shift belongs to the day on which the shift started. Ultimately, the objective of the RMP is to find a combination of employee roster lines that satisfy demand at minimum cost, where costs also reflect employee satisfaction with the roster. The employee-specific constraints that are included in the RMP are formulated in Section 4.3 and added to the basic RMP presented in this section.

The set $\mathcal{K}_e$ contains all available roster lines for employee e, and the binary decision variable λ_{ek} indicates if employee e is assigned to roster line k. By referring to the RMP as opposed to the MP, $\mathcal{K}_e$ only contains a subset of all feasible roster lines for each employee. The parameter C^K_{ek} states the total cost of assigning roster line k to employee e and is a sum of monetary costs and costs associated with employee satisfaction. These costs are discussed in greater detail in Sections 4.2 and 4.3. The parameter A_{ekds} indicates whether or not employee e works shift type s on day d in roster line k and is used to determine demand coverage for each shift. Integer variables w^+_{ds} and w^-_{ds} are used to record over- and under-coverage for each shift. Over- and under-coverage are limited to be at most $\overline{W}^+_{ds}$ and $\overline{W}^-_{ds}$ and are penalized with objective function cost coefficients C^+_{ds} and C^-_{ds} respectively. There are various conventions for handling over- and under-coverage in the literature (De Causmaecker and Berghe, 2002). Based on the discussion in Chapter 3, both over- and under-coverage are associated with costs and hard limits. This is a flexible formulation that allows for handling most practical cases by adjusting the relevant parameters. Note, however, that the alternative of only modeling under-coverage is evaluated in Section 6.6.1. The general RMP is a set partitioning problem formulated as (4.4) - (4.9).

$$\min \quad \sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}_e} C^K_{ek}\lambda_{ek} + \sum_{d\in\mathcal{D}}\sum_{s\in\mathcal{S}^W} C^+_{ds}w^+_{ds} + \sum_{d\in\mathcal{D}}\sum_{s\in\mathcal{S}^W} C^-_{ds}w^-_{ds} \tag{4.4}$$

$$\sum_{e\in\mathcal{E}}\sum_{k\in\mathcal{K}_e} A_{ekds}\lambda_{ek} + w^-_{ds} - w^+_{ds} = B_{ds} \qquad s\in\mathcal{S}^W, d\in\mathcal{D} \quad | \quad \pi_{ds}\in\mathbb{R} \tag{4.5}$$

$$\sum_{k\in\mathcal{K}_e} \lambda_{ek} = 1 \qquad e\in\mathcal{E} \quad | \quad \omega_e\in\mathbb{R} \tag{4.6}$$

$$\lambda_{ek}\in\{0,1\} \qquad e\in\mathcal{E}, k\in\mathcal{K}_e \tag{4.7}$$

$$0\le w^+_{ds}\le \overline{W}^+_{ds} \qquad s\in\mathcal{S}^W, d\in\mathcal{D} \tag{4.8}$$

$$0\le w^-_{ds}\le \overline{W}^-_{ds} \qquad s\in\mathcal{S}^W, d\in\mathcal{D} \tag{4.9}$$

The objective function (4.4) is the sum of costs for roster lines and penalties for over- and under-coverage. Constraints (4.5) ensure that all demand is covered or the appropriate over- or under-coverage is recorded. Through (4.6) - (4.7), exactly one roster line is assigned to each employee. The variable restrictions on coverage are enforced by (4.8) - (4.9). Note that by enforcing integer requirements on the λ-variables, integrality is automatically ensured for the over- and under-coverage variables in an optimal solution and must not be enforced explicitly. The dual variables π_{ds} and ω_e associated with constraints (4.5) and (4.6) respectively are only available when the binary requirements on the λ-variables in constraints (4.7) are relaxed.

4.2 Sub Problems

Based on the dual variables from an LP relaxed RMP solution, the SPs generate one or several columns with negative reduced cost, if they exist. There is one SP for each employee $e\in\mathcal{E}$, denoted SP(e). A solution of SP(e) corresponds to a roster line for employee e and may be added as a column to the RMP. The general formulation of SP(e) is described in this section, while the modeling of roster line constraints in both the RMP and SPs are described in Section 4.3.

The reduced cost of a roster line for employee e based on a solution of the RMP in Section 4.1 is given in eq. (4.10). The k-index is omitted since SP(e) aims at generating new roster lines that can be added to the set $\mathcal{K}_e$ with an assigned k-index. To differentiate between different skill levels, a set of skills $\mathcal{T}$ is introduced. The sets $\mathcal{T}^S_s \subseteq \mathcal{T}, s \in \mathcal{S}$ are defined to contain the skill levels that qualify for working shift type s and the sets $\mathcal{T}^E_e \subseteq \mathcal{T}, e \in \mathcal{E}$ contain the skill levels of employee e. To alleviate notation complexity, $\mathcal{S}_e \subseteq \mathcal{S}$ is hereafter used to denote the shift types that can be assigned to employee $e \in \mathcal{E}$ (i.e. the set of shift types for which the employee has at least one of the skills required to be assigned the shift[1]). Similarly, $\mathcal{S}^W_e$ contains the working shift types that can be assigned to employee $e \in \mathcal{E}$.

$$\bar{c}_e = C^K_e - \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}^W_e} A_{eds}\pi_{ds} - \omega_e \tag{4.10}$$

The direct cost C^K_e of a roster line for employee e is a sum of the cost per shift assigned and individual preference costs which are discussed in Section 4.3. For now it is assumed that $C^K_e = \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}_e} C_{eds}x_{eds}$ where C_{eds} is the cost of employee e working shift type s on day d and the binary variable x_{eds} indicates if employee e works shift type s on day d. The x-variables are the decision variables of SP(e). With $A_{eds} = x_{eds}$ for $s \in \mathcal{S}_e$ and $A_{eds} = 0$ otherwise, the general formulation of SP(e) is given by (4.11). The objective is to minimize the reduced cost while satisfying roster line constraints which are discussed in detail in Section 4.3. When constraints are added to the RMP, the objective function of the SPs are added terms corresponding to the changes in reduced cost because of the new dual variables. These added objective function terms are also discussed in Section 4.3.

$$\begin{aligned} \min \quad & \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}_e} C_{eds}x_{eds} - \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}^W_e} \pi_{ds}x_{eds} - \omega_e \\ \text{s.t.} \quad & \text{(Roster Line Constraints)} \\ & x_{eds} \in \{0,1\} \quad d \in \mathcal{D}, s \in \mathcal{S}_e \end{aligned} \tag{4.11}$$

SP(e) is formulated as an SPPRC. In Chapter 2, this was found to often be implemented with good results (Gamache et al., 1999; Maenhout and Vanhoucke, 2010; Dohn and Mason, 2013; Burke and Curtois, 2014). The notation used to formulate the SPPRCs in this thesis closely resembles that of Irnich and Desaulniers (2005). Although there is one SP for each employee, the graph and resource structures are the same for each employee. Therefore, the e-index is omitted in the associated notation. However, some parameters and sets and all SP solution variables are employee specific and require the e-index.

A graph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{A}_e)$ of nodes (vertices) $\mathcal{V}_e$ and arcs $\mathcal{A}_e$ is defined for each employee $e \in \mathcal{E}$. Each node $v \in \mathcal{V}_e$ represents a specific shift type s on a given day d, and can be written $v = (d, s)$. Start and end nodes are defined with artificial shift type 0 on days 0 and $|\mathcal{D}| + 1$ respectively. Arcs represent the transition from a shift assignment on one day to the next day, giving rise to an acyclic directed graph like the one illustrated in Figure 4.2. The assignment of exactly one shift per day is implied by the network. This enables the notation $a_{edij} \in \mathcal{A}_e$ for the arc between nodes $(d-1, i)$ and (d, j). To enable modeling of days off, an off shift type is introduced on each day. Note that for an employee e, the SPPRC network contains only nodes representing the shift types $s \in \mathcal{S}_e \subseteq \mathcal{S}$.

[1]$\mathcal{S}_e = \{s \in \mathcal{S} | \mathcal{T}^S_s \cap \mathcal{T}^E_e \neq \emptyset\}$.

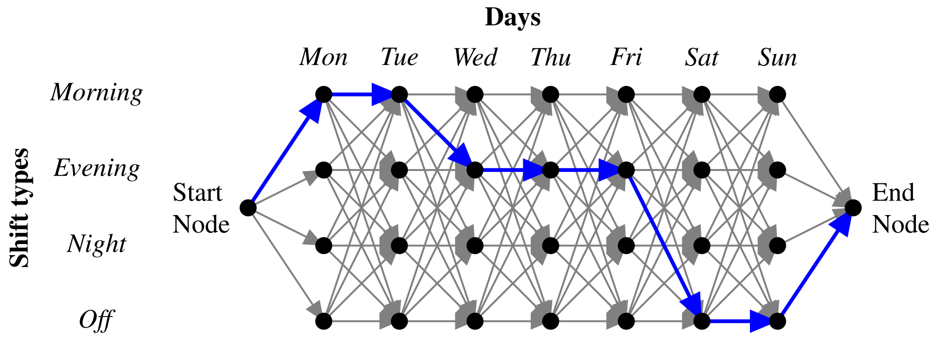


Figure 4.2: Example of graph with seven days and four shift types including the off shift type. A feasible path representing a week with two morning shifts followed by three evening shifts and then the weekend off is highlighted in blue.

To enable an SPPRC formulation, the binary variables y_{edij} are introduced for $e \in \mathcal{E}, d \in \mathcal{D} \cup \{|\mathcal{D}|+1\}, i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e \cup \{0\}$. $y_{edij} = 1$ implies that employee e works shift type i on day $d-1$ and shift type j on day d and thereby traverses arc a_{edij}. The transform from the y-variables of the SPPRC to the x-variables in the formulation of SP(e) is expressed by eq. (4.12). On the first day, the transform must account for all paths originating in the start node with shift type 0.

$$\begin{aligned} x_{eds} &= \sum_{i \in \mathcal{S}_e} y_{edis} \quad e \in \mathcal{E}, d \in \mathcal{D} \backslash \{1\}, s \in \mathcal{S}_e \\ x_{e1s} &= y_{e10s} \quad e \in \mathcal{E}, s \in \mathcal{S}_e \end{aligned} \tag{4.12}$$

Constraints (4.13)-(4.15) are added to SP(e) to ensure flow through the SPPRC network is consistent with the problem formulation. Zero net flow in each node except for the start and end node ensures the employee traverses from one shift to the next and is enforced by constraints (4.13). Constraint (4.14) requires the employee to leave the start node and be assigned some shift type on the first day. Similarly, constraint (4.15) ensures the employee finishes a shift type on the last day and makes the transition to the end node.

$$\sum_{i \in \mathcal{S}_e} y_{edis} - \sum_{j \in \mathcal{S}_e} y_{e(d+1)sj} = 0 \quad d \in \mathcal{D}, s \in \mathcal{S}_e \tag{4.13}$$

$$\sum_{j \in \mathcal{S}_e} y_{e10j} = 1 \tag{4.14}$$

$$\sum_{i \in \mathcal{S}_e} y_{e(|\mathcal{D}|+1)i0} = 1 \tag{4.15}$$

A path $P = (v_0, v_1, \ldots, v_p)$ through the $\mathcal{G}_e$ is a sequence of nodes with $(v_i, v_{i+1}) \in \mathcal{A}_e$ for all $i = 0, \ldots, p-1$. If $v_0 = (0,0)$ and $v_p = (|\mathcal{D}|+1, 0)$, the path represents a roster line for employee e. For convenience, the path $P_q = (v_0, \ldots, v_q)$ for $0 \leq q \leq p$ is called the sub path of P consisting of the first $q+1$ nodes in P. The notation $v(P)$ is used to denote the last node in path P. Each arc is associated with a cost C_{edij} equal to the cost of setting the value of variable y_{edij} to one. Before adding terms for roster line constraints as described in Section 4.3, the cost is based on the cost for the x-variables in the general SP formulation, giving the expression $C_{edij} = C_{edj} - \pi_{dj}$. The dual variable ω_e is subtracted from the cost of all arcs going out of the start node of the SPPRC for SP(e) to make sure this is included in all roster lines. The cost of some sub path P_q is denoted $C(P_q)$ and expressed recursively by eq. (4.16) where $(d_q, s_q) = v(P_q)$. The objective of the SPPRC is to find the shortest feasible path from the start node to the end node, i.e. the feasible roster line with the lowest cost.

$$\begin{aligned} &C(P_0) = 0 \\ &C(P_q) = C(P_{q-1}) + C_{ed_q s_{q-1} s_q} \quad 1 \leq q \leq p \end{aligned} \tag{4.16}$$

The description of feasible paths by using resources is key to the definition of the SPPRC. Resources are used to enforce constraints not handled by the network structure, and are presented in Section 4.3. If $\mathcal{R} = \{1, \ldots, R\}$ is the set of resources, the resource vector $T(P_q) = (T^1(P_q), \ldots, T^R(P_q))^T \in \mathbb{R}^R$ describes the value of each resource in node v_q of sub path P_q. For simplified notation, T^r_{ds} is used to denote the value of resource r in node $v = (d, s)$. Note that T^r_{ds} is dependent on the path taken, i.e. $T^r_{ds} = T^r_{ds}(P_q)$. The change in resource value of resource $r \in \mathcal{R}$ on arc $a_{edij} \in \mathcal{A}_e$ is given by the *resource extension function* (REF) $f^r_{dij} : \mathbb{R}^R \mapsto \mathbb{R}$. Only separable REFs, i.e. without interdependence between resources, are considered in this thesis. Thus, the REFs are given as $f^r_{dij} : \mathbb{R} \mapsto \mathbb{R}$. The resource value of resource r is given by the recursion in eq. (4.17). T^r_0 is the value of the resource in the start node.

$$\begin{aligned} &T^r(P_0) = T^r_0 \\ &T^r(P_q) = f_{d_q s_{q-1} s_q}(T^r(P_{q-1})) \quad 1 \leq q \leq p \end{aligned} \tag{4.17}$$

Each node $v = (d, s) \in \mathcal{V}_e, e \in \mathcal{E}$ has a resource window $[\underline{T}^r_v, \overline{T}^r_v]$ for each $r \in \mathcal{R}$ which limits the resource values that are feasible. In the start and end nodes, all resource windows are unbounded, meaning there are no restrictions outside the planning period. A path P is *resource-feasible* if for all nodes in the path, the resource values are within the associated resource windows of the respective nodes. In this thesis the term *resource* is used in a wider sense than by Irnich and Desaulniers (2005). REFs are not required to be linear non-decreasing which allows handling of many types of constraints, e.g. path structural constraints, with the same notation and nomenclature. Resources are discussed further in Section 4.3. A path P through the network is said to be feasible if it is resource-feasible.

4.3 Roster Line Constraints

As mentioned in Chapters 1 and 2, the decomposition suggested handles some employee-specific constraints in the RMP rather in the SPs as is common in research. The rationale for this is that some employee specific constraints may be better handled in a mixed integer programming model than an SPPRC framework. Potential benefits of the model decomposition proposed in the thesis include less computation time spent solving each SP and the ability to handle a diverse collection of constraints. On the other hand, moving employee specific constraints from the SPs to the RMP can lead to the generation of infeasible roster lines that can never be part of an optimal integer solution, but nonetheless increase the size of the RMP. A method to handle this is discussed in Section 5.5.7, and the performance of the model is evaluated in Chapter 6.

Results from the preparatory research project to this thesis (Coates and Pedersen, 2018) are used when determining which constraints are handled in the RMP and SPs. Compared to the preparatory project, the thesis considers the more complex HRP rather than the healthcare roster line problem. The results from the project regarding which constraints are best handled in the RMP and SP are thus not fully applicable to this thesis. However, much of the discussion about how constraints perform and interact across the MIP formulated RMP and the SPPRC formulated SPs is relevant. Table 4.1 gives an overview of which constraints are handled where in the thesis and resembles the best configuration identified in the preparatory project. An alternative allocation of constraints where also maximum consecutive days working is handled in the SPs is presented in Section 5.5.5 and tested in Section 6.5.4. In the following, the details about the modeling of all constraints are presented.

Table 4.1: Overview of constraint allocation between the RMP and SPs.

Constraints	Restricted Master Problem	Sub Problems
One shift per day		X
Maximum consecutive days working	X	
Minimum consecutive days working		X
Rest between shifts		X
Maximum days with rest penalty	X	
Weekends		X
Strict days off	X	
Workload	X	
Patterns	X	X

One shift per day

As discussed in Section 4.2, each employee is assigned exactly one shift each day. By including an off shift as a shift type, days off are included in the model. There are alternative modeling choices. For example, the employee can be assigned at most one shift each day, but not necessarily one. This requires an alternative SP network structure. Another option is allowing the assignment of more shifts, for example a morning and night shift, on the same day. A possible way to handle variations of this constraint is to introduce new shift types, e.g. representing a morning and evening shift on the same day. This is not considered further in this thesis. The assignment of one shift per day is handled in the SPs because it is already embedded in the SP network structure where all arcs go from a shift on one day to a shift on the next day.

Planned holidays and vacations are not considered directly in the model. If holidays and vacations are predictable at the time of rostering, specific off days may easily be enforced by removing all working shifts from the SP network on the respective days. If not known at the time of rostering, they are better handled in rescheduling problems.

Maximum consecutive days working
Constraints on maximum consecutive days working, i.e. maximum length of an on stretch, are enforced in the RMP. The preparatory research project showed that including both maximum and minimum consecutive days working in the SPs increases SP computation time significantly, while handling them in separate problems showed promising results. The best performance was achieved with maximum consecutive days working handled in the RMP and is therefore chosen here. No more than $\overline{N}$ days can be worked consecutively. For modeling purposes, the parameter A^W_{ekd} is introduced to indicate if, for employee e, roster line $k \in \mathcal{K}_e$ has a working shift on day d. Based on a roster line solution $\overline{\mathbf{x}}_{ek} = \{\overline{x}_{ekds} | d \in \mathcal{D}, s \in \mathcal{S}_e\}$, this can be calculated as $A^W_{ekd} = \sum_{s \in \mathcal{S}^W_e} \overline{x}_{ekds}$. Constraints (4.18) ensure that over any period of $\overline{N}+1$ consecutive days, no more than $\overline{N}$ working shifts are assigned.

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=d}^{d+\overline{N}} A^W_{ekd'} \lambda_{ek} \leq \overline{N} \qquad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - \overline{N} \quad \Big| \quad \alpha^W_{ed} \leq 0 \tag{4.18}$$

A set of shift groups $\mathcal{G}$ is defined to categorize similar shift types (e.g. morning shifts). The subset $\mathcal{S}_g \subseteq \mathcal{S}$ is used to denote the set of shifts belonging to shift group $g \in \mathcal{G}$. As with A^W_{ekd}, A^g_{ekd} indicates if roster line $k \in \mathcal{K}_e$ has a shift from shift group g on day d and is calculated from a roster line solution $\overline{\mathbf{x}}_{ek}$ as $A^g_{ekd} = \sum_{s \in \mathcal{S}^g_e} \overline{x}_{ekds}$ where $\mathcal{S}^g_e$ contains all shift types from group g that may be worked by the employee[2]. In a similar manner to constraints (4.18), constraints (4.19) prohibit an employee from working more than $\overline{N}^g$ shifts consecutively from shift group $g \in \mathcal{G}$.

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=d}^{d+\overline{N}^g} A^g_{ekd'} \lambda_{ek} \leq \overline{N}^g \qquad \begin{array}{l} e \in \mathcal{E}, g \in \mathcal{G}, d \in \mathcal{D}, \\ d \leq |\mathcal{D}| - \overline{N}^g \end{array} \quad \Bigg| \quad \alpha^g_{ed} \leq 0 \tag{4.19}$$

The introduction of constraints (4.18) and (4.19) in the RMP lead to the subtraction of terms (4.20) and (4.21) from the SP(e) objective function. $\alpha^W_{ed} \leq 0$ and $\alpha^g_{ed} \leq 0$ are the dual variables from constraints (4.18) and (4.19) respectively.

$$\sum_{\substack{d \in \mathcal{D} \\ d \leq |\mathcal{D}| - \overline{N}}} \alpha^W_{ed} \sum_{d'=d}^{d+\overline{N}} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{s \in \mathcal{S}^W_e} y_{ed'is} \tag{4.20}$$

$$\sum_{g \in \mathcal{G}} \sum_{\substack{d \in \mathcal{D} \\ d \leq |\mathcal{D}| - \overline{N}^g}} \alpha^g_{ed} \sum_{d'=d}^{d+\overline{N}^g} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{s \in \mathcal{S}^g_e} y_{ed'is} \tag{4.21}$$

Minimum consecutive days working
Constraints on minimum consecutive days working, i.e. minimum length of an on stretch,

[2] $\mathcal{S}^g_e = \{s \in \mathcal{S}^g | \mathcal{T}^S_s \cap \mathcal{T}^E_e \neq \emptyset\}, e \in \mathcal{E}, g \in \mathcal{G}$.

are enforced in the SPs with resources. If employee $e \in \mathcal{E}$ starts working an on stretch, at least $\underline{N}_e$ days must be worked consecutively. The parameter $\underline{N}_e$ is employee specific to allow for part time employees whose requirements on minimum on stretch length can be less strict than those of full time employees. For modeling purposes, the set $\mathcal{S}^O$ is defined to contain all shift types corresponding to off time. The off shift type has no skill requirement, thus $\mathcal{T}_s^S = \mathcal{T}$ for $s \in \mathcal{S}^O$.

The resource $T^{\underline{W}}$ is used to enforce the constraints and is stated in Box (4.22). Once an employee enters an on stretch, corresponding to a transition from an off shift to a working shift in the SP network, the resource $T^{\underline{W}}$ is set to one. This reflects that one working shift has been assigned. Whenever a transition between working shifts is made, $T^{\underline{W}}$ is incremented by one unit to record that an additional working shift has been assigned. The resource value is unaffected by transitions to off shifts such that the resource keeps tracks of the length of the previous on stretch assigned. By requiring a resource value of at least $\underline{N}_e$ upon making a transition to an off shift with the resource windows in Box (4.22), no off shift can be assigned unless the previous on stretch assigned was of length at least $\underline{N}_e$. The initial value of this resource for SP(e) is $\underline{N}_e$ to allow the assignment of an off shift on the first day.

$$\begin{array}{rl}
\text{Resource} & T^{\underline{W}} \\
\text{REF} & f_{dij}^{\underline{W}}(T^{\underline{W}}) = \begin{cases} T^{\underline{W}} + 1 & i \in \mathcal{S}_e^W, j \in \mathcal{S}_e^W, d \in \mathcal{D} \\ 1 & i \notin \mathcal{S}_e^W, j \in \mathcal{S}_e^W, d \in \mathcal{D} \\ T^{\underline{W}} & \text{otherwise} \end{cases} \\
\text{Initial value} & T_0^{\underline{W}} = \underline{N}_e \\
\text{Resource window} & T_{ds}^{\underline{W}} \in \begin{cases} [0, \infty) & d \in \mathcal{D}, s \in \mathcal{S}_e^W \\ [\underline{N}_e, \infty) & d \in \mathcal{D}, s \in \mathcal{S}^O \end{cases}
\end{array} \tag{4.22}$$

Similarly, if the employee is assigned to a shift from shift group $g \in \mathcal{G}$, at least $\underline{N}_e^g$ shifts from that shift group must be worked consecutively. The design of the corresponding resources $T^{\underline{W}^g}, g \in \mathcal{G}$ is similar to that of $T^{\underline{W}}$ and is formulated in Box (4.23).

$$\begin{array}{rl}
\text{Resource} & T^{\underline{W}^g}, \quad g \in \mathcal{G} \\
\text{REF} & f_{dij}^{\underline{W}^g}(T^{\underline{W}^g}) = \begin{cases} T^{\underline{W}^g} + 1 & i \in \mathcal{S}_e^g, j \in \mathcal{S}_e^g, d \in \mathcal{D} \\ 1 & i \notin \mathcal{S}_e^g, j \in \mathcal{S}_e^g, d \in \mathcal{D} \\ T^{\underline{W}^g} & \text{otherwise} \end{cases} \\
\text{Initial value} & T_0^{\underline{W}^g} = \underline{N}_e^g \\
\text{Resource window} & T_{ds}^{\underline{W}^g} \in \begin{cases} [0, \infty) & d \in \mathcal{D}, s \in \mathcal{S}_e^g \\ [\underline{N}_e^g, \infty) & d \in \mathcal{D}, s \notin \mathcal{S}_e^g \end{cases}
\end{array} \tag{4.23}$$

Rest between shifts
A shift type $s \in \mathcal{S}^W$ starts and ends at times $T^S_s \in \mathbb{R}^+$ and $T^E_s \in \mathbb{R}^+$ respectively. All time is modeled in hours. The end time is expressed relative to the day the shift started. For example, a shift with $T^S_s = 23$ and duration 8 hours has $T^E_s = 31$, meaning it starts at 23:00 and ends at 07:00 the following day.

To model requirements on rest between shifts stated in Section 3.2, two sets are introduced for each working shift type. The sets $\mathcal{S}^I_s$ contain all shift types s' that cannot follow shift type $s \in \mathcal{S}^W$ due to too little resting time. Note that for some shift types this set may be empty, meaning any shift type may be assigned on the following day. The parameter T^I denotes the required rest time (in hours) between successive shifts. Using the parameter $H = 24$ to denote the number of hours in a day, the sets are defined as $\mathcal{S}^I_s = \{s' \in \mathcal{S}^W \mid T^S_{s'} + H - T^E_s < T^I\}, s \in \mathcal{S}^W$. If the time between the end of a shift type s, T^E_s, and the start of a shift type on the following day, $T^S_{s'} + H$ (H is added to reference the time to the same day), is less than T^I, shift type s' is not allowed to follow directly after shift type s and is thus included in $\mathcal{S}^I_s$.

Similarly, to model soft requirements on rest between shifts, the sets $\mathcal{S}^R_s$ are defined to contain all shift types s' which, if assigned directly after shift type s, incur a penalty for little rest time. To reduce the size of $\mathcal{S}^R_s$, shifts s' that are in $\mathcal{S}^I_s$ are left out of $\mathcal{S}^R_s$, since s' will never be assigned following s immediately. With the parameter T^R denoting the minimum rest time between shifts without penalty, the sets are defined as $\mathcal{S}^R_s = \{s' \in \mathcal{S}^W \mid T^I \leq T^S_{s'} + H - T^E_s < T^R\}, s \in \mathcal{S}^W$.

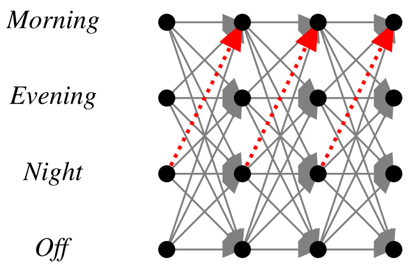
Modeling of the requirements on the rest time between shifts is divided between the SPs and the RMP. Required rest is implemented in the SP networks because it involves an uncomplicated network adjustment that reduces the problem size. It is handled by removing all arcs between any shift type s and shift types $s' \in \mathcal{S}^I_s$ on the following day. This is enforced by constraints (4.24) in SP(e) and illustrated in Figure 4.3a.

$$y_{edij} = 0 \qquad\qquad d \in \mathcal{D}, i \in \mathcal{S}_e, j \in \mathcal{S}^I_i \tag{4.24}$$

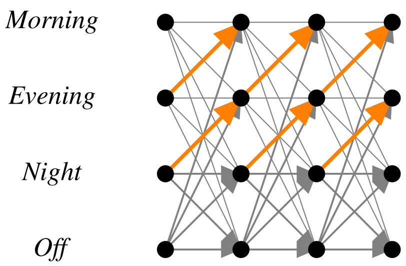
Shift transitions incurring a penalty for reduced rest are also modeled in the SP network because of an uncomplicated network adjustment. The penalty C^R is added to all arcs from any shift type s to a shift type $s' \in \mathcal{S}^R_s$ on the following day. This is illustrated in Figure 4.3b and leads to the term in eq. (4.25) being added to the objective function of SP(e). Correspondingly, for each penalized shift transition, C^R is added to the cost C^K_{ek} of a roster line in the RMP. Note that sufficient rest is always assumed on the first day.

$$\sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{S}^W_e} \sum_{j \in \mathcal{S}^R_i} C^R y_{edij} \tag{4.25}$$

Over any period of D^R days, at most $\overline{N}^R$ shifts may be assigned that incur a penalty due to little rest. This requirement is handled in the RMP based on results from the preparatory research project where handling in the SP slows down computation time significantly. The parameter V_{ekd} indicates if roster line $k \in \mathcal{K}_e$ has a shift assignment on day d that incurs a penalty. From an SP solution $\overline{\mathbf{y}}_{ek} = \{\overline{y}_{ekdij} | e \in \mathcal{E}, d \in \mathcal{D} \cup \{|\mathcal{D}| + 1\}, i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e \cup \{0\}\}$, the parameter may be calculated as $V_{ekd} = \sum_{i \in \mathcal{S}^W_e} \sum_{j \in \mathcal{S}^R_i} \overline{y}_{ekdij}$.



(a) Illegal transitions (shown in red) between shift types due to insufficient rest time between successive shifts are removed. A typical illegal transition is between a night shift and a morning shift on the following day.



(b) Transitions between shift types that incur a penalty due to reduced rest time (shown in orange) are added a penalty cost. Typical transitions include transitions from night shifts to evening shifts or evening shifts to morning shifts.

Figure 4.3: Handling of rest between shifts in the SP network.

Constraints (4.26) ensure that over any period of D^R consecutive days, no more than $\overline{N}^R$ shifts incurring a penalty for reduced rest are assigned.

$$\sum_{k\in\mathcal{K}_e}\sum_{d'=d}^{d+D^R-1} V_{ekd'}\lambda_{ek} \leq \overline{N}^R \quad e\in\mathcal{E}, d\in\mathcal{D}, d\leq|\mathcal{D}|-D^R+1 \quad | \quad \gamma_{ed}\leq 0 \quad (4.26)$$

The introduction of constraints (4.26) in the RMP lead to the subtraction of terms (4.27) from the SP(e) objective function. $\gamma_{ed} \leq 0$ are the respective dual variables.

$$\sum_{\substack{d\in\mathcal{D}\\ d\leq|\mathcal{D}|-D^R+1}} \gamma_{ed} \sum_{d'=d}^{d+D^R-1} \sum_{i\in\mathcal{S}_e^W} \sum_{j\in\mathcal{S}_{s_1}^R} y_{ed'ij} \quad (4.27)$$

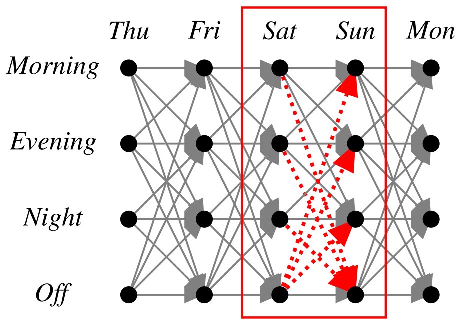
Weekends

The requirement from Section 3.2 that an employee works either both or none of the weekend days is modeled in the SP networks by removing all arcs from weekend working shifts to weekend off shifts and vice versa. This is enforced with constraints (4.28) and (4.29) in SP(e) and shown in Figure 4.4a. The set $\mathcal{B}$ contains all weekdays and the set $\mathcal{D}^b \subset \mathcal{D}$ contains all days on weekday $b\in\mathcal{B}$.

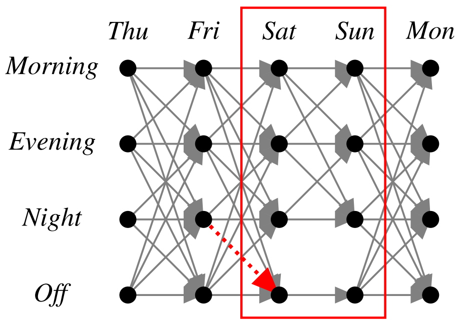
$$y_{edij} = 0 \qquad d\in\mathcal{D}^{\text{SUN}}, i\in\mathcal{S}_e^W, j\in\mathcal{S}^O \quad (4.28)$$

$$y_{edij} = 0 \qquad d\in\mathcal{D}^{\text{SUN}}, i\in\mathcal{S}^O, j\in\mathcal{S}_e^W \quad (4.29)$$

Similarly, to prohibit the assignment of a late Friday shift before a weekend off, all arcs from Friday shifts that end after midnight to an off shift on the following Saturday are removed. This is enforced by constraints (4.30) and shown in Figure 4.4b where it is assumed that the night shift starting on Friday ends after midnight. Recall that the day of



(a) Handling of constraint on working either both or no days in a weekend by removing arcs in the SP network (shown with red dotted lines).



(b) Handling of constraints prohibiting working a late shift on the Friday before a free weekend by removing arcs in the SP network (shown with a red dotted line).

Figure 4.4: Handling of weekend constraints in the SP network.

a shift is determined by the start time, and not end time, of the shift.

$$y_{edij} = 0 \qquad\qquad d \in \mathcal{D}^{\text{SAT}}, \{i \in \mathcal{S}_e | T_i^E > H\}, j \in \mathcal{S}^O \tag{4.30}$$

To model restrictions on the number of weekends worked over a period of W^W weeks, the set $\mathcal{W}$ is introduced and contains all weeks in the planning period. The set $\mathcal{D}_w^b$ contains the weekday $b \in \mathcal{B}$ in week $w \in \mathcal{W}$. $\underline{N}^W$ states the minimum number of weekends off during a period of W^W weeks, meaning the number of weekends worked is required to be less than $W^W - \underline{N}^W$ over W^W consecutive weeks. Because of promising results in the preparatory research project, this requirement is modeled in the SPs by the resource $T^{V(t)}$, counting the number of weekends worked. To consider all overlapping periods of W^W consecutive weeks, the resource is indexed by $t \in \{1, \ldots, W^W\}$.

$$\begin{array}{rl}
\text{Resource} & T^{V(t)}, \quad t \in \{1, \ldots, W^W\} \\
\text{REF} & f_{dij}^{V(t)}(T^{V(t)}) = \begin{cases} T^{V(t)} + 1 & d \in \mathcal{D}^{SAT}, \\ & i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e^W \\ 0 & w \in \mathcal{W}, \\ & w \bmod W^W = t-1, \\ & d \in \mathcal{D}_w^{MON}, \\ & i \in \mathcal{S}_e, j \in \mathcal{S}_e \\ T^{V(t)} & \text{otherwise} \end{cases} \\
\text{Initial value} & T_0^{V(t)} = 0 \\
\text{Resource window} & T_{ds}^{V(t)} \in [0, W^W - \underline{N}^W] \quad d \in \mathcal{D}, s \in \mathcal{S}_e
\end{array} \tag{4.31}$$

$T^{V(t)}$ counts the number of weekends worked and is initialized with value zero. Whenever a working shift is assigned on a Saturday the resource is incremented by one. Because of constraints (4.28) and (4.29), a working shift on a Saturday is equivalent to a full working weekend. The resource window ensures the resource never surpasses $W^W - \underline{N}^W$ and violates the requirement on the number of weekends worked. Because the requirements are over a period of W^W weeks, the resource is reset to zero on each Monday in the new period, expressed mathematically with each Monday on a week w with $w \bmod W^W = t-1$. The constraints are enforced on any period of W^W weeks meaning many of the periods overlap. To account for this, there is one resource $T^{V(t)}$ for each overlapping cycle beginning from week one to week W^W (i.e. $t \in \{1, \dots, W^W\}$).

Strict days off
To model strict days off one must consider more than two consecutive days at the time, which increases the complexity of modeling with resources in the SPs significantly. Therefore, these constraints are handled in the RMP. The variable $s_{ed} \in \{0,1\}$ is introduced to indicate if employee e has a strict day off on day d. The parameter A^O_{edk} resembles the previously defined A^W_{edk}, and indicates if employee e is assigned an off shift on day d in roster line $k \in \mathcal{K}_e$. The parameter is calculated as $A^O_{ekd} = \sum_{s \in \mathcal{S}^O} \overline{x}_{ekds}$. Constraints (4.32) hinder a strict day off if an employee does not have an off day.

$$s_{ed} - \sum_{k \in \mathcal{K}_e} A^O_{ekd} \lambda_{ek} \leq 0 \qquad e \in \mathcal{E}, d \in \mathcal{D} \quad \Big| \quad \epsilon^{\mathrm{O}}_{ed} \leq 0 \qquad (4.32)$$

Recall from the discussion of required rest between shifts that subsets of shift types may be defined to prohibit sequences of shift types. To enforce requirements on the time off between shifts yielding strict days off, two such subsets are defined for each $s \in \mathcal{S}^W$. The set $\mathcal{S}^{S1}_s = \{s' \in \mathcal{S}^W \mid T^S_{s'} + 2H - T^E_s < H^{S1}\}$ contains all shifts types s' that, if assigned on day $d+1$ when shift type s was assigned on day $d-1$, will not yield a strict day off on day d. The parameter H^{S1} denotes the required time off (in hours) between shifts for one strict day off to be recorded. If the time between the end of a working shift type s, T^E_s, and the beginning of some shift type s' two days later, $T^S_{s'} + 2H$, is less than T^{S1} hours, a strict day off will not be recorded between the shift types, and s' is included in $\mathcal{S}^{S1}_s$.

Similarly, $\mathcal{S}^{S2}_s = \{s' \in \mathcal{S}^W \mid T^S_{s'} + 3H - T^E_s < H^{S2}\}$ contains all shift types s' that cannot yield two strict days off if assigned three days after shift type s. Compared to for one strict day off the same logic applies, only with the new parameter H^{S2} denoting required time off between shifts for two strict days off, and $3H$ is added since there is one more day between the assignment of the shift types.

Constraints (4.33) ensure one strict day off is assigned only if the employee has the required time off between working shifts. They require that s_{ed} is set to zero if some employee e is assigned shift type s_1 on day $d-1$ and some shift type $s_2 \in \mathcal{S}^{S1}_{s_1}$ on day $d+1$.

$$\begin{array}{c|c} \displaystyle\sum_{k \in \mathcal{K}_e} (A_{ek(d-1)s_1} + A_{ek(d+1)s_2}) \lambda_{ek} + s_{ed} \leq 2 & \\ & \epsilon^{\mathrm{I}}_{eds_1s_2} \leq 0 \\ e \in \mathcal{E}, d \in \mathcal{D} \setminus \{1, |\mathcal{D}|\}, s_1 \in \mathcal{S}^W_e, s_2 \in \mathcal{S}^{S1}_{s_1} & \end{array} \qquad (4.33)$$

Note that constraints (4.33) are not defined for off shifts, as these do not limit the recording of strict days off.

Combined with constraints (4.32) and (4.33), constraints (4.34) prohibit recording two strict days off on days d and $d+1$ if a shift type s_1 was assigned on day $d-1$ and a shift type $s_2 \in \mathcal{S}^{S2}_{s_1}$ on day $d+2$.

$$\sum_{k\in\mathcal{K}_e} (A_{ek(d-1)s_1} + A_{ek(d+2)s_2})\lambda_{ek} + s_{ed} + s_{e(d+1)} \leq 3 \qquad \epsilon^{\text{II}}_{eds_1s_2} \leq 0 \tag{4.34}$$
$$e \in \mathcal{E}, d \in \mathcal{D}\backslash\{1, |\mathcal{D}|-1, |\mathcal{D}|\}, s_1 \in \mathcal{S}^W_e, s_2 \in \mathcal{S}^{S2}_{s_1}$$

Finally, requirements on the number of strict days off assigned are enforced by constraints (4.35). At least $\underline{N}^S$ strict days off are required over any period of D^S days.

$$\sum_{d'=d}^{d+D^S-1} s_{ed'} \geq \underline{N}^S \qquad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - D^S + 1 \tag{4.35}$$

The introduction of constraints (4.32)-(4.34) in the RMP lead to the subtraction of terms (4.36)-(4.38) from the objective function of SP(e). $\epsilon^O_{ed} \leq 0$, $\epsilon^{\text{I}}_{eds_1s_2} \leq 0$ and $\epsilon^{\text{II}}_{eds_1s_2} \leq 0$ are the dual variables from constraints (4.32), (4.33) and (4.34) respectively. Since the s-variables only appear in the RMP, constraints (4.35) do not lead to any terms in SP(e).

$$-\sum_{d\in\mathcal{D}} \epsilon^{\text{O}}_{ed} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in S^O} y_{edij} \tag{4.36}$$

$$\sum_{d\in\mathcal{D}\backslash\{1,|\mathcal{D}|\}} \sum_{j_1\in\mathcal{S}^W_e} \sum_{j_2\in\mathcal{S}^{S1}_{j_1}} \epsilon^{\text{I}}_{edj_1j_2} \sum_{i\in\mathcal{S}_e\cup\{0\}} \left(y_{e(d-1)ij_1} + y_{e(d+1)ij_2}\right) \tag{4.37}$$

$$\sum_{d\in\mathcal{D}\backslash\{1,|\mathcal{D}|-1,|\mathcal{D}|\}} \sum_{j_1\in\mathcal{S}^W_e} \sum_{j_2\in\mathcal{S}^{S2}_{j_1}} \epsilon^{\text{II}}_{edj_1j_2} \sum_{i\in\mathcal{S}_e\cup\{0\}} \left(y_{e(d-1)ij_1} + y_{e(d+2)ij_2}\right) \tag{4.38}$$

Workload
Requirements on employee workload are handled in the RMP since the desire to neither go over nor under the contracted number of hours typically slows down the computation time when solving an SPPRC with a dynamic programming labelling algorithm due to conflicting dominance criteria. The labelling algorithm and notion of dominance are discussed in Section 5.3.

The norm periods, over which requirements on the total workload apply, are modeled using the subset $\mathcal{D}^N \subset \mathcal{D}$ containing the first day of each norm period. Based on a roster line solution, the total workload over a norm period starting on day d, D_{ekd}, can be calculated as $D_{ekd} = \sum_{d'=d}^{d+N^N-1} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{s\in\mathcal{S}^W_e} T_s\overline{y}_{ekd'is}$. N^N is the number of days in the norm period and T_s is the effective workload of shift type s. Note that T_s may be unequal to the difference between T^E_s and T^S_s. This happens in cases where effective workload is not the same as total hours of the shift, e.g. if the lunch break is not included in the workload calculations. The parameter W^N denotes the number of weeks in a norm period and H^W_e the contracted weekly workload of employee e. To allow deviations from the contracted workload, variables u^+_{ed} and u^-_{ed} are introduced to denote the total over- and undertime for employee e over the norm period starting on day $d \in \mathcal{D}^N$. Constraints (4.39) ensure the contracted workload is assigned, or the correct over- or undertime is recorded.

$$\sum_{k\in\mathcal{K}_e} D_{ekd}\lambda_{ek} - u^+_{ed} + u^-_{ed} = W^N H^W_e \qquad e \in \mathcal{E}, d \in \mathcal{D}^N \quad | \quad \zeta_{ed} \in \mathbb{R} \tag{4.39}$$

Over- and undertime is penalized with hourly costs C^{N+}_e and C^{N-}_e respectively for employee $e \in \mathcal{E}$ in the RMP objective function by adding terms (4.40) and (4.41).

$$\sum_{e\in\mathcal{E}} C^{N+}_e \sum_{d\in\mathcal{D}^N} u^+_{ed} \tag{4.40}$$

$$\sum_{e\in\mathcal{E}} C^{N-}_e \sum_{d\in\mathcal{D}^N} u^-_{ed} \tag{4.41}$$

In addition to the penalties for over- and undertime, there are hard limits that cannot be exceeded. These are $\overline{U}^+_e$ for overtime and $\overline{U}^-_e$ for undertime for employee e, and enforced in the variable restrictions $0 \leq u^+_{ed} \leq \overline{U}^+_e$ and $0 \leq u^-_{ed} \leq \overline{U}^-_e$, $e \in \mathcal{E}, d \in \mathcal{D}$.

The introduction of constraints (4.39) in the RMP lead to the subtraction of terms (4.42) from the SP(e) objective function. The ζ_{ed}-variables are the dual variables of the constraints.

$$\sum_{d\in\mathcal{D}^N} \zeta_{ed} \sum_{d'=d}^{d+N^N-1} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in S^W_e} T_j y_{ed'ij} \tag{4.42}$$

Patterns

Pattern constraints are handled both in the SPs and the RMP. Patterns with lengths of three days or more are handled in the RMP as this showed best results in the preparatory research project. Two-day patterns are on the other hand handled in the SPs because they involve a simple network modification. The model is expected to scale well with added two-day patterns since it represents a restriction of SP solution spaces and thus reduces the number of solutions that are considered. The sets $\mathcal{P}^I_e$, $\mathcal{P}^P_e$ and $\mathcal{P}^R_e$, are introduced to model illegal, penalized and rewarded patterns respectively for employee $e \in \mathcal{E}$. The patterns each have a duration of D^P_{ep} days and are represented by the parameters $M_{epd'g}$, indicating if a shift type from shift group g is associated with day d' of pattern p for employee e. Note that d' is with reference the start of the pattern, meaning e.g. $d' = 2$ denotes the second day of the pattern. Some patterns may be specified to only start on certain weekdays. The set $\mathcal{B}_{ep}$ contains all weekdays on which pattern p for employee e may start. To record whether a pattern is worked by an employee, the binary variable m_{epd} is introduced for all patterns with length of three days or more. The variable indicates if employee e worked pattern p starting on day d (relative to the planning period). Since illegal patterns shall never be worked, m_{epd} is defined for $p \in \mathcal{P}^P_e \cup \mathcal{P}^R_e$, $D^P_{ep} \geq 3$.

First, the handling of two-day patterns in the SPs is described. Illegal patterns are handled by removing all arcs corresponding to the pattern, much like how rest and weekend requirements were handled. The constraints that are added to the SPs are stated in constraints (4.43). For penalized and rewarded patterns, the corresponding penalty, P_{ep}, or reward, R_{ep}, is added to the arc of the pattern and gives the added terms (4.44) for penalized patterns and (4.45) for rewarded patterns to the objective function of each SP. Each time such a pattern is worked, the corresponding penalty or reward is also added to or subtracted from the roster line cost C^K_{ek}.

$$\begin{gathered}\sum_{g\in\mathcal{G}}\sum_{g'\in\mathcal{G}}\sum_{i\in\mathcal{S}^g}\sum_{j\in\mathcal{S}^{g'}} M_{ep1g}M_{ep2g'}y_{e(d+1)ij}=0\\ p\in\mathcal{P}^I_e, D^P_{ep}=2, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\le|\mathcal{D}|-1\end{gathered}\tag{4.43}$$

$$\sum_{\substack{p\in\mathcal{P}^P_e\\ D^P_{ep}=2}}\sum_{\substack{d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b\\ d\le|\mathcal{D}|-1}}\sum_{g\in\mathcal{G}}\sum_{g'\in\mathcal{G}}\sum_{i\in\mathcal{S}^g}\sum_{j\in\mathcal{S}^{g'}} P_{ep}M_{ep1g}M_{ep2g'}y_{e(d+1)ij}\tag{4.44}$$

$$-\sum_{\substack{p\in\mathcal{P}^R_e\\ D^P_{ep}=2}}\sum_{\substack{d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b\\ d\le|\mathcal{D}|-1}}\sum_{g\in\mathcal{G}}\sum_{g'\in\mathcal{G}}\sum_{i\in\mathcal{S}^g}\sum_{j\in\mathcal{S}^{g'}} R_{ep}M_{ep1g}M_{ep2g'}y_{e(d+1)ij}\tag{4.45}$$

In the following, handling of patterns with a duration of at least three days in the RMP is described. An illegal pattern $p\in\mathcal{P}^I_e, D^P_{ep}\ge 3$ is prohibited through constraints (4.46) by considering periods of D^P_{ep} consecutive days and allowing at most $D^P_{ep}-1$ shifts in the pattern to be worked during the period. Constraints (4.46) are only enforced for days representing the weekdays on which the pattern can begin. An example of how the product of $M_{epd'g}$ and $A^g_{ek(d+d'-1)}$ indicates if employee e works a shift type from the shift group on the d'th day of pattern p starting on day d is shown after the presentation of constraints for all patterns are formulated.

$$\begin{gathered}\sum_{k\in\mathcal{K}_e}\sum_{d'=1}^{D^P_{ep}}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek}\le D^P_{ep}-1\\ e\in\mathcal{E}, p\in\mathcal{P}^I_e, D^P_{ep}\ge 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\le|\mathcal{D}|-D^P_{ep}+1\end{gathered}\quad\Bigg|\quad \theta^I_{epd}\le 0\tag{4.46}$$

Constraints (4.47) handle penalized patterns. Compared to constraints (4.46), the binary variable m_{edp} is added to the right hand side. This allows for the pattern to be worked if the binary variable is one. The term (4.48) is added to the objective function to give the penalty P_{ep} if penalized pattern p is worked by employee e starting on day d.

$$\begin{array}{c|c} \sum_{k\in\mathcal{K}_e}\sum_{d'=1}^{D^P_{ep}}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \leq D^P_{ep} + m_{epd} - 1 & \\ e\in\mathcal{E}, p\in\mathcal{P}^P_e, D^P_{ep}\geq 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1 & \theta^P_{epd}\leq 0 \end{array} \tag{4.47}$$

$$\sum_{e\in\mathcal{E}}\sum_{\substack{p\in\mathcal{P}^P_e\\ D^P_{ep}\geq 3}} P_{ep} \sum_{\substack{d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b\\ d\leq|\mathcal{D}|-D^P_{ep}+1}} m_{epd} \tag{4.48}$$

Constraints (4.49) handle rewarded patterns and resemble constraints (4.46) and (4.47), but with the inequality reversed. The variable m_{epd} is only allowed to take value one if the entire pattern is worked. The reward R_{ep} for employee e working pattern p is included in the RMP objective function by subtracting term (4.50).

$$\begin{array}{c|c} \sum_{k\in\mathcal{K}_e}\sum_{d'=1}^{D^P_{ep}}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \geq D^P_{ep}m_{epd} & \\ e\in\mathcal{E}, p\in\mathcal{P}^R_e, D^P_{ep}\geq 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1 & \theta^R_{epd}\geq 0 \end{array} \tag{4.49}$$

$$\sum_{e\in\mathcal{E}}\sum_{\substack{p\in\mathcal{P}^R_e\\ D^P_{ep}\geq 3}} R_{ep} \sum_{\substack{d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b\\ d\leq|\mathcal{D}|-D^P_{ep}+1}} m_{epd} \tag{4.50}$$

Tighter formulation for rewarded patterns

The formulation in (4.49) produces weak LP-bounds because the m-variables may take fractional values if only parts of the rewarded pattern is worked. A tighter formulation is presented in (4.51) that includes one constraint for each day of a rewarded pattern. If an integer λ-solution is found, the new formulation ensures that the m-variables are binary because the left hand side of the constraints is always binary. This tightens the LP bound and removes the need for branching on m-variables to find integer solutions in the solution process further explained in Section 5.4. Note that a similar reformulation is not required for penalized patterns since the corresponding m-variables are penalized in the objective function and have a lower bound of zero or one for an integer λ-solution in (4.47).

$$\begin{array}{c|c} \sum_{k\in\mathcal{K}_e}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \geq m_{epd} & \\ e\in\mathcal{E}, p\in\mathcal{P}^R_e, D^P_{ep}\geq 3 & \theta^R_{epdd'}\geq 0 \\ d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1, d'\in\{1,\ldots,D^P_{ep}\} & \end{array} \tag{4.51}$$

The introduction of constraints (4.46), (4.47) and (4.51) in the RMP lead to the subtraction of terms (4.52)-(4.54) from the SP(e) objective function. $\theta^I_{epd} \leq 0$, $\theta^P_{epd} \leq 0$ and $\theta^R_{epdd'} \geq 0$ are the dual variables from constraints (4.46), (4.47) and (4.51) respectively.

$$\sum_{\substack{p \in \mathcal{P}^I_e \\ D^P_{ep} \geq 3}} \sum_{\substack{d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - D^P_{ep} + 1}} \theta^I_{epd} \sum_{d'=1}^{D^P_{ep}} \sum_{g \in \mathcal{G}} M_{epd'g} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^g_e} y_{e(d+d'-1)ij} \tag{4.52}$$

$$\sum_{\substack{p \in \mathcal{P}^P_e \\ D^P_{ep} \geq 3}} \sum_{\substack{d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - D^P_{ep} + 1}} \theta^P_{epd} \sum_{d'=1}^{D^P_{ep}} \sum_{g \in \mathcal{G}} M_{epd'g} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^g_e} y_{e(d+d'-1)ij} \tag{4.53}$$

$$\sum_{\substack{p \in \mathcal{P}^R_e \\ D^P_{ep} \geq 3}} \sum_{\substack{d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - D^P_{ep} + 1}} \sum_{d'=1}^{D^P_{ep}} \theta^R_{epdd'} \sum_{g \in \mathcal{G}} M_{epd'g} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^g_e} y_{e(d+d'-1)ij} \tag{4.54}$$

In addition to registering the beginning of a working pattern, there must be some convention on how to handle overlapping of the same rewarded pattern. Constraints (4.55) ensure that a pattern cannot be recorded again before it is completed. If the employee for example works five consecutive morning shifts (M-M-M-M-M), only one of the two overlapping four day consecutive morning shift patterns (M-M-M-M) is rewarded. Without these constraints, the fifth morning shift worked in the example above would give unreasonably high reward. The m-variables only exist for $d \leq |\mathcal{D}| - D^P_{ep} + 1$ because a pattern cannot start so late that it is not finished before the planning period is over. In constraints (4.55), there is summation $D^P_{ep} - 2$ days ahead, thus the constraints are only defined for $d \leq |\mathcal{D}| - 2D^P_{ep} + 3$.

$$\sum_{d'=d}^{d+D^P_{ep}-2} m_{epd'} \leq 1 \qquad \begin{array}{r} e \in \mathcal{E}, p \in \mathcal{P}^R_e, D^P_{ep} \geq 3, d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - 2D^P_{ep} + 3 \end{array} \tag{4.55}$$

Since the m_{epd}-variables only appear in the RMP, constraints (4.55) do not lead to any terms in SP(e).

Some patterns may be sub-patterns of other super-patterns (as in the example of sub-pattern M-M-M-M of super-pattern M-M-M-M-M). The convention chosen to handle this is that the reward/penalty for the super-pattern is set to be the marginal benefit/cost relative to the sub-pattern. As an example, if the benefit of working four consecutive morning shifts (M-M-M-M) is considered to be $4\,000$ and the benefit of working five (M-M-M-M-M) consecutive morning shifts is considered to be $6\,000$, the reward for M-M-M-M is set to $4\,000$ and the reward for M-M-M-M-M is set to $2\,000$. When registering patterns with the m_{epd} variable, both patterns M-M-M-M and M-M-M-M-M are registered, and the total reward accumulates to $6\,000$.

When faced with overlapping patterns (for example D-D-E and D-E-N, overlapping on D-E), a possible solution is to define a super-pattern (in the example D-D-E-N) with reward/penalty corresponding to the perceived marginal benefit/cost. In some cases, the overlapping patterns may be registered individually without defining a super-pattern with benefit/cost correction. This depends on how working patterns are rewarded or penalized in practice.

An illustrative example
Consider the penalized pattern N-E-M, i.e. a night shift followed by an evening shift, then followed by a morning shift. If this is pattern $p = 1$ for all employees $e \in \mathcal{E}$ and has a penalty, constraints (4.47) for this pattern are as stated in constraints (4.56). It is assumed that this pattern can be started on any day during the week.

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=1}^{3} \sum_{g \in \mathcal{G}} M_{e1d'g} A^{g}_{ek(d+d'-1)} \lambda_{ek} \leq 2 + m_{e1d} \quad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - 2 \tag{4.56}$$

The left hand hand side will only count ones whenever a shift from the pattern is worked. All other shifts worked will be canceled by a zero M-parameter. All three day periods are checked, and the variable m_{e1d} is only forced to 1 if all three shifts in the pattern are worked, starting on day d. As an example of the constraints for this penalized pattern, consider the constraints for $d = 1$ stated in (4.57). The variables m_{e11} registering the pattern are only forced to 1 if the whole pattern is worked.

$$\sum_{k \in \mathcal{K}_e} (A^{N}_{ek1} + A^{E}_{ek2} + A^{M}_{ek3}) \lambda_{ek} \leq 2 + m_{e11} \quad e \in \mathcal{E} \tag{4.57}$$

The solution methods used to solve the mathematical model formulated in this chapter are discussed in Chapter 5.

Chapter 5

Solution Method

The solution methods applied to solve the mathematical model formulated in Chapter 4 are presented in this chapter. First, the column generation algorithm is introduced in Section 5.1. Thereafter, the solution methods used to solve the restricted master problem and sub problems are discussed in Sections 5.2 and 5.3 respectively. To obtain an integer solution, a branch and price framework is introduced in Section 5.4. Ultimately, a selection of speed-up techniques to reduce the solution time are presented in Section 5.5.

5.1 Column Generation

A CG algorithm is used to solve the decomposed model in Chapter 4 with the integer requirements of the RMP relaxed. The general algorithm is presented in the following and summarized in Algorithm 5.1. It is inspired by the description of CG by Lundgren et al. (2010).

In CG, columns are added to a restricted set of columns $\mathcal{K}$ until an optimal solution is found. In each iteration i, the relaxation of the is solved to obtain a solution $\lambda^{(i)}$ and dual variables $\Delta^{(i)}$. Then, each SP is solved with these dual variables to find one or more column(s) with negative reduced cost(s). To simplify notation, it is in Algorithm 5.1 assumed that the column with the most negative reduced cost is chosen. If a column with negative reduced cost is found, it is added to the set $\mathcal{K}$ in an attempt to improve the RMP solution. When no columns with negative reduced cost can be generated in the SPs, the RMP solution is LP optimal and the algorithm terminates.

As discussed in Chapter 4, there is one SP for each employee, meaning the set of SPs can be indexed by the set of employees, $\mathcal{E}$, i.e. $\{\text{SP}_e, e \in \mathcal{E}\}$. In a similar manner, the set $\mathcal{K}_e$ is used to denote the RMP columns associated with employee e. It follows that $\mathcal{K} = \bigcup_{e \in \mathcal{E}} \mathcal{K}_e$.

Algorithm 5.1: Column Generation Algorithm

```
1  Construct an initial set of columns $\mathcal{K}$ that create an LP feasible RMP
2  Initialize continue $\leftarrow$ True, $i \leftarrow 0$ and $k_e \leftarrow |\mathcal{K}_e|, e \in \mathcal{E}$
3  while continue $=$ True do
4      $i \leftarrow i + 1$
5      Solve the LP relaxation of the RMP with columns $\mathcal{K}$ and obtain the solution $\lambda^{(i)}$ and
         dual variables $\Delta^{(i)}$
6      continue $\leftarrow$ False
7      for $e \in \mathcal{E}$ do
8          Solve SP$_e$ with $\Delta^{(i)}$ and obtain new column $(e, k)$ with reduced cost $\overline{c}_{ek}^{(i)}$
9          if $\overline{c}_{ek}^{(i)} < 0$ then
10             Add column $(e, k)$ to $\mathcal{K}$
11             $k_e \leftarrow k_e + 1$
12             continue $\leftarrow$ True
13         end
14     end
15 end
16 Obtain optimal solution $\lambda^{(i)}$
```

An initial restricted set of columns $\mathcal{K}$ is needed prior to CG to ensure the RMP is LP feasible in the first iteration. This is critical because LP feasibility is required such that dual variables may be calculated from the RMP solution and columns may be generated. To create an initial set of columns, a construction heuristic is proposed. The heuristic uses CG with artificial variables that relax all hard constraints of the RMP, much like the suggested procedure by Dantzig and Wolfe (1961). Initial columns are generated by solving the SPs without dual variables. As this set of SP feasible columns is not necessarily RMP feasible, each potentially binding hard constraint in the RMP is relaxed with one positive and unbounded slack variable in the case of inequality constraints, or two in the cases of equality constraints. The RMP objective function is substituted with the sum of these slack variables. A CG approach similar to that in Algorithm 5.1 is then applied with these modified constraints and the new objective function. Once an RMP solution with objective zero is found, the solution is RMP feasible, and the columns with a positive value in the optimal solution are kept as an initial set of columns for the CG algorithm. All other columns are discarded.

5.2 Solving the Restricted Master Problem

The RMP formulated in Chapter 4 is solved with the *FICO Xpress* solver version 8.5 via the associated Python 3.7 interface. To solve the LP relaxation of the RMP in each iteration of CG, Xpress uses an algorithm based on dual simplex. When solving the RMP with integer requirements, Xpress applies a branch and bound method with cutting planes.

5.3 Solving the Sub Problems

Each SP was formulated as a shortest path problem with resource constrains on an acyclic directed graph in Section 4.2. A forward dynamic programming method using a labelling algorithm is proposed to solve the SPs in this section. This is one of the most common methods for solving SPPRCs together with alternatives like Lagrangian relaxation, constraint programming and heuristics (Irnich and Desaulniers, 2005). The labeling algorithm begins in the start node, and traverses through the network while following specific decision rules to ultimately reach the end node with minimum cost through a feasible path. An introduction to dynamic programming and the chosen labelling algorithm is given in the following. Thereafter the dominance criteria used to disregard non-optimal paths in the algorithm are discussed.

5.3.1 Dynamic Programming and Labelling

Dynamic programming is based on the *Principle of Optimality*, introduced by Bellman (1954). The principle states that "an optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions." It implies for the shortest path problem considered here that for any node on a shortest path from the start node to the end node, the shortest path from this node to the end node coincides with the shortest path from the start node to the end node. To find all shortest paths from the start node to any node in the network, and ultimately the end node, a recursive labelling algorithm is proposed. The labelling algorithm used in this thesis resembles that of Irnich and Desaulniers (2005). The standard version of the labelling algorithm applied is described in the following and displayed in Algorithm 5.2. This algorithm is slightly modified to enable a speed-up utilizing the acyclic properties of the graph in Section 5.5.3.

A label L is a function of a path P, denoted $L = \mathcal{L}(P)$. The label contains information about the current node and the path taken, as well as the cost and resource values accumulated over the path. A label L is represented by eq. (5.1), where $T(P)$ is the resource vector and v_p is the last node in the path P.

$$L = \mathcal{L}(P) = (C(P), T(P), v_p, P) \tag{5.1}$$

Let $\mathcal{U}$ be a set of unprocessed labels, and $\mathcal{P}$ a set of processed labels. The first step in the algorithm is to create an initial label $L_0 = (0, T(P_0), v_0, (v_0))$ with zero cost, initial resource vector $T(P_0)$ and start node v_0. This label is the first unprocessed label and is thus added to $\mathcal{U}$. $\mathcal{P}$ is initialized to be an empty set.

Until there are no more unprocessed labels in $\mathcal{U}$, a label in the network is selected and removed from $\mathcal{U}$. All feasible extensions of the label are then created by traversing the arcs departing from the current node of the label, updating the cost and resources according to the recursion formulas described in Section 4.3, and checking feasibility in the destination node. For an acyclic graph, always selecting the oldest among the unprocessed labels ensures that the selected label will not be dominated later in the algorithm. This is because

Algorithm 5.2: Labelling Algorithm

```
1  Set $\mathcal{U} \leftarrow \{L_0\}$ and $\mathcal{P} \leftarrow \emptyset$.
2  while $\mathcal{U} \neq \emptyset$ do
3      Select and remove one label $L$ from $\mathcal{U}$.
4      Extend $L$ to all feasible extensions $\{L'_i\}$.
5      for $L' \in \{L'_i\}$ do
6          if No label in $\mathcal{U}$ and/or $\mathcal{P}$ dominates $L'$ then
7              if $L'$ dominates labels in $\mathcal{U}$ or $\mathcal{P}$ then
8                  Remove dominated labels from $\mathcal{U}$ or $\mathcal{P}$.
9              end
10             Add $L'$ to $\mathcal{U}$.
11         end
12     end
13     Add $L$ to $\mathcal{P}$.
14 end
15 Filter $\mathcal{P}$ to find optimal solution at end node.
```

all labels created at a later stage by extension will be in a different node further out in the network.

The concept of dominance is central to the labelling algorithm because it prevents full enumeration of all possible paths from the start node to the end node. A label $\mathcal{L}(P^1)$ is said to dominate label $\mathcal{L}(P^2)$ if they are at the same node, $C(P^1) < C(P^2)$ and all feasible extensions of $\mathcal{L}(P^2)$ are also feasible extensions of $\mathcal{L}(P^1)$. In this case, P^2 can never be part of an optimal path because an extension of $\mathcal{L}(P^2)$ will be more costly than the same extension of $\mathcal{L}(P^1)$. Thus, $\mathcal{L}(P^2)$ is discarded. If $\mathcal{L}(P^1)$ dominates $\mathcal{L}(P^2)$, the notation $\mathcal{L}(P^1) \prec \mathcal{L}(P^2)$ is used.

After extending a label in line 4 of Algorithm 5.2, dominance is checked against the unprocessed and processed labels. Extended labels that are not found to be dominated by labels in $\mathcal{U}$ or $\mathcal{P}$ are added to $\mathcal{U}$. If any label in $\mathcal{U}$ or $\mathcal{P}$ is found to be dominated by an extended label, the dominated label is removed from the respective set. After having checked for dominance among all extended labels, the label that was extended is added to $\mathcal{P}$ as a processed label. When all labels in $\mathcal{U}$ have been considered, $\mathcal{P}$ contains all labels that cannot be dominated. The labels in $\mathcal{P}$ are non-dominated and represent feasible paths or roster lines if they are in the end node. Filtering out these labels yields solutions to the SP. Since resource values at the end of the planning period are of no importance, i.e. end effects are neglected, the label with the lowest cost in the end node is filtered out and renders the optimal solution.

5.3.2 Dominance Criteria

To complete the description of the labelling algorithm, it is necessary to specify which criteria must be satisfied to ensure that all feasible extensions of some label $\mathcal{L}(P^2)$ are also feasible extensions of another label $\mathcal{L}(P^1)$. These criteria will determine whether $\mathcal{L}(P^1)$ can dominate $\mathcal{L}(P^2)$. Recall from Section 4.2 that a path is feasible if it is resource-feasible. It is thus required that all resource-feasible extended paths from $\mathcal{L}(P^2)$ are also resource-feasible if extended from $\mathcal{L}(P^1)$. This yields dominance criteria for each of the resources introduced in Section 4.3. The resources extension functions (REFs) used are separable and non-decreasing in the resource value, leading to quite simple dominance criteria. Desaulniers et al. (1998) prove that for this type of resources, dominance reduces to a simple comparison of resource values in a label. In the following, the dominance criteria for each resource are discussed before all dominance criteria are summarized.

Minimum consecutive days working and working specific shift groups
The resources $T^{\underline{W}}$ and $T^{\underline{W}^g}, g \in \mathcal{G}$ count consecutive days and were introduced in Section 4.3 to enforce constraints on minimum consecutive days working and minimum consecutive days working a shift group. Since the resources are similar in structure, they are discussed together. For all extended paths of $\mathcal{L}(P^2)$ to also be feasible extensions of $\mathcal{L}(P^1)$, $\mathcal{L}(P^1)$ must have at least as high a value of $T^{\underline{W}}$ and $T^{\underline{W}^g}, g \in \mathcal{G}$, as $\mathcal{L}(P^2)$, or $\mathcal{L}(P^1)$ must have a value of $T^{\underline{W}}$ and $T^{\underline{W}^g}, g \in \mathcal{G}$, reflecting consecutive days working above the minimum requirements. This is specified by criteria (5.2) and (5.3).

$$T^{\underline{W}}(P^1) \geq \underline{N} \quad \vee \quad T^{\underline{W}}(P^1) \geq T^{\underline{W}}(P^2) \tag{5.2}$$

$$T^{\underline{W}^g}(P^1) \geq \underline{N}^g \quad \vee \quad T^{\underline{W}^g}(P^1) \geq T^{\underline{W}^g}(P^2) \quad g \in \mathcal{G} \tag{5.3}$$

The validity of these dominance criteria comes from that the resource windows for all nodes are bounded from below and unbounded from above for these resources. Together with the non-decreasing property, this implies that the magnitude of the resource value is not of importance as long as it is above the lower bound of the resource windows. As a result, if $\mathcal{L}(P^1)$ has $T^{\underline{W}}(P^1) \geq \underline{N}$ and $T^{\underline{W}^g}(P^1) \geq \underline{N}^g, g \in \mathcal{G}$, the resource value of $\mathcal{L}(P^2)$ does not limit which feasible extended paths from $\mathcal{L}(P^2)$ are also feasible extension of $\mathcal{L}(P^1)$. Also, if $\mathcal{L}(P^1)$ has weakly higher values of $T^{\underline{W}}$ and $T^{\underline{W}^g}, g \in \mathcal{G}$, than $\mathcal{L}(P^2)$, any feasible path taken by extending $\mathcal{L}(P^2)$ can also be taken by extending $\mathcal{L}(P^1)$. This follows from the fact that all lower bounded resource windows satisfied by extensions $\mathcal{L}(P^2)$ are also satisfied by extensions of $\mathcal{L}(P^1)$ following the same path.

Weekends
The resources $T^{V(t)}$, $t \in \{1, \dots, W^W\}$, were introduced in Section 4.3 to enforce constraints on the maximum number of weekends worked over a period of W^W weeks. Recall that the resources count the number of weekends worked and restrict this number by resource windows with an upper bound. Together with the fact that these are separable and non decreasing, it is proven by Desaulniers et al. (1998) that all extended paths of $\mathcal{L}(P^2)$ are feasible extensions of $\mathcal{L}(P^1)$ if $\mathcal{L}(P^1)$ has at most as high values of $T^{V(t)}$, $t \in \{1, \dots, W^W\}$, as $\mathcal{L}(P^2)$. This is specified by criteria (5.4).

$$T^{V(t)}(P^1) \leq T^{V(t)}(P^2) \quad t \in \{1, \dots, W^W\} \tag{5.4}$$

Recall from Section 5.3.1 that the first two dominance criteria are that the compared labels must be in the same node, and that the cost of $\mathcal{L}(P^1)$ is less than that of $\mathcal{L}(P^2)$. Combined with the dominance criteria related to resource-feasibility, the dominance criteria are as summarized in Box (5.5).

$$
\begin{aligned}
v(P^1) &= v(P^2) \\
C(P^1) &< C(P^2) \\
T^{\underline{W}}(P^1) \geq \underline{N} \quad &\vee \quad T^{\underline{W}}(P^1) \geq T^{\underline{W}}(P^2) \\
T^{\underline{W}^g}(P^1) \geq \underline{N}^g \quad &\vee \quad T^{\underline{W}^g}(P^1) \geq T^{\underline{W}^g}(P^2) \quad g \in \mathcal{G} \\
T^{V(e)}(P^1) &\leq T^{V(e)}(P^2) \quad e \in \{0, \ldots, W^W - 1\}
\end{aligned}
\tag{5.5}
$$

5.4 Branch and Price

When solving CG, the integer property of the RMP is relaxed. To find the integer feasible optimal solution of the healthcare rostering problem, a branch and price algorithm using constraint branching is applied. Branch and price is a specialized version of the branch and bound algorithm where CG is solved in each node of the branching tree. Barnhart et al. (1998) present a thorough and unifying article on the branch and price procedure and, as highlighted in Chapter 2, the method is widely used in the literature when solving rostering problems. For a general introduction to branch and bound, Lundgren et al. (2010) is recommended. In this section, the specifics of the applied branch and price algorithm are presented.

5.4.1 Algorithm

The general branch and price algorithm applied in this thesis is displayed in Algorithm 5.3. The algorithm is built on the branch and bound algorithm presented by Lundgren et al. (2010) which is based on the work of Land and Doig (1960) and Dakin (1965). In each node of the branch and bound tree, CG is used to solve the RMP to LP optimality. The use of CG with an RMP and associated pricing problem[1] gives branch and price its name. Barnhart et al. (1998) argue that although branch and price is a combination of the well studied branch and bound and CG frameworks, the procedure is not as straightforward as would be expected. They highlight that conventional branching on variables often prove ineffective, and that solving LPs to optimality in each node with CG might not be the best procedure. The algorithm is therefore slightly generalized in Algorithm 5.3 compared to the version in Lundgren et al. (2010) by not specifying the branching strategy. Also, line 21 is added to update the dual bound in each iteration. By carefully choosing the branching strategy as described in Section 5.4.2 and not solving LPs to optimality by heuristic speed-ups in Section 5.5, the issues pointed out by Barnhart et al. (1998) are addressed.

[1]In this thesis, the term sub problem is used for the pricing problem.

Algorithm 5.3: Branch and Price Algorithm

```
Set $\overline{z} \leftarrow \infty$, or if an integer feasible solution $\hat{\lambda}$ is known, set $\overline{z} \leftarrow c^T\hat{\lambda}$.
Initialize $i \leftarrow 0, n \leftarrow 0, terminate \leftarrow False$.
while terminate $= False$ do
    Solve problem $P_i$ by column generation. Obtain solution $\lambda^{P_i}$ and objective value $z^{P_i}$.
    Solve the RMP of $P_i$ with integer requirements to obtain integer solution $\lambda^{P_i,IP}$ with
      objective value $z^{P_i,IP}$ if feasible.
    if $P_i$ infeasible or $z^{P_i} > \overline{z}$ then
        Prune $P_i$.
    else if $\lambda^{P_i,IP}$ exists then
        if $z^{P_i,IP} < \overline{z}$ then
            $\overline{z} \leftarrow z^{P_i,IP}$, $\hat{\lambda} \leftarrow \lambda^{P_i,IP}$.
            Prune all problems $P_j$ with $z^{P_j} \geq \overline{z}$.
        end
        if $\lambda^{P_i} = \lambda^{P_i,IP}$ then
            Prune $P_i$.
        end
    else
        Branch to create two new problems $P_{n+1}$ and $P_{n+2}$.
        $n \leftarrow n + 2$
        Mark that $P_i$ is solved.
    end
    $\underline{z} \leftarrow \max\{z^{P_i} | P_i$ is in a leaf node$\}$.
    if termination criterion $= True$ then
        terminate $\leftarrow True$.
    else
        Update $i$ according to a chosen search strategy.
    end
end
```

The idea of branch and bound is to first solve the original problem with the integer requirements relaxed. This problem is referred to as the root node, and is solved with CG in branch and price. Should the solution be fractional, this solution is cut out of the solution space by branching the problem into two new problem nodes that divide the solution space in two without removing integer feasible solutions. This process continues until only integer solutions remain or some termination criterion is fulfilled. During the algorithm, the branching tree grows in size, and it is critical to prune nodes, i.e. stop searching an area of the tree if it has no potential of finding more optimal solutions. For more efficient notation, an illustration of the branching tree with important terms is displayed in Figure 5.1.

In the first steps of the algorithm, an upper bound $\overline{z}$ is set to the objective value of the best available integer feasible solution. If no such solution is known, the bound is set to ∞. The current node i is initialized with value 0, referring to the root node. In each iteration of the algorithm, the problem P_i is solved with CG. The root node problem, P_0, is to solve the model described in Chapter 4 with the integrality of the λ-variables relaxed. Subsequent problems, $P_i, i = 1, 2, \ldots$, start out with a subset of columns from the parent problem, and include added constraints to enforce branching.

If P_i has no feasible solution or the optimal objective value z^{P_i} is higher than the best upper bound, searching this node's children nodes cannot yield improving solutions because branching restricts the problem, and thus reduces the solution space. In this case, the node is pruned, and another area of the branch and bound tree is searched. If instead, solving the RMP with integer requirements yields an integer feasible solution $\lambda^{P_i,IP}$ for the set of columns generated that is better than the previously best found integer solution, the upper bound $\overline{z}$ and the best found solution $\hat{\lambda}$ are updated. The updated bound is stronger than before and is used for further pruning of existing nodes. If the LP optimal solution to problem P_i was integer, i.e. $\lambda^{P_i} = \lambda^{P_i,IP}$, the problem is pruned. In many cases, none of these pruning criteria are met, and the node must be branched into two new nodes available for further search. This creates two new problems on a lower level in the tree. The branching strategy is discussed in Section 5.4.2.

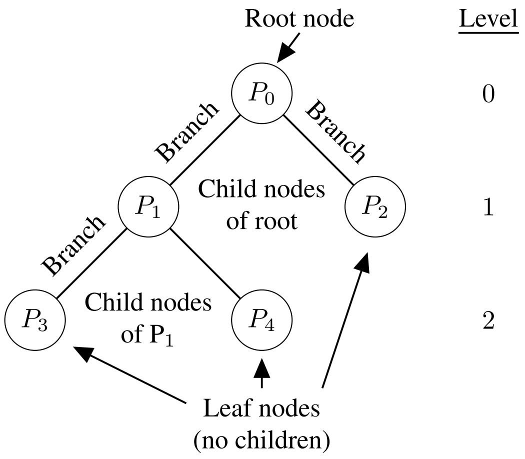


Figure 5.1: Illustration of the branching tree with relevant terms used to describe different tree elements.

After solving a node, the lower bound $\underline{z}$ is updated to be the lowest value of z^{P_i} across all leaf nodes. A leaf node is a node that has not been branched into new nodes. Because branching always restricts the problem, no better solution than this bound can be found. The lower bound can be used in the branch and price termination criterion. Termination criteria are discussed in Section 5.4.4. Unless one of the the termination criteria is fulfilled, a new node to process is selected based on a search strategy, as discussed in Section 5.4.3.

5.4.2 Branching Strategy

Central to an efficient branch and price algorithm is the branching strategy. In regular branch and bound, variable branching is the natural choice (Lundgren et al., 2010). For branch and price on binary problems, however, branching directly on the λ-variables is inefficient. The up-branch where a λ_{ek}-variable is forced to one implies assigning roster line k to employee e. This cuts away a large portion of the solution space. The down-branch on the other hand, contains all other possible solutions. This results in a highly unbalanced branching tree, which can lead to weak bounds and slow convergence of the branch and price procedure. Another issue with this branching strategy is that the down-branch is hard to enforce in the SPs because one specific roster line must be prevented to avoid it being generated again as a column in the RMP, only with a new index.

Several branching strategies are suggested in the literature. For instance, Maenhout and Vanhoucke (2010) propose a combination of branching on the residual problem and branching on the original variables by three alternative methods (0/1 branching, 1/2/.../$|S|$ branching and constraint branching). Maenhout and Vanhoucke also highlight the variety of branching variable selection methods available (e.g. most fractional variable and largest positive non-integer variable).

In this thesis, 0/1 branching on the original x-variables is used. This is a popular choice in personnel scheduling research (Maenhout and Vanhoucke, 2010). Dohn and Mason (2013) substantiate the method to be both efficient and quite simple. It involves branching on a specific shift assignment by requiring an employee to work the shift in the up-branch, and prohibiting the employee from working the same shift in the down-branch. A shift assignment represented by variable x_{eds} is specified in the RMP through the sum $\sum_{k \in \mathcal{K}_e} A_{ekds}\lambda_{ek}$, which is forced to zero in the down-branch and one in the up-branch, thus representing a form of constraint branching. This is illustrated in Figure 5.2. The x-variable closest to 0.5 is chosen for branching. The rationale is that several columns from the optimal solution of the parent node will then be kept in each child node.

Instead of adding the constraints to the RMP, the branching is enforced by removing all columns including the specific shift assignment in the down-branch, and removing all other columns in the up-branch for the given employee. In the SP for the relevant employee, the variable is forced to zero by removing the corresponding node in the SPPRC network, or to one by removing all nodes corresponding to other shift assignments on the same day for the relevant employee. If the removal of columns leads to a set of columns that make the RMP LP infeasible, the construction heuristic described in Section 5.1 is applied to generate new columns until a feasible RMP is achieved or an infeasible problem is proved and the node can be pruned.

For this branching strategy, columns in the RMP and nodes in one SP are removed in each branch, thereby reducing the problem size. Together with the simple and efficient implementation, this is a great benefit of the strategy. Also, the strategy reduces the risk of an unbalanced branch and bound tree compared to branching directly on the λ-variables.

Modification of the branching on the original x-variables by fixating more than one variable in a third branch is discussed in Section 5.5.

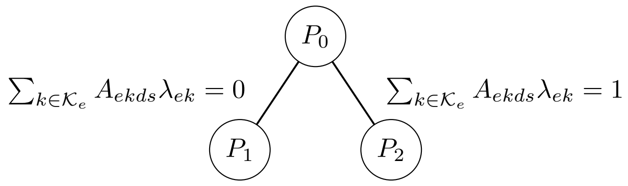

P_0
$\sum_{k\in\mathcal{K}_e} A_{ekds}\lambda_{ek} = 0$
$\sum_{k\in\mathcal{K}_e} A_{ekds}\lambda_{ek} = 1$
P_1
P_2

Figure 5.2: Illustration of branching on shift assignment (x_{eds})

5.4.3 Search Strategy

After branching, deciding which problem to solve next is critical, and is determined by the search strategy. According to Lundgren et al. (2010), standard search strategies include *depth first*, *breadth first* and *best first*. *Depth first* is associated with finding integer solution relatively quickly as the search will lead the branch and price algorithm to solve problems for which repeated branching has fixated several variables to integer values. However, branching restricts the solution space and may lead to searching deep in the branching tree without finding good solutions. This might require large computation time that could have been avoided with better pruning of nodes. This issue motivates a *breadth first* search where higher level nodes are solved first. Breadth first is found to work particularly well when integer solutions are easily found without substantial branching (Lundgren et al., 2010). *Best first* differentiates itself from depth first and breadth first by not considering the level of the node. Instead, this search strategy finds the node associated with the most promising lower bound. The idea is that a low lower bound may indicate an integer solution with a low objective value.

In this thesis, a depth first search is used until the first integer solution is found. When choosing between two children nodes in the depth first search, the node for which branching fixated a shift assignment to one is chosen. This is argued to be the simpler problem to solve since more arcs are removed from the SPPRC network when requiring certain shift assignments rather than prohibiting them.

After the first integer solution is found, the search strategy is changed to best first. This ensures the parts of the tree with highest potential for a good solution is searched. Additionally, a wider part of the solution space is searched compared to the *depth first* strategy. Lundgren et al. (2010) argue that a typical disadvantage is that the search might alternate between different parts of the tree, making it difficult to use knowledge from the parent node to solve the child node. However, this is not a major issue in branch and price, because knowledge about the parent node solution only affects the first iteration of CG in the child node, while previous RMP solutions are utilized in subsequent iterations.

5.4.4 Termination Criteria

The branch and price algorithm terminates when a certain optimality gap or a time limit is reached. Ideally, the model is solved to optimality. However, closing the gap entirely is commonly associated with large computation time expense, and an optimality gap stop criterion is commonly applied (Lundgren et al., 2010). The optimality gap is defined by eq. (5.6) based on the upper bound (UBD) and lower bound (LBD). In many cases, computation time is the limiting resource, and a time limit is therefore input to the model. The algorithm terminates at the best current solution if the time limit is reached.

$$\frac{|\text{UBD} - \text{LBD}|}{\max\{|\text{UBD}|, |\text{LBD}|\}} \tag{5.6}$$

5.5 Speed-Up Techniques

Due to long computation times experienced when solving large instances, a number of techniques are proposed to improve the solution method. These are hereafter called "speed-up techniques" and are discussed in the following.

5.5.1 Column Generation Stop Criterion

Stopping CG in each node before reaching optimality results in more frequent branching and can reduce computation time if considerable time is spent reaching optimality without improving the solution considerably (Desaulniers et al., 2002). In each iteration i of CG, a lower bound (LBD) of the optimal solution can be calculated by eq. (5.7), where $z_{RMP}^{(i)}$ is the RMP objective value and $\overline{c}_e^{(i)*}$ is the column with the lowest reduced cost for employee e upon solving all SPs to optimality.

$$LBD^{(i)} = z_{RMP}^{(i)} + \sum_{e \in \mathcal{E}} \overline{c}_e^{(i)*} \tag{5.7}$$

This is a lower bound because no variable can be pivoted into the basis of the RMP with a value larger than one. The CG stop criterion inspired by Lundgren et al. (2010) is stated in eq. (5.8), where UBD is the best upper bound found, i.e. $z_{RMP}^{(i)}$, and ϵ is an input parameter setting a threshold for the optimality gap. This criterion assures CG is stopped once the optimality gap falls below ϵ.

$$\left| \frac{UBD - LBD}{LBD} \right| \leq \epsilon \tag{5.8}$$

It should be noted that adding this stop criterion to the CG procedure turns the branch and price method into a heuristic. In line 14 of Algorithm 5.3, a node is pruned if the optimal LP solution of the RMP is integer. However, because of the optimality gap at the node, the final LP solution is not necessarily optimal and a better integer solution can in theory exist within the solution space of the node. Such better integer solutions may never be considered if the node is pruned. By setting ϵ sufficiently small in eq. (5.8), this effect can be controlled to avoid large optimality gaps in the branching tree.

After implementing some of the speed-ups suggested later in this section, a lower bound is not necessarily available in each iteration of CG because all SPs may not have been solved to optimality. To still be able to terminate the algorithm, an improvement criterion is added to the model. This criterion states that if the improvement in the RMP objective value during a specified number of iterations does not exceed a specified threshold, each SP is solved to optimality in the next iteration to calculate a lower bound and check the termination criterion. Details on the implementation of this improvement criterion are provided in Section 6.4.3.

As discussed in Section 5.4, child nodes represent restrictions of their parent node problem in the branching tree. This implies that the lower bound of the parent node problem may be used as an initial lower bound for the children node problems. This tightens the initial optimality gap of the child nodes and might reduce the computation time.

5.5.2 Retrieving Several Sub Problem Solutions

Instead of only adding the column with most negative reduced cost from each SP, several columns from each SP can be added in each iteration of CG to reach optimality quicker. Dohn and Mason (2013), for instance, find that several columns returned in each iteration gives the best results. Because dominance reduces the number of labels and hence the number of solutions available when solving the SPs, dominance is neglected in the end node when solving the SPs to promote more SP solutions. An upper limit on the number of SP solutions returned each time an SP is solved is input to the model. Sometimes this limit might not be reached because only columns found in the SPs with negative reduced cost are added to the RMP. Potential benefits of retrieving more SP solutions include a reduction in the number of iterations needed to reach optimality and thus more rapid optimality gap convergence. However, each CG iteration is likely to consume more time as the RMP grows in size.

5.5.3 Limited Label Extension

In CG, finding any column with a negative reduced cost will suffice to potentially improve the RMP solution. It may thus be more efficient to solve the SP heuristically to find solutions with negative reduced costs, rather than to find the solutions with the guaranteed most negative reduced costs. As discussed in Chapter 4, the minimum reduced cost is only needed in the last iteration of CG, when proving optimality of the RMP by guaranteeing that no roster line with negative reduced cost may be added as a column.

The labelling algorithm used to solve the SPs was discussed in Section 5.3. Recall that dominance and the discarding of non-optimal labels was central to the effectiveness of the algorithm. This speed-up limits the number of labels that are extended from each node in the SPPRC-networks to those with the lowest costs in each node, thus reducing the solution space of the SPs and the computation time. Consequentially, the solutions found by the labelling algorithm are not guaranteed to include the optimal solution. A similar approach is proposed by Burke and Curtois (2011) and Desaulniers et al. (2002). If no roster line with negative reduced cost is found in an SP after solving with such limited label extension, the number of extended labels from each node is increased and the SP is solved again. This continues until either a roster line with negative reduced cost is found or the extension limit is not binding for the optimal solution.

The limited label extension is implemented by altering Algorithm 5.2 between lines 3 and 4. First, the unprocessed labels in $\mathcal{U}$ are ordered chronologically. That is, the first label is the initial label L_0, the following n labels are the n extensions of L_0 and so on. Since all arcs in the SPPRC-network are between successive days, this asserts that upon selecting the first label L from $\mathcal{U}$, no more labels will be extended into the node of L. Thus, instead of extending L directly, the labels $\{L'\}$ in $\mathcal{U}$ that are in the same node as L are ordered according to their reduced cost, and only a subset of these labels with the most negative reduced cost are extended. The other labels in the node are discarded. The modified labelling algorithm with limited label extension is presented in Appendix B.

The increments of the label extension limit is input to the model and discussed along with the resulting effectiveness of the speed-up in Section 6.4.3.

5.5.4 Partial Column Generation

Partial CG refers to re-optimizing the RMP without solving all SPs in each iteration. Gamache et al. (1999) argue that if all SPs are solved in each iteration, they use the same dual variables and tend to generate roster lines covering the same shifts. Thus, a better approach might be to solve SPs in each iteration until a column with negative reduced cost is found. The first column with negative reduced cost to be found is then added to the RMP, and a new iteration is performed to update dual variables and solve updated SPs.

When only a subset of the SPs are solved in each iteration, a method for choosing the order of solving SPs is required. In this thesis, two alternatives are proposed and tested. The first alternative is simply to solve the SPs in a random order in each iteration, a method that hardly uses computation time for determining the order. The rationale is that the SP in which the best column may be found varies, motivating changing the order between iterations. Also, a random selection between employees and hence SPs is likely to explore a vast selection of SPs over the course of CG, thus having the chance of generating several roster line alternatives for each employee. The second order selection alternative is to solve relaxed versions of all SPs and solve them in ascending order of lower bounds. The rationale is that the SPs with most negative lower bound have the best chances of yielding columns with negative reduced cost. In the relaxed SPs, all resources are removed, yielding shortest path problems (SPPs) which can be solved efficiently with the labelling algorithm.

5.5.5 Handling Maximum Consecutive Days in the Sub Problems

The model presented in Chapter 4 handles some constraints in the RMP and some in the SPs. The allocation of constraints is based on the best configuration found for the related healthcare roster line problem discussed in the preparatory research project. However, the HRP in this thesis is more complex than the roster line problem and does not necessarily share the best configuration.

Also handling constraints on maximum consecutive days working in the SPs was found to be an alternative option with promising results in the preparatory project. This increases the computation time of the SPs because dominance is more often prevented in the dynamic programming algorithm. However, each column generated has a higher likelihood of being feasible, which might tighten the LP bound of the RMP, improve convergence and lead to better integer solutions.

To enforce constraints on maximum consecutive days working in the SPs, a new resource is introduced for each employee. The resource is denoted $T^{\overline{W}}$ and presented in Box (5.9). The resource keeps track of the number of consecutive days worked by increasing the value with one each time an arc going into a work shift is traversed, while it is set to zero initially and each time an off shift is assigned. By limiting the resource value to $\overline{N}$ in each node, no more days than this can be worked consecutively.

$$\boxed{\begin{array}{rll} \text{Resource} & T^{\overline{W}} & \\ \text{REF} & f^{\overline{W}}_{dij}(T^{\overline{W}}) = \begin{cases} T^{\overline{W}} + 1 & i \in \mathcal{S}_e, j \in \mathcal{S}^W_e, d \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases} & \\ \text{Initial value} & T^{\overline{W}}_0 = 0 & \\ \text{Resource window} & T^{\overline{W}}_{ds} \in \ [0, \overline{N}] \qquad d \in \mathcal{D}, s \in \mathcal{S}_e & \end{array}} \tag{5.9}$$

Assume label $\mathcal{L}(P^1)$ is to dominate $\mathcal{L}(P^2)$. When adding the new resource, an additional dominance criterion is added to those stated in Box (5.5). This new criterion is given by eq. (5.10). For all feasible extensions of $\mathcal{L}(P^2)$ to also be feasible for $\mathcal{L}(P^1)$, the resource value must be lower or equal for $\mathcal{L}(P^1)$. Otherwise, more consecutive working shifts may be extended for $\mathcal{L}(P^2)$ than for $\mathcal{L}(P^1)$.

$$T^{\overline{W}}(P^1) \leq T^{\overline{W}}(P^2) \tag{5.10}$$

5.5.6 Multiple Variable Aggressive Branching

As discussed in Section 5.4.2, 0/1 branching on the original x-variables is used in this thesis. In addition to branching on one shift assignment to create two branches, as suggested there, a third branch may be created in which one shift assignment is fixed for each employee. In this third branch, the highest non-integer x-variable for each employee is fixed to one. The search strategy is set to search this branch with multiple variable fixing before the two original branches. This speed-up slightly resembles the approach of Lusby et al. (2012), who argue that rostering problems contain a high level of degeneracy and apply a greedy roster line fixing where the most fractional weighting variables are fixed to one. Because many variables are fixed in each branch of the new branching strategy introduced here, the two original branches are kept to maintain the exactness of the branching tree. While the speed-up might show good ability to find integer solutions, one drawback is the increased size of the branching tree where the same solution space might be searched twice because the solutions in the additional branch also are feasible in other nodes in the branching tree.

5.5.7 Column Elimination

Because of the decomposition approach suggested in the thesis, where some employee specific constraints are handled in the RMP, columns that are generated in the SPs and take part in an optimal LP solution of the RMP might not be integer feasible. Column elimination is applied to such illegal columns prior to solving a node in the branching tree. No measures are taken to prevent these columns from being generated again in the SPs. However, because of the high degeneracy often seen in the RMP, the integer infeasible column elimination might change the solution and increase the fraction of integer feasible columns in the RMP.

In addition to the possible benefit of finding more integer solutions, a smaller set of columns might have other advantages as well. The RMP computation time tends to increase with the number of columns considered (Beliën and Demeulemeester, 2007). To reduce RMP computation time, Desaulniers et al. (2002) suggest eliminating a subset of the nonbasic columns in each node, for example selecting according to marginal cost or based on recent absence in the basis. They do, however, warn that too frequent elimination may halt convergence and that a number of nonbasic columns should always remain in the RMP. Although only integer infeasible columns are removed with the suggested speed-up here, which is therefore not a heuristic, convergence may be slower and cause an increase in the total computation time due to less available columns at the start of CG. The effects are tested in Section 6.5.2.

The solution methods described in this chapter are evaluated in the computational study in Chapter 6.

Chapter 6

Computational Study

In this chapter, a computational study of applying the solution methods described in Chapter 5 to the mathematical model formulated in Chapter 4 is presented. An important part of the underlying work of the thesis is implementation and this is discussed in Section 6.1. A detailed description of test instances used follows in Section 6.2. The instances form a sound base for testing solution methods in the rest of the chapter, beginning with the default solution method performance presented in Section 6.3. Next, the effect of column generation speed-ups are considered in Section 6.4 before speed-ups regarding branch and price are evaluated in Section 6.5 to form a suggested improved configuration that concludes the chapter with results presented in Section 6.6.

6.1 Implementation

In this section, the implementation of the model and solution methods proposed in Chapters 4 and 5 is discussed. All code is written in Python version 3.7. Python was chosen for two main reasons: (i) simple syntax not requiring much prior experience with programming and (ii) accessibility and popularity of the open source programming language. A disadvantage of the high-level Python implementation is less flexibility to customize the program to take full advantage of the problem structure and speed up the implementation. All code can be accessed from an online repository[1].

As discussed in Section 5.2, the RMP is solved using the *FICO Xpress* solver version 8.5 through the Xpress Python interface. The labelling, CG and branch and price algorithms are implemented from scratch with an object oriented approach. Implementation details, such as the choice of data structures and objects, are not discussed here. The interested reader is invited to explore this in the documented code online. However, an overview of

[1] https://github.com/sandercoates/HRP

the main logic of the implementation is presented in Figure 6.1. Note how in branching, a new set of columns and graphs are copied into each branched child node, along with a modified copy of the parent node Xpress problem object.

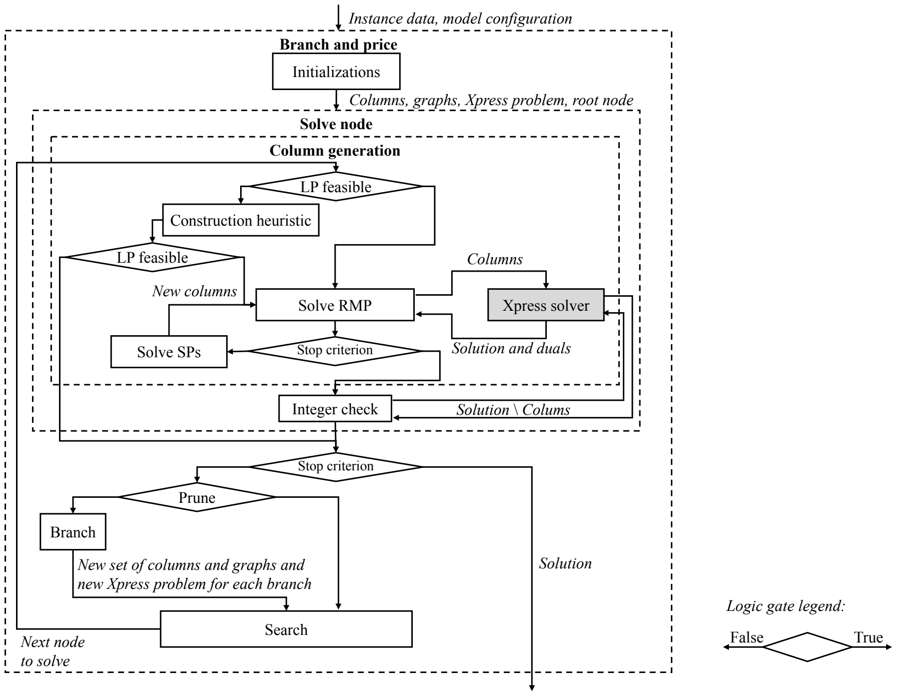


Figure 6.1: Overview of the implementation. Flow of information is annotated in italics. Logical gates have the true exit flow to the right and the false exit flow to the left. The commercial Xpress solver developed by FICO is shaded in gray to illustrate that the details of the solver implementation are not considered in this thesis.

Challenges with insufficient random access memory (RAM) were experienced in testing due to the storage of RMP column objects, Xpress problem objects and SP graphs in each unprocessed node of the branching tree. This was especially an issue in Section 6.5.1 when branching into three rather than two branches for each feasible processed problem. An alternative more lean implementation that would alleviate this memory burden is to keep only one Xpress problem object for the entire branching tree and instead only store information about which columns are accessible in each node. By variable fixing in each node according to the columns accessible, there is no need for neither RMP column objects nor Xpress problem objects in each node. However, it is uncertain whether the Xpress solver utilizes knowledge from previous solutions to the same extent in such an implementation, since the common RMP problem object is used to solve problems with potentially very different columns and solutions. Preliminary testing indicates that using knowledge about the previous solution reduces the computation time of solving RMPs. The alternative implementation was not tested in the thesis due to constrained time.

All tests were conducted on HP dl165 G6 servers with two AMD Opteron 2431 processors, 12 CPUs, 24 cores, 2.4 GHz and 24 GB RAM running on Linux. Parallel computing was not implemented. The test instances used to evaluate the performance of the thesis model and speed-ups are described in the following section.

6.2 Test Instances

Following the discussions of the problem in Chapters 1 and 3, the selection criterion for test instances is that they should resemble realistic rostering situations in Nordic healthcare institutions. In this section, the test instances used in the computational study are discussed. First, the most common benchmark instances used in the literature are presented in Section 6.2.1 before a set of instances is selected. As the benchmark instances do not contain all information needed for the model proposed in this thesis, the selected test instances are combined with estimates of the additional required parameters. This instance modification is described in Section 6.2.2. Finally, an overview of the test instances is presented in Section 6.2.3.

6.2.1 Benchmark Instances

Smet et al. (2013) present a selection of four public benchmark datasets that are popular and diverse: NSPLIB (Vanhoucke and Maenhout, 2007), Nottingham (Burke and Curtois, 2014), KAHO (Bilgin et al., 2012) and 2010 NRC (Haspeslagh et al., 2014).

The NSPLIB dataset consist of more than 30,000 artificial test instances generated to simulate nurse rostering problems. Although the dataset is rich in diversity, there is no inclusion of shift patterns of any sort, reducing the relevance to this thesis.

The Nottingham dataset consists of both real and artificial instances collected from a variety of international papers and joined in a common format. As such, the instances are very diverse and not in general suited for representing Nordic conditions.

The KAHO dataset may be argued to be the most realistic due to its construction on the basis of six wards from two Belgian hospitals. However, the rostering problem addressed differs from that modeled in this thesis. In the KAHO instances, an empty, full or partially full schedule may be input to the solution method. That is, certain shift assignments may be given. With such a non-empty input schedule, the problem may be compared to the nurse rescheduling problem, as studied by for instance Moz and Pato (2004). Another limitation of the KAHO dataset's relevance to this thesis is the lack of patterns.

The 2010 NRC (International Nurse Rostering Competition, INRC) dataset is the most cited among the four benchmark instances presented by Smet et al. (2013) and is chosen as a basis for developing test instances for this thesis. Note that the instances represent nurse rostering. While the model developed is intended to handle several types of healthcare employees, tests were only conducted for nurse rostering instances. The aims of the 2010 INRC was to stimulate the development of new solution approaches for the nurse rostering problem, reduce the gap between research and practice and encourage debate withing the rostering research community (Haspeslagh et al., 2014). To reduce the gap between research and practice, the INRC instances contain several real world constraints. These include a maximum and minimum number of shift assignments, limits on consecutive days working, weekend constraints, shift preferences and unwanted shift patterns. In addition, the instances include employees with different contracts specifying parameters like workload. It is worth mentioning that a second international nurse rostering competition is described by Ceschia et al. (2015), but has not yet received the same recognition as its precursor.

The INRC instances also include constraints not directly addressed by the model proposed in Chapter 4. The number of days off after consecutive night shifts was not found to be relevant at Nordic healthcare institutions after discussions with professionals and is thus left out. Similarly, no mention of requiring the same shift types through a working weekend was found in Norwegian regulation and the requirement is left out. It should be noted that there still are constraints ensuring a weekend is either entirely off or has a working shift both on Saturday and Sunday, which experience shows is important for employee satisfaction. By altering the benchmark instances of the INRC, a direct comparison between the results in this thesis and the competition is disabled. This disadvantage is accepted in exchange for the advantage of a closer simulation of the specifics expected in Nordic healthcare practice, and even more realistic problems.

The INRC dataset contains 60 instances of different size. The instances comprise three *tracks* (*sprint*, *middle distance* and *long distance*) inspired by the Olympics, making the distinction on the allowed solution time. As would be expected, the long distance instances are more challenging than the sprint instances, mainly due to more employees. Each track is again made up of three sets, *early*, *late* and *hidden*, only differing by the time of release when the competition was held. In this thesis, an arbitrary selection of the first five *sprint* instances, three *medium* instances and two *long* instances from the *late* set were selected and modified as described in the following section. Each instance was also extended to a new version with increased length of the planning period from four to twelve weeks. After instance modifications are discussed in Section 6.2.2, an overview of the final instances is given in Section 6.2.3.

6.2.2 Modification of Test Instances

The INRC benchmark instances form a solid basis for testing, but lack some complexity compared to real world instances in Nordic healthcare institutions. An important contribution of this thesis is to develop a model able to handle complicated constraints that bridge the gap between real world problems and academic models further. Thus, the benchmark instances are added parameters that fit them to the model proposed in Chapter 4 and make them more realistic in a Nordic situation. The main sources for parameter values are Norwegian labor regulations (Ministry of Labour and Social Affairs, 2005), the collective agreement between the Ministry of Local Government and Modernisation and LO Stat, Unio and YS Stat regarding a large part of the employees at Norwegian hospitals (Ministry of Local Government and Modernization et al. (2018), hereafter called *the collective* *agreement*), and discussions with professionals with experience from Norway and Denmark. These include nurses, doctors and this thesis' supervisors. The parameters added to the datasets are presented in this section following the same order as they appear in the mathematical model. Note that because Norwegian regulations have been most accessible to the authors, these have been important for the test instances. However, smaller variations between the Nordic countries are believed to have little impact on the computational results.

Cost parameters
Cost parameters in the objective function represent both monetary (e.g. salaries, payroll taxes and insurance) costs and costs reflecting employee satisfaction. Their magnitudes are critical to build a practically relevant model. The monetary costs are estimated based on Norwegian conditions, which are assumed similar to other Nordic countries. An hourly base cost is estimated in eq. (6.1), where C_e^H is a variable hourly cost of having employee e at work and S_e denotes the annual gross salary for employee $e \in \mathcal{E}$ which is based on the seniority of the employee and the collective agreement. 1.18 represents a scaling factor to adjust for vacation pay, pension contributions and payroll taxes, and $1\,695$ is the estimated total number of working hours through a year. Details about, and the rationale for, the hourly base cost for each employee is appended in Appendix C.

$$C_e^H = \left\lfloor \frac{1.18 S_e}{1\,695} \right\rceil \tag{6.1}$$

The cost of having an employee work a specific shift is input to the model. This is calculated as the hourly base cost, C_e^H, times the duration of the shift, subtracted 30 minutes for a meal break. Following the collective agreement, hours worked between 8 p.m. and 6 a.m. are compensated with 45 percent extra salary, while weekend hours worked between 0 a.m. Saturday and 0 a.m. Monday are compensated with an additional NOK 55.

In many of the benchmark instances, employees have requests for specific days or shifts off. These requests are treated as soft constraints through altering shift cost parameters. Fulfilling a request incurs a reward corresponding to the cost of a standard shift, i.e. a shift with duration of 7.5 hours in the middle of the day. This reward is intended to reflect the importance of fulfilling the request compared to minimizing monetary costs. Professionals' experience suggest that setting the exact reward for preference satisfaction is a demanding exercise, but that preferences are an important factor, and sometimes even the primary consideration in rostering. Preliminary testing suggests that the cost modification proposed here leads to fulfillment of most preferences in the optimal solution. The degree to which preferences are prioritized in the real world varies, and the model is flexible to changes by altering the cost parameters.

Demand coverage
The benchmark instances state a demand for employees with certain skills for each shift in the planning horizon. Limits and costs for over- and under-coverage are set based on discussions with experienced professionals. For under-coverage, it is assumed that at least half of the employees at work on a specific shift must be from the internal workforce. This ensures there are always some employees that know the department and internal

routines. Hiring external personnel is more costly than using the internal workforce. A rough estimate of profits and overhead costs going to staffing agencies resulted in the cost of under-coverage equal to the cost of the most expensive internal employee and an additional 30 percent. Over-coverage is viewed as disadvantageous due to capacity fluctuations causing inefficiencies and poor planning and is penalized with NOK 1 000 for each extra employee at work. The amount of over-coverage of a shift is unlimited, but will seldom be high in an optimal solution due to the added cost.

Consecutive days working
The maximum and minimum consecutive days working are specified in the INRC instances. No constraints are proposed on consecutive days working specific shift types. Benefits of working the same shift type consecutively are instead considered through modeling of rewarded patterns.

Rest between shifts
According to Norwegian labor regulations, an employee should always have minimum eleven hours of rest during a day. Experience shows this limit is not always followed. To allow small violations in turn for a penalty, the absolute minimum is set to eight hours, i.e. $T^I = 8$, while the limit for reduced rest is set to $T^R = 11$.

Working two consecutive shifts with less than T^R hours of rest in between incurs an additional cost of NOK 1 000, i.e. $C^R = 1\,000$. This reflects a substantial compensation for a stressful shift transition. The number of days with rest penalty is also limited to $\overline{N}^R = 8$ during a period of $D^R = 28$ days, ensuring preferences regarding rest are not violated too often.

Weekends
In the test instances, it is specified that at most three consecutive weekends may be worked. This is modeled by requiring at least $\underline{N}^W = 1$ weekend off over any period of $W^W = 4$ weeks for each employee.

Strict days off
Constraints regarding strict days off vary from country to country. The model has a flexible implementation of these constraints allowing for several variants. For the test instances in this thesis, the collective agreement form the basis for the parameter values. It is specified that within a week one should always have one period of at least 36 hours off. Parameters are set accordingly.

Workload
The workload of a shift generally equals the time between the start and end of the shift, subtracted 30 minutes for a meal break. Further, each hour is counted with an additional 15 minutes between 8 p.m. and 6 a.m., as per the collective agreement. The benchmark instances have different contracts with a given number of minimum and maximum assignments during the planning period for each employee. Contracted workload is input to the model and set to the midpoint between maximum and minimum number of assignments assuming a shift length of 7.5 hours. Similarly, the limits on under- and overtime are given as the difference from the midpoint to the end points assuming 7.5 hour shifts.

For every hour an employee works less than the contracted amount, the employee is compensated with the base hourly salary. This reflects that the salary is bounded from below even though the employee is assigned few shifts. For overtime, the employee gets the base salary plus 50 percent, equal to the compensation stated in the collective agreement for overtime on regular working hours.

Patterns

Working patterns are particularly important for the test instances used in the thesis. Recall from Section 4.3 that the model considers rewarded, penalized and illegal patterns. The benchmark instances provide some patterns, but these are on shift transitions, while the model in this thesis defines patterns for shift groups. Therefore, new penalized and illegal patterns are created that are inspired by the patterns from the benchmark instances. These patterns are mostly relatively short patterns of two or three days that are given a penalty in the model. The pattern going directly from a night shift to a morning shift is set to illegal. The patterns used in the test instances are summarized in Table 6.1. Each letter M, E and N refer to the shift groups morning, evening and night respectively. In addition to

Table 6.1: List of patterns in the test instances. Starting days refer to the days on which the pattern must start to be recorded. Shift group legend: M – Morning; E – Evening; N – Night. Pattern type legend: R – Rewarded; P – Penalized; I – Illegal.

Shift group sequence	Starting days	Type	Comment
E-M	All	P	Based on instance patterns
M-N	All	P	Based on instance patterns
N-M	All	I	Based on instance patterns
N-M	All	I	Based on instance patterns
M-M	All	R	Reward same shift group
E-E	All	R	Reward same shift group
N-N	All	R	Reward same shift group
N-E-M	All	P	Frequent changes in starting time
M-E-N	All	P	Frequent changes in starting time
M-E-N-E-M	All	I	Frequent changes in starting time
N-E-M-E-N	All	I	Frequent changes in starting time
M-M-M-M	Mon, Tue	R	Pattern for employees preferring morning shifts
M-M-M-M-M	Mon	R	Pattern for employees preferring morning shifts
E-E-E-E	Mon, Tue	R	Pattern for employees preferring late shifts
E-E-E-E-E	Mon	R	Pattern for employees preferring late shifts
N-N-N	Mon, Tue, Wed	R	Pattern for employees preferring late shifts

the patterns inspired by the benchmark instances, more patterns are added based on professionals' experience from real world healthcare rostering. One important consideration is that frequent changes in start and end time of shifts is disfavored. The two penalized patterns M-E-N and N-E-M are therefore added to the instances. The patterns M-E-N-E-M and N-E-M-E-N are set to illegal as they are thought to be especially tiring.

Some rewarded patterns are also added to the problem and reflect employee wishes. Each employee is either assigned rewarded patterns of four and five consecutive morning shifts between Monday and Friday, or four and five consecutive evening shifts and three consecutive night shifts between Monday and Friday. These patterns reflect stable working weeks and represent preference for mornings or evenings and night. Which group of patterns each employee is assigned is determined by a random draw, both groups equally probable.

The penalties and rewards for patterns should be in the same size order as the other cost components in the objective function. Let C_e^S denote the standard cost of a shift for employee e, given as $7.5C_e^H$. The penalty or reward for a pattern is given as a certain percentage of this. The longer the pattern, the higher the reward and penalty. Discussions with experienced personnel yielded the calculation of the penalty or reward of a pattern as $0.1nC_e^S$, where n is the length of the pattern in days. Note that this is a rough estimate. For overlapping rewarded patterns, rewards are added cumulatively, i.e. both reward for the patterns N-N and N-N-N is given if an employee works N-N-N and prefers both N-N and N-N-N. As discussed with regards to shift preferences, the magnitude of penalties and rewards gives a sensible prioritization of patterns compared to monetary costs. Preliminary testing confirms this, showing that the suggested rewards and penalties yield seldom assignment of penalized patterns and common assignment of rewarded patterns in the optimal solution.

6.2.3 Instance Overview

An overview of the instances used in the computational study is presented in Table 6.2. The number of shift types does not include off shifts. For each instance, a twelve week instance was created by duplicating the demand to extend the planning period. These are not included in the table, but will be denoted with an added *w12* at the end of the instance name to distinguish them from the four week instances. The full instances are available in Excel data sheets online[2].

6.3 Performance of the Default Configuration

In this section, the performance of the model presented in Chapter 4 solved with branch and price is evaluated. *Default configuration* refers to the initial configuration of the solution method before implementation of speed-ups described in Section 5.5. Results from the default configuration form a frame of reference for further testing, and indicate which parts of the model are most important to improve. After discussing and tuning the speed-up techniques in Sections 6.4 and 6.5, a final configuration is proposed and compared with the default configuration in Section 6.6 to gain an understanding of the overall effect of altering the solution method.

Table 6.3 displays the key results from testing the default configuration on all test instances with an eight hour time limit on computation time. The number of processed nodes refers

[2]https://github.com/sandercoates/HRP

to the number of nodes solved in the branching tree with CG. The upper bound represents the best integer solution found, while the lower bound is the best lower limit for an optimal solution. The optimality gap quantifies the relative difference between the upper and lower bounds as defined in Section 5.4.4, eq. (5.6). Results are only displayed for the four week instances and one 12 week instance because for all other instances, the root node was not solved within eight hours. Note how just the root node is solved for the 12 week instance included in Table 6.3.

Table 6.2: Overview of the test instances. Note that twelve week instances are annotated with *w12*, e.g. *inrc_sprint_late01w12*. The number of shift types does not include the off shift type.

Name	Employees	Planning period [weeks]	Shift types	Skills
inrc_sprint_late01	10	4, 12	4	1
inrc_sprint_late02	10	4, 12	3	1
inrc_sprint_late03	10	4, 12	4	1
inrc_sprint_late04	10	4, 12	4	1
inrc_sprint_late05	10	4, 12	4	1
inrc_medium_late01	30	4, 12	4	1
inrc_medium_late02	30	4, 12	4	1
inrc_medium_late03	30	4, 12	4	1
inrc_long_late01	50	4, 12	5	2
inrc_long_late02	50	4, 12	5	2

Table 6.3: Key results from the default configuration with an eight hours time limit. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6). For the instances not included in the table, the root node was not solved within the time limit.

Instance	Processed nodes	Upper bound $[\cdot 10^3]$	Lower bound $[\cdot 10^3]$	Optimality gap
inrc_sprint_late01	186	225	178	20.7 %
inrc_sprint_late02	410	152	115	24.3 %
inrc_sprint_late03	178	269	208	22.5 %
inrc_sprint_late04	157	253	208	17.8 %
inrc_sprint_late05	296	inf	191	inf
inrc_medium_late01	100	902	783	13.1 %
inrc_medium_late02	70	960	812	15.4 %
inrc_medium_late03	47	894	770	13.9 %
inrc_long_late01	14	inf	1,680	inf
inrc_long_late02	15	inf	1,651	inf
inrc_sprint_late02w12	1	inf	334	inf

From Table 6.3 it is evident that for most instances, the integer solution found may be far from optimality after eight hours of computation time, and that there is a clear need for speeding up the default configuration. The instance *inrc_sprint_late02* seems to be significantly quicker to solve than the other sprint instances. This is seen by the high number of nodes solved for the four week instance and by *inrc_sprint_late02w12* being the only 12 week instance with the root node solved in less than eight hours. The observation that *inrc_sprint_late02* is particularly easy to solve is consistent with the tests that are described later in this chapter. Notice that for the *inrc_sprint_late05*, *inrc_long_late01*, *inrc_long_late02* and *inrc_sprint_late02w12* instances, no integer solution is found within eight hours.

It is interesting to discuss the optimality gap of less than 16 percent found for the four week medium instances. An explanation for why better optimality gaps are achieved for these instances than the smaller sprint instances might be that there is higher demand for each shift and thus more flexibility to hire external personnel. This can ease the task of finding integer feasible solutions from an LP feasible set of columns.

For a reminder of the ultimate goal of the problem studied in the thesis, an example of the roster obtained by the default configuration for the *inrc_sprint_late01* instance is illustrated in Table 6.4 with shift assignments for all employees during the planning period. Note how hiring of external personnel is extensively used to counter under-coverage. This might be a reason for the optimality gap of over 20 percent, or might suggest that the cost of hiring external personnel in the test instances is low compared to other costs. Because under-coverage also has a hard limit of half of the demand for each shift, this is not considered a pressing issue. It is also worth considering the accommodation of employee preferences in the roster example. Employees 1, 2, 7 and 8 prefer evening and night shifts according to the *inrc_sprint_late01* instance. The remaining employees prefer morning shifts. From Table 6.4 it can be seen that these preferences are accommodated to a large extent.

To identify which parts of the solution procedure require most of computation time, the distribution of computation time over different categories is shown in Table 6.5. It is clear that the largest portion of time is spent solving the nodes. Branching also requires some computation time. This is, however, believed to mainly be because of the implementation. In branching, parts of the parent node RMP, columns and SPPRC graphs are copied into the children nodes. Experience suggests that this is a time consuming exercise and could have been implemented more efficiently. Other computations such as searching the tree and calculating bounds consume less than one percent of the total eight hours.

The time spent solving the nodes is broken down further in percentages of the time spent solving the nodes in Table 6.6. For all instances, a major part of the computation time is spent solving the SPs, as expected from the discussions in Chapters 2 and 5. However, a considerable part of the computation time is spent solving the RMP and the effort to improve should also be directed towards reducing RMP computation time. The *other* category in Table 6.6 includes testing whether the LP optimal solution from CG is also integer feasible, which is done by solving the RMP with binary requirements enforced. This explains the significant part of computation time spent in *other* for the *inrc_sprint_late02* in-

Table 6.4: Illustration of the roster generated for the *inrc_sprint_late01* instance by the default configuration. Black cells indicate employee assignment. Over-coverage is represented by positive and under-coverage by negative numbers.

Employee	Shift \ Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	6:30-14:30																												
	8:30-16:30																												
	14:30-22:30								■	■	■	■	■																
	22:30-6:30																■	■	■		■	■	■	■	■				
2	6:30-14:30																												
	8:30-16:30																												
	14:30-22:30												■	■	■		■	■	■	■									
	22:30-6:30	■	■	■	■																					■	■	■	■
3	6:30-14:30																								■	■		■	
	8:30-16:30																							■					■
	14:30-22:30					■	■	■																					
	22:30-6:30										■	■	■	■	■														
4	6:30-14:30			■		■		■															■	■					
	8:30-16:30	■	■				■		■		■	■	■														■		
	14:30-22:30																				■	■						■	■
	22:30-6:30																			■									
5	6:30-14:30			■	■	■								■	■					■	■				■		■		
	8:30-16:30		■																			■	■			■			
	14:30-22:30															■	■												
	22:30-6:30																												
6	6:30-14:30	■	■									■	■			■	■	■	■	■									
	8:30-16:30										■																		
	14:30-22:30																												
	22:30-6:30					■	■	■																					
7	6:30-14:30																												
	8:30-16:30																												
	14:30-22:30	■	■	■	■	■																	■	■	■	■	■		
	22:30-6:30								■	■																			
8	6:30-14:30									■	■	■										■		■					
	8:30-16:30			■	■	■															■				■				
	14:30-22:30																												
	22:30-6:30															■	■												
9	6:30-14:30						■																						■
	8:30-16:30							■									■	■	■	■								■	
	14:30-22:30																												
	22:30-6:30																												
10	6:30-14:30								■		■		■																
	8:30-16:30									■				■	■	■													
	14:30-22:30																												
	22:30-6:30																												
Demand	6:30-14:30	2	2	2	2	2	1	1	2	2	2	2	2	1	1	2	2	2	2	2	1	1	2	2	2	2	2	1	1
	14:30-22:30	2	2	2	2	2	1	1	2	2	2	2	2	1	1	2	2	2	2	2	1	1	2	2	2	2	2	1	1
	8:30-16:30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	22:30-6:30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Over-/ under- coverage	6:30-14:30	-1	-1	0	-1	0	0	0	-1	-1	0	0	0	0	0	-1	-1	-1	-1	0	0	0	-1	0	0	-1	-1	0	0
	14:30-22:30	-1	-1	-1	-1	0	0	0	-1	-1	-1	-1	0	0	0	-1	0	-1	-1	-1	0	0	-1	-1	-1	-1	-1	0	0
	8:30-16:30	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22:30-6:30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

stance. Recall that this instance is solved quicker than other sprint instances and that many more nodes are solved, requiring a higher number of integer feasibility checks. Sometimes, such a test requires much computation time, depending on the specific node.

Table 6.5: Distribution of total computation time for the default configuration. Total computation was limited to eight hours. *Other* includes searching the tree, calculating bounds and other overhead data management.

Instance	Solving nodes	Branching	Other
inrc_sprint_late01	87.2 %	12.6 %	0.2 %
inrc_sprint_late02	83.5 %	16.1 %	0.4 %
inrc_sprint_late03	86.0 %	13.7 %	0.3 %
inrc_sprint_late04	86.2 %	13.5 %	0.3 %
inrc_sprint_late05	81.3 %	18.3 %	0.4 %
inrc_medium_late01	89.0 %	10.5 %	0.5 %
inrc_medium_late02	87.1 %	12.4 %	0.5 %
inrc_medium_late03	92.1 %	7.4 %	0.6 %
inrc_long_late01	90.4 %	8.9 %	0.7 %
inrc_long_late02	90.8 %	8.5 %	0.7 %
inrc_sprint_late02w12	99.0 %	0.4 %	0.6 %

Table 6.6: Distribution of time solving nodes for the default configuration. All figures are percentages of the time spent solving nodes in the respective instances. The *other* category includes computation time spent solving the RMP with integer requirements to check integer feasibility of LP solution.

Instance	Solving the SPs	Solving the RMP	Construction heuristic	Updating the RMP	Other
inrc_long_late01	77.5 %	16.8 %	4.9 %	0.5 %	0.3 %
inrc_long_late02	74.9 %	21.0 %	3.3 %	0.5 %	0.3 %
inrc_medium_late01	86.5 %	9.3 %	2.6 %	0.5 %	1.1 %
inrc_medium_late02	85.1 %	12.3 %	1.2 %	0.6 %	0.8 %
inrc_medium_late03	90.0 %	8.3 %	0.2 %	0.5 %	1.0 %
inrc_sprint_late01	82.1 %	14.8 %	0.7 %	0.6 %	1.9 %
inrc_sprint_late02	70.9 %	16.0 %	1.7 %	0.7 %	10.8 %
inrc_sprint_late03	79.5 %	17.7 %	1.0 %	0.6 %	1.1 %
inrc_sprint_late04	78.2 %	17.6 %	2.5 %	0.6 %	1.1 %
inrc_sprint_late05	54.1 %	13.5 %	31.0 %	0.7 %	0.6 %
inrc_sprint_late02w12	85.9 %	13.8 %	0.0 %	0.2 %	0.1 %

Note that particularly much time is spent in the construction heuristic for the *inrc_sprint_late05* instance. The construction heuristic is used every time the initial RMP of a node to be solved by CG is infeasible after branching. Considering the lack of an integer solution and high number of nodes processed for *inrc_sprint_late05* in Table 6.3, it seems that for this instance, many of the branching nodes turn out to be infeasible. Proving infeasibility requires quite some time in the construction heuristic, but the total time of the node is often reduced because it can be pruned without solving CG.

Tables 6.3, 6.5 and 6.6 indicate a need for speed-ups to reduce computation time and find better bounds. Especially, reduction of the SP solution time and the number of nodes that must be solved to find good integer solutions is critical. The speed-up techniques proposed in Section 5.5 and their effect on computation time are discussed in the remainder of this chapter.

6.4 Column Generation Speed-Up

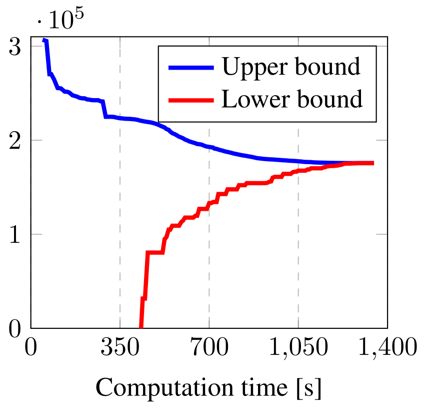
The results presented in Section 6.3 showed that the largest part of the computation time when solving the decomposed model from Chapter 4 is used to solve CG in each node of the branching tree. The most computationally expensive process is solving the SPs. In this section, the impact of applying several speed-up techniques to the CG procedure is discussed and the performance in the root node is evaluated. The emphasis is put on the four week instances, as well as the twelve week sprint instances to keep computation time within reasonable limits and allow testing on a great variety of configurations in the time available for writing the thesis. For all tests in this section, the time limit was set to four hours or 14 400 seconds. If the root node is not solved within this time, good solutions are unlikely to be found in the branching tree within the eight hours computation time limit of the branch and price algorithm.

6.4.1 Column Generation Stop Criterion

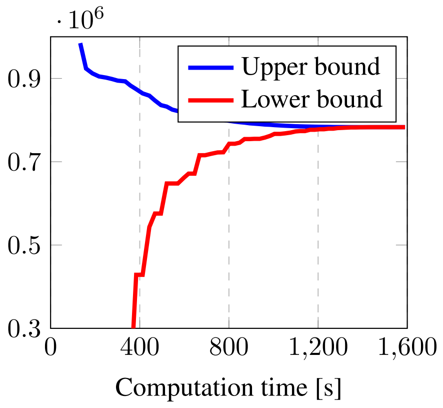
To better understand the performance of the CG algorithm, Figure 6.2 illustrates the development of the upper bound, or RMP objective function, and the lower bound for three instances of different size. The instances are selected arbitrarily as the first instances *inrc_sprint_late01*, *inrc_medium_late01* and *inrc_long_late01*. It is evident that the RMP objective function and the lower bound tends to have diminishing marginal improvement over time. This poor convergence, or tailing-off effect, is often seen in CG (Lübbecke and Desrosiers, 2005) and motivates a more thorough analysis to consider terminating CG before optimality is reached in each node.

Computation time and iterations needed to reach certain CG optimality gaps in the root node are displayed for all four week instances in Table 6.7. The optimality gap in the CG procedure was defined by eq. (5.8) in Section 5.5.1. No twelve week instances reached optimality within the time limit, and are therefore not presented. As expected, both the number of iterations and computation time increases to reach better gaps.

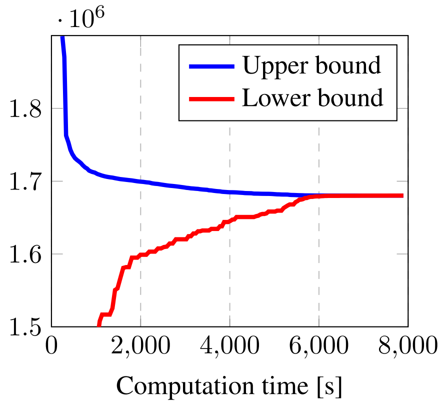
The relative time increase between the different CG optimality gaps is displayed in Table 6.8. To close the optimality gap from one percent requires a time increase in the range of 9 to 65 percent. This is significantly higher than the relative time increase between



(a) *inrc_sprint_late01*



(b) *inrc_medium_late01*



(c) *inrc_long_late01*

Figure 6.2: Development of upper and lower bound over time for selected instances in root node. The upper bound represents the RMP objective. The lower bound is calculated by eq. (5.7). Because the initial lower bound is negative indefinite, the vertical axes have been trimmed to emphasize the shape of the curves.

the other increments in CG gap. These findings reflect the tailing off effects seen in Figure 6.2, where the convergence is especially slow towards the end. Based on this, the CG optimality gap limit for future tests was set to one percent.

As explained in Section 5.5.1, introducing an optimality gap in the CG procedure in each node removes the exactness of the branch and price algorithm. However, preliminary testing showed that a one percent gap was considerably smaller than the gap found in the branch and price algorithm. Therefore, the gap limit of one percent seems to be a reasonable choice.

Table 6.7: Iterations and computation time to reach different CG optimality gaps in the root node. The CG optimality gap is defined in eq. (5.8).

	Iterations						Computation time [s]					
Instance \ CG gap	10 %	5 %	3 %	2 %	1 %	0 %	10 %	5 %	3 %	2 %	1 %	0 %
inrc_sprint_late01	111	124	133	137	144	162	991	1 086	1 148	1 176	1 222	1 347
inrc_sprint_late02	61	66	73	76	80	92	267	285	311	322	337	382
inrc_sprint_late03	123	140	152	157	166	188	1 073	1 197	1 281	1 315	1 374	1 523
inrc_sprint_late04	119	136	150	154	166	202	1 041	1 164	1 268	1 297	1 378	1 631
inrc_sprint_late05	116	134	143	150	158	177	1 106	1 232	1 294	1 340	1 393	1 517
inrc_medium_late01	26	31	35	39	42	63	801	914	1 004	1 099	1 164	1 590
inrc_medium_late02	26	35	50	66	72	102	762	960	1 295	1 632	1 756	2 342
inrc_medium_late03	21	28	35	42	54	94	642	816	981	1 152	1 438	2 377
inrc_long_late01	24	43	64	76	96	160	1 426	2 650	3 779	4 342	5 297	7 894
inrc_long_late02	23	42	56	68	94	169	1 334	2 594	3 458	4 107	5 516	8 737

Table 6.8: Relative computation time increase to reach different column generation optimality gaps in the root node. Percentages are calculated by considering the time increase taken to reach the gap in question, relative to the previous gap in focus. For example, $t^*_{2\%} = (t_{2\%} - t_{3\%})/t_{3\%}$, where t^* is the relative time increase and $t_\bullet$ the time taken to reach optimality gap $\bullet\%$.

	Computation time increase [%]				
Instance \ CG gap	5 %[1]	3 %	2 %	1 %	0 %
inrc_sprint_late01	10 %	6 %	2 %	4 %	10 %
inrc_sprint_late02	7 %	9 %	3 %	5 %	13 %
inrc_sprint_late03	11 %	7 %	3 %	4 %	11 %
inrc_sprint_late04	12 %	9 %	2 %	6 %	18 %
inrc_sprint_late05	11 %	5 %	4 %	4 %	9 %
inrc_medium_late01	14 %	10 %	9 %	6 %	37 %
inrc_medium_late02	26 %	35 %	26 %	8 %	33 %
inrc_medium_late03	27 %	20 %	17 %	25 %	65 %
inrc_long_late01	86 %	43 %	15 %	22 %	49 %
inrc_long_late02	94 %	33 %	19 %	34 %	58 %

[1]The relative time increase to reach the 5 percent optimality gap is calculated with reference to the 10 percent optimality gap.

6.4.2 Retrieving Several Sub Problem Solutions

As described in Section 5.5.2, the number of columns added from each SP in each iteration of CG can be larger than one. Input to the model is the maximum number of roster lines retrieved from each employee SP in each iteration, referred to as *# SP Solutions*. Recall that dominance is relaxed in the end node in the labelling algorithm to increase the number of available solutions. A preliminary test on the *inrc_sprint_late01* instance was conducted to select configurations to test. The computation times for different values of *# SP Solutions* in the preliminary test are displayed in Figure 6.3. No limit means retrieving all available columns with negative reduced cost. Note that the one percent stop criterion is kept from the previous speed-up.

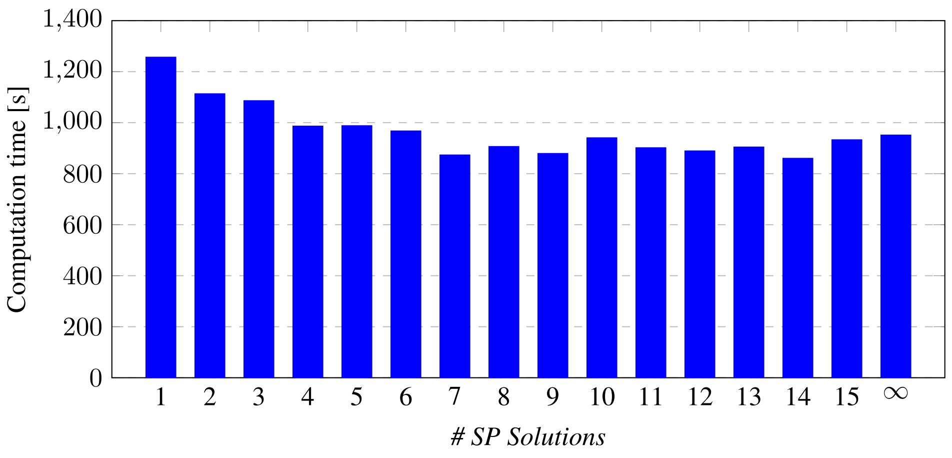


Figure 6.3: Computation time for different values of *# SP Solutions* in the preliminary testing on *inrc_sprint_late01*. The *no limit* option is denoted by ∞.

The results from *inrc_sprint_late01* indicate a clear benefit of retrieving more columns from each SP in each iteration on computation time in the root node. This could be expected since the SPs require most computation time, and retrieving more columns in each iteration can possibly reduce the number of iterations and thereby the number of SPs solved. It is, however, not obvious which value of *# SP Solutions* is the best because an increased number of columns also increases the size of the RMP in each iteration. It is worth mentioning that it seems that more than 15 solutions are rarely returned so that this limit is similar to an unlimited number of SP solutions. Based on the preliminary test, a limit of 5, 10 and no limit were selected for further testing, and compared with the default of only returning one column from each SP in each iteration.

In Figure 6.4, results from all four week instances are presented with different values of *# SP Solutions*. As expected, the number of iterations are reduced for an increased number of solutions retrieved in each iteration. However, the effect on computation time is not as obvious. Especially when going from a limit of 10 to no limit, there is no clear reduction in computation time. Still, there is clear benefit on computation time of retrieving more than one column from each SP in each iteration.

Table 6.9 displays the upper and lower bounds found after four hours of computation time for different limits on the number of columns retrieved in each iteration for the twelve week sprint instances. Among these, only *inrc_sprint_late02w12* was solved to a one percent gap. Especially for the lower bound, there is clear benefit of retrieving several solutions in each iteration. However, going from a limit of 10 to no limit seems to have negative impact on the bounds. It should be noted that these results are highly dependent on the relative speed of the RMP and SPs. For example, a more efficient RMP solver or a less complex master problem with fewer employee specific constraints is expected to shift results in favor of a higher number of SP solutions.

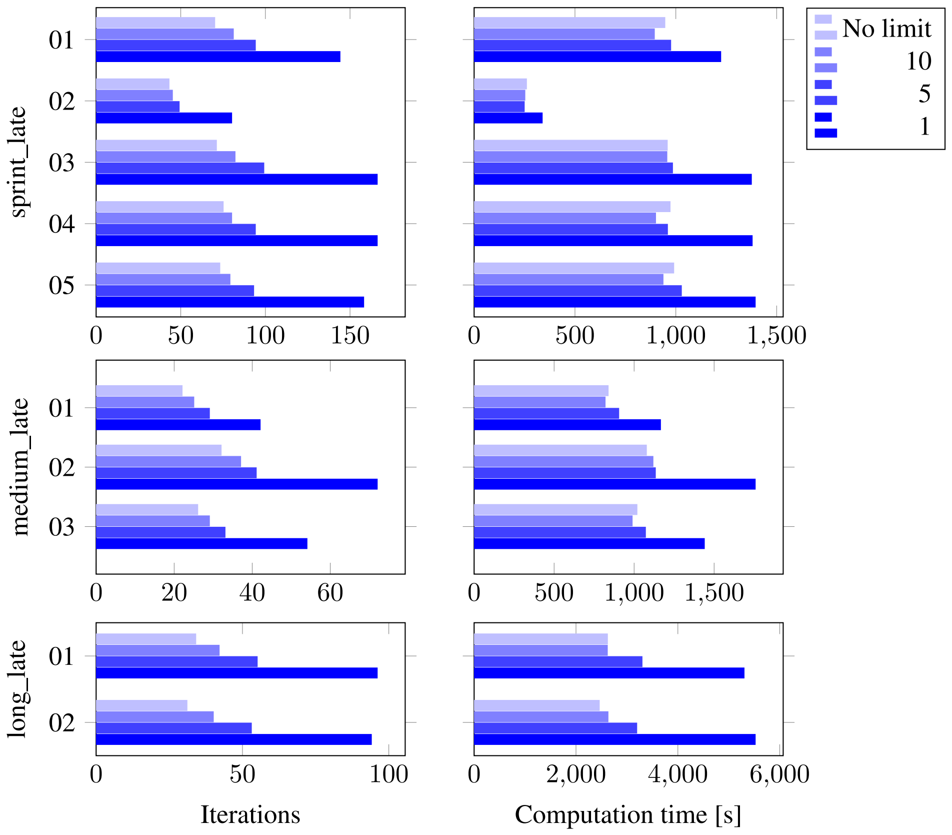


Figure 6.4: Number of CG iterations and computation time in the root node for four week instances with different limits on the number of columns returned from each SP in each iteration.

Table 6.9: Lower and upper bounds for different values of *# SP Solutions* at the four hour time limit. The upper bound is the RMP objective. The lower bound is calculated by eq. (5.7).

	Lower bound $[\cdot 10^3]$				Upper bound $[\cdot 10^3]$			
Instance \ # SP Solutions	1	5	10	No limit	1	5	10	No limit
inrc_sprint_late01w12	-313	-58	79	22	697	680	667	676
inrc_sprint_late02w12	322	330	327	326	336	335	335	336
inrc_sprint_late03w12	-162	-36	90	-25	794	780	772	777
inrc_sprint_late04w12	-40	90	126	47	792	772	766	776
inrc_sprint_late05w12	-234	-83	62	-5	754	735	728	733

Based on the results presented, a limit of 10 columns retrieved from each SP in each iteration of CG was chosen for further testing. This showed the best computation time and bounds among the configurations tested. Although no limit on SP solutions retrieved also performed well with regards to computation time, a limit of 10 seems most reasonable. In addition to the better bounds, imposing a limit manages the risk of the RMP growing too large.

For future research, it would be interesting to consider alternative schemes for selecting among the negative reduced cost SP solutions that are available upon solving the SPs with the labelling algorithm. The most negative reduced cost solutions were selected in the thesis. An alternative approach is, for example, to select solutions with the greatest difference in shift assignments. The rationale is then that solutions with similar reduced costs often reflect similar shift assignments and that their collective impact on the RMP is reduced. Choosing roster lines with little similarity instead can contribute to different parts of the roster.

6.4.3 Limited Label Extension

When testing limited label extension, the stop criterion of one percent and the upper limit of ten solutions retrieved from each SP is kept from the previous speed-ups. Because the implementation of limited label extension has consequences for the calculation of lower bounds and CG stop criterion discussed in Section 5.5.1 and Section 6.4.1, a CG improvement criterion is presented before the details of limited label extension are discussed. After evaluating limited label extension, the parameters of the improvement criterion are evaluated.

Introduction to the column generation improvement criterion
By limiting label extension in the labelling algorithm, as described in Section 5.5.3, the SPs are not guaranteed to be solved to optimality. Recall that this is because the solution space of the SPs are heuristically restricted by not extending all non-dominated labels in the labelling algorithm. Without guaranteed optimal SPs, a lower bound cannot be calculated to evaluate the stop criterion in eq. (5.7). Therefore, an improvement criterion is introduced in the CG algorithm. Whenever the marginal improvement of the CG upper bound, or RMP objective, falls below a given threshold, the limited label extension is relaxed to allow the calculation of a lower bound in the next iteration and check the CG stop criterion. Should the stop criterion not be fulfilled, limited label extension is resumed until the next time the improvement criterion calls for another lower bound calculation. Preliminary testing suggested that an improvement criterion of 0.05 percent per iteration was a reasonable choice. That is, if the current iteration RMP objective is not 0.05 percent lower than the previous iteration objective, the lower bound is calculated and the optimality gap checked. The suitability of the improvement criterion is justified later in this section.

Limited label extension
Input to the limited label extension speed-up is a vector of label extension limits. The vector specifies the increments for all employees. For example, the vector $[1, 2, 3]$ indicates that an SP should first be attempted solved with one label extended from each node in the SP network. If no solution with negative reduced cost is obtained and the limit was found to be restrictive, the limit is increased to two, and the SP is attempted solved again. This continues until a solution with negative reduced cost is found, the current limit is found to

not be restrictive, or the last element of the vector was tested. If a solution with negative reduced cost is still not found, the limit is removed for the relevant SP for the rest of the CG procedure in the relevant node.

Although the label extension limits input vector is common for all employees, the current limit for each employee may differ because some SPs might require an increased number of labels extended to find negative reduced cost solutions while other might be solved with only the first limit. The current label extension limits are reset to the first element of the input vector whenever CG is restarted, i.e. when initiating CG in a new node in the branch and price algorithm. If the limits are relaxed to calculate the lower bound, they are resumed where they left off after calculating the bound.

To settle a selection of label extension limits to test on several instances, preliminary tests were done on the *inrc_sprint_late01* instance. The input vectors $[1, 2, 3, 4, 5]$, $[2, 3, 4, 5]$, $[3, 4, 5]$, $[4, 5]$, $[5]$, $[10]$ and $[]$ were evaluated in the preliminary test. An empty vector leads to unlimited label extension and is the default configuration. These limit vectors were chosen such that the effect of the initial limit and the development of increments could be investigated.

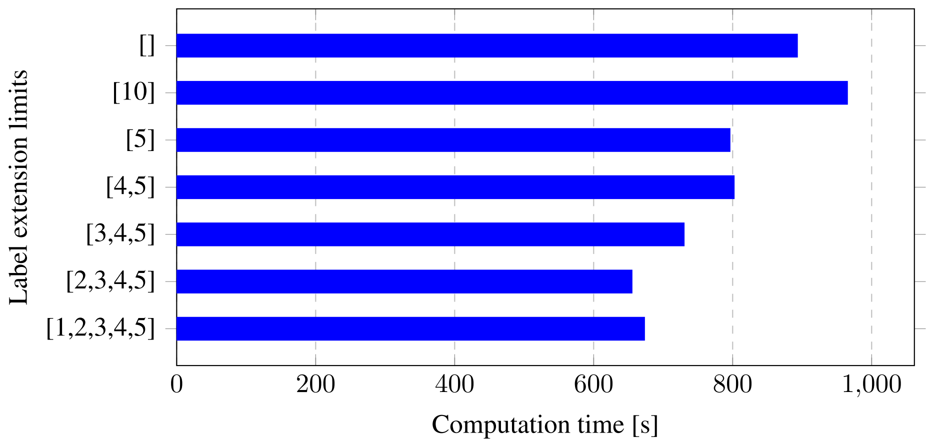


Figure 6.5: Computation times for different label extension limits in preliminary testing. The vectors specify the increments of label extension limits.

The computation times from the preliminary test are shown in Figure 6.5. These show a clear benefit of limited label extension. Also, from considering the extension limit used for each employee in each iteration, it was found that the limit was rarely increased. This indicates long input vectors are of little use since the limit rarely prevents a negative reduced cost solution. It may be hypothesized that the improvement criterion also influences this because the limits tend to be relaxed to allow lower bound calculation when there is little improvement to calculate lower bounds. This might make the SPs easier to solve afterwards due to updated dual variables when resuming the limited label extension.

Based on the results from preliminary testing, the vectors $[1, 2, 3]$, $[2, 3]$, $[3]$ and $[]$ were tested. The results of the main test are showed in Figure 6.6. For none of the twelve week instances other than *inrc_sprint_late02w12* was a one percent CG optimality gap reached. For the easier *inrc_sprint_late02w12* instance, a one percent gap was reached with all the alternative label extension limits, but not without a limit.

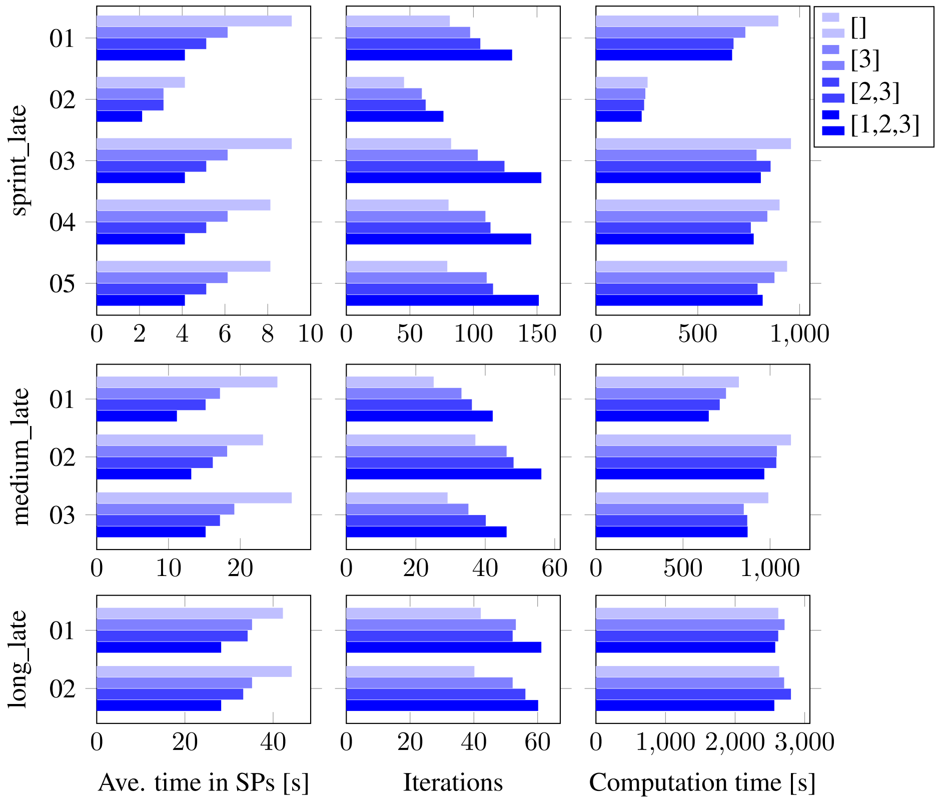


Figure 6.6: Average time in the SPs over all iterations, the number of iterations and the computation time needed to reach the one percent CG optimality gap for different label extension limits.

From Figure 6.6, it is clear that more limited label extension reduced the average computation time of the SPs. However, a stricter limit increases the number of iterations required to reach the one percent CG optimality gap. This was expected because the quality of the SP solutions, in terms of reduced cost, is lower than those obtained without limited label extension. By considering the total computation time, it seems that the average SP time reduction does more than compensates for the increase in the number of iterations and that overall, more limited label extension speeds up the CG.

The label extension limits input vector of $[1, 2, 3]$ does most often yield the best computation time in the root node. Especially for the larger instances, this configuration is superior. The *long_late* instances have more shift types, making the SPs more difficult to solve. This might be the reason why only extending the best label in each node has such a significant reduction in computation time. For the smaller *sprint* instances, $[2, 3]$ is also competitive, but this configuration struggles in the larger instances.

Table 6.10: Best integer solutions found for different label extension limits after four hours of computation time. None denotes that no integer solution was found using the columns that were generated after four hours. Instances for which no integer solution was obtained are excluded.

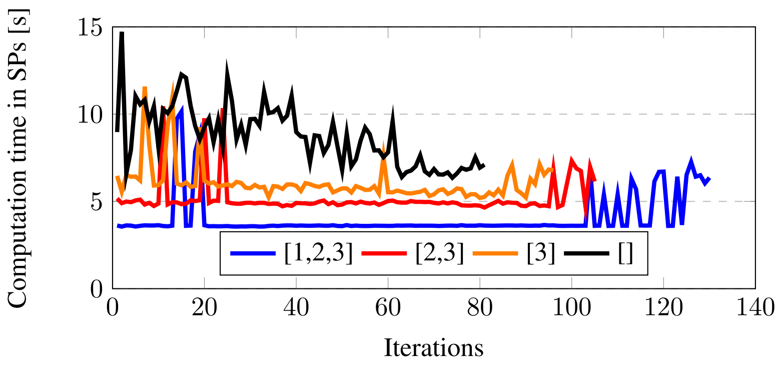
	Best integer solution $[\cdot 10^3]$			
Instance \ Label extension limits	[1, 2, 3]	[2, 3]	[3]	[]
inrc_medium_late01	None	None	None	922
inrc_medium_late02	None	987	968	949
inrc_medium_late03	None	None	995	964
inrc_sprint_late02	None	196	None	178

An overview of the best integer solutions obtained in the root node is given in Table 6.10. Instances for which an integer solution was not obtained are excluded. Considering the best integer solutions, it becomes evident that there is a trade-off between reduced computation time and good integer solutions when selecting limits on label extension. More limited label extension seems to reduce computation time, but less limited label extension seems to yield better integer solutions. Because branching to subsequent nodes will produce integer solutions and computation times are reduced significantly by label extension limits $[1, 2, 3]$, especially for larger instances, this configuration is chosen for further testing.

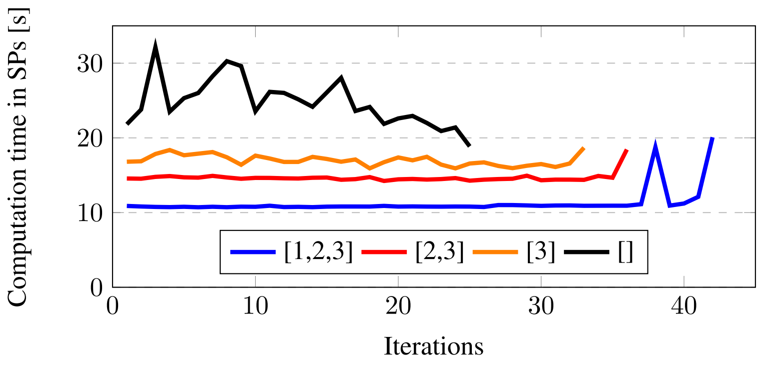
Evaluation of the column generation improvement criterion

Before evaluating partial CG, it is worth reconsidering the interaction between limited label extension and the CG improvement criterion used to calculate lower bounds and check the optimality gap stop criterion. In Figure 6.7, the computation time spent solving the SPs in each iteration of CG is plotted for the *inrc_sprint_late01*, *inrc_medium_late01* and *inrc_long_late01* instances across different label extension limits. First, it is clear that more limited labelling extension generally reduces the time spent solving SPs, as previously discussed for the average over all iterations. Second, it is again seen that more limited label extension increases the number of iterations needed to reach the one percent CG optimality gap. Third, the calculation of lower bounds is seen by sudden spikes in the SP computation time. This is explained by the need to solve all SPs to optimality, i.e. without limited label extension, to calculate a lower bound. Apart from a few unnecessary lower bound calculations in the early iterations for the *inrc_sprint_late01* instance, the lower bound is only calculated towards the end of CG. This supports the choice of improvement criterion made in the beginning of this section.

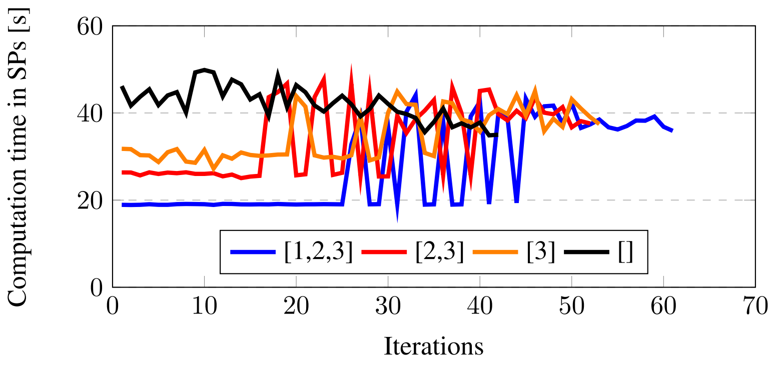
It may be argued that the improvement criterion could be more relaxed to calculate lower bounds earlier for the *inrc_medium_late01* instance and that this could lead to quicker termination of CG. On the other hand, the improvement criterion may be argued to be too relaxed for the *inrc_long_late01*, causing several calculations of the lower bound long before the optimality gap is sufficiently small to terminate. Thus, the improvement criterion is left as is. Fourth and finally, it seems that for unlimited label extension, SP solution time varies severely in the early iterations of CG. This may be due to unstable dual variables causing rapid changes in the SPs. Lübbecke and Desrosiers (2005) discuss such "heading



(a) *inrc_sprint_late01*



(b) *inrc_medium_late01*



(c) *inrc_long_late01*

Figure 6.7: Computation time spent solving the SPs in each iteration of CG across different label extension limits. CG was terminated once an optimality gap defined in eq. (5.8) of less than one percent was found.

in" effects and argue that stabilization techniques can be utilized to reduce the computation time. A simple stabilization technique is investigated in Section 6.6.1 where the demand coverage constraint is relaxed from an equality to an inequality constraint. Other than this, stabilization is not discussed further in this thesis.

6.4.4 Partial Column Generation

Several examples of applying partial column generation to rostering problems have been seen in literature (Gamache et al., 1999; Maenhout and Vanhoucke, 2010). Recall from Section 5.5.4 that in this thesis, partial CG refers to only solving SPs until one roster line with negative reduced cost is found. This roster line is added to the RMP which is solved again to obtain new dual variables. The rationale for partial CG is to save time by not solving similar SPs with the same dual variables. Thereby, the number of SPs solved can be reduced.

In Figure 6.8, the results from applying partial CG are compared to solving all SPs in each iteration. The best configuration from previous testing is applied. Two versions of partial CG were tested, based on the two options for choosing the order of solving SPs presented in Section 5.5.4. The option where each SP is solved without resources to compute lower bounds used for selecting the order is significantly slower than a random order for all instances, and does not have the desired effect of reducing the number of iterations. Because few resources are used in the SPs with the model configuration proposed in the thesis, the relative computation time of solving SPs without resources is quite large, making this strategy disadvantageous.

As expected, the number of iterations significantly increases when applying partial CG because fewer SPs are solved in each iteration. However, the average time per iteration decreases. As seen from Figure 6.8 however, the reduction in time per iteration is mostly not sufficient to outweigh the disadvantage of more iterations. It should also be mentioned that *inrc_print_02w12* did not reach the one percent CG gap with partial CG.

One of the reasons why partial CG does not outperform the option of solving all SPs in each iteration might be the enforcement of several employee specific roster line constraints in the RMP. First, this makes the SPs simpler and places more computational burden on the RMP instead. This shifts computation time from the SPs to the RMP and makes the number of iterations solving the RMP more important. Second, a large part of the dual variables in each iteration are employee specific. The dual variables relevant for one employee may therefore not be influenced much by adding new columns to the RMP for another employee. Thus, the positive effect of updating the RMP solution each time a new SP is solved in partial CG might be reduced compared to for the more common method of handling all employee specific constraints in the SPs.

Partial CG is discarded in further testing in this thesis. For future research, it would, however, be interesting to test alternative versions of partial CG where, for example, a subset of SPs are solved in each iteration instead of re-optimizing the RMP once one negative reduced cost solution is found. This might balance the effects of reduced time per iteration and increased number of iterations better.

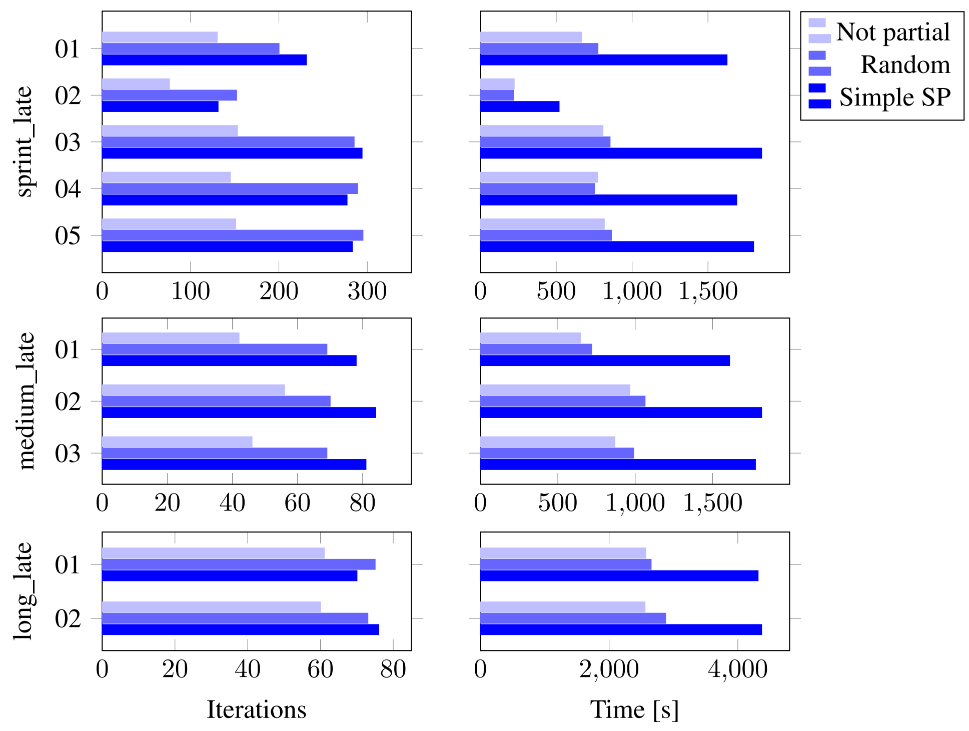


Figure 6.8: Number of iterations and computation time in the root node for four week instances for partial column generation. The order selection strategies *Random* and *Simple SP* (solving SPs without resources to select order) are displayed and referenced to results without partial column generation, or *Not partial*.

6.4.5 Handling Maximum Consecutive Days in the Sub Problems

An optional allocation of constraints discussed in Section 5.5.5 is to enforce constraints on maximum consecutive days working in the SPs with the resource $T^{\overline{W}}$ instead of in the RMP as suggested in Section 4.3. In the preparatory research project, enforcing constraints on maximum consecutive days working in the SPs was found to improve convergence, but at the same time increase the computation time in each iteration due to reduced dominance. Preliminary testing for this thesis indicated that a significant part of the roster lines generated in the SPs violate constraints on maximum consecutive days. This supports enforcing these constraints in the SPs to improve convergence.

The performance of this change in model configuration was tested in the root node to evaluate the effect on computation time and bounds. By moving constraints from the RMP to the SPs a tighter LP bound in the root node is expected.

In Figure 6.9, the number of iterations, average time solving SPs in each iteration and computation time in the root node are displayed. As expected, the average time per itera-

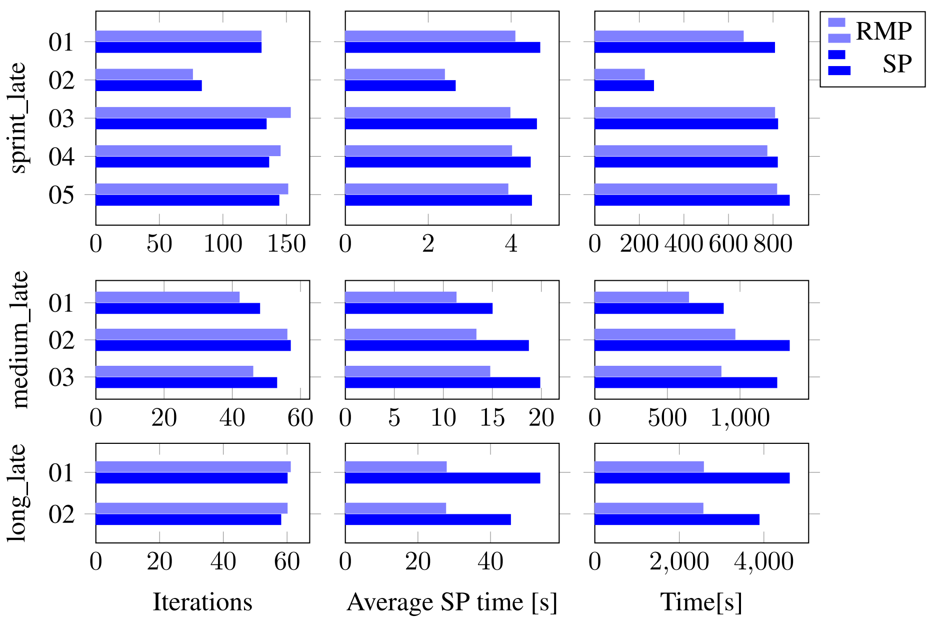


Figure 6.9: Number of iterations, average computation time solving SPs in each iteration and total computation time in the root node for four week instances. Results from handling maximum consecutive days working in the SPs and in the RMP are compared.

tion increases, but there is no significant reduction in the number of iterations required to solve the root node. Consequentially, computation time also increases. Table 6.11 shows the lower bound found in the root node for the different configurations. It is evident that the LP relaxation of the RMP is tighter when enforcing maximum consecutive days in the SPs. Also, as seen in Table 6.12, the new configuration was able to find integer feasible solutions in the root node for three instances, compared to none from before. Instances not included in the table did not provide any integer solutions in the root node.

Although computation time in the root node tends to increase when enforcing constraints on maximum consecutive days in the root node, the bounds produced are better, and it seems more likely to find integer feasible solutions. Therefore, no conclusion regarding the best configuration is made. Further tests are conducted in Section 6.5 to consider potential effects in the full branch and price algorithm.

Table 6.11: Lower bound found in the root node for four week instances when handling constraints on maximum consecutive days working in the RMP and the SPs.

	Lower bound $[\cdot 10^3]$		
Instance	Handling in RMP	Handling in SPs	Increase [%]
inrc_sprint_late01	174	179	2.6 %
inrc_sprint_late02	112	115	2.9 %
inrc_sprint_late03	204	208	1.6 %
inrc_sprint_late04	203	208	2.2 %
inrc_sprint_late05	190	193	1.5 %
inrc_medium_late01	779	784	0.6 %
inrc_medium_late02	805	819	1.7 %
inrc_medium_late03	764	769	0.7 %
inrc_long_late01	1 665	1 704	2.3 %
inrc_long_late02	1 637	1 676	2.4 %

Table 6.12: Best integer solution found in the root node for four week instances when handling constraints on maximum consecutive days working in the RMP and the SPs. None denotes that no integer solution was found. For the instances not included in the table, no integer solution was found.

	Best integer solution found $[\cdot 10^3]$	
Instance	Handling in RMP	Handling in SP
inrc_sprint_late02	None	157
inrc_medium_late01	None	948
inrc_medium_late02	None	878

6.5 Branch and Price Speed-Up

Based on the considerations in Section 6.4, a suggested best configuration of speed-up techniques was found. This configuration is summarized in Table 6.13. A significant reduction in computation time of the root node was accomplished, and Table 6.14 shows the results from implementing these speed-ups in the full branch and price algorithm. The results are compared with the default configuration presented in Section 6.3.

As expected, the reduction in computation time shown in the root node extended to the full algorithm, enabling solving of significantly more nodes in the branch and price tree within the time limit of eight hours with the new configuration.

Although the number of nodes solved increased significantly, the effect on bounds found was not as good as one could hope for. The lower bounds tend to be a bit weaker with the new configuration, probably because of the gap introduced in the CG procedure. For the medium instances, the best integer solutions found were slightly better, but no general conclusion on this matter can be made.

Table 6.13: Solution method configuration based on tests in the root node. Configuration parameters are described in Section 6.4.

Parameter	Value
CG stop criterion	0.01
Limit on SP solutions	10
CG improvement criterion	0.0005
Label extension limits	[1,2,3]
Partial column generation	FALSE
Handle max consecutive days in SP	FALSE

Table 6.14: Results from default configuration and best configuration found after root node testing. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6). For the instances not included in the table, the root node was not solved within the time limit of eight hours.

	Processed nodes		Upper bound $[\cdot 10^3]$		Lower bound $[\cdot 10^3]$		Optimality gap	
Instance	Default	Best CG	Default	Best CG	Default	Best CG	Default	Best CG
inrc_sprint_late01	186	419	225	243	178	178	20.7 %	26.7 %
inrc_sprint_late02	410	692	152	153	115	115	24.3 %	24.9 %
inrc_sprint_late03	178	420	269	259	208	208	22.5 %	19.7 %
inrc_sprint_late04	157	369	253	266	208	207	17.8 %	22.1 %
inrc_sprint_late05	296	460	inf	269	191	193	inf	28.3 %
inrc_medium_late01	100	283	902	893	783	780	13.1 %	12.7 %
inrc_medium_late02	70	234	960	935	812	805	15.4 %	13.9 %
inrc_medium_late03	47	244	894	899	770	765	13.9 %	14.9 %
inrc_long_late01	14	46	inf	inf	1,680	1,665	inf	inf
inrc_long_late02	15	42	inf	inf	1,651	1,637	inf	inf
inrc_sprint_late02w12	1	25	inf	inf	334	332	inf	inf

The benefit of solving more nodes gives more flexibility to test speed-ups concerning the full branch and price procedure. These might have better impact when the branching tree grows large. There is still need for improvements to show significant benefit of the speed-ups, and this motivates the continued testing presented in this section. If not otherwise stated, the best configuration from the root node is kept as a basis for further solution method speed-ups.

6.5.1 Multiple Variable Aggressive Branching

An alternative branching strategy was proposed in Section 5.5.6 that creates an additional branch where one x-variable is fixed for each employee. By fixing more shifts, it might be more likely to find integer solutions. However, the number of nodes in the tree increases due to the initial branch, and this might increase the computation time.

Testing showed that implementational issues arose with this new strategy because the RAM of the computers used was not sufficient to run the model for a full eight hours. Table 6.15 shows the results from the tests that were conducted, but had to be terminated before reaching the time limit. The strategy seems to produce good bounds compared to other configurations, especially when considering the time used, but the potential effect after eight hours without RAM issues cannot be predicted. These results indicate that the branching strategy might be an interesting subject for future research, but was not considered further in this thesis.

Table 6.15: Results from multiple variable aggressive branching. Note that the tests were terminated before the time limit was reached due to exceeding the RAM available. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

Instance	Computation time [s]	Processed nodes	Upper bound $[\cdot 10^3]$	Lower bound $[\cdot 10^3]$	Optimality gap
inrc_sprint_late01	24 173	350	214	176	17.7%
inrc_sprint_late02	19 853	540	166	113	31.8%
inrc_sprint_late03	23 382	290	241	206	14.5%
inrc_sprint_late04	24 885	310	272	206	24.6%
inrc_sprint_late05	23 863	350	233	192	17.7%
inrc_medium_late01	28 161	230	908	780	14.2%
inrc_medium_late02	24 947	150	939	805	14.2%
inrc_medium_late03	22 268	280	inf	764	inf
inrc_long_late01	25 134	150	inf	1 665	inf
inrc_long_late02	25 292	150	inf	1 637	inf

6.5.2 Column Elimination

Preliminary testing revealed that a large share of the columns generated by the SPs are integer infeasible, i.e. the corresponding roster lines may not be assigned in an integer feasible solution. This is explained by the handling of several employee specific constraints in the RMP, rather than in the SPs where integrality is ensured for each roster line through the network representation. By eliminating integer infeasible columns before solving a node as proposed in Section 5.5.7, a larger share of the RMP columns are expected to be integer feasible throughout the tree search. This is expected to yield a better chance of finding integer feasible solutions. A change in the computation time is also expected. One the one hand, the RMP size is reduced before solving each node and might reduce the computation time of each iteration. On the other hand, more CG iterations are expected because each node starts out with a smaller set of columns likely to be longer away from an LP optimal RMP solution.

The results from testing the best configuration so far with and without column elimination are displayed in Tables 6.16 and 6.17. By removing integer infeasible columns, fewer

Table 6.16: Processed nodes and time distribution with and without column elimination. Times are given as percentages of the total computation time of eight hours.

	Processed nodes		Time solving nodes		Time branching	
Instance \ Column elimination	Off	On	Off	On	Off	On
inrc_sprint_late01	419	169	69.5 %	93.3 %	30.4 %	6.3 %
inrc_sprint_late02	692	364	73.4 %	92.3 %	26.1 %	7.2 %
inrc_sprint_late03	420	172	66.6 %	92.5 %	33.0 %	6.9 %
inrc_sprint_late04	369	138	69.2 %	93.1 %	30.5 %	6.3 %
inrc_sprint_late05	460	150	68.7 %	93.2 %	30.9 %	6.1 %
inrc_medium_late01	283	106	77.8 %	95.2 %	21.6 %	4.1 %
inrc_medium_late02	234	65	67.4 %	92.7 %	32.0 %	6.4 %
inrc_medium_late03	244	68	71.6 %	94.6 %	27.8 %	4.5 %
inrc_long_late01	46	28	61.0 %	91.8 %	38.2 %	7.0 %
inrc_long_late02	42	25	65.6 %	92.8 %	33.7 %	6.1 %
inrc_sprint_late02w12	25	3	81.7 %	97.6 %	17.7 %	1.7 %

Table 6.17: Upper and lower bounds with and without column elimination. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

	Upper bound $[\cdot 10^3]$		Upper bound $[\cdot 10^3]$		Optimality gap	
Instance \ Column elimination	Off	On	Off	On	Off	On
inrc_sprint_late01	243	233	178	177	26.7 %	24.0 %
inrc_sprint_late02	153	157	115	115	24.9 %	26.9 %
inrc_sprint_late03	259	293	208	207	19.7 %	29.3 %
inrc_sprint_late04	266	249	207	206	22.1 %	17.4 %
inrc_sprint_late05	269	223	193	192	28.3 %	13.9 %
inrc_medium_late01	893	858	780	779	12.7 %	9.2 %
inrc_medium_late02	935	888	805	805	13.9 %	9.3 %
inrc_medium_late03	899	855	765	764	14.9 %	10.6 %
inrc_long_late01	inf	inf	1,665	1,665	inf	inf
inrc_long_late02	inf	inf	1,637	1,637	inf	inf
inrc_sprint_late02w12	inf	inf	332	332	inf	inf

nodes are processed in the branching tree. This suggests that the number of CG iterations required in each node increases significantly, thus increasing computation time in each node. Especially for the instance *inrc_sprint_late02w12* the number of nodes are significantly reduced. This might be because the longer planning period makes an SP solution less likely to be feasible with respect to all constraints, thereby causing more columns to be removed in each node. The computation time spent solving the nodes increases when applying column elimination and leaves less time for branching and solving subsequent nodes in the branching tree. As discussed in Section 6.4, the implementation of branching may be slow. However, by eliminating columns, this effect appears to be reduced.

For most instances, the upper bound improves. This seems to confirm that a larger set of columns are integer feasible in many of the nodes, yielding more integer feasible solutions in the branching tree. There is little change in the lower bound. Based on the tendency to produce better upper bounds and optimality gaps, column elimination is kept in the further testing.

6.5.3 Altered Column Generation Stop Criterion and Limited Label Extension

After improving the branch and price algorithm with column elimination, it is interesting to have another look at the CG speed-ups. Since the CG stop criterion and limited label extension discussed in Section 6.4 both effect the trade-off between short computation time and high quality solutions, they are tested again in the full branch and price solution. The main hypothesis is that a slower configuration by an inactive CG stop criterion and less limiting label extension will yield better bounds. To test this, the CG was solved to optimality in each node and the label extension limits were set to $[2, 3]$ since this choice was found to produce integer solutions while still performing well with respect to computation time.

The results for the altered configuration are shown in Table 6.18. As would be expected by a slower configuration, there is a reduction in the number of nodes processed compared to the previous best configuration. This substantiates the tailing off effect discussed with regards to the CG stop criterion. A slower CG configuration leaves less flexibility to improve the branch and price algorithm on a higher level because of the reduced size of the branching tree. The bounds are slightly better than before, but the effect on optimality gaps is small. On this basis, the CG stop criterion and limited label extension are not altered from the best configuration found in Section 6.4 in the further tests.

Table 6.18: Results with altered column generation stop criterion and limited label extension. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

Instance	Processed nodes	Upper bound $[\cdot 10^3]$	Lower bound $[\cdot 10^3]$	Optimality gap
inrc_sprint_late01	99	230	178	23,0 %
inrc_sprint_late02	221	147	115	21,6 %
inrc_sprint_late03	92	282	208	26,4 %
inrc_sprint_late04	96	262	207	21,0 %
inrc_sprint_late05	101	225	193	14,0 %
inrc_medium_late01	56	859	783	8,8 %
inrc_medium_late02	32	891	812	8,8 %
inrc_medium_late03	32	853	770	9,8 %
inrc_long_late01	6	inf	1 680	inf
inrc_long_late02	7	inf	1 651	inf
inrc_sprint_late02w12	1	inf	334	inf

6.5.4 Handling Maximum Consecutive Days in the Sub Problems

In Section 6.4, handling the constraints on maximum consecutive days working in the SPs was found to produce better bounds than the original handling in the RMP. However, due to increased computation time, the original model configuration was used for further testing. Considering the full branch and price algorithm, it is of interest to reevaluate this.

The results from handling maximum consecutive days working in the SPs are displayed in Table 6.19 and compared with the best configuration found so far. The lower bounds are improved for all four week instances with the new configuration. The improved lower bounds are in line with the improved root node bounds seen in Section 6.4. This is due to the tighter formulation of the LP relaxed RMP, leading to a smaller solution space more closely resembling the convex hull of the integer solution space. For the instance *inrc_sprint_late02w12*, the root node was not solved within the time limit of eight hours when handling maximum consecutive days working in the SPs. This points to a trade off between speeding up CG and obtaining better solutions in each node associated with this configuration.

Most instances have an improved upper bound when handling maximum consecutive days in the SPs compared to in the RMP. When moving constraints from the RMP to the SPs, there are probably more integer feasible columns in each LP solution. Thus, it seems the RMP has a better likelihood of finding good integer solutions.

Table 6.19: Processed nodes, upper and lower bound and optimality gap for handling of maximum consecutive days working in the RMP and SP across instances. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

	Processed nodes		Upper bound $[\cdot 10^3]$		Lower bound $[\cdot 10^3]$		Optimality gap	
Instance \Handling	RMP	SP	RMP	SP	RMP	SP	RMP	SP
inrc_sprint_late01	169	61	233	238	177	181	24.0 %	23.9 %
inrc_sprint_late02	364	251	157	137	115	117	26.9 %	14.3 %
inrc_sprint_late03	172	51	293	300	207	210	29.3 %	30.0 %
inrc_sprint_late04	138	120	249	259	206	211	17.4 %	18.7 %
inrc_sprint_late05	150	146	223	209	192	196	13.9 %	6.4 %
inrc_medium_late01	106	71	858	845	779	784	9.2 %	7.2 %
inrc_medium_late02	65	71	888	860	805	820	9.3 %	4.7 %
inrc_medium_late03	68	41	855	835	764	770	10.6 %	7.7 %
inrc_long_late01	28	8	inf	inf	1,665	1,704	inf	inf
inrc_long_late02	25	10	inf	inf	1,637	1,676	inf	inf
inrc_sprint_late02w12	3	1	inf	inf	332	-inf	inf	inf

6.6 Performance of the Improved Configuration

After evaluating all the suggested speed-ups in the previous sections, the improved configuration proposed in the thesis is as shown in Table 6.20. Some speed-ups, such as the CG stop criterion, limited label extension and column elimination, were found to be effective. Others, like partial CG and multiple variable aggressive branching, turned out to make little positive contribution.

Table 6.20: Improved solution method configuration. Configuration parameters are described in Sections 6.4 and 6.5.

Parameter	Value
CG stop criterion	0.01
Limit on SP solutions	10
CG improvement criterion	0.0005
Label extension limits	[1,2,3]
Partial column generation	FALSE
Handle max consecutive days in SP	TRUE
Multiple variable aggressive branching	FALSE
Column elimination	TRUE

In Table 6.21, the performance of the improved configuration is measured along with the interim best configuration identified after CG testing and the default configuration. The

CG speed-ups were seen to have large impact on the computation time per node, but did nevertheless not improve the bounds much. By also implementing column elimination and handling of maximum consecutive days in the SPs, however, the optimality gap was improved for most instances. Particularly good results are seen for the medium instances where optimality gaps were decreased with 44 to 70 percent and the total cost of the best roster solution was reduced with up to ten percent. Substantial improvement was also seen for the instance *inrc_sprint_late05* where the optimality gap improved from *inf* to 6.4 percent. For three of the sprint instances however, the optimality gaps increased, indicating the speed-ups applied do not always have a positive effect.

Table 6.21: Processed nodes, lower and upper bounds and optimality gap for different instances with the improved configuration. Default refers to the default configuration without speed-ups and interim refers to the best configuration identified after CG testing on the root node. Processed nodes is the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

	Processed nodes			Upper bound $[\cdot 10^3]$		
Instance \ Configuration	Default	Interim	Improved	Default	Interim	Improved
inrc_sprint_late01	186	419	61	225	243	238
inrc_sprint_late02	410	692	251	152	153	137
inrc_sprint_late03	178	420	51	269	259	300
inrc_sprint_late04	157	369	120	253	266	259
inrc_sprint_late05	296	460	146	inf	269	209
inrc_medium_late01	100	283	71	902	893	845
inrc_medium_late02	70	234	71	960	935	860
inrc_medium_late03	47	244	41	894	899	835
inrc_long_late01	14	46	8	inf	inf	inf
inrc_long_late02	15	42	10	inf	inf	inf
inrc_sprint_late02w12	1	25	1	inf	inf	inf

	Lower bound $[\cdot 10^3]$			Optimality gap		
Instance \ Configuration	Default	Interim	Improved	Default	Interim	Improved
inrc_sprint_late01	178	178	181	20.7 %	26.7 %	23.9 %
inrc_sprint_late02	115	115	117	24.3 %	24.9 %	14.3 %
inrc_sprint_late03	208	208	210	22.5 %	19.7 %	30.0 %
inrc_sprint_late04	208	207	211	17.8 %	22.1 %	18.7 %
inrc_sprint_late05	191	193	196	inf	28.3 %	6.4 %
inrc_medium_late01	783	780	784	13.1 %	12.7 %	7.2 %
inrc_medium_late02	812	805	820	15.4 %	13.9 %	4.7 %
inrc_medium_late03	770	765	770	13.9 %	14.9 %	7.7 %
inrc_long_late01	1680	1665	1704	inf	inf	inf
inrc_long_late02	1651	1637	1676	inf	inf	inf
inrc_sprint_late02w12	334	332	-inf	inf	inf	inf

Both column elimination and handling of maximum consecutive days in the SPs were found to reduce computation time in each node. Therefore, the CG speed-ups discussed in Section 6.4 are believed to have contributed greatly to the effect of these methods. By having a fast CG solver, several nodes were investigated in spite of the increased time per node for the altered methods on branch and price level.

The final model proposed shows a good ability to handle rostering instances of sizes comparable to what is faced in reality. The scalability in number of employees is a significant achievement, and the medium instances with 30 employees were solved to high quality solutions within eight hours. The initial purpose of the thesis was to also investigate scalability in time, but no twelve week instances were found to yield integer solutions. In retrospect, a test on instances of lengths between four and twelve weeks would be interesting, and is suggested as a subject for future research.

Before presenting the conclusion in Chapter 7, it is worth considering the results of an additional test that was conducted in a curious attempt to speed up the model further by altering not the solution method, but the model itself.

6.6.1 A Simple Stabilization Measure

As mentioned in Section 6.4.3, unstable dual variables is often a challenge in CG and can cause longer computation time (Lübbecke and Desrosiers, 2005). Stabilization techniques are not considered in greater detail in this thesis. However, a simple experiment presents itself to attempt to stabilize the dual variables. By changing the RMP demand coverage equality constraints to inequality constraints, the domains of the dual variables for the constraints are restricted to only positive values. As discussed in Chapter 2, the modeling of demand coverage by inequality constraints is a common choice in operations research. It must, however, be noted that this is a simplification of the model presented in this thesis that neglects over-coverage. Rationale for the inclusion of over-coverage was provided in Section 3.2.

The results of changing the demand coverage constraints to inequality constraints for the improved configuration described by Table 6.20 are displayed in Table 6.22. Twelve week instances are not included because bounds remained indefinite after eight hours computation time. As seen by the number of processed nodes, the constraint relaxation leads to quicker CG, indicating a positive effect of stabilization.

From Table 6.22 it can be seen that the upper bound is lowered for several instances. However, the results are inconclusive. The reason why the relaxed cover constraint is not always better and sometimes produces worse bounds might be bilateral. First, recall that there is no hard limit on the amount of over-coverage that is permitted for any shift in a roster in the instances considered. Therefore, the dual variable change associated with over-coverage for any one shift can at most equal the cost of over-coverage for the shift, thus limiting the possible fluctuations. Second, there are several other constraints in the RMP than the cover constraints also contributing to the dual variables. Therefore, changing the cover constraints only addresses a small part of the total dual variables in

Table 6.22: Processed nodes, upper and lower bound and optimality gap for modeling demand coverage with equality or inequality constraints across instances. The improved configuration as described in Table 6.20 is used for comparison. Processed nodes refers to the number of branching tree nodes that have been solved and/or pruned. The upper bound is the objective value of the best found integer solution and the lower bound is the optimistic bound for the objective value of the optimal integer solution. The optimality gap is calculated by eq. (5.6).

	Processed nodes		Upper bound $[\cdot 10^3]$		Lower bound $[\cdot 10^3]$		Optimality gap	
Instance \ Constraint	Equality	Inequality	Equality	Inequality	Equality	Inequality	Equality	Inequality
inrc_sprint_late01	61	51	238	224	181	181	23.9 %	19.2 %
inrc_sprint_late02	251	346	137	136	117	118	14.3 %	13.4 %
inrc_sprint_late03	51	88	300	267	210	210	30.0 %	21.1 %
inrc_sprint_late04	120	120	259	260	211	210	18.7 %	19.2 %
inrc_sprint_late05	146	161	209	227	196	196	6.4 %	13.7 %
inrc_medium_late01	71	78	845	840	784	784	7.2 %	6.6 %
inrc_medium_late02	71	62	860	861	820	820	4.7 %	4.8 %
inrc_medium_late03	41	40	835	837	770	771	7.7 %	8.0 %
inrc_long_late01	8	7	inf	inf	1,704	1,705	inf	inf
inrc_long_late02	10	12	inf	inf	1,676	1,677	inf	inf

the problem. A more thorough analysis of stabilization techniques is not conducted, but suggested as a subject of future research.

Chapter 7

Concluding Remarks

In this chapter, the conclusion of the thesis is presented in Section 7.1 and areas for future research are suggested in Section 7.2.

7.1 Conclusion

A new approach to solving the healthcare rostering problem that contributes to closing the gap between operations research theory and real world healthcare rostering practice has been proposed in this thesis. The need for improved methods for rostering was identified through a literature review and healthcare rostering was found to stand out as an application area of particular interest due to the high complexity and large institutions.

With particular inspiration from rostering in Nordic healthcare institutions, the HRP was described and a mathematical model was formulated with Dantzig-Wolfe decomposition to be solved with a branch and price algorithm. The inclusion of several employee preferences, such as working patterns stretching over more days, is an extension of the typical constraints modeled in the literature. The restricted master problem used in column generation was formulated as a mixed integer program and the employee specific sub problems as shortest path problems with resource constraints. To model a wider variety of constraints than is common in the literature without making the SPs too complex, some of the employee specific constraints were handled in the RMP instead of in the SPs. This allocation of constraints differentiates the model in this thesis from the common modeling in literature.

The proposed model was solved using a branch and price algorithm. In the CG procedure in each branching tree node, the LP-relaxed RMP was solved as a linear program by the simplex method while the SPPRC-formulated SPs were solved with dynamic programming using a labelling algorithm. The choice of solution method was motivated by approaches that have shown good results for similar problems in the literature. Inspired by the success of speed-up techniques in research, and motivated by the complexity of the proposed model, several speed-up techniques were proposed and evaluated. A modified set of commonly used benchmark test instances were used to assess the performance of the model. Modifications were intended to give a better mirroring of realistic rostering situations in Nordic healthcare institutions. The instances ranged from ten to 50 employees and four to twelve weeks long planning periods.

Applying the default branch and price solution method yielded no optimality gaps below ten percent within eight hours of computation time, and for one of the five smallest instances, *inrc_sprint_late05*, a feasible integer solution was not found. Thus, a need for speed-ups was revealed. In CG, increasing the number of solutions retrieved from each SP in each iteration and limiting the number of extended labels from each node when applying dynamic programming are examples of speed-ups implemented successfully. Reduction in computation time was also achieved by stopping CG before reaching optimality at a specified optimality gap. Partial CG was on the other hand not found to reduce computation times nor produce better solutions.

On a higher level in the branch and price algorithm, column elimination in each node was one of the speed-ups with the greatest effect. A change in the model configuration, in which additional employee specific constraints were handled in the SPs contributed to better bounds although it increased computation time in each node. A new multiple variable aggressive branching strategy was proposed, but unsuccessful testing and inconclusive results invite further research.

After applying the best combination of speed-ups found, improvements in most optimality gaps were found. Especially for the *inrc_medium_late* instances, the gaps were decreased with 44 to 70 percent compared to the default model. In the case of *inrc_sprint_late05*, where no optimality gap was found with the default configuration, a 6.4 percent gap was found after applying speed-ups. For some instances, however, the speed-ups increased the optimality gap up to 7.5 percentage points higher, reminding that speed-ups will sometimes improve and other times worsen solutions. The model showed good scalability in number of employees, and high quality rosters were obtained for instances with up to 30 employees.

In conclusion, this thesis has contributed to healthcare rostering research by producing a detailed description of a new model able to handle a wide range of realistic rostering rules and proposed and tested solution methods to solve the model. In evaluating the model, a collection of speed-up techniques have been tested and may be used for reference when considering implementing speed-ups for similar problems. The results from applying branch and price and several speed-up techniques show promising results, but further research is required to be able to solve instances with more than 30 employees or longer planning periods than four weeks.

7.2 Future Research

Through the thesis work, several areas of future research have been identified. The literature review in Chapter 2 revealed the need for improved rostering methods in healthcare in general to enable implementation of automated rostering systems in real world applications. With basis in the model and methods proposed in the thesis, several areas of future research that may contribute towards the goal of real world automated rostering systems have presented themselves.

One such area of future research is continued work on the optimal combination of the speed-ups suggested in this thesis for the decomposed model. Due to constrained time, only a selection of solution method configurations were tested and the improvement was shown to be considerable. Interaction effects were however not studied in detail and there are several more combinations of speed-ups that can be tested.

Another relevant area for future research is the implementation of the model and solution methods proposed. Implementation was described in Section 6.1 and does most likely not constitute the most efficient approach. The effect of improved implementation done by more experiences programmers could give a better impression of the efficiency of the solution methods proposed compared to other research. In this case, the results regarding effects of the suggested solution methods can still be used in the modeling design and choice of solution methods.

The new decomposition proposed in Chapter 4 where some employee specific constraints are handled in the RMP invites further investigation of the most ideal allocation of constraints. This thesis in large makes use of the results from the preparatory research project for the model decomposition. However, it was found in Section 6.5.4 that moving constraints on the maximum number of consecutive days working from the RMP to the SPs improved the solutions found. Other options exist for the allocation of constraints between the RMP and the SPs, and building on the main decomposition idea from this thesis provides a good start for developing even more efficient model formulations.

Investigating new decomposition formulations is in large dependent on the actual problem description considered. For future research, it is interesting to test the model proposed in this thesis on a wider set of test instances, and evaluate the flexibility of the model. The scalability in, for example, length of the planning period is a subject of future research. Depending on the instances, changes in constraints considered and their formulation is an area of possible improvement.

The last domain of future research suggested is the continued work on speed-up techniques that might improve the branch and price algorithm. Several options exist that were not tested in this thesis. These include stabilization techniques, other branching strategies and dual variable fathoming to mention a few. Combination of the branch and price procedure with heuristics to e.g. find good initial feasible solutions are also interesting to consider.

References

Abobaker, R. A., Ayob, M., and Hadwan, M. (2011). Greedy constructive heuristic and local search algorithm for solving nurse rostering problems. In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 194–198. IEEE.

Aickelin, U. and Dowsland, K. A. (2000). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of scheduling*, 3(3):139–153.

Asta, S., Özcan, E., and Curtois, T. (2016). A tensor based hyper-heuristic for nurse rostering. *Knowledge-based systems*, 98:185–199.

Baker, K. R. (1976). Workforce allocation in cyclical scheduling problems: A survey. *Journal of the Operational Research Society*, 27(1):155–167.

Bard, J. F. and Purnomo, H. W. (2005). Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164:510–534.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329.

Beliën, J. and Demeulemeester, E. (2006). Scheduling trainees at a hospital department using a branch-and-price approach. *European journal of operational research*, 175(1):258–278.

Beliën, J. and Demeulemeester, E. (2007). On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem. *Annals of Operations Research*, 155(1):143–166.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–515.

Bilgin, B., De Causmaecker, P., Rossie, B., and Berghe, G. V. (2012). Local search neighbourhoods for dealing with a novel nurse rostering model. *Annals of Operations Research*, 194(1):33–57.

Burke, E. K. and Curtois, T. (2011). New computational results for nurse rostering benchmark instances. Technical report, Technical report.

Burke, E. K. and Curtois, T. (2014). New approaches to nurse rostering benchmark instances. *European Journal of Operational Research*, 237(1):71–81.

Burke, E. K., De Causmaecker, P., Petrovic, S., and Berghe, G. V. (2002). A multi criteria meta-heuristic approach to nurse rostering. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1197–1202. IEEE.

Burke, E. K., De Causmaecker, P., Vanden Berghe, G., and Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499.

Burke, E. K., Kendall, G., and Soubeiga, E. (2003). A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, 9(6):451–470.

Ceschia, S., Dang, N. T. T., De Causmaecker, P., Haspeslagh, S., and Schaerf, A. (2015). Second international nurse rostering competition (inrc-ii)—problem description and rules—. *arXiv preprint arXiv:1501.04177.*

Cheang, B., Li, H., Lim, A., and Rodrigues, B. (2003). Nurse rostering problems—a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460.

Coates, S. S. H. and Pedersen, V. (2018). Solving the roster line problem in healthcare with column generation and dynamic programming. Unpublished. Project Report, Norwegian University of Science and Technology.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The computer journal*, 8(3):250–255.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations research*, 8(1):101–111.

Dantzig, G. B. and Wolfe, P. (1961). The decomposition algorithm for linear programs. *Econometrica: Journal of the Econometric Society*, pages 767–778.

De Causmaecker, P. and Berghe, G. V. (2002). Relaxation of coverage constraints in hospital personnel rostering. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 129–147. Springer.

De Causmaecker, P. and Berghe, G. V. (2004). Novel meta-heuristic approaches to nurse rostering problems in belgian hospitals.

De Grano, M. L., Medeiros, D., and Eitel, D. (2009). Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Management Science*, 12(3):228.

Den Norske Legeforening (2017). Arbeidstid for sykehusleger. Retrieved May 28 2019 from https://beta.legeforeningen.no/jus-og-arbeidsliv/arbeidsforhold/arbeidstid/arbeidstid-for-sykehusleger/.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2002). Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. In *Essays and surveys in metaheuristics*, pages 309–324. Springer.

Desaulniers, G., Desrosiers, J., and Solomon, M. M. (2006). *Column generation*, volume 5. Springer Science & Business Media.

Desaulniers, G., Desrosiers, J., Solomon, M. M., Soumis, F., Villeneuve, D., et al. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In *Fleet management and logistics*, pages 57–93. Springer.

Dohn, A. and Mason, A. (2013). Branch-and-price for staff rostering: An efficient implementation using generic programming and nested column generation. *European Journal of Operational Research*, 230(1):157–169.

Edie, L. C. (1954). Traffic delays at toll boths. *Journal of the Operations Research Society of America*, 2(2):107–138.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations research*, 47(2):247–263.

Gérard, M., Clautiaux, F., and Sadykov, R. (2016). Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. *European Journal of Operational Research*, 252:1019–1030.

Hans, E. W., Van Houdenhoven, M., and Hulshof, P. J. (2012). A framework for healthcare planning and control. In *Handbook of healthcare system scheduling*, pages 303–320. Springer.

Haspeslagh, S., De Causmaecker, P., Schaerf, A., and Stølevik, M. (2014). The first international nurse rostering competition 2010. *Annals of Operations Research*, 218(1):221–236.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In *Column Generation*, chapter 2. Springer.

Jaumard, B., Semet, F., and Vovor, T. (1998). A generalized linear programming model for nurse scheduling. *European journal of operational research*, 107(1):1–18.

Jensen, L., Horsted, C., Lunde, A., and Hansen, M. B. (2008). *Vagtplanlægning i det danske sygehusvæsen*. Syddansk Universitet.

Kellogg, D. L. and Walczak, S. (2007). Nurse scheduling: from academia to implementation or not? *Interfaces*, 37(4):355–369.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520.

Lübbecke, M. E. (2005). Dual variable based fathoming in dynamic programs for column generation. *European Journal of Operational Research*, 162(1):122–125.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations research*, 53(6):1007–1023.

Lundgren, J., Rönnqvist, M., and Värbrand, P. (2010). *Optimization, Studentlitteratur*. Studentlitteratur AB, Lund.

Lusby, R., Dohn, A., Range, T. M., and Larsen, J. (2012). A column generation-based heuristic for rostering with work patterns. *Journal of the Operational Research Society*, 63:261–277.

Maenhout, B. and Vanhoucke, M. (2010). Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling*, 13(1):77–93.

Mason, A. J. and Smith, M. C. (1998). A nested column generator for solving rostering problems with integer programming. In *International conference on optimisation: techniques and applications*, pages 827–834. Curtin University of Technology Perth, Australia.

Ministry of Labour and Social Affairs (2005). Lov om arbeidsmiljø, arbeidstid og stillingsvern mv. (arbeidsmiljøloven). Retrieved April 26 2019 from https://lovdata.no/dokument/NL/lov/2005-06-17-62.

Ministry of Local Government and Modernization, LO Stat, Unio, and YS Stat (2018). Hovedtariffavtalen i staten 1. mai 2018 – 30. april 2020 for LO Stat, Unio og YS Stat. Retrieved April 26 2019 from https://www.regjeringen.no/contentassets/43efadcb4e394a5fa57176b00f7b07ea/2018/hovedtariffavtalen_2018-20_lostat_unio_ysstat.pdf.

Moz, M. and Pato, M. V. (2004). Solving the problem of rerostering nurse schedules with hard constraints: New multicommodity flow models. *Annals of Operations Research*, 128(1-4):179–197.

Norsk Sykepleierforbund (2019). Daglig og ukentlig fritid og pauser. Retrieved May 28 2019 from https://www.nsf.no/vis-artikkel/113741/17074/Daglig-og-ukentlig-fritid-og-pauser.

Ovchinnikov, A. and Milner, J. (2008). Spreadsheet model helps to assign medical residents at the university of vermont's college of medicine. *Interfaces*, 38(4):311–323.

Rahimian, E., Akartunalı, K., and Levine, J. (2017). A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research*, 258(2):411–423.

Rönnberg, E. and Larsson, T. (2010). Automating the self-scheduling process of nurses in swedish healthcare: a pilot study. *Health Care Management Science*, 13:35–53.

Smet, P., Bilgin, B., Causmaecker, P. D., and Vanden Berghe, G. (2014). Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*, 218:303–326.

Smet, P., De Causmaecker, P., Bilgin, B., and Berghe, G. V. (2013). Nurse rostering: a complex example of personnel scheduling with perspectives. In *Automated Scheduling and Planning*, pages 129–153. Springer.

Smet, P., Ernst, A. T., and Vanden Berghe, G. (2016). Heuristic decomposition approaches for an integrated task scheduling and personnel rostering problem. *Computers Operations Research*, 76:60–72.

Trilling, L., Guinet, A., and Le Magny, D. (2006). Nurse scheduling using integer linear programming and constraint programming. *IFAC Proceedings Volumes*, 39(3):671–676.

Václavík, R., Novák, A., Šcha, P., and Hanzálek, Z. (2018). Accelerating the branch-and-price algorithm using machine learning. *European Journal of Operational Research*, 271(3):1055–1069.

Valouxis, C., Gogos, C., Goulas, G., Alefragis, P., and Housos, E. (2012). A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2):425–433.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *European journal of operational research*, 226(3):367–385.

Van der Veen, E., Hans, E. W., Post, G. F., and Veltman, B. (2015). Shift rostering using decomposition: assign weekend shifts first. *Journal of Scheduling*, 18:29–43.

Vanden Berghe, G. (2002). *An Advanced Model and Novel Meta-Heuristic Solution Methods to Personnel Scheduling in Healthcare*. PhD thesis, University of Gent, Belgium.

Vanhoucke, M. and Maenhout, B. (2007). Nsplib—a nurse scheduling problem library: a tool to evaluate (meta-) heuristic procedures. In *Operational research for health policy: making better decisions, proceedings of the 31st annual meeting of the working group on operations research applied to health services*, pages 151–165.

Appendix A

Compact Formulation of the Mathematical Model

A compact formulation of the mathematical model presented in Chapter 4 is presented in this appendix and intended to give a better overview of the model. First, sets and parameters applicable to both the restricted master problem and the sub problems are collected in Sections A.1 and A.2. The RMP is presented in Section A.3 before the SPs are summarized in Section A.4.

A.1 Sets

$\mathcal{E}$	Employees
$\mathcal{K}_e$	Available roster lines for employee $e \in \mathcal{E}$
$\mathcal{D}$	Days in planning period
$\mathcal{W}$	Weeks in planning period
$\mathcal{S}$	Shift types
$\mathcal{S}^W$	Shift types representing working shifts, $\mathcal{S}^W \subseteq \mathcal{S}$
$\mathcal{S}^O$	Shift types representing a day off, $\mathcal{S}^O \subseteq \mathcal{S}$
$\mathcal{G}$	Working shift groups (e.g. *Morning, Evening, Night*)
$\mathcal{S}^g$	Shift types categorized to shift group g, $g \in \mathcal{G}$
$\mathcal{T}$	Skill levels
$\mathcal{T}_s^S$	Skill levels that qualify for working shift type $s \in \mathcal{S}$
$\mathcal{T}_e^E$	Skill levels for employee $e \in \mathcal{E}$
$\mathcal{S}_e$	Shift types that can be assigned to employee $e \in \mathcal{E}$
$\mathcal{S}_e^W$	Working shift types that can be assigned to employee $e \in \mathcal{E}$
$\mathcal{S}_s^I$	Shift types not allowed to follow shift type $s \in \mathcal{S}$ due to little rest
$\mathcal{S}_s^R$	Shift types that incur a penalty for little rest if following shift type $s \in \mathcal{S}$
$\mathcal{S}_s^{S1}$	Shift types that cannot follow shift type $s \in \mathcal{S}$ with a strict day off in between
$\mathcal{S}_s^{S2}$	Shift types that cannot follow shift type $s \in \mathcal{S}$ with two strict days off in between
$\mathcal{P}_e^P$	Shift patterns that are penalized for employee $e \in \mathcal{E}$
$\mathcal{P}_e^R$	Shift patterns that are rewarded for employee $e \in \mathcal{E}$
$\mathcal{P}_e^I$	Shift patterns that are illegal for employee $e \in \mathcal{E}$
$\mathcal{D}_w$	Days in week $w \in \mathcal{W}$
$\mathcal{B}$	Set of weekdays ($\mathcal{B} = \{$MON, TUE, WED, THU, FRI, SAT, SUN$\}$)
$\mathcal{D}^b$	Days in planning period that are on weekday $b \in \mathcal{B}$
$\mathcal{D}_w^b$	Day $b \in \mathcal{B}$ in week $w \in \mathcal{W}$
$\mathcal{D}^N$	First day in each norm period for workload calculation
$\mathcal{B}_{ep}$	Weekdays when pattern $p \in \{\mathcal{P}_e^I, \mathcal{P}_e^P, \mathcal{P}_e^R\}$ is allowed to begin, $\mathcal{B}_{ep} \subseteq \mathcal{B}$

A.2 Parameters

Costs

C_{ek}^K	$\in \mathbb{R}$	Cost of assigning roster line $k \in \mathcal{K}_e$ to employee $e \in \mathcal{E}$
C_{ds}^+	$\in \mathbb{R}^+$	Cost per employee for over-coverage on shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$
C_{ds}^-	$\in \mathbb{R}^+$	Cost per employee for under-coverage on shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$
C_{eds}	$\in \mathbb{R}$	Cost of employee $e \in \mathcal{E}$ working shift $s \in \mathcal{S}$ on day $d \in \mathcal{D}$
C^R	$\in \mathbb{R}^+$	Cost for each day with reduced rest
C_e^{N-}	$\in \mathbb{R}^+$	Cost per hour working less than contracted in norm period for employee $e \in \mathcal{E}$
C_e^{N+}	$\in \mathbb{R}^+$	Cost per hour working more than contracted in norm period for employee $e \in \mathcal{E}$

Variable parameters

$\overline{W}^{+}_{ds}$	$\in \mathbb{Z}^{+}$	Limit on over-coverage for shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$
$\overline{W}^{-}_{ds}$	$\in \mathbb{Z}^{+}$	Limit on under-coverage for shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$
$\overline{U}^{+}_{e}$	$\in \mathbb{R}^{+}$	Limit on overtime during a norm period
$\overline{U}^{-}_{e}$	$\in \mathbb{R}^{+}$	Limit on undertime during a norm period

Roster line attributes

A_{ekds}	$\in \{0,1\}$	Indicates if employee $e \in \mathcal{E}$ works shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$ in roster line $k \in \mathcal{K}_e$
A^{W}_{ekd}	$\in \{0,1\}$	Indicates if employee $e \in \mathcal{E}$ has a working shift on day $d \in \mathcal{D}$ in roster line $k \in \mathcal{K}_e$
A^{O}_{ekd}	$\in \{0,1\}$	Indicates if employee $e \in \mathcal{E}$ has a day off on day $d \in \mathcal{D}$ in roster line $k \in \mathcal{K}_e$
A^{g}_{ekd}	$\in \{0,1\}$	Indicates if employee $e \in \mathcal{E}$ is assigned shift group $g \in \mathcal{G}$ on day $d \in \mathcal{D}$ in roster line $k \in \mathcal{K}_e$
V_{ekd}	$\in \{0,1\}$	Indicates if employee $e \in \mathcal{E}$ has a shift assignment on day $d \in \mathcal{D}$ that incurs a rest penalty in roster line $k \in \mathcal{K}_e$
D_{ekd}	$\in \mathbb{R}^{+}$	Workload for employee $e \in \mathcal{E}$ in roster line $k \in \mathcal{K}_e$ during norm period starting on day $d \in \mathcal{D}^{N}$

Demand

B_{ds}	$\in \mathbb{Z}^{+}$	Demand for employees working shift type $s \in \mathcal{S}$ on day $d \in \mathcal{D}$

Time

T^{S}_{s}	$\in \mathbb{R}^{+}$	Start time of shift type $s \in \mathcal{S}^{W}$
T^{E}_{s}	$\in \mathbb{R}^{+}$	End time of shift type $s \in \mathcal{S}^{W}$
T_{s}	$\in \mathbb{R}^{+}$	Workload of shift type $s \in \mathcal{S}^{W}$
H	$= 24$	Hours in a day

Assorted constraint parameters

$\overline{N}$	$\in \mathbb{N}$	Maximum length of an on stretch
$\overline{N}^{g}$	$\in \mathbb{N}$	Maximum days working consecutive shifts in shift group $g \in \mathcal{G}$
$\underline{N}_{e}$	$\in \mathbb{N}$	Minimum length of an on stretch for employee $e \in \mathcal{E}$
$\underline{N}^{g}_{e}$	$\in \mathbb{N}$	Minimum days working consecutive shifts in shift group $g \in \mathcal{G}$ for employee $e \in \mathcal{E}$
W^{W}	$\in \mathbb{N}$	Number of weeks in period subject to constraints on number of free weekends
$\underline{N}^{W}$	$\in \mathbb{N}_0$	Minimum number of weekends off during a period of W^{W} weeks
D^{S}	$\in \mathbb{N}$	Number of days in period with requirements on strict days off
$\underline{N}^{S}$	$\in \mathbb{N}_0$	Minimum number of strict days off in D^{S} days
D^{R}	$\in \mathbb{N}$	Number of days in period with requirements on resting days
$\overline{N}^{R}$	$\in \mathbb{N}_0$	Maximum number of days with little rest in D^{R} days
H^{S1}	$\in \mathbb{R}^{+}$	Number of hours off required for one strict day off
H^{S2}	$\in \mathbb{R}^{+}$	Number of hours off required for two strict days off
N^{N}	$\in \mathbb{N}$	Number of days in norm period for workload calculation
W^{N}	$\in \mathbb{N}$	Number of weeks in norm period for workload calculation ($= \frac{N^{N}}{7}$)
H^{W}_{e}	$\in \mathbb{R}^{+}$	Number of contracted working hours per week for employee $e \in \mathcal{E}$

Patterns

P_{ep}	$\in \mathbb{R}^+$	Penalty for working pattern $p \in \mathcal{P}^P$ for employee $e \in \mathcal{E}$
R_{ep}	$\in \mathbb{R}^+$	Reward for working pattern $p \in \mathcal{P}^R$ for employee $e \in \mathcal{E}$
D^P_{ep}	$\in \mathbb{N}$	Duration (number of days) of pattern $p \in \{\mathcal{P}^P, \mathcal{P}^R, \mathcal{P}^I\}$ for employee $e \in \mathcal{E}$
$M_{epd'g}$	$\in \{0,1\}$	1 if the shift group on day d' in pattern p for employee e is in group $g \in \mathcal{G}$

A.3 Restricted Master Problem

The RMP is formulated as a set partitioning problem with added constraints regarding individual roster lines. Because not all available roster lines are generated, the term "restricted" is used. This section presents the RMP step-by-step, starting with the variables, then the objective function and finally the constraints in the problem.

A.3.1 Variables

λ_{ek}	$\in \{0,1\}$	$e \in \mathcal{E}, k \in \mathcal{K}_e$	Variables indicating if employee e works roster line k
w^+_{ds}	$\in [0, \overline{W}^+_{ds}]$	$d \in \mathcal{D}, s \in \mathcal{S}$	Over-coverage of shift type s on day d
w^-_{ds}	$\in [0, \overline{W}^-_{ds}]$	$d \in \mathcal{D}, s \in \mathcal{S}$	Under-coverage of shift type s on day d
s_{ed}	$\in \{0,1\}$	$e \in \mathcal{E}, d \in \mathcal{D}$	Variables indicating if employee e has a strict
u^+_{ed}	$\in [0, \overline{U}^+_e]$	$e \in \mathcal{E}, d \in \mathcal{D}^N$	Overtime in norm period starting on day d
u^-_{ed}	$\in [0, \overline{U}^-_e]$	$e \in \mathcal{E}, d \in \mathcal{D}^N$	Undertime in norm period starting on day d day off on day d
m_{epd}	$\in \{0,1\}$	$e \in \mathcal{E}, p \in \mathcal{P}^P_e \cup \mathcal{P}^R_e$ $D^P_{ep} \geq 3, d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b$ $d \leq \lvert\mathcal{D}\rvert - D^P_{ep} + 1$	Variable indicating if for employee e pattern p is worked that started on day d

A.3.2 Objective Function

The objective function minimizes the costs of roster line assignment and over- or under-coverage of each shift. Costs for over- and undertime and pattern penalties are also added, and pattern rewards are subtracted.

$$\begin{aligned}
\min \quad & \sum_{e \in \mathcal{E}} \sum_{k \in \mathcal{K}_e} C^K_{ek} \lambda_{ek} + \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}^W} C^+_{ds} w^+_{ds} + \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}^W} C^-_{ds} w^-_{ds} \\
& + \sum_{e \in \mathcal{E}} C^{N+}_e \sum_{d \in \mathcal{D}^N} u^+_{ed} + \sum_{e \in \mathcal{E}} C^{N-}_e \sum_{d \in \mathcal{D}^N} u^-_{ed} \\
& + \sum_{e \in \mathcal{E}} \sum_{\substack{p \in \mathcal{P}^P_e \\ D^P_{ep} \geq 3}} P_{ep} \sum_{\substack{d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - D^P_{ep} + 1}} m_{epd} - \sum_{e \in \mathcal{E}} \sum_{\substack{p \in \mathcal{P}^R_e \\ D^P_{ep} \geq 3}} R_{ep} \sum_{\substack{d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}| - D^P_{ep} + 1}} m_{epd}
\end{aligned}$$

A.3.3 Rostering Constraints

The rostering constraints are the constraints regarding demand coverage and roster line assignment. Note that the binary requirement on the λ-variables must be relaxed to obtain the dual variables π_{ds} and ω_e.

$$\begin{aligned}
\sum_{e \in \mathcal{E}} \sum_{k \in \mathcal{K}_e} A_{ekds} \lambda_{ek} + w^-_{ds} - w^+_{ds} = B_{ds} \qquad & s \in \mathcal{S}^W, d \in \mathcal{D} \quad | \quad \pi_{ds} \in \mathbb{R} \\
\sum_{k \in \mathcal{K}_e} \lambda_{ek} = 1 \qquad & e \in \mathcal{E} \quad | \quad \omega_e \in \mathbb{R} \\
\lambda_{ek} \in \{0, 1\} \qquad & e \in \mathcal{E}, k \in \mathcal{K}_e \\
0 \leq w^+_{ds} \leq \overline{W}^+_{ds} \qquad & s \in \mathcal{S}^W, d \in \mathcal{D} \\
0 \leq w^-_{ds} \leq \overline{W}^-_{ds} \qquad & s \in \mathcal{S}^W, d \in \mathcal{D}
\end{aligned}$$

A.3.4 Roster Line Constraints

The roster line constraints regard the construction of a roster line for a single employee. Some roster line constraints are enforced in the RMP and are described here. Where applicable, the belonging dual variables are defined to the right of each constraint.

Maximum consecutive days working

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=d}^{d+\overline{N}} A^W_{ekd'} \lambda_{ek} \leq \overline{N} \qquad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - \overline{N} \quad | \quad \alpha^W_{ed} \leq 0$$

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=d}^{d+\overline{N}^g} A^g_{ekd'} \lambda_{ek} \leq \overline{N}^g \qquad \begin{array}{l} e \in \mathcal{E}, g \in \mathcal{G}, d \in \mathcal{D}, \\ d \leq |\mathcal{D}| - \overline{N}^g \end{array} \quad \Bigg| \quad \alpha^g_{ed} \leq 0$$

Number of days with reduced rest

$$\sum_{k \in \mathcal{K}_e} \sum_{d'=d}^{d+D^R-1} V_{ekd'} \lambda_{ek} \leq \overline{N}^R \qquad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - D^R + 1 \quad | \quad \gamma_{ed} \leq 0$$

Strict days off

$$\mathcal{S}_s^{S1} = \{s' \in \mathcal{S}^W \mid T_{s'}^S + 2H - T_s^E < H^{S1}\}$$
$$\mathcal{S}_s^{S2} = \{s' \in \mathcal{S}^W \mid T_{s'}^S + 3H - T_s^E < H^{S2}\}$$

$$s_{ed} - \sum_{k \in \mathcal{K}_e} A_{ekd}^O \lambda_{ek} \leq 0 \quad e \in \mathcal{E}, d \in \mathcal{D} \qquad | \quad \epsilon_{ed}^{\mathrm{O}} \leq 0$$

$$\sum_{k \in \mathcal{K}_e} (A_{ek(d-1)s_1} + A_{ek(d+1)s_2}) \lambda_{ek} + s_{ed} \leq 2 \qquad \Bigg| \quad \epsilon_{eds_1s_2}^{\mathrm{I}} \leq 0$$
$$e \in \mathcal{E}, d \in \mathcal{D} \backslash \{1, |\mathcal{D}|\}, s_1 \in \mathcal{S}_e^W, s_2 \in \mathcal{S}_{s_1}^{S1}$$

$$\sum_{k \in \mathcal{K}_e} (A_{ek(d-1)s_1} + A_{ek(d+2)s_2}) \lambda_{ek} + s_{ed} + s_{e(d+1)} \leq 3 \qquad \Bigg| \quad \epsilon_{eds_1s_2}^{\mathrm{II}} \leq 0$$
$$e \in \mathcal{E}, d \in \mathcal{D} \backslash \{1, |\mathcal{D}| - 1, |\mathcal{D}|\}, s_1 \in \mathcal{S}_e^W, s_2 \in \mathcal{S}_{s_1}^{S2}$$

$$\sum_{d'=d}^{d+D^S-1} s_{ed'} \geq \underline{N}^S \quad e \in \mathcal{E}, d \in \mathcal{D}, d \leq |\mathcal{D}| - D^S + 1$$

$$s_{ed} \in \{0, 1\} \quad e \in \mathcal{E}, d \in \mathcal{D}$$

Workload

$$\sum_{k \in \mathcal{K}_e} D_{ekd} \lambda_{ek} - u_{ed}^+ + u_{ed}^- = W^N H_e^W \qquad e \in \mathcal{E}, d \in \mathcal{D}^N \quad | \quad \zeta_{ed} \in \mathbb{R}$$
$$0 \leq u_{ed}^+ \leq \overline{U}_e^+ \qquad e \in \mathcal{E}, d \in \mathcal{D}^N$$
$$0 \leq u_{ed}^- \leq \overline{U}_e^- \qquad e \in \mathcal{E}, d \in \mathcal{D}^N$$

Patterns

$$\sum_{k\in\mathcal{K}_e}\sum_{d'=1}^{D^P_{ep}}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \leq D^P_{ep}-1 \qquad \theta^I_{epd}\leq 0$$
$$e\in\mathcal{E}, p\in\mathcal{P}^I_e, D^P_{ep}\geq 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1$$

$$\sum_{k\in\mathcal{K}_e}\sum_{d'=1}^{D^P_{ep}}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \leq D^P_{ep}+m_{epd}-1 \qquad \theta^P_{epd}\leq 0$$
$$e\in\mathcal{E}, p\in\mathcal{P}^P_e, D^P_{ep}\geq 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1$$

$$\sum_{k\in\mathcal{K}_e}\sum_{g\in\mathcal{G}} M_{epd'g}A^g_{ek(d+d'-1)}\lambda_{ek} \geq m_{epd} \qquad \theta^R_{epdd'}\geq 0$$
$$e\in\mathcal{E}, p\in\mathcal{P}^R_e, D^P_{ep}\geq 3$$
$$d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1, d'\in\{1,\dots,D^P_{ep}\}$$

$$\sum_{d'=d}^{d+D^P_{ep}-2} m_{epd'}\leq 1 \qquad e\in\mathcal{E}, p\in\mathcal{P}^R_e, D^P_{ep}\geq 3, d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b$$
$$d\leq|\mathcal{D}|-2D^P_{ep}+3$$

$$e\in\mathcal{E}, p\in\mathcal{P}^P_e\cup\mathcal{P}^R_e, D^P_{ep}\geq 3$$
$$m_{epd}\in\{0,1\} \qquad d\in\bigcup_{b\in\mathcal{B}_{ep}}\mathcal{D}^b, d\leq|\mathcal{D}|-D^P_{ep}+1$$

A.4 Sub Problems

The SPs are formulated as shortest path problems with resource constraints. The formulation of the problem for an employee $e\in\mathcal{E}$ follows. A general introduction to the network, cost recursion and resources is given in Section 4.2. First in this section, the SPs are defined using the flow variables y_{edij}. Thereafter, the calculation of relevant RMP parameters based on an SP solution is given and the transform from an SP solution to an x-solution is stated. The SPs assume a known RMP solution with dual variables is given.

A.4.1 Variables

$y_{edij} \quad \in \{0,1\} \quad e \in \mathcal{E}, d \in \mathcal{D} \cup \{0, |\mathcal{D}|+1\}, i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e \cup \{0\}$

Variables indicating if employee e is assigned shift type i on day $(d-1)$ and shift type j on day d

A.4.2 Flow constraints

$$\sum_{i \in \mathcal{S}_e} y_{edis} - \sum_{j \in \mathcal{S}_e} y_{e(d+1)sj} = 0 \quad d \in \mathcal{D}, s \in \mathcal{S}_e$$

$$\sum_{j \in \mathcal{S}_e} y_{e10j} = 1$$

$$\sum_{i \in \mathcal{S}_e} y_{e(|\mathcal{D}|+1)i0} = 1$$

A.4.3 Objective Function

The objective function includes direct costs of a roster line as well as terms from the dual variables of an RMP solution. In the expression given, the direct costs are collected first before the dual variable terms are added. The cost of an arc in the SPPRC network is given as the sum of objective coefficients for the corresponding flow variable.

$$
\begin{aligned}
\min \quad & \sum_{d\in\mathcal{D}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e} C_{edj} y_{edij} + \sum_{d\in\mathcal{D}} \sum_{i\in\mathcal{S}_e^W} \sum_{j\in\mathcal{S}_i^R} C^R y_{edij} \\
& + \sum_{\substack{p\in\mathcal{P}_e^P \\ D_{ep}^P=2}} \sum_{\substack{d\in \bigcup\limits_{b\in\mathcal{B}_{ep}} \mathcal{D}^b \\ d\leq|\mathcal{D}|-1}} \sum_{g\in\mathcal{G}} \sum_{g'\in\mathcal{G}} \sum_{i\in\mathcal{S}^g} \sum_{j\in\mathcal{S}^{g'}} P_{ep} M_{ep1g} M_{ep2g'} y_{e(d+1)ij} \\
& - \sum_{\substack{p\in\mathcal{P}_e^R \\ D_{ep}^P=2}} \sum_{\substack{d\in \bigcup\limits_{b\in\mathcal{B}_{ep}} \mathcal{D}^b \\ d\leq|\mathcal{D}|-1}} \sum_{g\in\mathcal{G}} \sum_{g'\in\mathcal{G}} \sum_{i\in\mathcal{S}^g} \sum_{j\in\mathcal{S}^{g'}} R_{ep} M_{ep1g} M_{ep2g'} y_{e(d+1)ij} \\
& - \sum_{d\in\mathcal{D}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e^W} \pi_{dj} y_{edij} - \omega_e - \sum_{\substack{d\in\mathcal{D} \\ d\leq|\mathcal{D}|-\overline{N}}} \alpha_{ed}^W \sum_{d'=d}^{d+\overline{N}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{s\in\mathcal{S}_e^W} y_{ed'is} \\
& - \sum_{g\in\mathcal{G}} \sum_{\substack{d\in\mathcal{D} \\ d\leq|\mathcal{D}|-\overline{N}^g}} \alpha_{ed}^g \sum_{d'=d}^{d+\overline{N}^g} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{s\in\mathcal{S}_e^g} y_{ed'is} \\
& - \sum_{\substack{d\in\mathcal{D} \\ d\leq|\mathcal{D}|-D^R+1}} \gamma_{ed} \sum_{d'=d}^{d+D^R-1} \sum_{i\in\mathcal{S}_e^W} \sum_{j\in\mathcal{S}_{s_1}^R} y_{ed'ij} + \sum_{d\in\mathcal{D}} \epsilon_{ed}^{\mathrm{O}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in S^O} y_{edij} \\
& - \sum_{d\in\mathcal{D}\setminus\{1,|\mathcal{D}|\}} \sum_{j_1\in\mathcal{S}_e^W} \sum_{j_2\in\mathcal{S}_{j_1}^{S1}} \epsilon_{edj_1j_2}^{\mathrm{I}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \big(y_{e(d-1)ij_1} + y_{e(d+1)ij_2}\big) \\
& - \sum_{d\in\mathcal{D}\setminus\{1,|\mathcal{D}|-1,|\mathcal{D}|\}} \sum_{j_1\in\mathcal{S}_e^W} \sum_{j_2\in\mathcal{S}_{j_1}^{S2}} \epsilon_{edj_1j_2}^{\mathrm{II}} \sum_{i\in\mathcal{S}_e\cup\{0\}} \big(y_{e(d-1)ij_1} + y_{e(d+2)ij_2}\big) \\
& - \sum_{d\in\mathcal{D}^N} \zeta_{ed} \sum_{d'=d}^{d+N^N-1} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e^W} T_j y_{ed'ij} \\
& - \sum_{\substack{p\in\mathcal{P}_e^I \\ D_{ep}^P\geq 3}} \sum_{\substack{d\in \bigcup\limits_{b\in\mathcal{B}_{ep}} \mathcal{D}^b \\ d\leq|\mathcal{D}|-D_{ep}^P+1}} \theta_{epd}^I \sum_{d'=1}^{D_{ep}^P} \sum_{g\in\mathcal{G}} M_{epd'g} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e^g} y_{e(d+d'-1)ij} \\
& - \sum_{\substack{p\in\mathcal{P}_e^P \\ D_{ep}^P\geq 3}} \sum_{\substack{d\in \bigcup\limits_{b\in\mathcal{B}_{ep}} \mathcal{D}^b \\ d\leq|\mathcal{D}|-D_{ep}^P+1}} \theta_{epd}^P \sum_{d'=1}^{D_{ep}^P} \sum_{g\in\mathcal{G}} M_{epd'g} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e^g} y_{e(d+d'-1)ij} \\
& \sum_{\substack{p\in\mathcal{P}_e^R \\ D_{ep}^P\geq 3}} \sum_{\substack{d\in \bigcup\limits_{b\in\mathcal{B}_{ep}} \mathcal{D}^b \\ d\leq|\mathcal{D}|-D_{ep}^P+1}} \sum_{d'=1}^{D_{ep}^P} \theta_{epdd'}^R \sum_{g\in\mathcal{G}} M_{epd'g} \sum_{i\in\mathcal{S}_e\cup\{0\}} \sum_{j\in\mathcal{S}_e^g} y_{e(d+d'-1)ij}
\end{aligned}
$$

A.4.4 Graph Modification Constraints

Graph modification constraints involve removing arcs in the SPPRC network, and is done by fixing the corresponding flow variable to zero. The constraints concern rest between shifts, weekends and patterns.

Rest between shifts

$$y_{edij} = 0 \qquad d \in \mathcal{D}, i \in \mathcal{S}_e, j \in \mathcal{S}_i^I$$

Weekends

$$\begin{aligned} y_{edij} &= 0 & d &\in \mathcal{D}^{\text{SUN}}, i \in \mathcal{S}_e^W, j \in \mathcal{S}^O \\ y_{edij} &= 0 & d &\in \mathcal{D}^{\text{SUN}}, i \in \mathcal{S}^O, j \in \mathcal{S}_e^W \\ y_{edij} &= 0 & d &\in \mathcal{D}^{\text{SAT}}, \{i \in \mathcal{S}_e | T_i^E > H\}, j \in \mathcal{S}^O \end{aligned}$$

Illegal patterns

$$\sum_{g \in \mathcal{G}} \sum_{g' \in \mathcal{G}} \sum_{i \in \mathcal{S}^g} \sum_{j \in \mathcal{S}^{g'}} M_{ep1g} M_{ep2g'} y_{e(d+1)ij} = 0$$
$$p \in \mathcal{P}_e^I, D_{ep}^P = 2, d \in \bigcup_{b \in \mathcal{B}_{ep}} \mathcal{D}^b, d \leq |\mathcal{D}| - 1$$

A.4.5 Resources

Some constraints presented in Section 4.3 were formulated using resources, and these are summarized in the following. The enforcement of such constraints to ensure feasible paths were described in Section 4.2.

Resource	$T^{\underline{W}}$
REF	$f_{dij}^{\underline{W}}(T^{\underline{W}}) = \begin{cases} T^{\underline{W}} + 1 & i \in \mathcal{S}_e^W, j \in \mathcal{S}_e^W, d \in \mathcal{D} \\ 1 & i \notin \mathcal{S}_e^W, j \in \mathcal{S}_e^W, d \in \mathcal{D} \\ T^{\underline{W}} & \text{otherwise} \end{cases}$
Initial value	$T_0^{\underline{W}} = \underline{N}_e$
Resource window	$T_{ds}^{\underline{W}} \in \begin{cases} [0, \infty) & d \in \mathcal{D}, s \in \mathcal{S}_e^W \\ [\underline{N}_e, \infty) & d \in \mathcal{D}, s \in \mathcal{S}^O \end{cases}$

$$
\begin{array}{rl}
\text{Resource} & T^{\underline{W}^g}, \quad g \in \mathcal{G} \\
\text{REF} & f_{dij}^{\underline{W}^g}(T^{\underline{W}^g}) = \begin{cases} T^{\underline{W}^g} + 1 & i \in \mathcal{S}_e^g, j \in \mathcal{S}_e^g, d \in \mathcal{D} \\ 1 & i \notin \mathcal{S}_e^g, j \in \mathcal{S}_e^g, d \in \mathcal{D} \\ T^{\underline{W}^g} & \text{otherwise} \end{cases} \\
\text{Initial value} & T_0^{\underline{W}^g} = \underline{N}_e^g \\
\text{Resource window} & T_{ds}^{\underline{W}^g} \in \begin{cases} [0, \infty) & d \in \mathcal{D}, s \in \mathcal{S}_e^g \\ [\underline{N}_e^g, \infty) & d \in \mathcal{D}, s \notin \mathcal{S}_e^g \end{cases}
\end{array}
$$

$$
\begin{array}{rl}
\text{Resource} & T^{V(t)}, \quad t \in \{1, \dots, W^W\} \\
\text{REF} & f_{dij}^{V(t)}(T^{V(t)}) = \begin{cases} T^{V(t)} + 1 & \begin{array}{l} d \in \mathcal{D}^{SAT}, \\ i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e^W \end{array} \\ 0 & \begin{array}{l} w \in \mathcal{W}, \\ w \bmod W^W = t - 1, \\ d \in \mathcal{D}_w^{MON}, \\ i \in \mathcal{S}_e, j \in \mathcal{S}_e \end{array} \\ T^{V(t)} & \text{otherwise} \end{cases} \\
\text{Initial value} & T_0^{V(t)} = 0 \\
\text{Resource window} & T_{ds}^{V(t)} \in [0, W^W - \underline{N}^W] \quad d \in \mathcal{D}, s \in \mathcal{S}_e
\end{array}
$$

A.4.6 Roster Line Parameters

When solving an SP, a solution represented with y-variables is obtained. This represents a possible new column in the RMP which may be added with a new k-index. The relevant parameters for this new column is calculated as follows. Note that an SP solution for employee e with index k is denoted $\overline{\mathbf{y}}_{ek} = \{\overline{y}_{ekdij} | e \in \mathcal{E}, d \in \mathcal{D} \cup \{|\mathcal{D}| + 1\}, i \in \mathcal{S}_e \cup \{0\}, j \in \mathcal{S}_e \cup \{0\}\}$.

Cost of a roster line

$$\begin{aligned}
C^K_{ek} = &\sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}_e} C_{edj} \overline{y}_{ekdij} + \sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{S}^W_e} \sum_{j \in \mathcal{S}^R_i} C^R \overline{y}_{ekdij} \\
&+ \sum_{\substack{p \in \mathcal{P}^P_e \\ D^P_{ep}=2}} \sum_{\substack{d \in \bigcup\limits_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}|-1}} \sum_{g \in \mathcal{G}} \sum_{g' \in \mathcal{G}} \sum_{i \in \mathcal{S}^g} \sum_{j \in \mathcal{S}^{g'}} P_{ep} M_{ep1g} M_{ep2g'} \overline{y}_{ek(d+1)ij} \\
&- \sum_{\substack{p \in \mathcal{P}^R_e \\ D^P_{ep}=2}} \sum_{\substack{d \in \bigcup\limits_{b \in \mathcal{B}_{ep}} \mathcal{D}^b \\ d \leq |\mathcal{D}|-1}} \sum_{g \in \mathcal{G}} \sum_{g' \in \mathcal{G}} \sum_{i \in \mathcal{S}^g} \sum_{j \in \mathcal{S}^{g'}} R_{ep} M_{ep1g} M_{ep2g'} \overline{y}_{ek(d+1)ij}
\end{aligned}$$

Roster line attributes

$$A_{ekds} = \sum_{i \in \mathcal{S}_e \cup \{0\}} \overline{y}_{ekdis} \qquad d \in \mathcal{D}, s \in \mathcal{S}_e$$

$$A^W_{ekd} = \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^W_e} \overline{y}_{ekdij} \qquad d \in \mathcal{D}$$

$$A^O_{ekd} = \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^O} \overline{y}_{ekdij} \qquad d \in \mathcal{D}$$

$$A^g_{ekd} = \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{j \in \mathcal{S}^g_e} \overline{y}_{ekdij} \qquad d \in \mathcal{D}, g \in \mathcal{G}$$

$$V_{ekd} = \sum_{i \in \mathcal{S}^W_e} \sum_{j \in \mathcal{S}^R_i} \overline{y}_{ekdij} \qquad d \in \mathcal{D} \backslash \{1\}$$

$$D_{ekd} = \sum_{d'=d}^{d+N^N-1} \sum_{i \in \mathcal{S}_e \cup \{0\}} \sum_{s \in \mathcal{S}^W_e} T_s \overline{y}_{ekd'is} \qquad d \in \mathcal{D}^N$$

Transform to x-variables
A solution with flow variables has a simple transform to the more understandable binary x-variables. A variable x_{eds} indicates if employee $e \in \mathcal{E}$ is assigned shift type $s \in \mathcal{S}_e$ on day $d \in \mathcal{D}$. The transform from is given below.

$$x_{eds} = \sum_{i \in \mathcal{S}_e} y_{edis} \qquad e \in \mathcal{E}, d \in \mathcal{D} \backslash \{1\}, s \in \mathcal{S}_e$$

$$x_{e1s} = y_{e10s} \qquad e \in \mathcal{E}, s \in \mathcal{S}_e$$

Appendix B

Limited Label Extension Algorithm

A modified labelling algorithm is presented in Algorithm B.1 that enables the limited label extension speed-up presented in Section 5.5.3. An extension limit $\bar{n}$ stating the maximum number of labels extended from each node is input to the algorithm. The algorithm assumes an acyclic directed graph.

Algorithm B.1: Labelling algorithm with limited label extension

```
1  Set extension limit $\bar{n}$.
2  Set $\mathcal{U} \leftarrow [L_0]$ and $\mathcal{P} \leftarrow \emptyset$.
3  while $\mathcal{U} \neq \emptyset$ do
4      Move the first label $L$ from $\mathcal{U}$ to a temporary list $\mathcal{U}'$.
5      Move all labels $L'$ that are in the node same node as $L$ from $\mathcal{U}$ to $\mathcal{U}'$.
6      Sort $\mathcal{U}'$ and remove all but the $\bar{n}$ labels with lowest reduced cost.
7      for $L' \in \mathcal{U}'$ do
8          Extend $L'$ to all feasible extensions $\{L''_i\}$.
9          for $L'' \in \{L''_i\}$ do
10             if No label in $\mathcal{U}$ and/or $\mathcal{P}$ dominates $L''$ then
11                 if $L''$ dominates labels in $\mathcal{U}$ or $\mathcal{P}$ then
12                     Remove dominated labels from $\mathcal{U}$ or $\mathcal{P}$.
13                 end
14                 Add $L''$ to the end of $\mathcal{U}$.
15             end
16         end
17         Add $L'$ to $\mathcal{P}$.
18     end
19 end
20 Filter $\mathcal{P}$ to find optimal solution at end node.
```

Appendix C

Cost Calculation for Test Instances

Details on the cost calculation used to determine cost coefficients in the test instances used in Chapter 6 are provided in this appendix.

Cost parameters in the objective function represent both monetary (e.g. salaries, payroll taxes and insurance) and preference costs, and their magnitudes are critical to build a practically relevant model. The monetary costs are in large based on the collective agreement, constituting salary for a large part of the Norwegian nurse staff. The agreement states a relation between seniority, skill level and annual salary for nurses at a hospital.

Costs are not specified in the INRC benchmark instances. Often in rostering, fixed salary costs are assumed since the workforce composition is usually constant over the planning period and salary costs are thus considered sunk costs. However, it is argued in this thesis that setting a specific variable cost for each hour worked by an employee makes the instances more realistic. By applying variable costs, time specific shift costs (e.g. night or day), overtime costs and similar costs are more easily calculated and understood based on a common base hourly variable cost. To ensure that the variable salary cost approach in the thesis is consistent with the fixed minimum salary cost often faced by healthcare institutions, a penalty equal to the missed hourly pay is given for every hour an employee works less than his or her contracted number of hours. This ensures that the salary costs always exceed the minimum fixed costs while also including variable costs of for instance overtime.

The hourly variable cost of having an employee at work is estimated from an annual base salary for each employee. To calculate the base salaries, the seniority of each employee is drawn randomly from a uniform distribution between three and 25 years for each instance. The choice of distribution bounds is assumed to have minimal effect on the model because it only alters the costs slightly. Three years is set as the lower bound since three years of education is common for Norwegian nurses and count as seniority. The upper bound of 25 is believed to reflect the higher end of the age distribution at a hospital.

In the collective agreement, salary table LR15 states the implied annual base salary for each employee working as a regular nurse. LR15 provides a series of alternatives for the relation between seniority and income. Alternative five was chosen arbitrarily from the salary table and has little impact on the model due to minor variation across the alternatives. According to the collective agreement, head nurses earn salary stage 50.

The annual gross salary for employee $e \in \mathcal{E}$ is denoted by S_e. An annual base salary cost is calculated for each employee by assuming 47 out of 52 weeks on duty, an additional 12 percent in vacation pay and thus a salary of $1.12 \cdot \frac{47}{52} S_e$. For the employer, an additional cost of 14.1 percent is added as payroll taxes according to Norwegian labor regulation. Also, it is assumed that a two percent add-on cost for pension contributions applies. All other monetary costs of employment, e.g. insurance, materials and training, are assumed sunk costs in relation to the HRP and are not considered further. In conclusion, the total annual salary cost of an employee is $1.02 \cdot 1.141 \cdot 1.12 \cdot \frac{47}{52} S_e \approx 1.18 S_e$. The annual salary serves as a minimum cost, excluding the compensation for e.g. overtime and reduced rest.

A base hourly cost for an employee can then be found. 365 days per year less 104 weekend days, 10 public holidays, and 25 vacation days, leaves 226 regular working days assumed for the hourly cost calculation. A standard contract amounts to 37.5 working hours per week or 7.5 hours per weekday according to Norwegian labor regulations. Multiplying the number of working days with the duration per day, the total number of working hours through one year is 1 695. Equation (C.1) displays the final formula for a base hourly employee cost C_e^H based on a gross salary S_e. Hourly costs are rounded to the nearest integer.

$$C_e^H = \left\lfloor \frac{1.18 S_e}{1\,695} \right\rceil \tag{C.1}$$

C_e^H is a variable hourly cost of having employee e at work and provides a reference for all cost calculations. How this is used to determine the specific costs, e.g. of shifts and preferences, is stated in Section 6.2.2.