# Personnel Preferences in Personnel Planning and Scheduling

Egbert van der Veen

# Personnel Preferences in Personnel Planning and Scheduling

Egbert van der Veen

**Dissertation committee**

| | |
|---|---|
| Chairman & secretary: | Prof. dr. ir. A.J. Mouthaan |
| Promoters: | Prof. dr. R.J. Boucherie<br>Prof. dr. ir. E.W. Hans |
| Assistant promoter: | Dr. B. Veltman |
| Members: | Prof. dr. J. van Hillegersberg<br>Prof. dr. J.L. Hurink<br>Prof. dr. G.G. van Merode<br>Prof. dr. G.T. Timmer<br>Dr. G. Vanden Berghe<br>Prof. dr. S. de Vries |

This research is financially supported by ORTEC, and partly by the Dutch Technology Foundation STW by means of the project 'Logistical Design for Optimal Care' (No. 08140)

Ph.D. thesis, University of Twente, Enschede, the Netherlands
Center for Telematics and Information Technology (No.13-245, ISSN 1381-3617)
Center for Healthcare Operations Improvement and Research
Publisher: Egbert van der Veen
Printed by: Ipskamp Drukkers BV, Enschede, the Netherlands
Cover design: Bart Timmer



ISBN 978-90-365-1152-0
DOI http://dx.doi.org/10.3990/1.9789036511520

ORTEC
OPTIMIZE YOUR WORLD

CTIT

CHOIR

# Personnel Preferences in Personnel Planning and Scheduling

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 22 november 2013 om 14:45 uur

door

Egbert van der Veen

geboren op 20 november 1984
te Leeuwarden

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. R.J. Boucherie
Prof. dr. ir. E.W. Hans

en de assistent-promotor:
Dr. B. Veltman

"Knowledge is of no value unless you put it into practice."

— *Heber J. Grant*

HOOFDSTUK 1

# Voorwoord

Na viereneenhalf jaar is het eindresultaat daar! Een proefschrift dat de naam draagt van één auteur, maar dat het werk is van velen. De afgelopen jaren zijn intensief en enerverend geweest. Ik heb mooie herinneringen aan de goede en inspirerende samenwerkingen die hebben bijgedragen aan de totstandkoming van dit proefschrift. Zonder de illusie te koesteren uitputtend te zijn, wil ik graag gebruik maken van de gelegenheid een aantal mensen te bedanken voor hun waardevolle bijdrage.

Bart, onze samenwerking begon tijdens mijn afstuderen. Toen jij, tijdens het afronden van mijn masterscriptie, vroeg of ik weleens gedacht had om te promoveren, was mijn eerste gedachte: "Lijkt me niets voor mij". Uiteraard heb ik me weleens afgevraagd waar ik aan begonnen ben, maar spijt heb ik niet gehad. Bedankt voor deze geweldige kans, die je mij geboden hebt. Naast je onmiskenbare bijdrage aan dit proefschrift, ben jij een baken in het commerciële bestaan dat ik naast mijn promoveren heb. Jij hebt mij daarin op weg geholpen naar professionele volwassenheid.

Erwin, zonder twijfel ben jij een van de meest enthousiaste en gedreven mensen die ik ken. Daarnaast is jouw talent om een 'commercieel sausje' over een wetenschappelijke tekst te gieten onovertroffen. Je hebt mij geleerd hoe je wetenschappelijke teksten beter verkoopt en hebt daarbij van mij een betere schrijver gemaakt. Je vertel- en moppentaptalenten kenmerken je daarnaast als iemand die vol in het leven staat.

Richard, nadat ik in het eerste jaar vooral mijn best deed mijn kweekvijver met potentiële onderzoeksonderwerpen zo vol mogelijk te krijgen, heb jij mij op het juiste moment laten focussen. Een uitspraak van jou die ik nooit zal vergeten is dat "geen keuze ook een keuze is". Jouw oog voor het constateren van incoherente

verbanden heeft mij geleerd de grote lijnen en de argumentatie in artikelen en in dit proefschrift zorgvuldiger en scherper neer te zetten.

Uiteraard een speciaal woord van dank voor de commissieleden. Sven de Vries, without knowing at the time, you were already involved with my PhD dissertation when supervising my master's thesis that later turned out to be the foundation for Chapter 8 of this dissertation. Johann Hurink, onze discussies over wiskundige probleemformuleringen hebben geholpen de modellen beter te kiezen en de keuzes gedegen te motiveren. Ik bewonder je kwaliteiten om artikelen te structureren en logisch op te bouwen, welke zeker geholpen hebben ons 'zelfrooster-onderzoek' (hoofdstuk 9 van dit proefschrift) goed op te schrijven. Ton Mouthaan, Jos van Hillegersberg, Frits van Merode, Gerrit Timmer en Greet Vanden Berghe: ik voel me vereerd dat jullie plaatsnemen in mijn promotiecommissie.

Een bijzonder woord van dank gaat uit naar mijn collega's van CHOIR van de Universiteit Twente.

Peter Vanberkel, thanks for setting an excellent example for the CHOIR PhD's that followed you. For me, you are the example of a devoted scientist.

Maartje Zonderland, ik bewonder je uitzonderlijke talent om zwakheden in de praktijk en theorie op te merken en hoe je vanuit de theorie in de vaak weerbarstige ziekenhuispraktijk verbeteringen weet te realiseren.

Nikky, een jaar na het afronden van jouw proefschrift is onze voortdurende strijd om het hardste lachsalvo op te wekken tijdens een presentatie nog steeds onbeslist. Nu nodig ik jou uit voor een volgende ronde.

Peter Hulshof, dat je ondanks je switch naar een leven als strategieconsultant, halverwege je promotie, in staat bent gebleken je promotie af te ronden, vind ik knap.

Theresia, dat we nooit samen aan een artikel gewerkt hebben is ergens vreemd, maar wat niet is kan nog komen. Onze samenwerking was bij tijden intensief, met name in het eerste jaar, bij het volgen van LNMB vakken. Ook in de afrondende fase van onze promoties was de samenwerking waardevol. De vrijdagen waren bij tijden misschien iets te gezellig, maar zullen mij zeker bijblijven.

Aleida, de gesprekken in de auto samen op weg naar Fryslân ga ik missen. Je ontegenzeggelijke talent om prijzen in de wacht te slepen vind ik bewonderenswaardig.

Maartje van de Vrugt, veel hebben wij niet samengewerkt, maar gezellig was het altijd wel op vrijdagen en congressen. Je werk laat een wiskundig talent zien, wat buitengewoon verenigd is met het zijn van een sociaal persoon.

Nardo, onze samenwerking was kort. Desondanks constateer ik bij jou een gefocusedheid die ik knap vind.

De grote hoeveelheid co-auteurs waarmee is samengewerkt, onderstrepen de

invloed van samenwerking op de totstandkoming van dit proefschrift. De personen die als co-auteur hebben bijgedragen aan één of meerdere hoofdstukken uit dit proefschrift ben ik ontzettend dankbaar. Erwin, Bart, Leo Berrevoets, Bart Berden en Windi Winasti, met jullie hulp is hoofdstuk 3 tot stand gekomen. Na een flinke dataverzamelinspanning en meerdere ritjes van en naar Nijmegen hebben we een mooi onderzoek neergezet. Richard en Jan-Kees, samen hebben we laten zien dat stochastische operations research prima gebruikt kan worden om deterministische problemen op te lossen. Sophie, uit jouw masterscriptie is hoofdstuk 5 voortgekomen. Ik dank jou, Gerhard en Tim voor het tot stand komen van dit hoofdstuk en het bijbehorende artikel. Frédérique, uit jouw masterscriptie is hoofdstuk 6 voortgekomen. In die tijd deelden wij een kamer, wat mijn proefschrift zeker ook veel goeds heeft gebracht. Ik dank jou, Erwin, Gerhard en Bart voor het tot stand komen van dit hoofdstuk en het bijbehorende artikel. Sven en Bart, dankzij jullie begeleiding tijdens het schrijven van mijn eigen masterscriptie bleek deze scriptie voldoende voedingsbodem voor zowel een mooi artikel als hoofdstuk 7 van dit proefschrift. Suzanne, het pionierswerk voor hoofdstuk 8 heb jij voor je rekening genomen. Dank hiervoor en ook voor het zijn van een gezellige kamergenoot. Uit jouw masterscriptie is met hulp van Johann en Marco uiteindelijk een mooi hoofdstuk en artikel voortgekomen.

Mijn collega's bij ORTEC dank ik voor de gezellige tijd die ik heb gehad. Hierbij bedank ik in het bijzonder mijn leidinggevenden Merlijn en Monique voor de vrijheid die jullie mij hebben gegeven om mijn commerciële werkzaamheden en promotieonderzoek op elkaar af te stemmen. Dat dit onder jullie hoede als vanzelfsprekend verliep, is niet vanzelfsprekend. Merlijn, de ontspannende avondjes pool heb ik zeer gewaardeerd; moge er nog vele volgen. Monique, direct en indirect heb jij een belangrijke rol gespeeld bij de totstandkoming van een groot deel van mijn proefschrift. Jouw kritische blik op stijl en woordkeuze is de leesbaarheid van dit proefschrift zeker ten goede gekomen.

Ten slotte, maar zeker niet ten minste, dank ik mijn vrienden, ouders, broers en grootouders voor de mentale ondersteuning. De ontspannende avonden en weekendjes hebben mij geholpen te blijven beseffen dat het leven van een promovendus niet alleen hoeft te bestaan uit het schrijven van een proefschrift. Harmen en Jan, bedankt dat jullie mij als paranimfen ter zijde staan.

Lieve Sanne. Ik kan vele redenen noemen om je te bedanken, maar het belangrijkste is dat je er gewoon altijd voor me bent.

Egbert
Gouda, oktober 2013

# Contents

CHAPTER 2

# Research Relevance and Outline

## 2.1 Introduction

In many organizations, particularly in the service industry, personnel wages account for the major part of the operational cost [185]. Efficient personnel scheduling may thus significantly reduce costs. This particularly holds for the healthcare sector, where expenditures have been rising, and are expected to rise even further due to, among others, the aging population. The aging population spurs the need for more healthcare personnel, while the relative size of the working population is decreasing, see the population pyramid of the Netherlands in Figure 2.1 and the old age dependency ratio for the coming decades in Figure 2.2. In the Netherlands, about 68% of the healthcare expenses are being spend on personnel wages [112], and total healthcare expenses are expected to grow from 12% of GDP in 2012 (i.e., 92.7 billion euros) to 19-31% in 2040 [84, 85].

In the 21st century, organizations face a more heterogeneous workforce than in the previous century, which requires careful consideration of requests and preferences of individual employees. From our own experience in the healthcare sector, a lot of effort is required to develop efficient personnel schedules. However, typically when the work schedules are provided to the personnel, staff members immediately start exchanging shifts. This indicates that individual personnel preferences are insufficiently taken into account during the personnel scheduling.

On the one hand, the highly heterogeneous workforce is a result of employee preferences to work part-time or to have fixed days off. Represented by labor unions, employees negotiate with organizations about the collective labor agreements, which necessitate that companies provide different contracts, such as full-time and part-time. Moreover, personnel nowadays is more diverse with respect to socio-demographic aspects such as age, gender, and race, which also plays an important role in employee management [52]. On the other hand, organizations try

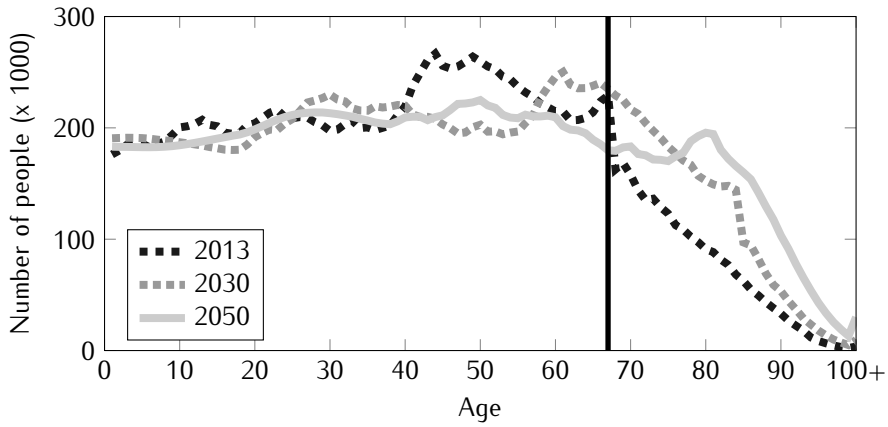



Figure 2.1: Population pyramid of the Netherlands [83]

to schedule personnel to match demand as efficiently as possible. This requires flexibility in the scheduling of personnel, particularly when demand fluctuates.

In this dissertation, we address several personnel planning and scheduling challenges that explicitly address preferences and characteristics of individual employees. We design algorithmic support for these challenges. The algorithms help to cope with the diversity between employees as well as improve cost control. The algorithms may help organizations to increase job satisfaction and profit margins.

The outline of this chapter is as follows: Sections 2.2 and 2.3 give a general introduction in the research field and research methods, respectively. Section 2.2 introduces the research topic of this dissertation, Section 2.4 describes the research environment and Section 2.5 presents an outline of the dissertation.

## 2.2 Personnel preferences in personnel planning and scheduling

We experience a gap between the need in practice and the existing theory on personnel planning and scheduling. This gap is also recognized by a literature study that analyzes the implementation success of personnel scheduling support in practice [164]. Especially regarding personnel preferences, the recent literature

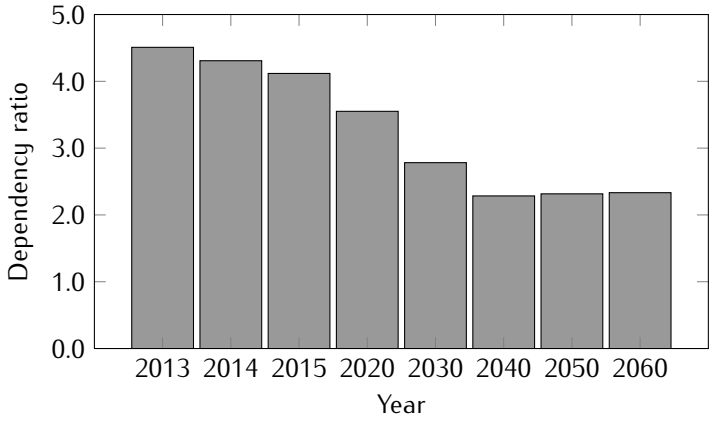



Figure 2.2: Ratio of the population aged 20-66 over the population aged 67+ [83]

review by Bergh et al. 2013 [255] states: "there are still some great opportunities in finding algorithms that efficiently cope with employee preferences".

The goal of this dissertation is to investigate, design and develop personnel planning and scheduling algorithms that in particular focus on personnel preferences on various levels of planning. Thereby, this dissertation contributes to bridging the experienced gap between practice and theory. Moreover, the algorithms proposed and designed in this dissertations are based on practical applications, and resulted in a number of practical implementations.

In order to cope with personnel preferences in personnel planning and scheduling, many human planners decompose scheduling problems into subproblems. For example, when creating work schedules planners often start by constructing weekend schedules or days off schedules since employee preferences predominantly focus on weekends and days off. Another way of coping with employee preferences is self-scheduling. In self-scheduling employees propose their own schedules, catered to their own preferences.

With respect to personnel planning, we propose a method that distributes workforce capacity on an individual level over the year. Allowing employees to work more in one week and less in another week is also known as annualized hours. The proposed method considers individual preferences on weekly working hours (for details, see Chapters 4 and Chapter 5). In addition, we propose methods that support the natural planning process by offering decomposition algorithms, such as a days off scheduling and a weekend shift scheduling algorithm (Chapter 6 and

Chapter 7). Furthermore, we investigate the potential of integrating two important personnel scheduling decisions, so-called shift scheduling and shift rostering, thereby aiming to better cope with personnel preferences and characteristics in the generated work schedules (Chapter 8). Finally, some organizations allow employees to propose their own preferred schedule, which is known as self-scheduling. We propose a method that supports the planner to create feasible work schedules from the individual work schedules proposed by the employees (Chapter 9).

## 2.3 The role of Operations Research

*Operations Research* is a discipline in applied mathematics that develops and applies quantitative techniques to support decision making in business processes. Operations Research originates from the Second World War, where military planners used it to support decision making. Since then, Operations Research analyzes real world optimization problems in various contexts, such as transportation, supply chain management, telecommunications, and personnel planning and scheduling. Typically, an operations research application starts from a real-life business challenge. This business challenge is translated into a mathematical problem and subsequently a mathematical model is designed to solve this. The mathematical solution obtained from the model is then translated into a practical answer to the business challenge.

Tailoring operations research techniques to practical applications can be a real challenge. Nevertheless, the research field of operations research is not only concerned with applications, but also involves the development of advanced mathematics. A famous open mathematical problem is the so-called $\mathcal{P} = \mathcal{NP}$ conjecture. This conjecture is one out of 7 mathematical problems that is awarded a million dollar prize by the Clay Mathematics Institute to the first person that solves it [114].

In this dissertation operations research techniques are developed for and applied to the field of personnel planning and scheduling. Operations Research for personnel planning and scheduling started in the 1950s, with Leslie Edie's "Traffic Delays at Toll Booths" in 1954 [116], and the response, "A Comment on Edie's 'Traffic Delays at Toll Booths' ", to this by George Dantzig in 1954 [106]. Since these, a significant amount of operations research literature is devoted to personnel rostering, which stems from the over 1100 references listed by various comprehensive literature reviews [71, 123, 255].

Most of the early literature (1950s-1970s) focuses on either finding feasible shift rosters under a set of (hard) constraints or minimizing the number of

employees needed to cover a given set of shifts. In other words, they focus on employers' needs. More recent literature (1980s-2000s) additionally considers employee preferences. The models in this literature try to balance or align the goals of employers and employees. Literature reviews [71, 86, 164] show that there has been a strong focus on nurse rostering problems. Nurse rostering characterizes itself by shift being scheduled 24/7 and taking into account many personnel preferences, e.g., about which weekends employees prefer not to work, and days off requests.

## 2.4 Research environment

Real-life applications are at the basis of the research in this dissertation. We address several personnel planning and scheduling applications from various organizations. The research in this dissertation is performed in collaboration between ORTEC and the research group CHOIR of the University of Twente, of which we give brief descriptions in this section. The shared interest in efficient personnel scheduling in healthcare is a basis for the collaboration between ORTEC and CHOIR, of which this dissertation is a result.

#### ORTEC - www.ortec.com

ORTEC is a Dutch software supplier and consulting firm that offers off-the-shelf software solutions as well as consulting services for a wide range of business optimization problems. ORTEC offers software and consulting services to increase service quality in supply chains, improve revenues and profit margin via yield management, and create better working conditions and higher levels of job satisfaction by better employee scheduling [200]. ORTEC is a multinational with global presence, operating in various industries; one of these is healthcare. In the Netherlands, where this research is based, ORTEC is market leader in workforce scheduling.

#### CHOIR - www.utwente.nl/choir/en/

CHOIR (Center for Healthcare Operations Improvement and Research) is a research group of the University of Twente, the Netherlands. CHOIR aims to improve healthcare operations by developing tailored operations research models for healthcare optimization problems from practice [89]. Within this research area,

CHOIR also addresses employee scheduling problems in healthcare. The author participates in CHOIR's research on this topic.

## 2.5 Outline of the dissertation

This dissertation is organized into nine chapters, starting with this introduction.

Chapter 3 provides a terminology that is used throughout this dissertation. Furthermore, it addresses the personnel preferences and characteristics considered in the literature and discusses how these preferences and characteristics are typically incorporated in scheduling algorithms.

Chapter 4 and Chapter 5 study annualized hours applications. Annualized hours allow organizations to measure working time per year, instead of per month or per week, relaxing the restriction for employees to work the same number of hours every week. Chapter 4 proposes a mathematical programming formulation that allows to flexibly model various personnel contract types, and Chapter 5 proposes a Cross-Entropy implementation to determine a cost-efficient workforce. The Cross-Entropy implementation is designed to provide high-quality solutions in short computation times, which is attractive from a user point of view.

Chapter 6 and Chapter 7 both apply a two-phase decomposition approach to personnel scheduling applications.

The decomposition approach in Chapter 6 first creates a days off schedule, indicating working days and days off for each employee. The second phase assigns shifts to the working days. Hence, this decomposition specifically addresses constraints and preferences regarding days off. The decomposition is applied to public benchmark instances.

Chapter 7 proposes a decomposition approach that first schedules weekend shifts, and secondly assigns weekday shifts. This decomposition is motivated by our experience that in many settings employees' shift preferences predominantly focus on the weekends, since many social activities happen during weekends. Since specifically scheduling weekend shifts has not been studied before, we introduce a problem specific heuristic for this. We demonstrate our decomposition on generated and real-life instances.

In Chapter 8, we discuss an approach that integrates two different personnel scheduling decisions: shift scheduling and shift rostering. With this approach personnel preferences are already considered in shift scheduling, the phase that determines when shifts should start and end. For this approach two formulations are compared and solved. Especially our Branch-and-Price formulation is designed to be flexible with regard to scheduling specific shifts for specific personnel

members.

In Chapter 9, a self-scheduling application is addressed. In self-scheduling, employees propose their own schedules to match a staffing demand specified by the employer. Since these individually composed schedules often do not perfectly match with the specified demand, a planner or manager has to adapt the schedules. We present an approach that aims to divide the burden of shift reassignments 'fair' among employees. We discuss computational results and indicate how various model parameters influence scheduling performance indicators.

Chapter 10 concludes this dissertation with discussion and outlook for future research.

CHAPTER 3

# Terminology and Literature Survey

## 3.1 Introduction

Preferences and characteristics of individual employees should be carefully considered in personnel planning and scheduling, as motivated in Chapter 2. In this chapter, we review and discuss how preferences and characteristics of individual employees are handled in the literature. In addition, since in the literature many synonyms are used for the same planning and scheduling decisions, this chapter introduces the terminology that is used throughout this dissertation.

In [255], over 300 literature references on personnel planning and scheduling from 2004 and later are categorized according to, among others, contract types, scheduling constraints and the modeling techniques being used. In this chapter, we review these 300 literature references and highlight the literature that considers characteristics of individual employees, i.e., uses constraints or objectives for which parameter values can be set for individual employees. For the literature from the period before 2004, the reader is referred to the comprehensive reviews in [71, 123].

In the literature review, we consider personnel scheduling applications in which employees are assigned to 'shifts'. Note that the terminology we introduce is not restricted to this constraint. A shift is defined as a combination of consecutive work activities and breaks on scheduled moments in time. Such personnel scheduling applications are for example found in healthcare and security services. We exclude industries for which mathematical problem formulations are much different, such as cyclical scheduling in manufacturing settings. Crew pairing and rostering, which is typical in transportation settings, and task assignment, which aims to optimally assign tasks to a set of scheduled employees, are excluded as well.

This chapter is structured as follows. Section 3.2 introduces our terminology and Section 3.3 discusses the various personnel preferences and characteristics that are considered in the literature. Section 3.4 discusses how these preferences

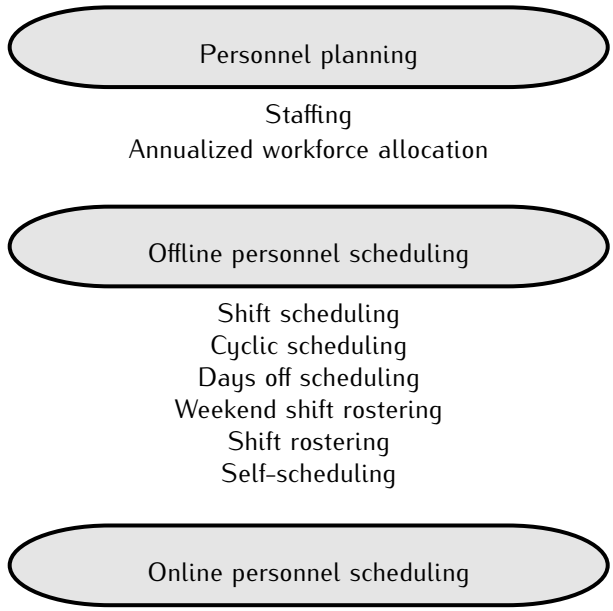



Figure 3.1: Personnel planning and scheduling terminology

and characteristics are modeled, and conclusions and discussion are presented in Section 3.5.

## 3.2 Terminology

This section proposes a terminology for personnel planning and scheduling decisions, which is used throughout this dissertation. We defined our terminology by evaluating the terminology from various personnel planning and scheduling literature reviews and categorization papers [15, 29, 53, 71, 86, 98, 109, 117, 123, 124, 139, 152, 157, 164, 215, 216, 220, 236, 239, 247, 255].

In Sections 3.2.1-3.2.3, we discuss the various personnel planning and scheduling decisions included in our terminology. In this discussion, we also address the terminology used in the literature, were we will see that in the literature, many synonyms are used for the same scheduling decision. The terminology is schematically summarized in Figure 3.1.

### 3.2.1 Personnel planning

Personnel planning concerns decision making for which workforce capacity and demand are known on an aggregated level, but where workforce capacity is still flexible to a certain extent. We divide personnel planning decisions into two categories: *Staffing* and *Capacity allocation*. Table 3.1 provides an overview of the literature synonyms for personnel planning decisions.

Table 3.1: Synonyms for personnel planning decisions used in literature reviews

| Planning level | Decision |
|---|---|
| Staffing | Manpower planning [53]<br>Staffing [71, 150, 164]<br>Total manpower requirements [247] |
| Capacity allocation | Annualized hours [15, 98]<br>Annualizing hours [98]<br>Flexiyear [15] |

**Staffing** Staffing decisions consider the composition of a workforce. Demand is given on an aggregated level of, e.g., a week or a month for a planning horizon of, e.g., a year. To match this demand, staffing considers skill-mix decisions, such as which skills should to be hired or trained employees have, and contract-mix decisions addressing hiring and firing decisions. Staffing decisions may both consider specific employees or anonymous employees.

**Annualized workforce allocation** Annualized workforce allocation considers the distribution of available workforce capacity over some time horizon of typically a year, where demand again is given on an aggregated level. An example of this is the application of annualized hours. Annualized hours, as used in labor legislation in, for example, Britain [223], France [138], Switzerland [148], and the Netherlands [256], allow organizations to measure working time per year, instead of per month or per week. This enables organizations to let employees work more hours in some periods, and less in others. In this dissertation, annualized hours applications are studied in Chapter 4 and Chapter 5. In Chapter 4, we present a literature review of both the staffing and the annualized hours literature. The staffing and annualized hours literature that considers preferences or specific characteristics of individual employees are discussed in the literature review of the current chapter as well.

### 3.2.2 Offline personnel scheduling

For offline personnel scheduling, workforce capacity and demand are given. Workforce capacity is described by a set of employees, their skills and their working hours. Demand is specified by 'staffing levels', expressing for example that "between 7:30 AM and 9:00 AM, two senior nurses should be available on the South Ward". Personnel scheduling involves creating work schedules by specifying days and time intervals during which personnel is required to work. For this, personnel scheduling considers different, but interrelated, scheduling decisions. These scheduling decisions include: *Shift scheduling*, *Days off scheduling*, *Cyclical scheduling*, *Weekend shift rostering*, *Shift rostering* and *Self-scheduling*. We discuss these scheduling decisions subsequently.

Note that the scheduling decisions, *Cyclical scheduling*, *Days off scheduling* and *Weekend shift rostering*, focus on constructing specific parts of the work schedules. As motivated in Chapter 2, focusing on creating specific parts of the work schedules supports the natural planning process. A days off scheduling algorithm and a weekend shift scheduling algorithm are proposed in Chapter 6 and Chapter 7, respectively. In Chapter 9, we propose an algorithm that supports self-scheduling.

Table 3.2 provides an overview of the literature synonyms for personnel scheduling decisions. As may be observed from Table 3.2, literature reviews actually do not use the term 'shift rostering'. However, 'rostering' is used as part of the terms 'Nurse rostering, 'Personnel rostering', 'Stint based rostering' and 'Tour rostering'. In addition, since we use 'shift scheduling' to indicate a different scheduling decision, our terminology uses 'shift rostering'.

**Shift scheduling** Shift scheduling defines shifts that should be staffed for a period of, for example, a day, a week or a month. Recall from Section 3.1 that we defined shifts as "a combination of consecutive work activities and breaks on scheduled moments in time". These shifts should respect a set of constraints and are supposed to cover given staffing levels, expressing the required number of employees in each time slot, as efficiently as possible. In addition to the required number of employees, staffing levels may also specify required skill levels. Thus, shift scheduling defines a set of shifts, which are not yet assigned to employees. Shift scheduling only defines the shifts that are required to be staffed.

**Cyclical scheduling** A cyclical work schedule establishes that shifts are performed in cyclical (rotating) patterns. A work schedule is specified for a certain planning horizon, and after this period the schedule is repeated. A cyclic schedule may be specified for either all or a subset of the employees of a department.

**Days off scheduling** Days off scheduling constructs schedules that indicate working days and days off for each employee. The specific shifts performed by employees on working days are determined at a later stage. A days off schedule should satisfy labor legislation, specifying for example the maximum number of consecutive working days. In addition, days off scheduling should ensure that sufficient employees are available to be assigned to shifts. Days off scheduling is the main topic of Chapter 6.

**Weekend shift rostering** Weekend shift rostering addresses the assignment of weekend shifts to employees. The weekday shifts are assigned to employees in a later stage. In Chapter 7, we introduce an algorithm for weekend shift rostering. Also weekend shift rostering has to comply with labor legislation specifying for example constraints on the number of consecutive working weekends.

**Shift rostering** Shift rostering is concerned with the assignment of employees to shifts. On a planning horizon of typically a couple of weeks or a month, for each day and employee it should be specified which shift the employee performs, such a schedule we refer to as a *work schedule*. Shift rostering is subject to labor legislation specifying constraints on assignment of a single shift, but also on combinations of shifts.

**Self-scheduling** With self-scheduling, employees propose the work schedule they prefer to work during a given planning horizon. Since these proposed schedules possibly do not match the shift staffing demand as specified by the organization, the planning problem is to reassign shifts in order to match the specified shift staffing demand. In Chapter 9, we propose a method that supports the planner to create feasible work schedules from the individual work schedules proposed by the employees.

### 3.2.3 Online personnel scheduling

Online personnel scheduling addresses reassignment and replacement decisions for late disturbances in workload or unexpected absences due to, for example, illnesses. Online personnel scheduling is normally performed on an ad hoc basis on a planning horizon that considers the next couple of days. Again, also online personnel scheduling has to respect labor legislation.

## 3.3 Personnel preferences and characteristics

In this section, we discuss the personnel preferences and characteristics that are considered in the set of 300 personnel planning and scheduling literature references from the comprehensive review in [255]. However, note that we exclude literature that considers 'pure' shift scheduling, since this literature does not consider characteristics of individual employees, and focuses only on determining a set a shifts that efficiently covers demand.

The personnel preferences and characteristics discussed in the literature are addressed one by one. For each of the characteristics, short descriptions and the literature examples are provided below.

**Skills** In most personnel scheduling literature, skills are considered. Skills express whether an employee is allowed to perform a specific shift. For example, the benchmark instances provided in [105], which are used for experimental studies in [62, 65, 67, 68, 172], contain skills restrictions, as well as the benchmark instances of the international nurse rostering competition issued by the PATAT 2010 conference [146], which are used for experimental studies in [19, 50, 64, 175, 197, 253].

Often, hierarchical skills are considered, which imply that a higher skilled employee is allowed to perform lower skill shifts, but not vice versa [10, 33, 136, 206, 233]. Skill restrictions are often implied as hard constraints, however skill restrictions modeled as soft constraints also occur. For example, some of the benchmark instances of the international nurse rostering competition issued by the PATAT 2010 conference [146] allow employees to be assigned to shifts for which they are not skilled. Of course, such shift assignments are not preferred. Research in [49] considers 'secondary skills'. Preferably, employees are assigned to a shift that requires their 'primary skill', but 'secondary skill assignments are allowed if required. The days off scheduling application studied in [16] restricts the set of allowed days off schedules dependent on the skills of the employees. Seniority restrictions, as discussed later on in this section, may also be seen as skill restrictions.

**Days off requests** Days off requests specify that an employee requests not to work on a specific day, or on a specific part of a day. Days off requests are mostly modeled as soft constraints. Examples are found in [203] and the benchmark instances provided in [105, 146] and the corresponding literature using these benchmark instances for experimental results.

**Shift requests** Shift requests specify that an employee requests to work (or not to work) a specific shift on a specific day. Shift requests are mostly modeled as soft constraints. Examples are found in [44] and in the benchmark instances provided in [105, 146]. In self-scheduling applications, where employees propose their preferred schedules, shift requests are inherent [18, 26, 110, 225]. In addition to proposing schedules, some literature lets employees specify 'importance' of shift requests, where 'strong' shift requests or more important to satisfy [110, 225].

**Annualized hours constraints** In annualized hours applications it is common to have constraints that are defined for individual employees, as in the annualized hours applications studied in Chapter 4 and Chapter 5 and in [101]. There, among others, individual contract hours, and minimum and maximum working hours per planning period may be defined.

**Work schedule constraints** Work schedule constraints specify allowed (hard constraints) or preferred (soft constraints) work schedule 'properties', such as shift sequence constraints and the number of shifts during the planning horizon. Many authors specify work schedule constraints per contract or employee type, and assign a contract to or designate an employee type for each employee. Almost all personnel scheduling literature considers work schedule constraints.

**Cooperation constraints** Cooperation constraints specify allowed or forbidden co-operations between employees, implying that some employees should or should not work on the same day. Examples are found in [71, 211, 243, 261].

**Prefixed assignment** A prefixed assignment is the assignment of an employee to a specific shift that must be performed on a specific day. Prefixed assignments are found in the instances of [105].

**Availability** Availability constraints specify whether employees are available or not on specific days. Employees may be unavailable due to, for example, vacation [21, 130, 132, 168, 169, 170, 251], absences [140, 192], or fixed days off [248, 268]. In [8] unavailabilities are considered, but the authors do not specify underlying reasons for the unavailability. The online personnel scheduling application in [192] aims to rebuild the work schedule when at least one employee informs that he is unable to perform one or more future shifts.

**Seniority** In many applications that consider seniority, the senior employees get a higher priority of being assigned to their preferred shifts [8, 149, 190]. In [249], constraints are implied on the minimum number of senior employees that should be available for some specific shift, which is the equivalence of a skill restriction.

**Wages** Minimizing personnel wages is considered in some literature as part of the objective, next to minimizing violations of soft constraints. Often either hourly wages and overtime wages, or both, are considered [58, 59, 96, 111, 127, 145, 263, 267].

**Hiring, firing and training cost** In personnel planning applications, hiring, firing and training cost may be considered, with the objective to minimize these under a set of constraints [127].

**Productivity** Maximizing personnel productivity is considered in some literature as part of the objective, next to minimizing violations of soft constraints. Productivity of employees may dependent on the department or location an employee is assigned to [60, 76], or on the type of employee [23, 101, 244].

For the discussed preferences and characteristics, we observe that in many papers parameter values are set for groups or 'types' of employees. Employees are grouped into sets, and for each of these sets for example skills or wages are defined, that are valid for all employees in the corresponding set.

## 3.4 Modeling

In general, mathematical models for personnel planning and scheduling problems have the objective to minimize planning costs subject to a set of hard and soft constraints. There are three types of constraints:

1. Sequence constraints, e.g., labor legislation specifying that an employee is not allowed to work more than 6 shifts a week.
2. Coverage constraints, e.g., shift X should be scheduled Y times
3. Assignment constraints, e.g., skill restrictions and unavailabilities

Constraints from these categories may be formulated as either hard or soft. Hard constraints express strict rules that the schedules must satisfy, such as labor

legislation. Soft constraints express scheduling rules that should be satisfied as much as possible. Soft constraints are often used to express employee preferences; violations lead to penalty costs.

Instead of providing an overview of the mathematical methods used by the various literature references, we discuss how these mathematical methods handle characteristics of individual employees in personnel planning and scheduling. For a complete overview of mathematical methods used in the literature the reader is referred to [255].

This section discusses mathematical methods used in personnel planning and scheduling literature and how these methods are used to model personnel preferences and characteristics. The main mathematical methods used in personnel planning and scheduling are mathematical programming and heuristics, which are discussed in Section 3.4.1 and Section 3.4.2, respectively.

### 3.4.1 Mathematical programming

Mathematical programming optimizes some objective function over a set of allowed input values. The set of allowed input values is restricted by a set of constraints. Commonly used forms of mathematical programming are *linear* programming, where the objective function and the constraints are restricted to be linear, and *(mixed) integer programming* where a (subset of) the input values is restricted to be integer. For a more elaborate linear programming introduction, see [91].

In the literature, mathematical programming is often used to model personnel scheduling problems. Common formulations that are used can roughly be divided into: explicit formulations and implicit formulations.

1. *Explicit formulations*. In these formulations, work schedules are defined *explicitly*, i.e., for the entire planning horizon a sequence of shifts and days off is specified. The objective is to select a work schedule for each employee such that the scheduling demand is covered. Employees are not allowed to be assigned to work schedules that violate any of their hard scheduling constraints. Two categories of explicit formulations are considered:

   a. Preference cost per work schedule. Literature examples: [111, 118, 142, 184, 243]. Mathematically this is formulated as follows:

      Given a set of work schedules $S$. For each $s \in S$, let $x_s$ denote the number of times work schedule $s$ is performed. For each work schedule $s \in S$, let $c_s \geq 0$ denote a cost expressing the 'burden' of performing

work schedule $s$. The objective is then to minimize $\sum_{s \in S} c_s x_s$ under a set of coverage constraints.

b. Individual preference cost per work schedule. Literature examples: [4, 6, 7, 34, 57, 59, 76, 253]. Mathematically this is formulated as follows:

Given a set of employees $I$ and a set of work schedules $S$. For each $i \in I, s \in S$, let $x_{is}$ indicate whether employee $i$ performs work schedule $s$ ($x_{is} = 1$) or not ($x_{is} = 0$). For each $i \in I, s \in S$, let $c_{is} \geq 0$ denote a cost expressing the 'burden' of employee $i$ of performing work schedule $s$. The objective is then to minimize $\sum_{i \in I} \sum_{s \in S} c_{is} x_{is}$ under a set of coverage constraints.

2. *Implicit formulations*. In these formulations, work schedules are defined *implicitly*. For each shift, it is decided whether an employee works or not. The objective is to minimize cost implied by violations of soft constraints. Hard constraints express combinations of allowed shift assignments. Two categories of implicit formulations are considered:

a. Preference cost per shift. Literature examples: [20, 126, 137, 171, 221, 240, 248, 269]. Mathematically this is formulated as follows:

Given a set of shifts $K$ and a set of employees $I$. For each $i \in I, k \in K$, $x_{ik}$ indicate whether employee $i$ works shift $k$ ($x_{ik} = 1$) or not ($x_{ik} = 0$). For each shift $k$, let $c_k \geq 0$ denote a cost expressing the 'burden' of working shift $k$. The objective is then to minimize $\sum_{i \in I} \sum_{k \in K} c_k x_{ik}$ under a set of sequence, coverage and assignment constraints.

b. Individual preference cost per shift. Literature examples: [9, 10, 72, 110, 190, 206, 224, 263, 267]. Mathematically this is formulated as follows:

Given a set of shifts $K$, a set of employees $I$ and a set of work schedules $J$. For each $i \in I, k \in K$, let $x_{ik}$ indicate whether employee $i$ works shift $k$ ($x_{ik} = 1$) or not ($x_{ik} = 0$). For each, $i \in I, k \in K$, let $c_{ik} \geq 0$ denote a cost expressing the 'dissatisfaction' of employee $i$ to work shift $k$. The objective is then to minimize $\sum_{i \in I} \sum_{k \in K} c_{ik} x_{ik}$ under a set of sequence, coverage and assignment constraints.

### 3.4.2 Heuristics

For many real-life optimization problems it is often not possible to find an optimal solution in reasonable time. A heuristic optimization method aims to find high-quality, but not guaranteed optimal, solutions in reasonable computation times.

In personnel planning and scheduling several alternative heuristic optimization methods are commonly used. In this section, we briefly discuss these methods and address how these methods consider specific personnel preferences and characteristics. An elaborate description of many heuristic optimization methods used in personnel scheduling is found in [70].

We divide the heuristic optimization methods that are used in personnel planning and scheduling in *constructive heuristics* and *meta-heuristics*. Constructive heuristics focus on a specific part of the optimization problem, whereas meta-heuristics define a more general search scheme and make none or few assumptions about the underlying optimization problem.

### Constructive heuristics

Constructive heuristics that focus on one or more of the specific preferences and characteristics are discussed here.

The heuristic proposed in [149] specifically focuses on *skills* of personnel. Personnel shift assignments are prioritized based on the commonality of commonality of the skills in the skill set of an employee. The heuristic method in [161], as well as some mathematical programming methods, focus on the assignment of *days off*. The construction algorithm used in [136] focuses at the number of allowed day and night shifts per employee.

We consider decomposition methods also as a kind of construction heuristics. Decomposition methods split the optimization problem into multiple subproblems that are solved sequentially. The order in which the subproblems are solved determine the priority being put on certain constraints or objectives of the optimization problem. For example, the decomposition method of [245] first assigns the night shifts, whereas [33] assigns days off before assigning the shift and subsequently assigning breaks and activities within the shifts. The decomposition proposed in [78] first addresses cover requirements and days off, and subsequently addresses the workload distribution.

### Meta-heuristics

Meta-heuristics define a search strategy to search the solution space. Meta-heuristics often make use of one or multiple neighborhood operators. Neighborhood operators search alternative solutions within a 'neighborhood' of the current solution. Neighborhood structures can vary from very simple to very complex. The meta-heuristic scheme defines when to use which neighborhood structure and whether the alternative solution found in the neighborhood is accepted or rejected.

First, we discuss some of the commonly used neighborhood operations. Second, we discuss meta-heuristics commonly used in personnel planning and scheduling. Whenever applicable, we outline how the literature handles specific personnel preferences and characteristics in either the neighborhood operations or in the meta-heuristic scheme.

The $k$-opt($n$) neighborhood operator swaps $k$ sequences of $n$ shifts between employees. For example, the 2-opt(1) neighborhood, as used in [92, 175, 194, 219], assigns employee A to a shift that is currently performed by employee B and vice versa. Other common examples are 1-opt(1) [67, 175, 219], which unassigns some employee from a shift and assigns it to another employee, and 2-opt($n$) that swaps a sequence of shifts between two employees. In [180] a 'day-neighborhood' is proposed that determines the optimal work schedule for that specific day, where the work schedule for the other days is considered as being fixed. In [70, 136, 180] an 'employee-neighborhood' is considered, where the optimal work schedule for a specific employee is determined, given the work schedules of the other employees.

The meta-heuristic schemes commonly used in personnel planning and scheduling are:

1. *Variable neighborhood search* (VNS), [49, 72]. VNS is based on the idea of changing neighborhoods within a local search to identify better local optima [72]. For example, iteratively changing the value of $k$ in with a $k$-opt($n$) neighborhood structure.

2. *Iterated local search* (ILS), [47, 62, 66, 68]. In each iteration of the ILS meta-heuristic, part of schedule is unassigned, after which the shifts are reassigned with the aim to generate an improved work schedule. For these reassignments, ILS often makes use of neighborhood search operators. The specific ILS implementation determines for which part of the schedule shifts are unassigned.

3. *Memetic algorithms*, [43, 211]. Memetic algorithms store information on which neighborhood operations worked well in the past to resolve violations in specific situations. To describe such situations, the literature considers for example skill-coverage and preference satisfaction [43, 211].

4. *Tabu search* (TS), [92, 266]. TS selects the neighborhood operation that gives the best objective function value in the neighborhood, excluding the current solution. This implies that tabu search might select a solution that is worse than the current solution. To prevent cycling between solutions, tabu search keeps a list of previous solutions which are declared 'tabu' and

may not be chosen in a number of iteration. TS, as applied in [266], only allows swaps between employees of the same seniority.

5. *Genetic algorithms* (GA), [130, 181, 261]. GA first generates a set of different initial schedules. From this set of schedules new, improved, schedules are generated by applying *cross-over* and *mutation* operations. Cross-over operations combine two or more schedules into a new schedule, and mutation operations apply a local change in a single schedule. A detailed description of various cross-over and mutation operations is found in [181]. In [261] a mutation operator is proposed that first unassigns all stand-alone shifts and subsequently reassign them. This is useful if stand-alone shifts are not preferred. A mutation operator that considers priorities put on vacations is considered in [251]. In [19] a mutation operator is proposed that unassigns a shift randomly, but when reassigning looks specifically at a number soft constraints, such as employee shift requests.

6. *Other*. Next to the described common meta-heuristic schemes, we mention the so-called particle swarm optimization implementation in [235]. Without going into the details of the particle swarm optimization meta-heuristic, we mention a number of interesting neighborhood operators used in [235]. In [235], neighborhood operators are used that reassign shifts both from employees that work on their preferred off days and from employees that work more than their requested working hours. Moreover, [235] proposes a neighborhood operator that reassigns shifts from employees that work on a lower skill level than their actual skill level.

A special class of heuristics we want to mention are hyper-heuristics. In contrast to meta-heuristics, hyper-heuristics do not search through a search space of solutions, but through a search space of heuristics. Hyper-heuristics are iterative methods that, in each iteration, select and apply a heuristic from a set of heuristics [24, 50, 237]. A hyper-heuristic framework that includes a heuristic that is designed to divide night and day shifts evenly is discussed in [24]. Other hyper-heuristics mainly include heuristics that randomly reassign shifts.

## 3.5 Conclusions and discussion

In this chapter, we have discussed how the literature considers preferences and characteristics of individual employees in personnel planning and scheduling decisions. First, in Section 3.2, we have introduced a terminology for personnel

planning and scheduling decisions. Next, in Section 3.3, we have provided an overview of the various personnel preferences and characteristics that are considered in the literature and finally, in Section 3.4, we have outlined how these preferences and characteristics are modeled in mathematical optimization methods.

As outlined and motivated in Chapter 2, the goal of this dissertation is to investigate, design and develop personnel planning and scheduling algorithms that in particular focus on personnel preferences. In the literature, personnel preferences are often modeled as soft constraints. The relative importance of each soft constraints in relation to the other soft constraints is set through user-defined weights. However, in practice it is often hard to properly set the weights of soft constraints.

In order to cope with personnel preferences in personnel planning and scheduling, many human planners decompose scheduling problems into subproblems. Therefore, we believe that a promising research direction for personnel planning and scheduling methods is to explicitly focus on one or some of the employee preferences. If, in practice, the weight of one soft constraint dominates another, this may be handled by decomposition techniques that focus on this soft constraint in one of the first phases of the decomposition. Two of the studies in this dissertation propose decomposition techniques (Chapter 6 and 7) that focus on a specific set of soft constraints.

An additional promising research direction is to develop scheduling algorithms that support self-scheduling. In self-scheduling, personnel preferences are of course inherently considered. In Chapter 9, we design an algorithm that supports the self-scheduling process. Next to this, we introduce preferences and characteristics of individual employees in shift scheduling by integrating shift scheduling and shift rostering in Chapter 8.

Next to the mentioned personnel *scheduling* applications, we also consider a personnel *planning* application, since we believe it is important to carefully consider personnel preferences and characteristics in personnel planning as well. In Chapter 4 and 5, we study annualized hours applications in which, especially in Chapter 4, we introduce a high degree of flexibility with regard to modeling specific individual contract types.

Table 3.2: Synonyms for personnel scheduling decisions used in literature reviews

| Planning decision | Literature terminology |
|---|---|
| Shift scheduling | Crew pairing [220]<br>Duty generation [124]<br>Shift scheduling [15, 29, 124, 139]<br>Staff shift scheduling [152]<br>Temporal manpower requirements [247]<br>Time-of-day scheduling [15] |
| Cyclical scheduling | Cyclical scheduling [29, 71, 86, 139, 199, 220, 236]<br>Fixed scheduling [71]<br>Cyclical rostering [260] |
| Days off scheduling | Day-off scheduling [29]<br>Days off scheduling [124, 139, 255]<br>Day-of-week scheduling [255]<br>Days-of-week scheduling [15] |
| Shift rostering | Crew sing problem [220]<br>Employee scheduling [71]<br>Hospital nurse scheduling [157]<br>Hospital personnel scheduling [71]<br>Line of work construction [124]<br>Nurse rostering [71, 86, 109, 164]<br>Nurse scheduling [71, 86, 124, 164]<br>Personnel rostering [109]<br>Personnel scheduling [71, 109, 255]<br>Personnel shift allocation [239]<br>Shift assignment [152]<br>Shift scheduling [124, 247, 255]<br>Staff assignment / Roster assignment [124]<br>Assigning staff to line of work [124]<br>Stint based rostering [124]<br>Tour rostering [86]<br>Tour scheduling [15, 139, 255]<br>Workforce staffing / Workforce scheduling [236] |
| Weekend shift rostering | (None) |
| Self-scheduling | Self-rostering / Self-scheduling [71]<br>Interactive scheduling [71] |
| Online personnel scheduling | Disruption management / Staff rescheduling [152] |

CHAPTER 4

# Cost-Efficient Staffing under Annualized Hours

## 4.1 Introduction

In many industries, demand for skilled workers varies throughout the year, for example due to seasonal influences. In addition, workforce capacity varies due to, for instance, vacations, illnesses, and other scheduled and unscheduled unavailability. A good contract-mix and skill-mix, and flexibility within employee contracts such as the annualized hours regime, enable organizations to efficiently match workforce demand and availability. As stressed in Chapter 2, such an efficient matching of workforce demand and supply is especially important to labor-intensive industries such as healthcare and professional customer services.

To match workforce demand efficiently, this chapter integrates a capacity allocation and a staffing problem, see Chapter 3 for definitions of these problems. The staffing problem studied in this chapter, aims to select from a given set of candidates, with their individual skills and contracts, the 'optimal' subset of employees to cover the workforce demand. Here, we define optimal as cost-efficient. Employee costs depend on their contract type. Contract types we consider in this chapter are full-time, part-time, min-max contracts, and subcontractors. This staffing problem is integrated with an annualized hours application. We refer to this as the staffing under annualized hours problem. Although workforce demand is uncertain to some extent, we consider a deterministic variant, and assume that operational demand deviations can be captured by letting employees work extra or hiring subcontractors.

The contribution of this research is threefold. First, we develop a model that integrates two personnel planning problems, as discussed in the previous paragraph. Second, with our model, several practical issues can be addressed, such as vacation planning, skill-mix decisions, and hiring and firing policies. Third, we

apply the model to a case study of the Emergency Department of the University Hospital St. Radboud Nijmegen in the Netherlands, and illustrate for some of the business questions how the model addresses them. For this case study, applying annualized hours yield a possible annual savings of 5.2% or €86000 on personnel cost within this single department of the hospital. This case study, where the annualized hours regime was not applied prior to the start of this study, also motivated this research.

This chapter is structured as follows. Section 4.2 discusses the related literature. In Section 4.3 we give a formal problem description, and in Section 4.4 we present a mathematical programming formulation of this problem. The mathematical program turns out to be very flexible with regard to contract types, which we discuss in Section 4.4.2. In Section 4.5 we address a number of business questions that can be answered using our model, and Section 4.6 discusses the application of our model to the case study. Conclusions are found in Section 4.7.

## 4.2 Literature review

This section first discusses the staffing literature, and, after that, the annualized hours literature is discussed.

The classical staffing problem as in [212] determines the number of employees needed to cover a given workload, where employees are considered equal, whereas we consider employees with different contract hours. In [51, 88, 258] staffing problems in production planning settings are studied, which focus mainly on profits that are induced by production capacities and demand. In [173] a staffing problem is studied in which employees can be hired and fired per period. Demand and employee working hours are given per period. In addition, shortages are allowed, but lead to a penalty. The decisions in [173] are mainly about when to hire and fire employees, without considering annualized hours. In [122] a staffing problem is solved where demands are expressed in shifts per period, under constraints on both the number and sequences of shifts employees can work, but without considering annualized hours.

In the annualized hours literature, the workforce demand is often specified in hours of work that need to be staffed during some planning period, which also holds for the annualized hours application studied in this chapter. However, some authors specify demand in shifts that need to be staffed [21, 22, 158, 159]. In [21, 22] mathematical programming approaches are proposed to solve an annualized hours problem, with demands expressed in the number of shifts, and constraints on the number and sequences of shifts employees can work. In [158, 159] even work

schedules for a one year are created explicitly.

Most annualized hours models in the literature consider a deterministic demand. However, [176] considers a stochastic demand, and optimizes over multiple demand scenarios, each having a probability of occurrence. In this chapter demand is considered to be deterministic. Furthermore, we discretize workforce demand into skills, which is also done in [95, 97, 102, 138, 178].

In most annualized hours literature, the planning horizon of one year is discretized into planning periods of weeks or days, and only a single employee contract type is considered. Multiple contract types are considered in [148, 176, 178], who distinguish full-time and part-time employees, i.e., they consider employees with different contract hours. In addition to full-time and part-time contracts, [96, 97, 99, 100, 102] also consider subcontractors. In this chapter, we consider full-time contracts, part-time contracts, min-max contracts and subcontractors. In addition, we consider hiring and firing of employees, which is also done in [148]. A classification scheme for annualized hours problems ifs proposed in [98].

Next to annualized hours, the literature considers the related concept of Working Time Accounts (WTAs), see [94, 101, 177, 179, 206, 207]. A WTA holds a balance of the cumulative difference between an employee's contract hours and the hours worked. The objective is then to find an assignment where the WTA stays between specified boundaries. In Section 4.4, we outline that WTAs can also be incorporated in our model.

The most used solution method in the literature is also used in this research: mathematical programming. In [158, 159] special purpose algorithms are developed for their variants of the annualized hours problem. In Chapter 5, we propose a stochastic optimization method, known as Cross-Entropy optimization.

The contribution of this research is that it considers modeling annualized hours in combination with multi-skill and multiple contract staffing problem, while minimizing salary cost.

## 4.3 Problem description

The objective of the problem studied in this chapter is to select the least cost-expensive subset of employees that stays within the bounds implied by annualized hours, and covers demand. Demand for work is given in terms of skills and numbers of hours of work required per skill and time slot. Employees can only perform work for which they are sufficiently skilled. The cost of an employee is represented by his salary. A salary is specified in an employee's contract, which also specifies

skills and working hours. We assume a salary has a fixed and a variable part. The fixed part is paid for the number of working hours specified in the contract, and a variable part is paid for additional hours. The salary of full-time employees normally only has a fixed part, whereas subcontractors have a variable fee per hour. Section 4.4.2 describes the various contract types we consider.

The planning horizon of one year is discretized into time slots of, e.g., a week or a day. We have constraints with respect to minimum and maximum working time, for every time slot and for the complete planning horizon. In addition, we specify sub-horizons of, for example, 4 or 13 weeks, for which we also imply constraints with respect to minimum and maximum working time. These constraints are employee specific and are determined by an employee's contract type.

We emphasize that the problem studied here is a tactical and not an operational problem. The solution to the problem studied in this chapter is the number of hours that employees should work per skill and per time slot. We do not aim to construct actual shift rosters on a weekly (or daily) basis.

## 4.4 Modeling

This section discusses our modeling of the annualized hours problem. We model our problem as a Mixed-Integer Linear Program (MILP). Section 4.4.1 discusses the MILP, and motivates why MILP is used as modeling technique. Section 4.4.2 discusses how various employee contracts are modeled in the proposed MILP, and Section 4.4.3 discusses possible model extensions.

### 4.4.1 Mathematical programming

The problem studied in this chapter, as discussed in Section 4.3, can be seen as a deterministic assignment problem with capacity constraints. The solution to this problem specifies which part of the total demand is covered by which employees (assignment), while complying with capacity constraints on the employee working hours. To solve this problem exactly we propose an MILP.

The notation used in this chapter is defined as follows:

**Sets, parameters:**

| | |
|---|---|
| $I$ | set of employees, indexed by $i$ |
| $J$ | set of skills, indexed by $j$ |
| $J_i \subseteq J$ | set of skills of employee $i$ |
| $T$ | set of time slots, indexed by $t$ |
| $V$ | set of subsets of $T$, indexed by $v$ |
| $T_v \subseteq T$ | subset $v$ of time slots for which working hour constraints have to be enforced |
| $d_{jt}$ | demand for skill $j$ in time slot $t$ (in hours) |
| $c_i^{\text{fix}}$ | fixed cost of employee $i$ |
| $c_i^{\text{var}}$ | variable cost of employee $i$ |
| $l_{it}$ , $u_{it}$ | minimum, maximum working hours of employee $i$ in time slot $t$ |
| $l_i^v$ , $u_i^v$ | minimum, maximum total working hours of employee $i$ in the time slots of $T_v$ |
| $l_i$ , $u_i$ | minimum, maximum total working hours of employee $i$ during the entire planning horizon $T$ |

**Variables:**

| | |
|---|---|
| $x_{ijt}$ | number of hours employee $i$ works on skill $j$ during time slot $t$ |
| $x_{it}$ | number of hours employee $i$ works during time slot $t$, i.e., $x_{it} = \sum_{j \in J} x_{ijt}$ |
| $y_i$ | 1 if employee $i$ is selected in the workforce, 0 if not |
| $TC$ | total cost of all employees |
| $TC^{\text{fix}}$ | sum of fixed cost of all employees |
| $TC^{\text{var}}$ | sum of variable cost of all employees |

The fixed cost, $c_i^{\text{fix}}$, is incurred if the employee is part of the workforce. Variable costs are incurred if the employee works more than the minimum hours that he should work, $l_i$, at a rate of $c_i^{\text{var}}$ for every excess hour. Note that, using $l_i$ here, and not $l_{it}$, is the basis of the annualized hours regime.

The objective function is:

$$\min TC = TC^{\text{fix}} + TC^{\text{var}}, \tag{4.4.1a}$$

First, we make sure demand is met in every time slot, and for every skill category:

$$\sum_{i \in I} x_{ijt} \geq d_{jt} \quad j \in J, t \in T, \tag{4.4.1b}$$

and that employees are only assigned to work they are skilled for:

$$x_{ijt} = 0 \quad i \in I, j \notin J_i, t \in T. \tag{4.4.1c}$$

Next, we define the auxiliary variable $x_{it}$ as follows:

$$x_{it} = \sum_{j \in J} x_{ijt} \quad i \in I, t \in T. \tag{4.4.1d}$$

The next constraint ensures that in every time slot, the working hours of every employee are between the lower and the upper bound:

$$l_{it} y_i \leq x_{it} \leq u_{it} y_i \quad i \in I, t \in T. \tag{4.4.1e}$$

We multiply $l_{it}$ with $y_i$, since the bounds only hold if employee $i$ is part of the workforce. Furthermore, by multiplying $u_{it}$ with $y_i$ we enforce $y_i = 0 \Rightarrow x_{it} = 0$ ($t \in T$).

For every $v \in V$, we have similar lower and upper bounds on the total working time:

$$l_i^v y_i \leq \sum_{t \in T_v} x_{it} \leq u_i^v y_i \quad i \in I, v \in V. \tag{4.4.1f}$$

Constraint (4.4.1f) allows us to model for example constraints on the minimum and maximum working time in 4 or 13 week periods.

The next constraint ensures that for every employee the working hours are between the lower and upper bound of the planning horizon:

$$l_i y_i \leq \sum_{t \in T} x_{it} \leq u_i y_i \quad i \in I. \tag{4.4.1g}$$

The fixed cost are defined as:

$$TC^{\text{fix}} = \sum_{i \in I} c_i^{\text{fix}} y_i, \tag{4.4.1h}$$

and, the variable cost are defined as:

$$TC^{\text{var}} = \sum_{i \in I} c_i^{\text{var}} \left( \left( \sum_{t \in T} x_{it} \right) - l_i y_i \right). \tag{4.4.1i}$$

Note that $TC^{\text{var}}$ is non-negative, due to Constraint (4.4.1g).

The MILP is completed with the following definition constraints:

$$x_{ijt} \geq 0 \quad i \in I, j \in J, t \in T, \tag{4.4.1j}$$

and:

$$y_i \in \{0, 1\} \quad i \in I. \tag{4.4.1k}$$

Now, the MILP used in this chapter is given by (4.4.1).

If we want to ensure that some subset of employees is part of the workforce, e.g., because these are employees with long-term contracts who cannot easily be fired, this can easily be added to our model. Define $I^{\text{employ}}$ as the set of employees that must be part of the workforce, and add the following constraint to the model:

$$y_i = 1 \quad i \in I^{\text{employ}}. \tag{4.4.2}$$

Note that Working Time Accounts (WTA), discussed in Section 4.2, can be modeled in (4.4.1) using appropriate values for $V$, $l_i^v$, and $u_i^v$.

### 4.4.2 Modeling employee contracts

The MILP model (4.4.1) allows to model various contract types. This section illustrates how to model various contract types, by adjusting model parameters. Section 4.4.2 considers full-time and part-time contracts, Section 4.4.2 considers min-max contracts, and Section 4.4.2 considers subcontractors. Section 4.4.2 describes how the model can be adapted so that it determines the optimal part-time factor for employees not part of $I^{\text{employ}}$.

#### Full-time and part-time employees

We assume that full-time and part-time employees either work in all time slots or not at all. These employees are paid a fixed employment cost $c_i^{\text{fix}}$ for a fixed number of hours per year. Hence, the variable hourly cost $c_i^{\text{var}}$ equals 0. To ensure that full-time and part-time employees work exactly their annual contract hours, we set

$$l_i = u_i = \sum_{t \in T} (h_{it} - a_{it}), \tag{4.4.3}$$

where $h_{it}$ denotes the employee's contract hours in time slot $t$, and $a_{it}$ absences, which are implied by, e.g., vacations, education, and predicted illnesses. Note that $h_{it}$ generally is the same in each time slot $t$.

#### Min-max employees

A common contract type in Dutch healthcare, is a so-called min-max contract. A min-max contract specifies per time slot the min(imum) number of hours employees are paid, and thus should work. In addition, it specifies a max(imum) number of

hours the employee may be called to work. The additional hours are paid at an hourly rate of $c_i^{\text{var}}$. In our model, $l_{it}$ and $u_{it}$ equal the min and max hours, respectively. Furthermore, $l_i = \sum_{t \in T} l_{it}$, which implies that a variable cost $c_i^{\text{var}}$ is incurred for every hour worked more than $l_i$.

#### Subcontractors

Subcontractors are considered to only have variable cost. Hours worked by subcontractor $i$ are paid at an hourly rate of $c_i^{\text{var}}$. In practice, subcontractors can either be internal, as part of a flexpool department, or external. In general, $c_i^{\text{var}}$ is higher for external subcontractors, and subcontractors are more expensive than full-time, part-time, and min-max employees.

#### Non-fixed part-time factor

For a full-time or part-time employee $i$ that is not part of $I^{\text{employ}}$ we can adapt the model to determine the optimal part-time factor. To this end, we use the same lower and upper bounds $l_{it}$ and $u_{it}$ as for a full-time employee, and we relax the integrality constraint on $y_i$, i.e., we let $y_i \in [0, 1]$. Then we let the model determine the optimal part-time factor, $y_i$, for this employee. To prohibit that $y_i$ attains an undesirable low value, i.e., smaller than some given fraction $p_i$, we add some constraints where we let the binary variable $z_i$ indicate whether employee $i$ is employed ($z_i = 1$) or not ($z_i = 0$):

$$z_i \geq y_i \tag{4.4.4a}$$
$$y_i \geq p_i z_i \tag{4.4.4b}$$
$$z_i \in \{0, 1\}. \tag{4.4.4c}$$

### 4.4.3 Model extensions

Model (4.4.1) aims to find a cost-efficient workforce that covers demand with working hours $x_{it}$ in time slot $t$ between $l_{it}$ and $u_{it}$ for each employee $i$. For an employee $i$ with $l_{it} = 35$ and $u_{it} = 45$, working hours may alternate between 35 and 45, whereas a solution where employee $i$ works 40 hours every week might also offer a feasible, and a more preferred, solution. The following cost component may be included in objective function (4.4.1a) to enforce this:

$$\lambda_1 \cdot \sum_{i \in I} \sum_{t \in T} (x_{it} - l_{it})^2 , \tag{4.4.5}$$

where $\lambda_1$ is a weight that determines the importance of this component. Note that adding this component makes model (4.4.1) quadratic instead of linear.

The following component may be used to steer on skill preferences of employees, where $c_{ij}$ denotes an artificial cost of employee $i$ working on skill $j$:

$$\lambda_2 \cdot \sum_{i \in I} \sum_{j \in J} \sum_{t \in T} c_{ij} x_{ijt}, \tag{4.4.6}$$

where $\lambda_2$ is a weight. Since full-time and part-time employees are hired for the entire year, this may imply overcapacity on an annual level. The following component may be used to distribute this overcapacity evenly over the year:

$$\lambda_3 \cdot \sum_{t \in T} \left( \sum_{i \in I} x_{it} - \sum_{j \in J} d_{jt} \right)^2, \tag{4.4.7}$$

where $\lambda_3$ is a weight.

If we want to prohibit that the cost components discussed in this section influence the employee selection in model (4.4.1), we first run model (4.4.1) without these cost components, add the selected employees $i$ to $I^{\text{employ}}$, and then rerun the model.

## 4.5 Business questions

Our model can be used to address various business questions. It can be used to answer questions regarding contract-mix and skill-mix planning, vacation and education planning, efficiently using annualized hours, subcontracting, and it can provide rostering support. We address these in the following paragraphs. In Section 4.6 we present a case study that addresses some of these business questions.

**Contract-mix and skill-mix planning** The model can be used to optimize the contract mix and skill mix of the workforce. To this end, one can experiment with the composition of the set $I$. The experiments help to address questions such as: by the end of the year two employees retire, what to do? Replace them by employees with the same contracts and skills, replace them by other employees, or do not replace them at all?

**Vacation and education planning** Many employees prefer to take their vacation during the summer season. The model can support planners to determine whether, given a vacation plan, there are sufficient employees to cover the (predicted) workload during vacation periods. In addition, the model supports education planning, and determines whether it is feasible to schedule a training for some employees in the same week.

**Annualized hours** One of the main goals of the model is to return a feasible annualized hours plan. We discuss this in detail in our case study in Section 4.6.

**Subcontracting vs. additional hiring** Covering the complete workload with contracted employees is not necessarily the most cost-efficient. Hiring subcontractors for peaks in the workload, or at times when many employees are unavailable, e.g., due to vacations, might be cheaper. We discuss this in more detail in the case study in Section 4.6.

**Rostering support** Using annualized hours implies 'saving' hours in weeks when employee availability is high, and 'spending' hours when the opposite holds. The model outcome provides planners information in which weeks to save hours and in which weeks to spend extra hours. This information can also be used to set budgets on the number of hours employees may be scheduled during a scheduling period. In addition, this information can be used to respond adequately to unexpected employee absences such as illnesses. Suppose an employee calls in sick during the morning. It is then not necessarily cost-efficient to call in a contracted employee. Assigning a contracted employee to work extra in April might imply that he cannot work extra during summer. When subcontracting is cheaper in April than in the summer, and more personnel is needed during the summer, it may be more cost-efficient to subcontract in April and schedule the contracted employees in the summer. Our model supports such analyses.

## 4.6 Case study

As mentioned in the introduction, annualized hours and other forms of workforce flexibility may contribute significantly to cost reductions in service industries such as healthcare. The Emergency Department of the University Hospital St. Radboud Nijmegen, the Netherlands, was interested in exploring the impact of introducing annualized hours in their personnel planning. Thus, we applied our model to a

case study of the Emergency Department. We created a decision support system in an MS Excel workbook, where the model's parameter values can be specified. The MILP model (4.4.1) is modeled in Aimms, which imports data from Excel. Aimms uses Cplex 12.2 to solve the MILP.

In Section 4.6.1 we describe the data and the experimental setup, and Section 4.6.2 presents the experimental results.

### 4.6.1 Data description and experimental setup

The department has 32 employees with a mixture of full-time and part-time contracts. In addition, the department uses subcontractors. A full-time employee has an annual salary of €60000, and the hourly tariff of subcontractors is approximately 1.7 times the hourly salary of contracted employees. The planning horizon of one year is discretized into 52 one-week time slots. Nine 8-hour and three 9-hour single-skilled shifts need to be staffed every day. Hence, 99 hours need to be staffed in a day, 693 hours in a week, and 36036 hours in 52 weeks (a year). For every employee we know his contract hours, i.e., the number of hours the employee is expected to work during a week, which we denote by $h_{it}$. Note that for most employees $h_{it}$ is the same in every time slot $t$. Due to absenteeism, e.g., vacations, illness, and staff meetings, the hours an employee is available for work is less than his contract hours. We refer to available hours as the net availability of an employee. The net availability of employee $i$, in time slot $t$, is defined as the difference between his contract hours $h_{it}$ and his absences $a_{it}$. We performed an extensive analysis to determine the expected net availability of the employees. The combined gross and the combined net availability of all employees are shown per week as the upper dashed line and the solid line in Figure 4.1, respectively. The lower dashed line represents the demand of 693 hours per week.

From Figure 4.1 we observe that in 13 of the 52 weeks demand is larger than the net availability of the employees. Hence, the net availability has a major influence on the workforce's ability to cover demand. Figure 4.1 illustrates that the workforce's ability to cover demand in this case study is greatly influenced by fluctuations in workforce availability. Using graphs such as the one in Figure 4.1 helped the hospital to better understand situations of overstaffing and understaffing.

When applying annualized hours, as discussed in Section 4.1, one is allowed to deviate from the contracted (weekly) hours, as long as on an annual level an employee does not work more than his annual contract hours. To study the effect of annualized hours, we introduce $\rho_i$, which denotes the 'annualized hours

percentage', and we let $l_{it}$ and $u_{it}$:

$$l_{it} = (1 - \rho_i)(h_{it} - a_{it}), \tag{4.6.1}$$

$$u_{it} = (1 + \rho_i)(h_{it} - a_{it}). \tag{4.6.2}$$

Hence, $[1-\rho_i, 1+\rho_i]$ specifies the bandwidth around the employee's net available hours $(h_{it} - a_{it})$ within which we allow deviations from the net available hours.

Figure 4.1 shows the bandwidths around the net available hours of the employees for $\rho_i \equiv 0.1$ and $\rho_i \equiv 0.2$ as the shaded area and the crosshatched area, respectively.

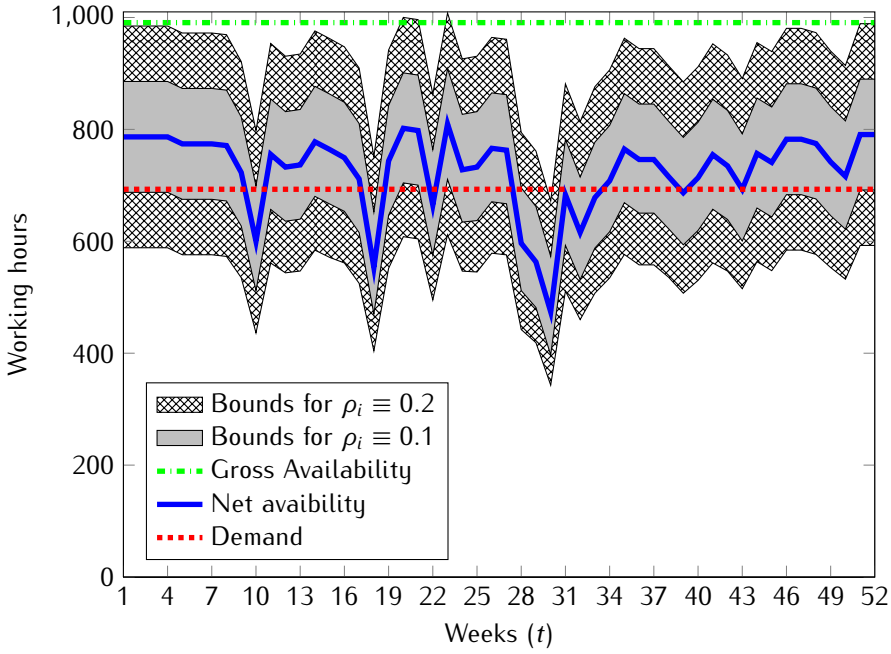



Figure 4.1: Demand, availability and annualized hours bounds for $\rho_i \equiv 0.1$ and $\rho_i \equiv 0.2$

From Figure 4.1, we observe that even for $\rho_i = 0.20$ the net available hours of the employees together cannot cover all workforce demand, since in week 18 there is still a difference of 18 hours between the workforce demand and the upper bound of the employees' net availability.

In order to analyze the effects of the salaries of full-time and part-time employees versus subcontractor tariffs, we let the salaries of full-time and part-time employees equal their annual contract hours. Thus, we let an employee have a unit 'cost' per contracted hour. Hence, for an employee with 36 contract hours per week, we set an annual 'salary' of 1872 ($= 52 \cdot 36$). For a full-time employee we thus set $c_i^{\text{fix}} = 1872$, and $c_i^{\text{var}} = 0$. However, since $l_i = u_i$, see Section 4.4.2, we never incur variable cost for full-time and part-time employees. Note that, due to absences, the actual cost per productive hour of full-time and part-time employees is larger than 1. Table 4.1 presents an overview of the full-time and part-time contracts, the contract hours, the 'salaries' and the number of employees that have a certain contract.

Table 4.1: Overview of employee contract hours and salaries

| Contract hrs | Employees | 'Salary' |
|---|---|---|
| 36.00 | 9 | 1872.00 |
| 32.00 | 10 | 1664.00 |
| 34.11 | 2 | 1773.72 |
| 30.32 | 4 | 1576.64 |
| 28.80 | 1 | 1497.60 |
| 26.53 | 1 | 1379.56 |
| 25.36 | 1 | 1318.72 |
| 24.63 | 1 | 1280.76 |
| 24.00 | 1 | 1248.00 |
| 16.00 | 2 | 832.00 |

In Table 4.1 we see, e.g., that 10 employees have a contract for 32 hours and a 'salary' of 1664. In total there are 32 employees, and together they have a 'salary' of 51523.

The main goal of the case study is to gain insight in the effects of annualized hours. Therefore, we vary $\rho_i$ between 0.00 and 0.25, and analyze the effect on the selected workforce. Furthermore, we set $I^{\text{employ}} = \emptyset$; so we let the model choose whom to employ. In addition, we analyze the effect the subcontractor cost has on the optimal solution. We vary the variable cost of subcontractors between 1 and 5, for every value of $\rho_i$. Moreover, we study the effects of $\rho_i$ in the case we do not have subcontractors.

### 4.6.2 Experimental results

This section presents the experimental results. We do not discuss computation time in detail, since the maximum observed computation time is 10 seconds, which is negligible for this kind of analysis and decision making.

Table 4.2 presents results on the cost of the optimal solutions. Note that we divided the total cost ($TC$) of the optimal solution by 36036, which is the cost when all demand is covered by subcontractors with $c_i^{\text{var}} = 1$. Note that, if $c_i^{\text{var}} = 1$ the optimal solution is to only hire subcontractors, since, as noted before, the full-time and part-time employees in fact cost more than 1 per productive hour due to their absences. By dividing the total cost ($TC$) by 36036, it is easier to compare the various solutions.

Table 4.2: Total cost ($TC$) divided by 36036

| | $c_i^{\text{var}}$ subcontractors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\rho_i$ | 1 | 1.5 | 1.7 | 2 | 2.5 | 3 | 4 | 5 | $\infty$ |
| 0% | 1.00 | 1.33 | 1.35 | 1.37 | 1.40 | 1.43 | 1.48 | 1.51 | $\infty$ |
| 5% | 1.00 | 1.33 | 1.34 | 1.35 | 1.37 | 1.38 | 1.40 | 1.42 | $\infty$ |
| 10% | 1.00 | 1.33 | 1.34 | 1.35 | 1.36 | 1.36 | 1.37 | 1.38 | $\infty$ |
| 15% | 1.00 | 1.33 | 1.34 | 1.34 | 1.35 | 1.35 | 1.35 | 1.36 | $\infty$ |
| 20% | 1.00 | 1.33 | 1.34 | 1.34 | 1.34 | 1.34 | 1.35 | 1.35 | $\infty$ |
| 25% | 1.00 | 1.33 | 1.34 | 1.34 | 1.34 | 1.34 | 1.34 | 1.34 | 1.35 |

As mentioned, the current workforce has a 'salary' of 51523, which divided by 36036 equals 1.43. From Table 4.2, we observe that in most situations the model's solution is cheaper than employing the current workforce. If subcontractors are not used ($c_i^{\text{var}} = \infty$), the current workforce is only able to cover the total demand if $\rho_i = 0.25$, since this is the only situation where $TC < \infty$. Moreover, without any subcontracting, annualized hours yields a possible savings of 5.2% or €86000. Furthermore, although for a subcontractor tariff of 1.7 the effect is minor, we observe that the optimal solution cost decreases for increasing $\rho_i$. This is to be expected, since a larger $\rho_i$ implies larger flexibility for the full-time and part-time employees to cover the workload, and hence the dependence on (expensive) subcontractors decreases. We also observe this when consider the total subcontractor cost in the optimal solutions, which is shown in Table 4.3.

Additionally, from Tables 4.2 and 4.3, we observe that for small $\rho_i$, the dependence of the organization on the cost of subcontracting is larger. For example, if subcontracting is three times as expensive as regular employees ($c_i^{\text{var}} = 3$), the

Table 4.3: Total subcontractor cost for varying $\rho_i$ and subcontractor cost

| | $c_i^{\text{var}}$ subcontractors | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\rho_i$ | 1 | 1.5 | 1.7 | 2 | 2.5 | 3 | 4 | 5 | $\infty$ |
| 0% | 36036 | 8273 | 5473 | 4636 | 5795 | 4990 | 5964 | 5979 | $\infty$ |
| 5% | 36036 | 8273 | 1871 | 2202 | 2752 | 2237 | 2982 | 3489 | $\infty$ |
| 10% | 36036 | 8273 | 1857 | 2185 | 887 | 1064 | 1419 | 1465 | $\infty$ |
| 15% | 36036 | 8273 | 1857 | 578 | 722 | 397 | 529 | 426 | $\infty$ |
| 20% | 36036 | 8273 | 1857 | 119 | 149 | 179 | 239 | 299 | $\infty$ |
| 25% | 36036 | 8273 | 64 | 75 | 94 | 113 | 44 | 55 | 0 |

total workforce cost is 7% larger for $\rho_i = 0\%$ compared to the siutation where $\rho_i = 25\%$. If $c_i^{\text{var}} = 5$, this increases to 13%, and if we do not allow subcontracting ($c_i^{\text{var}} = \infty$) then there is not even a feasible solution in which all demand is covered for $\rho_i < 25\%$. Hence, if the workforce is less flexible, the workforce cost is more sensitive to changes in subcontractor costs, which is an external factor that is often not influencable by the organization. Thus, using annualized hours reduces financial risk. Moreover, the results show that it is not always cost-efficient to cover all demand with contracted employees, since for most test instances subcontractors cover part of the workforce demand.

## 4.7 Conclusions

In this chapter, we studied a staffing problem that explicitly considers the annualized hours regime. We modeled this as a mathematical program. Our model includes various contract types, such as full-time, part-time, min-max, and subcontractors, which are modeled in a generic and flexible way: the model only has a single, generic contract type, and contract types can be included in the problem instances by setting the correct model parameter values. The objective of our model is to cost-efficiently match workforce capacity to demand by exploiting flexibility in employee contracts and by using annualized hours.

We discussed how annualized hours are incorporated in our model. Furthermore, we discussed how the model can be used to address various business questions regarding contract-mix and skill-mix planning, vacation and education planning, subcontracting policies, and how it can be used to provide rostering support.

We applied our model in a case study of the Emergency Department of the University Hospital St. Radboud Nijmegen, the Netherlands. Experimental results

show that the annualized hours regime decreases the dependency of the department on subcontractor tariffs, and thus reduces financial risk. In addition, without any subcontracting, annualized hours offers a possible savings of 5.2% or €86000 for this department. Moreover, results show that covering all workforce demand with contracted employees is not necessarily cost-efficient. If there is a mismatch between capacity and demand, due to peaks in demand or peaks in employee absences, assigning subcontractors to these peaks might be cheaper than contracting additional full-time or part-time employees.

CHAPTER 5

# Staffing under Annualized Hours Using Cross-Entropy Optimization

## 5.1 Introduction

In this chapter, we integrate a staffing and an annualized hours problem. We refer to this as the staffing under annualized hours (SUAH) problem. For the SUAH problem, it is not straightforward to define efficient neighborhood operators on the underlying combinatorial optimization problem, which we motivate in Section 5.3. Therefore, this chapter investigates the potential of Cross-Entropy optimization to solve the SUAH problem, since in contrast to other metaheuristics, Cross-Entropy optimization does not require a well-defined neighborhood structure. Since Cross-Entropy optimization has shown to work well on related optimization problems, we are going to investigate how Cross-Entropy Optimization can be tailored to solve SUAH.

Cross-Entropy (CE) optimization is widely used for rare event simulation, but is also used to solve combinatorial optimization problems. To solve a combinatorial optimization problem, CE iteratively generates solutions. In each iteration, a set of solutions is generated, and based on the best solutions in this set, the generation scheme is updated, with the aim to generate better solutions in the next iteration. For a general introduction to Cross-Entropy optimization, we refer to [228]. In Section 5.4, we provide a brief description of Cross-Entropy optimization.

CE optimization is successfully applied to problems related to SUAH, such as the multidimensional knapsack problem (see, e.g., [128]) and the capacitated facility location problem (see, e.g., [80]). The objectives of multidimensional knapsack and capacitated facility location are to select an 'optimal' set of items and an 'optimal' set of (supply) locations respectively. In [80] the capacitated facility location problem is solved using CE and in [82, 113, 230] CE is used to solve (variants of) knapsack problems. In addition to successful CE applications to related problems,

metaheuristics, such as tabu search and simulated annealing, have also shown to work well on knapsack problems [128]. An overview of the related literature is found in Section 5.2.

The results of our CE implementation show that, across a broad range of instances, CE considerably outperforms a mathematical programming formulation solved with Cplex. Specifically instances with 10 or more planning periods, the periods over which the annualized hours problem distributes available capacity, are solved relatively quickly by CE. In practice, the planning horizon often contains dozens of planning periods, for example, if the planning horizon is subdivided into weeks, there are 52 planning periods in a year. The CE implementation solves all our instances in matters of seconds and finds solutions close to or very close to optimal. Within an imposed time limit of an hour, occasionally CE even finds better solutions than Cplex. Since CE is designed for single-objective unconstrained optimization problems, we included a straightforward repair function in our approach to guarantee feasible solutions. Given its solution speed and quality, CE is the preferred method for quickly analyzing multiple input scenarios. Also CE is well-suited for assessing the effect of changes in parameter values, for example vacation requests or illnesses. If optimal solutions are required, as in Chapter 4, mathematical programming should be considered.

This chapter is structured as follows. Section 5.2 discusses the literature. Section 5.3 gives a mathematical description of SUAH, and further motivates why we use CE optimization to solve it. Next, Section 5.4 describes the CE optimization technique, and Section 5.5 discusses how we employ CE to solve the SUAH problem. Section 5.6 gives numerical results, and conclusions and recommendations are presented in Section 5.7.

## 5.2 Literature

For the literature on staffing and annualized hours the reader is referred to Section 4.2. Here, we discuss applications of Cross-Entropy optimization to combinatorial optimization problems.

CE is applied to a variety of combinatorial optimization problems. Some of these problems are closely related to our problem as outlined in Section 5.3. In [113] CE is used to solve a single-dimensional knapsack problem. In [82] CE is again applied to the single-dimensional knapsack problem, but then with setups: a fixed cost is incurred if an item is selected and a variable profit is earned depending on the 'integer' usage of this item. In [80] large-scale capacitated facility location problems are solved by CE. This problem is similar to ours, see

Section 5.3. Although the literature does not report on CE applications to solve the *multidimensional* knapsack problem, the MATLAB File Exchange Central holds a script for this problem [204], see Section 5.6.

Applications of CE to combinatorial optimization problems that are less related to our problem are found in, e.g., [81] that solves a capacitated lot-sizing problem with setup times. Moreover, [108, 147, 228, 231] solve traveling salesman problems (TSP) using CE, and [108, 229, 230] solve the max-cut problem using CE. In [230], a variant of CE is proposed, and applied to numerous combinatorial optimization problems. In [17, 79, 151] network reliability optimization problems are solved by CE, and [45] addresses a multi-objective optimization problem.

## 5.3 Problem description

Consider a workforce planning problem where the planning horizon is discretized into a set of time slots $T$, indexed by $t$. Employing staff is subject to cost and availability constraints. The set of employees is represented by $I$ and indexed by $i$. Employees are employed throughout the whole planning horizon at a cost $c_i$. The number of working hours employee $i$ is available during the whole planning horizon is $w_i$. Furthermore, for each period $t \in T$, there is a demand for a number of working hours, $d_t$. In addition, we have lower and upper bounds on the number of hours employee $i$ *should* and *can* work during period $t$, denoted by $l_{it}$ and $u_{it}$, respectively. Here, $l_{it}$ represents the minimal number of hours employee $i$ is paid for and $u_{it}$ represents the maximum number of hours employee $i$ is available in period $t$. It is assumed that $u_{it} \geq l_{it} \geq 0$. The objective is to select a subset of the employees and determine their working hours during every period $t$, such that demand is met in each period, and the number of hours the selected employees work is within the capacity restrictions. Furthermore, the cost of the employees should be minimized. We refer to this problem as the staffing under annualized hours (SUAH) problem.

Let $x_i$ indicate whether employee $i$ is part of the workforce ($x_i = 1$) or not ($x_i = 0$), and let $x_{it}$ denote the number of hours employee $i$ works in period $t$. The objective is to minimize the cost of the employed workforce:

$$\sum_{i \in I} c_i x_i. \tag{5.3.1a}$$

Three constraints are implied on the selection of employees. First, in every period

$t$ demand has to be met:

$$\sum_{i \in I} x_{it} \geq d_t \quad t \in T. \tag{5.3.1b}$$

Second, the number of hours an employee works throughout the planning horizon equals $w_i$ if the employee is part of the workforce, and 0 otherwise:

$$\sum_{t \in T} x_{it} = w_i x_i \quad i \in I. \tag{5.3.1c}$$

Third, per period, per employee, $x_{it}$ must lie within the given lower and upper bounds, if the employee is part of the workforce, and equal 0 otherwise:

$$l_{it} x_i \leq x_{it} \leq u_{it} x_i \quad i \in I, t \in T. \tag{5.3.1d}$$

Finally, for $x_i$ we have:

$$x_i \in \{0, 1\} \quad i \in I. \tag{5.3.1e}$$

Model (5.3.1) now states a mixed-integer linear programming formulation (MILP) of SUAH.

Note that SUAH is closely related to the capacitated facility location problem (CFLP), if we regard the facilities and destinations of CFLP as the employees and planning periods of SUAH. However, there are two important differences. First, in SUAH there is no transportation cost $c_{it}$. Second, and more importantly, SUAH has lower ($l_{it}$) and upper ($u_{it}$) bounds on the amount transported from location $i$ to destination $t$. Adding these bounds to CFLP increases the complexity. For CFLP to be feasible it is sufficient to have $\sum_{i \in I} w_i x_i \geq \sum_{t \in T} d_t$, whereas for SUAH it is not.

The multidimensional knapsack problem (MKP) is the special case of (5.3.1) where $l_{it} = u_{it}$. For MKP, optimal and near optimal solutions lie in the boundary of the feasible region, see [129]. For SUAH this also holds, since removing one of the employees from an optimal solution makes the solution infeasible. If it would be possible to remove an employee from an optimal solution without giving up feasibility, a cheaper solution is found, contradicting that the solution found is optimal. Hence, it is difficult to define a neighborhood with an efficient neighborhood search strategy, since improved solutions can only be found by removing one of the employees from the solution and inserting another employee into the solution. In addition, the values of $w_{ij}$ also have to be updated if the values of the $x_i$ change. In Section 5.5, we show that, if the values of $x_i$ are given, the $w_{ij}$ values can be easily found by solving a network flow problem. While network flow problems can be solved efficiently, solving many network flow formulations is very

time consuming. Given these characteristics of SUAH, we choose to apply Cross-Entropy optimization, which, in contrast to many other meta-heuristics, does not require a well-defined search neighborhood.

We present a detailed description of Cross-Entropy (CE) optimization in Section 5.4. CE has proven to be successful on variants of (single) knapsack problems, see [82, 113, 230], and on CFLP, see [80]. Given the close relation between SUAH and MKP, and the success of CE on knapsack problems, we choose to apply the CE optimization metaheuristic.

## 5.4 Cross-Entropy optimization

This section discusses the Cross-Entropy optimization (CE) method. As motivated in Section 5.3, we use CE to solve the staffing under annualized hours (SUAH) problem. This section gives a general description of the CE optimization technique. The details of our CE implementation are discussed in Section 5.5.

The CE method was initially developed during the late 1990s [228]. Loosely speaking, CE is a self-learning importance sampling (IS) method. Importance sampling is a well-known technique for rare event simulation. However, CE is also applied to many combinatorial optimization problems. A general introduction to CE, that is more elaborate than presented here, is found in [108, 231]. CE is an iterative method that generates samples of solutions, and in every iteration based on 'good' solutions, the sampling distribution's parameters are updated such that better solutions are generated in the next iteration.

To make this more formal, we closely follow the approach and notation of [108]. In this chapter we consider the general 0-1 binary minimization problem, with performance evaluation function $S : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \{0,1\}^m$ represents the feasible region. We want to minimize $S$ over $\mathcal{X}$, i.e., find $x^* \in \mathcal{X}$ such that:

$$S(x^*) = \min_{x \in \mathcal{X}} S(x) = \gamma^*. \tag{5.4.1}$$

CE aims to find $x^*$ by iteratively generating random samples from the family of random variables $X^{(v)}$ with probability distribution function $f(x; v), x \in \mathcal{X}, v \in \mathcal{V}$, where $\mathcal{V}$ is a set of parameter values. In this chapter, we let $f(x; v)$ be a pdf of the Bernoulli distribution:

$$f(x; v) = \prod_{i \in I} v_i^{x_i} (1 - v_i)^{1 - x_i}. \tag{5.4.2}$$

Now, for $x \in \mathcal{X}$, $v \in \mathcal{V}$, and $\gamma \in \mathbb{R}$, let:

$$l_v(\gamma) = \mathbb{P}_v\{S(X^{(v)}) \leq \gamma\} = \mathbb{E}_v I_{\{S(X^{(v)}) \leq \gamma\}} = \sum_{x \in \mathcal{X}} I_{\{S(x) \leq \gamma\}} f(x; v), \tag{5.4.3}$$

where the indicator function $I_A = 1$ if $A$ occurs, and 0 otherwise.

CE iterates over the set $\mathcal{V}$ to estimate $\gamma^*$ and $x^*$, and the corresponding 0-1 vector $v^* \in \mathcal{V}$, such that $X^{(v^*)} = x^*$. Hence, $X^{(v^*)}$ is a deterministic random variable with all mass at $x^*$. The estimation problem (5.4.3) is often referred to as the *associated stochastic problem* (ASP).

For $N$ i.i.d. random variables $X_1, \ldots, X_N$ distributed as $X^{(v)}$, an unbiased estimator of $l_v(\gamma)$ is:

$$\frac{1}{N} \sum_{k=1}^{N} I_{\{S(X_k) \leq \gamma\}}. \tag{5.4.4}$$

However, when $\{S(X_i) \leq \gamma\}$ is a rare event, a huge number of samples has to be generated to estimate $l_v(\gamma)$. An alternative to this, which is used in CE, is based on *importance sampling* (IS). Take a random sample $X_1, \ldots, X_N$ with a different density $g(x)$ on $\mathcal{X}$, and evaluate $l_v(\gamma)$ using the likelihood-ratio (LR) estimator:

$$\widehat{l_v(\gamma)} = \frac{1}{N} \sum_{k=1}^{N} I_{\{S(X_k) \leq \gamma\}} \frac{f(X_k; v)}{g(X_k)}. \tag{5.4.5}$$

The best way to estimate $l_v(\gamma)$, see [108], is to use a change of measure. The optimal $g^*(x)$ would be:

$$g^*(x) = \frac{I_{\{S(x) \leq \gamma\}} f(x; v)}{l_v(\gamma)}, \tag{5.4.6}$$

since inserting (5.4.6) in (5.4.5) gives $\widehat{l_v(\gamma)} = l_v(\gamma)$.

This is infeasible as $g^*(x)$ depends on the unknown $l_v(\gamma)$. Furthermore, it is convenient to choose $g^*(x)$ from the family of probability distributions $f(x; v)$. The idea in CE optimization is to choose $v$ such that the distance between $g^*(x)$ and $f(x; v)$ is minimized. A convenient measure of distance between two densities $g(x)$ and $h(x)$ is the CE distance, also referred to as the *Kullback-Leibler* distance. The CE distance between $g(x)$ and $h(x)$ is defined as:

$$\begin{aligned} \mathcal{D}(g(x); h(x)) &= \mathbb{E}_{g(x)} \left[ \log \left( \frac{g(x)}{h(x)} \right) \right] \\ &= \int g(x) \log g(x) \mathrm{d}x - \int g(x) \log h(x) \mathrm{d}x. \end{aligned} \tag{5.4.7}$$

Minimizing the CE distance between $g^*(x)$ and $f(x;v)$ over $v$ is equivalent to:

$$\max_v \int_{x\in\mathcal{X}} g^*(x)\log f(x;v)\mathrm{d}x. \tag{5.4.8}$$

Substituting $g^*(x)$ from (5.4.6) in (5.4.8) gives:

$$\max_v \int_{x\in\mathcal{X}} \frac{I_{\{S(x)\leq\gamma\}}f(x;u)}{l_u(\gamma)}\log f(x;v)\mathrm{d}x, \tag{5.4.9}$$

which is equivalent to:

$$\max_v \mathbb{E}_u I_{\{S(X^{(u)})\leq\gamma\}}\log f(X^{(u)};v). \tag{5.4.10}$$

Given a sample $X_1,\ldots,X_N$, which are generated under pmf $f(x;u)$, we can estimate $v$ from:

$$\hat{v} = \arg\max_v \frac{1}{N}\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}\}}\log f(X_k;v), \tag{5.4.11}$$

where we let $\hat{\gamma}$ be the $(1-\rho)$-quantile of the performances, i.e., sort the $S(X_i)$ on decreasing value: $S_{(1)},\ldots,S_{(N)}$ and let:

$$\hat{\gamma} = S_{(\lfloor(1-\rho)N\rfloor)}. \tag{5.4.12}$$

In [108] is shown that, if $\mathcal{V}$ is the set of Bernoulli($v$) distributions, the CE distance (5.4.7) is minimized for:

$$\hat{v}_i = \frac{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}\}}X_{k,i}}{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}\}}}, \tag{5.4.13}$$

where $X_{k,i}$ and $\hat{v}_i$ denote the $i$-th element of $X_k$ and $\hat{v}$, respectively.

Note that updating rule (5.4.13) appeals to intuition, since it sets $\hat{v}_i$ as the fraction of the number of times that element $i$ is present in the $1-\rho$ sample quantile. Note that - except for the indicator functions - (5.4.13) equals the maximum likelihood estimator for $\hat{v}$.

In order to approximate $\gamma^*$ and $v^*$ we update $\gamma$ and $v$ iteratively as in equation (5.4.12) and (5.4.13), respectively, by using the algorithmic representation of CE optimization as in Algorithm 5.1.

In many CE implementations updating rule (5.4.13) is replaced by:

$$\hat{v}_{\tau,i} = \lambda\frac{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}_\tau\}}X_{k,i}}{\sum_{k=1}^{N} I_{\{S(X_k)\leq\hat{\gamma}_\tau\}}} + (1-\lambda)\hat{v}_{\tau-1,i}, \tag{5.4.15}$$

**Algorithm 5.1** Cross-Entropy algorithm (for optimization)

1. Let $\hat{v}_0 = u$, for some initial distribution parameter $u$. Let $N$ denote the sample size, let $d \in \mathbb{N} \setminus \{0\}$, and $0 < \rho < 1$ be given. Set $\tau = 1$ (iteration counter).
2. Generate a sample $X_1, \ldots, X_N$ from the density $f(x; \hat{v}_{\tau-1})$ and compute the sample $(1-\rho)$-quantile $\hat{\gamma}_t$ of the performances, as in (5.4.12)
3. Use the same sample $X_1, \ldots, X_N$ and choose $\hat{v}_\tau \in \mathcal{V}$ such that the CE distance $D(f(x; \hat{v}_{\tau-1}), f(x; \hat{v}_\tau))$ as defined in (5.4.7) between $f(x; \hat{v}_{\tau-1})$ and $f(x; \hat{v}_\tau)$ is minimized.
4. If:
$$\hat{\gamma}_\tau = \hat{\gamma}_{\tau-1} = \ldots = \hat{\gamma}_{\tau-d}, \tag{5.4.14}$$
then stop. Otherwise set $\tau = \tau + 1$ and return to 2.

for some $0 < \lambda < 1$, in order to prevent the CE method from converging too quickly [108]. In addition, [103, 186] let $\lambda$ depend on $\tau$, i.e., replace $\lambda$ by $\lambda_\tau$ in (5.4.15), to achieve this. We want to prohibit that CE converges too quickly, because once an entry of $\hat{v}_\tau$ is fixed to 0 or 1, only sample solutions with (if the entry is fixed to 1) or without (if the entry is fixed to 0) the corresponding element are generated. Another way to control the convergence of CE is to let $\rho$ depend on $\tau$ [113]. In [103, 186] conditions for $\lambda_\tau$ are presented for the CE method to convergence to an optimal solution of the combinatorial optimization problem asymptotically with probability 1. Furthermore, [103] discusses necessary and sufficient conditions on $\lambda_\tau$ for $f(x; \hat{v}_\tau)$ to converge with probability 1 to a unit mass located at some solution candidate $x$.

In the literature several other improvements to CE exist. In [230] it is suggested to use:
$$\hat{v}_i = \frac{\sum_{k=1}^{N} S(X_k) I_{\{S(X_k) \leq \hat{\gamma}\}} X_{k,i}}{\sum_{k=1}^{N} S(X_k) I_{\{S(X_k) \leq \hat{\gamma}\}}}, \tag{5.4.16}$$
instead of (5.4.13). It is argued in [230] that this updating rule is, in general, at least as fast and accurate as (5.4.13) and it has a stronger mathematical foundation. In [80] it is suggested to apply, in every iteration, local search to the best elements of the sample solution. According to [80], better quality solutions are obtained for their problem, but at the cost of additional computation time.

In its basic form CE cannot handle constraint optimization problems, and is used to solve single objective optimization problems. When applying CE to constrained combinatorial optimization problems, such as SUAH, the generation of infeasible solutions has to be prevented. There are two main approaches to deal with this. A first approach is to penalize infeasibility, see, e.g., [17, 79, 103]. A second approach is to repair infeasible solutions, see, e.g., [82, 113]. In our implementation, we choose to penalize infeasibility and to apply repair functions, see Section 5.5.

## 5.5 Solution approach

The objective of model (5.3.1) is to select the most cost-efficient set of employees that is able to cover demand. We use Cross-Entropy (CE) optimization to solve this problem. For our problem we use CE to select the employees. If the set of selected employees is known, an easy network flow problem remains, see Section 5.5.1. Similar approaches are found in, e.g., [81, 82], where the idea is to reduce complexity by letting CE intelligently guess which 'items' to select. In Section 5.5.2 we describe how CE is used to select a set of employees, i.e., determine values for $x_i$. In Section 5.5.3 we discuss how we use feasibility conditions to incorporate demand constraints in the employee selection. However, since this approach does not necessarily lead to feasible solutions either, which is needed for practical applications, repair functions are used to guarantee feasible solutions, see Section 5.5.4. Section 5.5.5 discusses some details of our MATLAB implementation of CE.

### 5.5.1 Annualized hours for given employees

If the values of $x_i$ are known, the annualized hours problem is easily solved, since for given values of $x_i$ model (5.3.1) can be rewritten such that there are no lower bounds on $x_{it}$, except for nonnegativity. Let $M = \{i | x_i = 1\}$ and $\bar{x}_{it} = x_{it} - l_{it}$, $\bar{u}_{it} = u_{it} - l_{it}$, $\bar{w}_i = w_i - \sum_{t \in T} l_{it}$ and $\bar{d}_t = \bar{d}_t - \sum_{t \in T} l_{it}$ then (5.3.1) reduces to:

$$\text{find} \quad \bar{x}_{it} \tag{5.5.1a}$$

$$\text{s.t.} \quad \sum_{i \in M} \bar{x}_{it} \geq \bar{d}_t \quad t \in T \tag{5.5.1b}$$

$$\sum_{t \in T} \bar{x}_{it} = \bar{w}_i \quad i \in M \tag{5.5.1c}$$

$$0 \leq \bar{x}_{it} \leq \bar{u}_{it} \quad i \in M, t \in T. \tag{5.5.1d}$$

Instead of a mixed-integer linear program we now have a linear program, which is solvable by polynomial time algorithms. Moreover, (5.5.1) is a network *feasibility* problem [125, 133], for which special purpose algorithms exist.

### 5.5.2 Initialization

This section describes how our CE method is initialized. In many CE implementations that involve binary variables, $v_0$ is set to $\left(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right)$, see, e.g., [81, 151]. However, in our implementation we choose to set

$$v_0 = \left( \frac{\sum_{t \in T} d_t}{\sum_{i \in I} w_i}, \ldots, \frac{\sum_{t \in T} d_t}{\sum_{i \in I} w_i} \right). \tag{5.5.2}$$

The fraction $\sum_{t \in T} d_t / \sum_{i \in I} w_i$ equals the *expected* number of items needed for a solution to be feasible, if we ignore per time period demand constraints and only look at total capacity and total demand. Therefore, every generated solution has *in expectation* the number of items needed for a solution to be feasible. This way, the CE implementation converges faster and the probability to get stuck at infeasible solutions decreases.

### 5.5.3 Feasibility conditions

After a set of items is selected, feasibility can be checked by solving (5.5.1). Although this problem is polynomially solvable, it has to be solved for at least the best $\rho$ percent samples in every iteration of the CE algorithm, which is quite time consuming if it has to be done often.

Therefore, we incorporate two feasibility conditions on the employee selection of CE. Firstly, the total permitted number of employee working hours should exceed the total demand:

$$\sum_{i \in I} w_i x_i \geq \sum_{t \in T} d_t. \tag{5.5.3}$$

Secondly, the maximum permitted number of employee working hours in every period $t$ should exceed $d_t$:

$$\sum_{i \in I} u_{it} x_i \geq d_t \quad \text{for } t \in T. \tag{5.5.4}$$

Unfortunately, feasibility conditions (5.5.3) and (5.5.4) are necessary, but not sufficient. The only sufficient feasibility condition for a network problem that we know of, is to solve it using a network flow formulation or an LP formulation.

We incorporate (5.5.3) and (5.5.4) into the CE implementation, by penalizing violations in the objective function. The objective function is then given by:

$$S(x) = \sum_{i \in I} c_i x_i + \beta_1 \left( \sum_{t \in T} d_t - \sum_{i \in I} w_i x_i \right)^+ + \beta_2 \sum_{t \in T} \left( d_t - \sum_{i \in I} u_{it} x_i \right)^+, \quad (5.5.5)$$

where $(x)^+ = \max\{x, 0\}$ and $\beta_1, \beta_2 > 0$. The first part of the objective function represents the cost of the workforce, the second and the third part represent penalties caused by violating (5.5.3) and (5.5.4), respectively.

### 5.5.4 Repair functions

The approach of Section 5.5.3 does not guarantee solutions to be feasible, even for large values of $\beta_1, \beta_2$ in (5.5.5) CE might produce infeasible solutions. By adding a repair function that 'repairs' infeasible solutions, we can guarantee solutions to be feasible.

For this, we introduce an extension to model (5.5.1) that incorporates slack variables $\delta_t$ in the demand constraints as follows:

$$\begin{aligned}
\min \quad & \sum_{t \in T} \delta_t && (5.5.6a)\\
\text{s.t.} \quad & \sum_{i \in M} \bar{x}_{it} + \delta_t \geq \bar{d}_t \quad t \in T && (5.5.6b)\\
& \sum_{t \in T} \bar{x}_{it} = \bar{w}_i \quad i \in M && (5.5.6c)\\
& 0 \leq \bar{x}_{it} \leq \bar{u}_{it} \quad i \in M, t \in T && (5.5.6d)\\
& 0 \leq \delta_t \quad t \in T. && (5.5.6e)
\end{aligned}$$

The objective of (5.5.6) represents the demand that the employees in $M$ together are unable to cover. If this is larger than 0 the solution is infeasible and we invoke our repair strategy.

For this, let:

$$t' = \arg\max_{t \in T} \delta_t, \quad (5.5.7)$$

then $t'$ denotes the index of the demand constraint for which we have the largest slack. Let:

$$i' = \arg\max_{i \notin M} \left\{ \frac{c_i}{u_{it'} - l_{it'}} \right\}, \quad (5.5.8)$$

then, from the set of employees not in $M$, employee $i'$ can contribute to covering the demand in demand constraint $t'$ most cost-efficiently.

Algorithm 5.2 represents the repair strategy we invoke.

**Algorithm 5.2** Repair strategy

1. Let $M$ be the solution of CE. If it is feasible: done. Otherwise, go to 2.
2. Let $\delta$ be the solution to (5.5.6). Let $t'$ be the solution of (5.5.7), and $i'$ be the solution of (5.5.8). Let $M' = M \cup \{i'\}$. If $M'$ offers a feasible solution: done. Otherwise, reiterate using $M'$.

### 5.5.5 Software implementation

The CE method is implemented in MATLAB. For a fair comparison between CE and (not fine-tuned) CPLEX, a MATLAB script from [204], designed to solve the basic multidimensional knapsack problem with CE, is adapted to make it suitable to solve SUAH. Our adaptations to [204] include incorporating initialization rule (5.5.2), and modifying the objective function into (5.5.5). In our implementation we choose to set:

$$\beta_1 = \beta_2 = \frac{2 + 2\sum_{i \in I} c_i}{\max\limits_{i \in I} \max\limits_{t \in T} (l_{it} + u_{it})}, \tag{5.5.9}$$

which is analoguous to the (single) $\beta$ parameter used in [204], so as to stay close to the basic CE implementation of [204]. The intuition behind these parameter values is that if $\max_{i \in I} \max_{t \in T} (l_{it} + u_{it})$ is small, it is more likely that solutions are obtained where $\sum_{i \in I} u_{it} x_i$ is smaller than $d_t$ for some $t$. To prohibit CE from getting stuck at these solutions we want $\beta_1, \beta_2$ to be larger for instances where $\max_{i \in I} \max_{t \in T} (l_{it} + u_{it})$ is small. The other way around, when $\max_{i \in I} \max_{t \in T} (l_{it} + u_{it})$ is larger, $\beta_1, \beta_2$ are smaller, since for these instances it is less likely that solutions are obtained where $\sum_{i \in I} u_{it} x_i$ is smaller than $d_t$. Determining good values for $\beta_1, \beta_2$ is a trade-off. On the one hand, large $\beta_1, \beta_2$ make it more likely to find feasible solutions, but, on the other hand, they make it less likely to find a (near) optimal solution; larger $\beta_1, \beta_2$ 'push' CE harder away from the boundary of the feasible region, where the optimal solutions lie.

## 5.6 Experimental results

This section discusses the experimental results. The Cross-Entropy (CE) implementation described in Section 5.5 is compared with a mixed-integer linear programming implementation (MILP) of (5.3.1), since solving the MILP gives an optimal solution, which enables us to report on the solution quality of CE.

The MILP instances are encoded as MPS-files and solved using a precompiled Cplex 11.0 binary. We let Cplex stop when the instances are solved to optimality, that is when the integrality gap is less than 0.01%, or if the solving time exceeds 3600 seconds. We did not perform extensive tuning on the parameters of Cplex. The auto-tuner of Cplex did not suggest parameter values that would be useful in general for solving the SUAH instances. Furthermore, we did no fine-tuning of CE parameters such that we compare two not fine-tuned implementations. All experiments are performed on an Intel Centrino Duo CPU 2.20 GHz, 3.0 GB of RAM.

Section 5.6.1 discusses the instances on which we test both implementations. After that, Section 5.6.2 and Section 5.6.3, evaluate the solving time and the solution quality of both implementations, respectively.

### 5.6.1 Test instances

The implementation is tested on generated instances. We generate multidimensional knapsack problem (MKP) instances that are transformed into SUAH instances. A multidimensional knapsack *covering* problem has the objective function as in (5.3.1a) and it has $|T|$ constraints of the form:

$$a_{1t}x_1 + a_{2t}x_2 + \ldots + a_{|I|t}x_{|I|} \geq d_t \quad t \in T. \tag{5.6.1}$$

The MKP instances are generated with the method of [90]. The values of $a_{it}$ are discrete uniform random numbers from $[0, 1000]$. Furthermore, $d_t = \alpha \sum_{i \in I} a_{it}$, where $\alpha$ is a *tightness* ratio specifying approximately the fraction of the total number of items that are needed for a solution to be feasible. We generate problem instances with the number of variables $|I|$ set to $10, 20, \ldots, 100$, and the number of constraints $|T|$ set to $5, 10, \ldots, 50$. For every $|I|$-$|T|$ combination 30 problem instances are generated. Just as in [90], we have $\alpha = \frac{1}{4}$ for the first ten problems instances, $\alpha = \frac{1}{2}$ for the next ten instances, and $\alpha = \frac{3}{4}$ for the remaining ten instances. The objective function coefficients $c_i$ are generated as:

$$c_i = \sum_{t \in T} \frac{a_{it}}{|T|} + 500 q_i \quad i \in I \tag{5.6.2}$$

where $q_i$ is a real uniform random number from $(0, 1)$. According to [90] instances where the $c_i$ and the $a_{it}$ are correlated are harder to solve. We generate ten SUAH instances from every MKP instance by setting $l_{it} = (1-p)a_{it}$ and $u_{it} = (1+p)a_{it}$ for $p = \frac{1}{10}, \frac{2}{10}, \ldots, 1$, and setting $w_i = \sum_{t \in T} a_{it}$. Here, $p$ is the annualized hours parameter indicating the percentage we are allowed to over-staff or under-staff in a planning period. In total, we thus generate $10 \cdot 10 \cdot 30 \cdot 10 = 30000$ instances.

We use these instances instead of the instances of [90] that are made publicly available [39], since our instances better fit with our underlying practical application. For SUAH it is typical to have up to 100 employees (variables) and up to 50 planning periods (constraints), e.g., in a planning horizon of one year we have 52 planning periods of one week.

### 5.6.2 Solving time

This section examines the effect of the number of variables $|I|$, the number of constraints $|T|$, the density parameter $\alpha$, and the annualized hours parameter $p$ on the solving time of both CE and MILP.

#### Effect of $|I|$ on solving time

Figure 5.1 shows the effect of the number of variables $|I|$ on solving times of CE and Cplex for $|T| = 40$. In the left figure solving times are averaged over $\alpha$ and $p$, in the figure on the right medians are computed.

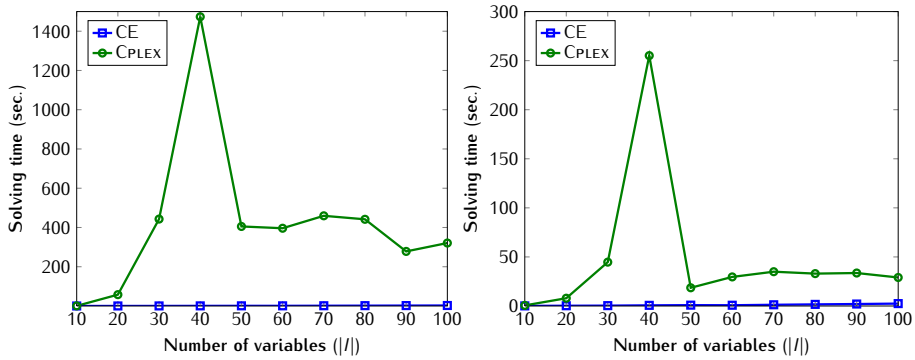



Figure 5.1: Mean (left) and median (right) solving time as function of $|I|$ for $|T| = 40$

From Figure 5.1, we observe that Cplex needs far more solving time than CE. Interesting to note is that for $|I| = 40$ Cplex needs on average the most time. We

think Cplex is able to relatively quickly obtain a solution for small $|I|$, since the number of solutions is limited, and that for larger $|I|$ ($|I| \geq 50$) there are many good solutions, which makes it easier for Cplex to find one. Also note that the average solving time of Cplex is strongly influenced by a small number of instances that are hard to solve; the difference between CE and Cplex is smaller when we look at the medians. We also observe that the CE solving time is almost constant. For other values of $|T|$, for which graphs are not shown, similar effects were observed.

### Effect of $|T|$ on solving time

Figure 5.2 shows the effect of the number of constraints $|T|$ on the solving time for $|I| = 30$. Again, in the left figure solving times are averaged over $\alpha$ and $p$, and the right figure shows medians.

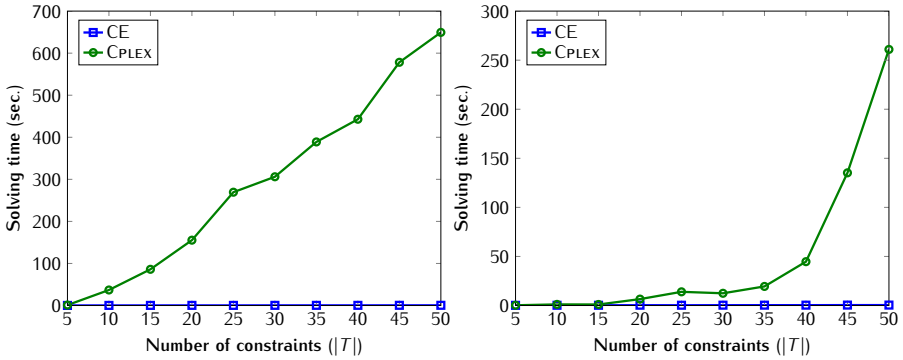



Figure 5.2: Mean (left) and median (right) solving time as function of $|T|$ for $|I| = 30$

Figure 5.2 shows that both the mean and median solving time of Cplex increases with $|T|$, which is caused by Cplex requiring more computation time per simplex iteration for increasing number of constraints. For CE both the mean and the median solving time stay about the same when $|T|$ increases, which is logical since $|T|$ only influences the computational effort needed to compute (5.5.5), which is, from a computational point, only a minor part of the CE implementation.

### Effect of $\alpha$ on solving time

Table 5.1 shows the effect of the density parameter $\alpha$ on the solving time.

Table 5.1 shows CE needs the most time for $\alpha = \frac{1}{2}$. For these instances about half of the total items are selected in the optimal solution, and the number of

Table 5.1: Effect of $\alpha$ on solving time

| | solving time CE | | | | solving time CPLEX | | | |
|---|---|---|---|---|---|---|---|---|
| $\alpha$ | min | max | mean | median | min | max | mean | median |
| 1/4 | 0.1 | 3.7 | 0.9 | 0.7 | 0.0 | 3600.0 | 463.1 | 19.8 |
| 1/2 | 0.1 | 5.5 | 1.1 | 0.9 | 0.0 | 3600.0 | 293.1 | 7.1 |
| 3/4 | 0.1 | 4.1 | 0.9 | 0.7 | 0.0 | 3600.0 | 98.1 | 4.2 |

ways to select $x$ items out of a set of $y$ items is the largest for $x = \frac{1}{2}y$. Hence, in expectation, CE needs more iterations before no improvements are found anymore. The solving time CPLEX needs decreases for increasing $\alpha$. For smaller $\alpha$ less items are selected in the optimal solution, which implies a smaller objective function value and hence the margin for the integrality gap is smaller. We think this makes it harder to find a solution.

### Effect of $p$ on solving time

Figure 5.3 shows the effect of the value of the annualized hours parameter $p$ on the solving time. In the left figure solving times are averaged over $|I|$, $|T|$, and $\alpha$, and in the right figure medians are computed.

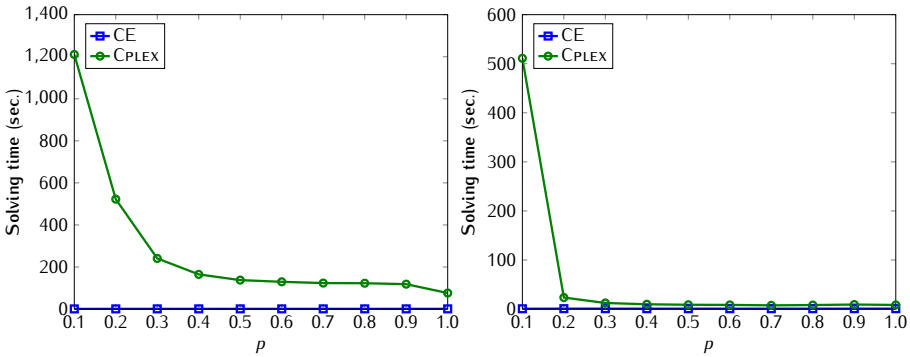



Figure 5.3: Mean (left) and median (right) solving time as function of $p$

We observe that the MILP instances are solved faster as $p$ increases. We think this is a consequence of the larger difference between $l_{it}$ and $u_{it}$ for larger $p$. This makes the bounds on $x_{it}$ less tight, making it easier to find a solution. For CE both the mean and median solving time remain approximately the same for increasing

$p$. This is logical since the computational effort of CE is independent of the *values* of $l_{it}$ and $u_{it}$.

#### Unsolved MILP instances

Table 5.2 shows, for every $|I|$-$|T|$ combination, the number of MILP instances for which Cplex is unable to find an optimal solution within an hour.

Table 5.2: Cplex unsolved instances

| | $\|I\|$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\|T\|$ | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 1 | 4 |
| 10 | 0 | 0 | 0 | 11 | 5 | 4 | 5 | 7 | 0 | 1 |
| 15 | 0 | 0 | 0 | 13 | 11 | 14 | 7 | 17 | 21 | 12 |
| 20 | 0 | 0 | 1 | 27 | 14 | 18 | 21 | 12 | 9 | 6 |
| 25 | 0 | 0 | 4 | 40 | 27 | 16 | 18 | 13 | 23 | 9 |
| 30 | 0 | 0 | 11 | 68 | 24 | 20 | 17 | 27 | 25 | 24 |
| 35 | 0 | 0 | 16 | 69 | 26 | 23 | 22 | 30 | 26 | 12 |
| 40 | 0 | 0 | 19 | 103 | 28 | 26 | 31 | 23 | 15 | 20 |
| 45 | 0 | 0 | 24 | 91 | 32 | 33 | 43 | 30 | 17 | 28 |
| 50 | 0 | 0 | 24 | 91 | 32 | 39 | 52 | 31 | 33 | 38 |

For small $|I|$ ($|I| \leq 30$) and small $|T|$ ($|T| \leq 10$), Cplex solved most instances within the hour. However, for larger $|I|$ and $|T|$ many instances are not solved within the hour. Interesting to note is that for $|I| = 40$ Cplex has the most unsolved instances. We think Cplex is able to quickly obtain a solution for small $|I|$ since the number of solutions is limited, and for larger $|I|$ there are many solutions, which makes it easier to find a solution.

### 5.6.3 Solution quality

This section compares the quality of the solutions produced by CE and Cplex. The solution quality is defined as the objective value of the solution divided by the linear programming relaxation solution. For less than 1% of the instances CE produces an infeasible solution. These instances are ignored in the comparisons.

The largest observed integrality gap for Cplex is 9.6%. The average and median integrality gap are 0.06% and 0.01%, respectively. The worst observed performance of CE is 44.8%, which is relatively large, however the average and median are

0.77% and 0.12%, respectively. In practice, the error in the input data is likely to be larger. Excluding the solutions for which the repair function needs to be applied, the worst performance of CE compared to the relaxation is 16.2%. More advanced repair functions probably improve on this worst performance figure.

### Effect of $|I|$ and $|T|$ on solution quality

In Figure 5.4, solution qualities are averaged over $\alpha$ and $p$. The left figure shows the effect of the number of variables $|I|$ on the solution quality for $|T| = 15$, and the right figure shows, for $|I| = 30$, the effect of the number of constraints $|T|$ on the mean solution quality.

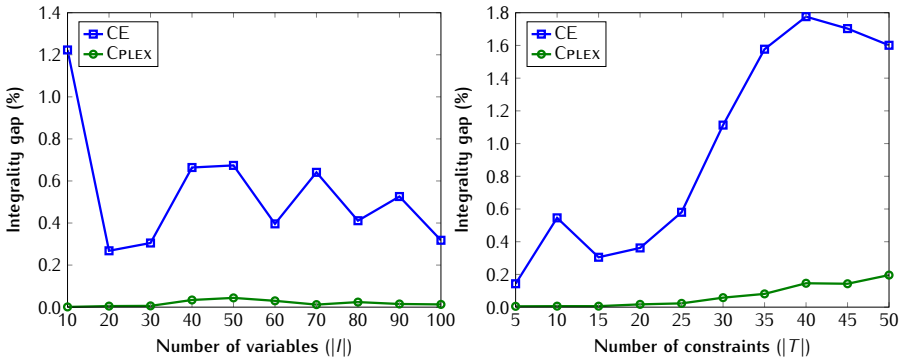



Figure 5.4: Mean solution quality as function of $|I|$ for $|T| = 15$ (left) and as function of $|T|$ for $|I| = 30$ (right)

From Figure 5.4, we observe that neither the value of $|I|$ nor the value of $|T|$ has a clear effect on the solution quality. Although we observe that the solution quality of CPLEX is on average better, the effects are small.

### Effect of $\alpha$ on solution quality

Table 5.3 shows the effect of the density parameter $\alpha$ on the solution quality. For each value of $\alpha$ the best, worst, mean, and median solution quality are shown.

As Table 5.3 shows the solution quality increases as $\alpha$ increases if looking at the worst and mean solution quality. We think this is caused by the fact that for larger $\alpha$ more items need to be selected in the optimal solution, which gives more freedom in the problem instance and makes it easier to find a good solution.

Table 5.3: Effect of $\alpha$ on solution quality (integrality gap)

| | CE | | | | Cplex | | | |
|---|---|---|---|---|---|---|---|---|
| $\alpha$ | best | worst | mean | median | best | worst | mean | median |
| 0.25 | 0.0% | 44.8% | 1.3% | 0.2% | 0.0% | 9.6% | 0.13% | 0.0% |
| 0.50 | 0.0% | 23.0% | 0.6% | 0.1% | 0.0% | 2.7% | 0.03% | 0.0% |
| 0.75 | 0.0% | 18.3% | 0.4% | 0.1% | 0.0% | 1.0% | 0.01% | 0.0% |

### Effect of $p$ on solution quality

In Figure 5.5, solution qualities are averaged over $|I|$, $|T|$, and $\alpha$, and shown as a function of the annualized hours parameter $p$.

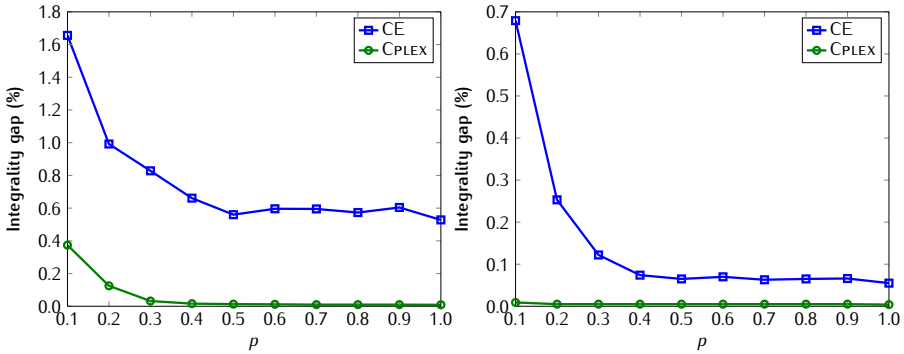



Figure 5.5: Mean (left) and median (right) solution quality as function of $p$

From Figure 5.5, we observe that the solution quality is better for increasing $p$ for both CE and Cplex. We think this is a consequence of the larger difference between $l_{it}$ and $u_{it}$ in the instances where $p$ is larger. In these cases, the bounds on $x_{it}$ are less tight. Provided that total capacity exceeds total demand, this makes it easier to find a solution, since there are less restrictions on the distribution of total capacity over all planning periods.

## 5.7 Conclusions and discussion

This chapter has studied the staffing under annualized hours (SUAH) problem. SUAH determines the optimal contract-mix of a workforce while applying annual-

ized hours. We have investigated the potential of Cross-Entropy (CE) optimization, a technique known from rare event simulation, to solve the SUAH problem. In contrast to other metaheuristics, CE does not require a well-defined neighborhood structure. Since it is not trivial to define an efficient neighborhood structure for SUAH, which is a key determinant for the success of most metaheuristics, we prefer CE over other metaheuristics. We have implemented CE in MATLAB. Our CE implementation includes a penalty function and a repair function to guarantee solutions to be feasible. In addition to the CE implementation, we have modeled the SUAH problem as a mathematical program that is solved by CPLEX.

For practical applications, opportunities for improving our CE implementation exist. First, simulations to optimize the penalty parameters of the model could be run. In addition, improved or alternative repair functions may be implemented. Second, updating schemes for the model parameters $\lambda_\tau$, $\rho_\tau$ and $N_\tau$ could be used. In [103, 113, 186] some success with this is mentioned, and conditions on $\lambda_\tau$ for asymptotic optimality of CE are derived in [103, 186]. We have implemented updating rule (5.4.16) of [230], but this had no noticeable positive effect for our CE implementation.

We have generated 30000 test instances. For most instances CPLEX produces the best solution, but CE solves the instances much faster than CPLEX. On average, CE is able to produce high-quality solutions in matters of seconds, whereas CPLEX requires a couple of minutes. Moreover, for 1738 instances (5.8%), CPLEX is not able to find the optimal solution within our time limit of one hour. Especially for larger instances that are in size comparable to real-life instances, CPLEX requires more time to solve the instances, if solved at all within an hour. This makes our CE implementation well-suited for data validation and scenario-analyses, such as evaluating the consequences of vacation requests and last-minute illnesses. In addition, for applications that have the SUAH problem, or a related problem, as a subproblem it is worthwhile to consider CE to solve these subproblems.

CHAPTER 6

# Shift Rostering Using Decomposition: Assign Days Off First

## 6.1 Introduction

In many practical situations, scheduling days off is a separate step in the shift assignment process. For employees, it is preferable or even requisite, see [198], to know their working days a long time ahead, so they can plan their leisure activities. The exact working hours are not essential to know in the long term. In addition, vacation requests can be taken into account long before the actual work schedules are created, which may alert the planner for possible capacity problems.

In this chapter, we study a decomposition approach for the shift rostering problem, that first solves the days off scheduling problem, which assigns working days and days off to a set of employees, and secondly assigns specific shifts, e.g., early or late, to working days in the days off schedule. In addition, we study an extended model that includes night shifts in the first phase of the decomposition. These decomposition methods reduce the complexity of the shift rostering problem by decomposing the problem into subproblems that are easier to solve. Since a decomposition approach implies a possible loss of solution quality, we want to mitigate this effect by solving the subproblems to optimality. To achieve this, we model the subproblems using mathematical programming formulations, which are solved using Cplex.

The contribution of this research is an analysis of the potential and pitfalls of days off decomposition. We compare two approaches: one with and one without night shifts included in the first phase of the decomposition. To let the decomposition be the only source of loss in solution quality, both phases of the decomposition are solved to optimality. The decomposition approaches are evaluated on 25 public nurse rostering benchmark instances [105]. The proposed mathematical programs may also be used to only solve the days off scheduling part, for practical situations

where working days should be scheduled a long time ahead.

This chapter is structured as follows. First, Section 6.2 discusses the related literature, and after that Section 6.3 presents a problem description. In Section 6.4, we present our decomposition approach, and Section 6.5 discusses computational results. Section 6.6 presents the conclusions.

## 6.2 Literature review

There is a vast amount of literature on personnel scheduling, see Chapter 3. Within the personnel scheduling literature days off scheduling has received considerable attention. In this section, we review the days off scheduling literature and indicate how the decomposition approach as proposed in this chapter contributes to the existing literature.

In the literature there has been a particular focus on *cyclic* days off scheduling. Algorithms to determine weekly repeating days off schedules for which employees have in each week 5 consecutive working days and 2 consecutive days off are proposed in [11, 12, 27, 36, 37, 41, 227, 246]. The objective is to determine the minimum size workforce or the minimum cost workforce. These days off scheduling problems are polynomially solvable, since they can be modeled as network flow problems [37]. The research in [28] extends on this by including part-time workers, which may work less than 5 consecutive shifts and [13] determines the minimum workforce in the situation where in each 3-week period an employee has 14 consecutive working days and 7 consecutive days off.

Research presented in [31, 55, 153, 174] determines the minimum workforce size for cyclic days off scheduling with various days off policies and workforce demand being $N$ on weekdays and $n$ during weekends. The research in [120] extends on this by implying that an employee has at least $A$ out of $B$ weekends off, and [122] include part-time workers who work fewer shifts than full-time workers. The research in [30] also includes constraints on the number of working weekends, but assumes staff requirements are the same each day, whereas [73, 75, 209] allow staff requirements to be different *each* day. The research in [185, 226] first determine a set of schedules, given a set of constraints, which are afterwards assigned to employees, and [121, 156, 196] determine the minimum cost workforce for hierarchical multi-skilled workforces. With hierarchical skills there is an ordering of skills such that a higher skilled more expensive worker can always do the work of a lower-skilled less-expensive worker.

Acyclic days off scheduling has also received considerable attention. A days off scheduling problem with a scheduling horizon of one year, enforcing constraints

on the number of shifts per week and during the scheduling horizon, is considered in [21, 22, 159]. The research in [14, 191] consider constraints on the number of (consecutive) working days and days off under the objective to minimize the workforce size, whereas [119] minimizes the workforce cost, and [40, 188] minimize the penalty implied by violations of scheduling criteria. A feasibility problem under numerous days off scheduling constraints is considered in [104].

In this chapter, we solve the shift rostering problem by first solving the days off scheduling problem and then assigning shifts to employees on their working days. This is also done in [38, 107, 131]. The research in [38] uses heuristics to solve the subproblems, whereas we use mathematical programming to get optimal solutions for the subproblems. In [107, 131] the subproblems are solved by enumerating days off patterns and solving partitioning problems. In [107] the days off scheduling problem is solved to optimality, however the shift assignment phase is not necessarily solved to optimality. Due to the heuristic nature of their methods, [38, 107] include a rescheduling method in their approaches. In [1] the shift rostering problem is decomposed into three phases: days off assignment, night shifts assignment, and morning and evening shift assignment. Each phase is solved to optimality. Our approach integrates this first and second phase. Unfortunately, [1] does not report any computational results. The research in [205] also decomposes the shift rostering problem in three phases. First, employees that are going to work night shifts are determined, second the days off schedule is generated, and third shifts are assigned to employees on working days. In [205] two alternative local search methods are compared to solve these phases.

Some authors combine days off scheduling with shift scheduling [25, 42]. Shift scheduling determines the set of shifts that should be assigned to employees. The research in [25, 42] start with shift scheduling, then generate a days off schedule, and finally assign shifts to employees on their working days.

## 6.3 Problem description

This section describes the shift rostering problem (see Section 6.3.1) and the set of benchmark instances to which we apply our approach (see Section 6.3.2).

### 6.3.1 The shift rostering problem

The basic data in the shift rostering problem that is considered in this chapter is given by:

- A *time period*, discretized in a set of time slots $T$, indexed by $t$. In the chapter, the length of a time slot $t$ is a day, and we consider a planning horizon of multiple weeks.
- A set of *employees* $I$, indexed by $i$.
- A set of *skills* $J$, indexed by $j$. The set of skills of employee $i$ is represented by $J_i \subseteq J$.
- A set of *shift types*, $K$, indexed by $k$. A shift type is a time interval which starts at a fixed time during the day and in which work is performed. Employees are only allowed to perform shifts for which they are sufficiently skilled.
- A set of *shifts*. A shift is of one of the shift types with a given start day.

The objective of the shift rostering problem is to assign the set of shifts to the set of employees, while respecting a number of constraints:

- **Single shift per day.** Each employee works at most one shift per day.
- **Cover requirements**. The problem instance describes *per day* a minimum, maximum, or preferred number of shifts on a day.
- **Employee constraints**. Each employee's shift assignment must satisfy a specified set of labor rules. In addition, employee specific work agreements and requests must be taken into account as well.

We formulate the shift rostering problem as a mathematical program. Since the main decision is to assign on each day a shift type to an employee, it is natural to introduce the binary variables $x_{ikt}$ representing this decision:

$$x_{ikt} = \begin{cases} 1 & \text{if employee } i \text{ performs shift type } k \text{ on day } t \\ 0 & \text{otherwise} \end{cases} \tag{6.3.1}$$

The constraint that an employee is assigned to at most one shift on each day translates into:

$$\sum_{k \in K} x_{ikt} \leq 1 \quad i \in I, t \in T \tag{6.3.2}$$

Formulating all constraints appearing in the benchmark instances as linear constraints is tedious but straightforward. The details of these formulations are not described here. However, to provide the reader insight in these formulations, Section 6.4.2 discusses several constraints for the days off scheduling problem.

### 6.3.2 The benchmark instances

The Employee Scheduling Benchmark instances, see [105], were collected over a period of several years by a number of researchers investigating the shift rostering problem. They represent a diverse collection of challenging, real-world instances. Because they were drawn from many sources they vary in dimensions such as the number of staff, shifts types, skills and length of the planning period. More challengingly from a modeling and computational aspect however, they also vary in the number and types of constraints present in each instance. This is due to their real-world nature and the fact that each employer has different requirements often affected by national legislation, union or industry specific labor legislation. Each instance is often further complicated by the presence of employee specific contracts such as part-time or night shift workers, and employee specific requests for, e.g., days off. For detailed information, we refer the reader to [105] where full details are available on each specific instance.

At this moment there are 27 instances available, of which 25 are used in this research, see Table 6.1. Table 6.1 provides an overview of the instances' dimensions, i.e., the number of employees, the number of shift types, and the number of days in the scheduling horizon. In addition, Table 6.1 provides the best known objective function values (column 'best') for these instances. The best known solutions are known to be optimal in all cases except MER[1]. The best solutions, found by different researchers and different techniques, are also available at [105]. The instance Musa is not used in our research, as it contains only 1 shift type, and HED01 was not included because it uses conditional constraints, which we did not implement.

## 6.4 Solution approach

This section outlines our solution approach to the shift rostering problem. We solve the shift rostering using a decomposition that first solves a days off scheduling problem. In days off scheduling only days off and working days are determined, so employees are not assigned to specific shifts. The idea behind the decompositions is that the position of the stints, i.e., the consecutive days with work for an employee, is dominant for the shift rostering problem; once the working days are known, we hope that for each working day a shift type may be chosen such that the remaining shift rostering constraints are met.

[1] For MER a lower bound of 7079 was established.

Table 6.1: Test instances retrieved from the Employee Scheduling Benchmark Data Sets

| Instance | employees | shift types | days | best |
|---|---|---|---|---|
| Azaiez | 13 | 2 | 28 | 0 |
| BCDT-Sep | 20 | 4 | 30 | 100 |
| BCV-3.46.2 | 46 | 3 | 26 | 894 |
| BCV-4.13.1 | 13 | 4 | 29 | 10 |
| CHILD | 41 | 5 | 42 | 149 |
| ERMGH | 41 | 4 | 48 | 779 |
| ERRVH | 51 | 8 | 48 | 2001 |
| GPost | 8 | 2 | 28 | 5 |
| GPost-B | 8 | 2 | 28 | 3 |
| Ikegami-2Shift-1 | 28 | 3 | 30 | 0 |
| Ikegami-3Shift-1 | 25 | 4 | 30 | 2 |
| Ikegami-3Shift-1.1 | 25 | 4 | 30 | 3 |
| Ikegami-3Shift-1.2 | 25 | 4 | 30 | 3 |
| LLR | 27 | 3 | 7 | 301 |
| MER | 54 | 12 | 48 | 7081 |
| Millar-2Shift-1 | 8 | 2 | 14 | 0 |
| Millar-2Shift-1.1 | 8 | 2 | 14 | 0 |
| ORTEC01 | 16 | 4 | 31 | 270 |
| ORTEC02 | 16 | 5 | 31 | 270 |
| Ozkarahan | 14 | 2 | 7 | 0 |
| QMC-1 | 19 | 9 | 28 | 13 |
| QMC-2 | 19 | 3 | 28 | 29 |
| SINTEF | 24 | 5 | 21 | 0 |
| Valouxis-1 | 16 | 3 | 28 | 20 |
| WHPP | 30 | 3 | 14 | 5 |

In the decomposition, the days off schedule is used as input to assign employees to specific shifts. Of course, employees may only be assigned to shifts on working days in the days off schedule. First, Section 6.4.1 outlines how we formulate and solve the days off scheduling problem for the benchmark instances described in Section 6.3.2. Section 6.4.1 describes an extension to our decomposition that also includes night shifts in the days off scheduling problem.

In order to apply these decompositions, the shift rostering instances have to be reduced to days off scheduling instances, which is discussed in Section 6.4.2.

### 6.4.1 Relations between the models

#### Days off scheduling

The basic decision in the days off scheduling problem is to decide for each day in the schedule of an employee whether it is a working day. This is represented by the variables $y_{it}$:

$$y_{it} = \begin{cases} 1 & \text{if employee } i \text{ works on day } t \\ 0 & \text{if employee } i \text{ has a day off on day } t \end{cases}. \tag{6.4.1}$$

The basic relation with the shift rostering problem is:

$$\sum_{k \in K} x_{ikt} = y_{it} \quad i \in I, t \in T. \tag{6.4.2}$$

The second phase in the decomposition is exactly the original shift rostering problem with the additional relations of equation (6.4.2).

#### Night shift scheduling

Constraints on night shifts can lead to very specific constraints on the position of stints, see also [134]. An example is a required separation of at least 2 days between two stints if the first stint ends with a night shift. Hence, we also investigate an extension of the basic days off scheduling model in which the working day ($y_{it} = 1$) is specialized to working a selected shift type, which we call the *night shift* ($N$). For this, we introduce the binary variable $z_{it}$:

$$z_{it} = \begin{cases} 1 & \text{if employee } i \text{ works shift type } N \text{ on day } t \\ 0 & \text{otherwise} \end{cases} \tag{6.4.3}$$

In the days off scheduling problem, we add the following constraints:

$$z_{it} \leq y_{it} \quad i \in I, t \in T, \tag{6.4.4}$$

and for the shift rostering problem the next constraints are added:

$$x_{iNt} = z_{it} \quad i \in I, t \in T. \tag{6.4.5}$$

Equation (6.4.4) expresses that an employee can only work a night shift on working days. Equation (6.4.5) expresses that the choice for a night shift in the first phase should be respected in the second phase.

We refer to the decomposition approaches as:

**(On, Off):** Assigns working days and days off in the first phase.
**(Day, Night, Off):** Assigns working days, night shifts, and days off in the first phase.

### 6.4.2 Modeling the constraints

From a higher level, we can say that we reduce the original shift rostering problem to a shift rostering problem with 2 shift types (On, Off) and 3 shift types (Day, Night, Off), respectively. Hence, to solve the first and second phase of the decomposition similar mathematical programming formulations can be used, where the work schedules created in the second phase should obey the decisions of the first phase. Although the mathematical programming formulations are similar, the reduction in the first phase is not always straightforward. Our basic principle is to reformulate the constraints of the shift rostering instances to necessary constraints for the days off scheduling problem; in this way we are able to assess the consequences of the decomposition in a uniform way. The next subsections describe how different types of constraints are handled in the days off scheduling problems.

#### Requests and preassigned shifts

Employees may have preassigned shifts, requests for specific shift types on specific days, or requests for days off. Work requests or shift on requests result in working days in the days off scheduling problem. Shift *off* requests are ignored, since the employee might work another shift on the same day. For the implementation it is important to incorporate preassigned shifts and requests in the possible matches of the pattern constraints (see below). For example, if there is a preassigned shift we can evaluate whether this shift matches a certain pattern or not, even in the days off scheduling phase.

#### Cover requirements

Cover requirements express the minimum and maximum number of employees that are required to be available either per shift type or per time interval. In the latter case, shift types assigned to the employees should be such that these limits are respected.

In both cases the information can be more detailed expressing that the employees that are required to be available should have certain skills. However, employees having certain skills can count for more than one skill cover requirement

at the same time. To determine the minimum and maximum number of available employees that satisfy the cover requirements, we formulate a small mathematical program. These bounds express per day the bounds on the number of employees with certain skills that are required to be available. For the details, the reader is referred to [259].

### Pattern constraints

Pattern constraints express a wide variety of constraints for individual schedules. A pattern consists of a description of sequences of shifts that form a *match* for the pattern. Moreover, the pattern contains information for which period of the schedule the pattern needs to be considered, e.g., for the full scheduling period or only the periods starting on Saturday. We illustrate this by an example that is taken from the XML representation of the Azaiez instance:

```
<Match>
 <Max>
   <Count>4</Count>
   <Weight>10000</Weight>
 </Max>
 <Pattern>
    <StartDay>Saturday</StartDay>
    <Shift>$</Shift>
  </Pattern>
  <Pattern>
    <StartDay>Sunday</StartDay>
    <Shift>$</Shift>
  </Pattern>
<Match>
```

The first pattern is for Saturdays; each day that an employee has any shift ('$') on a Saturday we have a match. Similarly, the second pattern is for Sundays. Hence, the match counts the number of Saturdays and Sundays on which a shift starts. In the Azaiez instance, we see that an employee should do at most 4 (Count) weekend shifts; violation of this constraint results in a penalty of 10000 (Weight) per extra shift. Pattern constraints seem complicated at first, but they are able to model a wide variety of constraints that appear in practice. They can model constraints on, e.g., the maximum number of shifts per week, shift sequences, or the minimum number of consecutive shifts.

If a pattern constraint is specified for the any shift type ('$'), we can use it in the

days off schedule. Pattern constraints specified for a specific shift type cannot be incorporated in the construction of a days off schedule. In the case of (Day, Night, Off), we can also incorporate pattern constraints for the night shift as well. For example, it is common that there are constraints on the total number of night shifts and the length of night shift sequences. Moreover, night shifts are usually at the end of a stint, which can pose some restrictions on its position, for example that such stint should not end on Friday.

#### Workload requirements

Workload requirements describe the number of hours an employee should work in the planning period. Clearly, these are difficult to use in the days off schedule if different shift types have different lengths. Since we use only necessary conditions, the best we can do is to calculate upper and lower bounds on the working days, and add those conditions as constraints to the mathematical program. For example if employee $i$ has a maximum workload of 120 hours, and the shortest shift contains 7 hours of work, then we add the constraint

$$7 \cdot \sum_{t \in T} y_{it} \leq 120,$$

implying that the employee will have at most 17 working days.

In the second phase of the decomposition, we have full information, so that we can use the correct workload requirements.

## 6.5 Results

To evaluate our decomposition approach, we solve the mathematical programming formulations of the shift rostering instances in Table 6.1 using Cplex 12.2 with the time limit set to one hour on a Dell Optiplex 990 (64-bit, 3.4 GHz, 4 GB RAM, 8 cores). As the objective value is integer, we set the absolute gap to 0.999 without loosing the optimality guarantee. The two phases of the decomposition are also solved using Cplex, with the time limit set to 30 minutes for each phase.

Cplex finishes with 4 different statuses: status 101 and 102, indicating that an optimal solution was found, status 107, indicating that the time limit was reached without (guaranteed) optimal solution and status 109 indicating that Cplex ran out of memory[2].

[2] see http://www-01.ibm.com/support/docview.wss?uid=swg21399943

Table 6.2: Results on instances with 2 shift types

| | Direct | | (On, Off) | |
|---|---|---|---|---|
| Instance | cost | time (sec) | cost | % time saving |
| Azaiez | 0* | 7.4 | 6 | 76.7 |
| GPost | 5 | 117 | 4002 | 96.4 |
| GPost-B | 3* | 428 | 2002 | 92.5 |
| Millar-2Shift-1 | 0* | 0.2 | 0* | 66.2 |
| Millar-2Shift-1.1 | 0* | 0.2 | 0* | 61.6 |
| Ozkarahan | 0* | 0.2 | 800 | 90.4 |
| Ozkarahan (skills) | 0* | 0.1 | 0* | 87.2 |

Table 6.2 and Table 6.3 present results for instances with 2 shift types and instances with more than 2 shift types, respectively. To the instances with 2 shifts types only the (On, Off) approach is applied, since the (Day, Night, Off) approach would be the original shift rostering problem. For 15 instances the direct approach finds the optimal solution within one hour; these are the solution values that are marked with * in the second column. Moreover, 12 instances are solved within one minute, see third column.

First, we analyze the results of the (On, Off) decomposition approach to the instances with 2 shift types, and after that we analyze the results of our decomposition approaches to the instances with more than 2 shift types.

### Instances with 2 shift types

The fourth column in Table 6.2 gives the solution values of the (On, Off) decomposition, while column 5 states the time saving, i.e., 100% minus the percentage of the time needed to solve the decomposition divided by the time needed to solve the direct approach. For example, the time saving of 96.4% for the GPost instance indicates that the decomposition is solved in 4.2 seconds (3.6% of 117 seconds).

We observe that the decomposition is solved faster than the direct approach. However, except for the Millar instances the results are not competitive. The reasons are slightly different for each instance. The Azaiez instance requires separate stints for day shifts and night shifts, information that can not be incorporated in the (On, Off) decomposition. The GPost instances require that some specific stints end at specific days, for example (sub)stints of night shifts should end on specific days, see [134]. The Ozkarahan instance contains skills in the cover requirements,

which is modeled by certain forbidden patterns for employees. If we remodel this using skills for the employees then the decomposition finds the optimal solution, see the final row in Table 6.2.

These two aspects, the special role of night shifts and the complication with skills, reappear in the instances with more than two shift types. The special role of night shifts motivated us to extend the (On, Off) approach to the (Day, Night, Off) approach.

### Instances with more than 2 shift types

For the instances with more than 2 shift types, the results of both decomposition approaches are presented in Table 6.3. Again results are compared with the direct approach. The first 5 columns in Table 6.3 are the same as the five columns in Table 6.2. Column 6 indicates the shift that is chosen as 'night shift'. For two instances (BCV-3.46.2 and QMC-1) choosing N as the night shift did not give the best result. For these instances, we added extra rows for the 'night shift' choice that gave the best solution in the decomposition approach. Instance BCV-4.13.1 does not have a night shift at all, and choosing shift DH as the 'night shift' leads to the optimal solution. Column 7 and 8 give the result of the (Day, Night, Off) decomposition and the time saving, respectively.

For 9 instances, the direct approach finds an optimal solution, whereas the decomposition finds optimal solutions for 3 instances. Note that in the cases where the optimal solution is not in the direct approach, and the time listed is less than 3600 seconds, the process ran out of memory. The decomposition yields better solutions in 5 instances, and in 2 instances the solution quality is comparable (LLR and QMC-2). For 9 out of 19 instances the decomposition approach gives satisfactory solutions. The time saving is in most cases substantial: in 11 of the instances the saving is 90 percent or more. However, for 3 instances the decomposition takes more time and yields worse solutions. Instance MER is the biggest instance. On this instance none of the models give a satisfactory result. The direct model ends after 628 seconds (out of memory). The decomposition models run longer without going out of memory, explaining the fact that there is a negative time saving here.

In several instances (BCV-3.46.2, ERMGH, ERRVH, all Ikegami-3Sh instances, and MER) the (On, Off)-approach yields better solutions than the (Day, Night, Off)-decomposition. In Sections 6.5.1 and 6.5.2, the decomposition approaches are analyzed in detail.

Table 6.3: Results on instances with more than 2 shift types

| | Direct | | (On, Off) | | (Day, Night, Off) | | |
|---|---|---|---|---|---|---|---|
| Instance | cost | time (sec) | cost | % time saving | night shift | cost | % time saving |
| BCDT-Sep | 104539 | 3602 | 104229 | 99.9 | N | 2350 | 85.2 |
| BCV-3.46.2 | 894* | 535 | 1215 | 99.7 | N | 3430 | 82.8 |
| BCV-3.46.2 | – | – | – | – | L | 2374 | 99.8 |
| BCV-4.13.1 | 10* | 1.9 | 17 | 65.9 | DH | 10* | 54.2 |
| CHILD | 149* | 47 | 5066 | 86.9 | N | 3847 | 89.9 |
| ERMGH | 779* | 1.7 | 929 | 27.9 | N | 1116 | -119.0 |
| ERRVH | 2204 | 385 | 12832 | 97.9 | N | 22796 | 91.3 |
| Ikegami-2Sh-1 | 0* | 27 | 4316 | 97.7 | N | 0* | 23.1 |
| Ikegami-3Sh-1 | 110 | 3601 | 807 | 90.8 | N | 3884 | 99.8 |
| Ikegami-3Sh-1.1 | 6 | 3601 | 994 | 94.8 | N | 2728 | 90.8 |
| Ikegami-3Sh-1.2 | 23 | 3602 | 994 | 90.8 | N | 1960 | 85.4 |
| LLR | 301* | 1.2 | 312 | 80.2 | N | 308 | 92.6 |
| MER | 56561 | 628 | 122022 | -187.0 | N | 142133 | -16.8 |
| ORTEC01 | 3527 | 1365 | 2270 | 94.1 | N | 280 | 92.4 |
| ORTEC02 | 2836 | 2385 | 1275 | 96.5 | N | 275 | 96.5 |
| QMC-1 | 13* | 3.2 | 29046 | 77.7 | N | 21070 | 38.8 |
| QMC-1 | – | – | – | – | O | 177 | 86.9 |
| QMC-2 | 29* | 0.4 | 1045 | 30.8 | N | 33 | -92.1 |
| SINTEF | 0* | 1.6 | 12202 | 77.9 | N | 17 | 77.9 |
| Valouxis-1 | 140 | 1461 | 20* | 64.5 | N | 20* | 93.9 |
| WHPP | 3008 | 155 | 16000 | 99.5 | N | 3001 | 77.7 |

### 6.5.1 Assessment of the decomposition approaches

In this section we investigate the quality of solutions produced by the decompositions approaches. Basically there are two orthogonal explanations for poor quality solutions of the decomposition approach:

- **The essence of the instance could be caught by days off scheduling.** Hence some information, hidden in a combination of factors, was not taken into account in *the implementation of* the days off scheduling problem. A way to improve this is to reformulate certain constraints, or reformulate the instance, such that the solutions we are interested in, still correspond to low costs. This is instance-dependent, and difficult to automate.

Table 6.4: Analysis of instances

| Instance | aggr | shift | cover | night | seq | run |
|---|---|---|---|---|---|---|
| BCDT-Sep | | V | | × | | × |
| BCV-3.46.2 | × | DH | | | × | × |
| BCV-4.13.1 | | DH | | | | |
| CHILD | × | ET | | | × | |
| ERMGH | × | | | | | × |
| ERRVH | | | | × | | |
| Ikegami-3Shift-1 | × | O | × | | × | × |
| Ikegami-3Shift-1.1 | × | O | × | | × | × |
| Ikegami-3Shift-1.2 | × | O | × | | × | × |
| LLR | | | | | | |
| MER | | | | | | × |
| ORTEC01 | | | | | | × |
| ORTEC02 | | V | | | | × |
| Ozkarahan | | | × | | | |
| QMC-1 | | O | | | | |
| QMC-2 | | | | | × | |
| SINTEF | | | | | × | |
| Valouxis-1 | × | | | | | |
| WHPP | × | | | | | × |

- **The essence of the instance is lost in the days off schedule.** The instance may contain aspects that can not be handled in the days off schedule and are of decisive importance. The night shift was such an aspect which led to the (Day, Night, Off)-approach. Other aspects are shifts with specific requirements, cover requirements with skills, and strong preferences for shift sequences.

Our findings are summarized in Table 6.4. The contents of the columns 'aggr', 'shift', 'cover', 'night', 'seq' and 'run' are addressed in Section 6.5.2.

### 6.5.2 Detailed analysis of the results

#### Aggregate constraints

The '×' in column 'aggr' of Table 6.4 indicates that a set of constraints in the instance was taken together to an equivalent instance with one aggregated con-

straint. Many instances contain a number of constraints expressing that the night shift can not be followed by shift types E, or D, or L, etc. The aggregated constraint expresses that the night shift can not be following by the *shift group* consisting of the types E, D, L, etc. In our tests we used these aggregated versions which are also available at [105].

### Special shifts

Several special shifts appear in the instances, which usually complicate the days off scheduling. There are two classes of these shifts:

- **Absence shifts**. These shifts will not contribute to the cover requirements, but can count as work (in the Ikegami instances and ORTEC02) or not (QMC-1 and BCDT-Sep). The cover requirements on day level (see Section 6.4.2) can be adjusted for these shifts. However, there might exist patterns for all shifts, except the absence shifts, for example: "A night can only be followed by a night shift or a special shift". As a consequence we lose these patterns in the first phase of days off scheduling, which for the example will lead to scheduling night shifts in the middle of stints.
- **Skilled shifts**. The BCV instances and CHILD contain shifts that should preferably be assigned to specific employees; information that can not be included in the first phase of days off scheduling. Consequently we create stints for these specific employees that do not match these shifts. In BCV-4.13.1 we see that the DH shift type gives the best result in the (Day, Night, Off)-approach; remember that BCV-4.13.1 does not contain a night shift.

### Cover requirements

Section 6.4.2 described how cover requirements in combination with skills or time units are handled. Unfortunately this is not always the best approach. In particular, the Ikegami instances contain complicated skill cover requirements and in combination with the strong preferences on shift sequences the results are unsatisfactory in these instances. Indeed the cover models we solve to calculate the number of required employees might lead to the same group of employees for all days, which unfortunately will not be feasible to schedule. On the other hand, we have introduced more skill details in the cover requirements of the instance Ozkarahan, leading to a better result on that instance, see Section 6.5. These covers are related to skills for disjoint groups of employees, which explains why it works very well here.

### The night shift (reprise)

One of the main conclusions of this study is that night shifts need special attention. However, it is sometimes difficult to incorporate all effects of night shifts. In particular, several instances allow the night shift to be followed by some specific other shift (a leave shift, a late shift or another night shift), see also 6.5.2. In such cases it might be better to ignore the night shift, and use the (On, Off)-approach instead, such as in BCV-3.46.2, ERRVH and the Ikegami-3Shifts cases.

### Shift sequences

Usually there are patterns describing which shift sequences are preferable. In some cases the violation of these preferences can not be avoided in the second phase of the decomposition, for example (again) in the Ikegami-3Shifts cases.

### Run aborted

As highlighted there are some instances where the (Day, Night, Off)-approach stopped before reaching optimality. In these cases the run was aborted, due to the time restriction or because of running out of memory. For the instance MER none of the approaches finds a reasonable solution. Due to the memory requirements the direct approach ends after 628 seconds, explaining why the decompositions take longer. In some other instances (ORTEC01 and ORTEC02) the solver almost reached optimality, or at least better solutions than the direct approach (BCDT-Sep and WHPP).

## 6.6 Conclusions

We studied the effects of solving the shift rostering problem by decomposition approaches. The first approach, (On, Off), is a two-phase decomposition approach that first assigns employees to working days and days off, and secondly assigns employees to shifts on the working days. The second approach, (Day, Night, Off), additionally includes night shifts in the first phase of the decomposition. Both phases of the decomposition approaches are solved using ILP, and evaluated using public shift rostering benchmark instances. The results of the decomposition approaches are compared against solving an ILP formulation of the shift rostering problem directly.

We see that the decomposition has a large impact on the solving time: for most instances the solving time is reduced by more than 80% or 90%. However,

the solutions do not give such a clear answer. One can roughly say that in 1/3 of the instances the (On, Off)-approach gives good results, in 1/3 of the instances the (Day, Night, Off)-approach gives good results, and in 1/3 of the instances this decomposition does not give competitive results. Especially for the (Day, Night, Off)-approach we have to pay attention to problem specific aspects.

We discussed several improvements for our implementations. Significant improvements could be obtained by aggregating scheduling constraints and by alternative modeling of skill-related constraints. This remodeling enabled us to effectively consider these constraints in the first phase of the decompositions resulting in improved overall results. Moreover, in our approach we currently use necessary constraints but including additional, stronger, constraints may also improve the results.

Since the first phase usually has multiple optimal solutions, we analyzed the effect different solutions of the first phase had on the outcome of the second phase. This had no significant positive or negative effect on the originally obtained solutions, which implies that the presented results are a representative sample. In view of the observation that Cplex solves 15 instances to optimality within one hour, it is in some situations easier to generate the complete work schedule without decomposing the problem, and use this to publish the days off schedule, as suggested in [25, 42].

In general, we conclude that applying the decomposition approaches significantly improves the required computation time, but they should be implemented with care. For example choosing the 'best' night shift, in the (Day, Night, Off)-approach requires some attention. However, by tailoring the decomposition approach to the instance (class) at hand, we expect our decomposition approach to be successful for many shift rostering instances.

CHAPTER 7

# Shift Rostering Using Decomposition: Assign Weekend Shifts First

## 7.1 Introduction

For many people most social activities are scheduled during the weekend, which makes working in the weekend less attractive for them. Various industries, such as healthcare and security services, offer services on a 24/7 basis, which implies that certain employees need to work during weekends. Scheduling personnel during weekends is challenging, since most employers like to consider the preferences of each individual on the one hand, but on the other hand also staffing demands need to be covered. Furthermore, work schedules have to respect labor legislation, and shifts have to be distributed in an equitable way among employees, which complicates the matter even further.

The existing literature recognizes the importance of finding good or fair assignments of weekend shifts to employees; some papers propose (soft) constraints to cope with preferences related to weekends shifts, see, e.g., [71]. Still, to our knowledge, the literature does not consider methods specifically designed to construct weekend shifts rosters. Of course, weekend work schedules can be created using a general shift rostering method. However, this approach ignores problem specific information. Therefore, we design a weekend shift rostering algorithm that is tailored to use this information.

The research in this chapter was motivated by practical experiences of customers of ORTEC. Many planners, when assigning shifts manually, decompose the shift rostering process into two or more steps. They first assign weekend shifts, and secondly they assign the weekday shifts. This shows that many planners consider weekend-related shift rostering preferences as more important than other preferences. We use the same assumption in this research, and apply the same decomposition to assign the weekend and weekday shifts.

Our main contribution is offering a decomposition approach that first assigns weekend shifts before the weekday shifts. Our main interest lies in analyzing whether such an approach is useful, also from an optimization point of view. First, we formally introduce the Weekend Rostering Problem (WRP) in Section 7.2. A problem specific heuristic designed to solve the Weekend Rostering Problem (WRP) is described in Section 7.3. After the heuristic created a weekend shifts schedule the weekday shifts are assigned using a commercially implemented algorithm, see [66], [214]. In Section 7.4 we show the first-phase heuristic to be effective both on artificially generated instances and real-life instances. In addition, Section 7.4 discusses the effects of our two-phase approach on the weekend work schedule, as well as on the complete roster. Section 7.5 presents conclusions and discussion.

Table 7.1: Constraints related to weekend shift rostering found in the literature

| Constraint | Literature references |
|---|---|
| Max # shifts during weekends (soft) | [9, 10, 68, 118, 249] |
| Max # shifts during weekends (hard) | [9, 10, 20, 110, 118, 249] |
| Work complete weekends (hard) | [35, 43, 66, 118, 141, 165, 167, 170, 184, 217, 224, 254, 266] |
| Work complete weekends (soft) | [19, 34, 38, 43, 48, 50, 56, 56, 61, 63, 64, 66, 66, 67, 68, 69, 72, 73, 74, 87, 131, 134, 141, 144, 160, 162, 166, 171, 175, 185, 187, 188, 195, 197, 218, 219, 226, 238, 248, 250, 253, 261] |
| Min/max # weekends in # weeks (hard) | [18, 19, 20, 34, 35, 43, 49, 63, 63, 66, 67, 72, 73, 74, 75, 110, 122, 134, 155, 161, 162, 162, 165, 166, 167, 168, 170, 171, 175, 184, 185, 188, 188, 189, 202, 217, 219, 224, 226, 261, 267, 268, 270] |
| Min/max # weekends in # weeks (soft) | [43, 50, 56, 64, 67, 69, 169, 187, 197, 248, 253, 265]. |
| Identical shifts in weekend (hard) | [67, 141, 184] |
| Identical shifts in weekend (soft) | [19, 50, 56, 63, 64, 67, 141, 175, 197, 253] |
| Fair distribution of weekend shifts in various scheduling periods | [225] |
| No night shift before free weekend (soft) | [50, 56, 64, 67, 187, 197] |

## 7.2 Problem assumptions and formulation

In this section we define the Weekend Rostering Problem (WRP), discuss the assumptions of the WRP and address the constraints that are considered in the WRP. In the WRP we are only interested in assigning weekend shifts. In this chapter, weekend shifts are the shifts on the two 'main' days of the weekend, possibly extended with shifts on the evening preceding the weekend and the morning after the weekend. For ease of discussion, we let Saturday and Sunday be the 'main' weekend days in the research performed in this chapter. Of course, depending on the application one might just as well choose any combination of two consecutive days as the weekend days.

As indicated in Section 7.1, we consider the WRP as a first phase of the shift rostering problem. In general, shift rostering problems strive to assign shifts to employees. Shifts are time periods specifying working time as opposed to rest time. Furthermore, demand constraints define the number of times each shift should be assigned. Demand constraints are the outcome of the *shift scheduling* phase, which precedes the *shift rostering* phase. Shift scheduling designs shifts to efficiently cover staffing levels, which in turn are the outcome of the *staffing* phase. Based on the predicted workload, staffing levels are determined, indicating how many employees need to be present on any specific time and day. For a detailed description of these scheduling processes, see Chapter 3.

The objective of the WRP is to assign as many weekend shifts as possible, while satisfying a set of hard constraints, and subsequently minimizing the total penalty cost associated with violations of a set of soft constraints. We will now discuss the hard and soft constraints that are used in the WRP.

In practice, many hard constraints are implied on rosters by labor legislation and labor agreements. The following four hard constraints are relevant for the WRP, and considered in most shift rostering literature; the literature review in [255] illustrates for papers from the period 2004-2012 which constraints are considered. This reveals that from that period 59 papers consider constraints related to weekend planning, however none of them proposes a special purpose weekend planner algorithm.

In [255], literature is classified according to the shift rostering constraints being considered. For constraints related to weekend shift scheduling we have extended this classification and list the corresponding papers in Table 7.1. For each constraint is indicated whether it is considered to be hard or soft. Compared to [255], we additionally include the literature from the period before 2004, and we include three additional constraints related to weekend shift schedules. These constraints are: the maximum number of weekend shifts during the scheduling horizon, a fair

distribution of weekend shifts over various scheduling horizons, and the constraint that no night shift should be scheduled before a free weekend.

For the weekend rostering problem as studied in this chapter, the following hard constraints are included:

**Constraint C1** (Availability). *Employees may not work if they are unavailable. Employees are unavailable if they are on leave, for example, or assigned to another shift.*

**Constraint C2** (Skills). *Shift may only be assigned to employees that have all skills required to work the shift.*

**Constraint C3** (Rest between shifts). *Between two subsequent shifts there should be a rest period of a given minimum length.*

The next constraint implies restrictions on the number of working weekends in a given period:

**Constraint C4** (Weekends in $p$ weeks). *An employee may only work $k$ out of $p$ weekends, where both $k$ and $p$ may be employee dependent.*

In the literature, constraints on the maximum number of consecutive working weekends $m$ are often separately mentioned. Note however, that such a constraint can also be ensured via hard constraint C4: working at most $m$ weekends consecutively is the same as working at most $m$ out of $m + 1$ weekends. Therefore, the constraints are grouped together in Table 7.1.

Next to these hard constraints, the WRP also considers seven soft constraints. This list of soft constraints is not supposed to be a complete list of soft constraints that can be implied on the assignment of weekend shifts.

The first soft constraint concerns the scheduling of 'complete' off weekends. Working only on one of the weekend days is unattractive for two reasons. First, there are labor rules regarding the number of times an employee works in weekends during a specified period. Working either the complete weekend, or not at all, enables organizations to assign more weekend shifts to employees in total. Second, employee preferences are either to work the complete weekend or to have the complete weekend off. We encounter this in the real-life instances of Section 7.4.1 on which we test the WRP algorithm designed in Section 7.3.3. Therefore, the following soft constraint is used to assign complete weekends off:

**Soft constraint S1** (Complete weekends off). *An employee should preferably either work the complete weekend, or have the complete weekend off.*

The next two soft constraints concern organizational preferences.

**Soft constraint S2.** *Weekend shifts should be equitably distributed among employees, i.e., proportional to the contract hours of the employees.*

**Soft constraint S3.** *Weekend shift types, such as early, late and night, should be equitably distributed among employees*

The next four constraints concern employee preferences.

**Soft constraint S4.** *Work or do not work during a specific weekend.*

**Soft constraint S5.** *Work or do not work a specific shift on a specific day in a specific weekend.*

**Soft constraint S6.** *Work or do not work two specific shifts consecutively.*

**Soft constraint S7.** *Work at most k weekends in p weeks.*

Note that soft constraint S7 is the equivalent of hard constraint C4.

Each of the soft constraints has a (user definable) penalty cost associated with the violation of the constraint. The penalty cost may differ per employee, and constraints can be added to a problem instance multiple times with different parameter and penalty values. If a shift assignment violates multiple soft constraints penalty costs are incurred for every violation.

Now that we have stated, described, and motivated the constraints used in the WRP, the next section introduces a model to solve the WRP.

## 7.3 Solution approach and modeling

### 7.3.1 Introduction

To assign the weekday *and* weekend shifts, i.e., to solve the shift rostering problem, we apply a decomposition approach. The first phase, the Weekend Rostering Problem (WRP), assigns the weekend shifts, which are, as indicated in Section 7.2, the shifts on Saturday and Sunday, and optionally Friday evening and Monday morning. The second phase completes the work schedule by assigning the weekday shifts. For the second phase we use a commercial algorithm, see Section 7.3.4. With the WRP, we introduce a new decomposition approach for shift rostering problems. In Section 7.3.2, we discuss the use of other decomposition approaches to solve shift rostering problems. In Section 7.3.3, we explain how we solve the WRP.

### 7.3.2 Decomposition in shift rostering

This section discusses some decomposition approaches to solve the shift rostering problem.

In Chapter 6, we extensively discuss days off scheduling. Days off scheduling first decides when employees should work, and next, in a reduced solution space, shifts are assigned. For more details on our days off decomposition method and the related literature, the reader is referred to Chapter 6. Another decomposition example is the use of shift patterns. Shift patterns specify a sequence of shifts that are worked consecutively. In the methods proposed in [5], [115], and [160] shift patterns must be created manually, after which they are assigned by scheduling algorithms. The algorithms of [56], and [68] first generate these shift patterns and then assign them to employees. The idea is shift patterns already comply with many constraints and preferences on consecutive shifts, which makes the assignment easier and more efficient.

### 7.3.3 The weekend rostering problem

The WRP is solved using a heuristic solution approach. A solution is constructed with a greedy 3-step heuristic. Step 1, described in Section 7.3.3, creates 'weekend shift combinations': combinations of Saturday and Sunday shifts for one particular weekend. These are possibly extended with Friday night and Monday morning shifts, dependent on whether these shifts are considered weekend shifts in the particular problem instance. Step 2, described in Section 7.3.3, assigns the weekend shift combinations to employees, this second step is based on a preliminary research in [262]. The first two steps of our heuristic are illustrated with an example in Section 7.3.3. Finally, in Step 3, described in Section 7.3.3, we apply local search to improve the assignment of Step 2.

#### Weekend shift combinations

We create shift combinations per weekend by solving a minimum-cost transportation problem. The idea is to choose shift combinations, such that as many shifts as possible can be assigned. Let $P$ and $Q$ be sets of nodes corresponding to the different types of shifts on Saturday and Sunday, respectively. For $p \in P$, let $s_p$ be the number of Saturday shifts of type $p$ that must be covered, and, for $q \in Q$, let $d_q$ be the number of Sunday shifts of type $q$ that must be covered. Let $I$ be the set of employees, and let $I^{pq} \subseteq I$ denote the subset of employees that is allowed to work combination $(p, q)$. An employee is not allowed to work combination $(p, q)$ if

working it would violate one of the hard constraints of Section 7.2. Let $c_{pq}$ denote the transportation cost from node $p$ to node $q$. To define $c_{pq}$ we distinguish three cases: either 'many', 'few' or no employees are allowed to work combination $(p, q)$. First, if no employees are allowed to work combination $(p, q)$, we do not want it to be selected. Second, if 'few' employees are allowed to work combination $(p, q)$, the combination may be selected only when it is necessary, i.e., when otherwise there is no solution to the transportation problem. Third, if 'many' employees are allowed to work combination $(p, q)$ we want to relate $c_{pq}$ to the penalty cost associated with violating the soft constraints.

If no employee is allowed to work combination $(p, q)$ we set $c_{pq} = \infty$, hence:

$$c_{pq} = \infty \text{ if } I^{pq} = \emptyset. \tag{7.3.1}$$

However, when $|I^{pq}| > 0$, we want to let $c_{pq}$ depend on the size of $I^{pq}$. If only 'few' employees are allowed to work combination $(p, q)$ we want $c_{pq}$ to be high. We say that 'few' employees are allowed to work combination $(p, q)$ if:

$$\frac{|I^{pq}|}{|I|} < \frac{\min\{s_p, d_q\}}{\max\{\sum_{p\in P} s_p, \sum_{q\in Q} d_q\}}. \tag{7.3.2}$$

Here, the numerator in the fraction on the right hand side denotes the maximum number of times combination $(p, q)$ can be assigned, the denominator denotes the minimum number of employees that need to work during the particular weekend. If the right hand side in (7.3.2) is larger than the left hand side, which denotes the fraction of employees that are allowed to work combination $(p, q)$, we say that 'few' employees are allowed to work combination $(p, q)$. In this case we define $c_{pq}$ as $M \cdot |I \setminus I^{pq}|$, where $M$ is a 'big' number. Hence:

$$c_{pq} = M \cdot |I \setminus I^{pq}| \text{ if } \frac{|I^{pq}|}{|I|} < \frac{\min\{s_p, d_q\}}{\max\{\sum_{p\in P} s_p, \sum_{q\in Q} d_q\}}. \tag{7.3.3}$$

If 'many' employees are allowed to work shift combination $(p, q)$, i.e., if (7.3.2) does not hold, we relate $c_{pq}$ to the penalty costs associated with the soft constraints. Let $S^i_{pq}$ denote the penalty cost associated with the soft constraints of assigning combination $(p, q)$ to employee $i$. Then:

$$c_{pq} = \sum_{i\in I} S^i_{pq} \text{ if } \frac{|I^{pq}|}{|I|} \geq \frac{\min\{s_p, d_q\}}{\max\{\sum_{p\in P} s_p, \sum_{q\in Q} d_q\}}. \tag{7.3.4}$$

Summarizing we have:

$$c_{pq} = \begin{cases} \sum_{i \in I} S^i_{pq} & \text{if } \frac{|I^{pq}|}{|I|} \geq \frac{\min\{s_p, d_q\}}{\max\{\sum_{p \in P} s_p, \sum_{q \in Q} d_q\}} \\ M \cdot |I \setminus I^{pq}| & \text{if } \frac{|I^{pq}|}{|I|} < \frac{\min\{s_p, d_q\}}{\max\{\sum_{p \in P} s_p, \sum_{q \in Q} d_q\}} \\ \infty & \text{if } I^{pq} = \emptyset. \end{cases} \tag{7.3.5}$$

The transportation problem defined by the described parameters is solved by the network flow formulation discussed in [241].

The solution $x_{pq}$ of the transportation problem indicates for each shift combination $(p, q)$ the number of times it should be assigned. These shift combinations are extended with the remaining Friday shifts by solving additional transport problems. To this end, the Friday shifts are considered as the set of $P$ shifts whereas the combinations of Saturday and Sunday shifts are considered as the set of $Q$ shifts. This problem is then initialized and solved as described above. For Monday shifts we apply an analogous procedure.

#### Assign shift combinations

Now that we have the set of shift combinations per weekend that we want to assign, we describe how these shifts are assigned to employees. The assignment algorithm is based on a preliminary study performed in [262].

The heuristic first selects a shift combination (part 1) and then an employee (part 2). The shift combination is selected using the following scheme:

1a. Select the 'least flexible' shift combination: the shift combination $(p, q)$ for which the ratio between "the number of times combination $(p, q)$ should be assigned" ($x_{pq}$) and "the number of employees allowed to work combination $(p, q)$" ($|I^{pq}|$) is the largest. This shift combination is presumed to be the hardest to assign to an employee. In case of a tie, go to 1b.

1b. Of the remaining combinations, select the one for which $x_{pq}$ is the smallest. These are presumed to be the hardest to assign. In case of a tie, go to 1c.

1c. Of the remaining combinations, select the combination that is earliest in time. Since some of the constraints consider assignments in previous weekends, shifts that are earlier in time are presumed to be harder to assign. In case of a tie, go to 1d.

1d. Randomly select from the remaining combinations.

When a shift combination is selected, an employee is selected via the following scheme:

2a. Select the 'least busy' employee: the employee working the fewest number of weekends relative to his contract hours. This way, all employees will work (approximately) the same number of weekends. In case of a tie, go to 2b.

2b. Select the employee for which $S^i_{pq}$ is the smallest. In case of a tie, go to 2c.

2c. Select the 'least flexible' employee: the employee that has the least number of remaining shift combinations that the employee is allowed to work. In case of a tie, go to 2d.

2d. Randomly select from the remaining employees.

#### Illustrative example

The example in this section consists of two parts. The first part illustrates the creation of weekend shift combinations, and the second part illustrates the assignment of the weekend shift combinations.

#### Create weekend shift combinations for one weekend

On both Saturday and Sunday we have one $A$ and one $B$ shift , so $I = J = \{A, B\}$. We have 4 employees, so $I = \{1, 2, 3, 4\}$. Employee 1 is not allowed to work shift $B$ at all, employee 3 is not allowed to work shift $B$ on Saturday. We let $S_{AA} = S_{BB} = 1$ and $S_{AB} = S_{BA} = 2$. We omit the superscript of the penalty costs, since in this example they are the same for all employees.

First, note that:

$$\frac{\min\{s_p, d_q\}}{\max\{\sum_{p\in P} s_p, \sum_{q\in Q} d_q\}} = \frac{1}{2} \quad \text{for } p \in P, q \in Q. \tag{7.3.6}$$

Second, $|I^{AA}| = 4$, $|I^{AB}| = 3$, $|I^{BA}| = 2$, $|I^{BB}| = 2$, and $|I| = 4$. Hence, for $(A, A)$ we have:

$$\frac{|I^{AA}|}{|I|} = \frac{4}{4} = 1 > \frac{\min\{s_A, d_A\}}{\max\{\sum_{p\in P} s_p, \sum_{q\in Q} d_q\}} = \frac{1}{2}. \tag{7.3.7}$$

So, for combination $(A, A)$ we say that 'many' employees are available, hence:

$$c_{AA} = S_{AA} = 1. \tag{7.3.8}$$

Similar, for $(A, B)$ we find that 'many' employees are available, hence $c_{AB} = S_{AB} = 2$, and for $(B, A)$ and $(B, B)$ we find that 'few' employees are available, hence $c_{BA} = M \cdot |I \setminus I^{BA}| = M \cdot 2$, and $c_{BB} = M \cdot |I \setminus I^{BB}| = M \cdot 2$.

The solution of the transportation problem is then $x_{AA} = x_{BB} = 1$, $x_{AB} = x_{BA} = 0$. So, we create one shift combination $(A, A)$ and one shift combination $(B, B)$, but no shift combinations $(B, A)$ or $(A, B)$.

**Assign weekend shift combinations**

After shift combinations are created they are assigned to employees in the assignment phase. To illustrate this: assume we have a scheduling horizon containing two weekends. For the first weekend, the shift combinations as generated in the previous example are used, see also Table 7.2. The shift combinations for the second weekend and the corresponding available employees are listed in Table 7.2 as well.

Table 7.2: Assigning shift combinations

| Weekend | Combination | $x_{pq}$ | $I^{pq}$ | $S_{pq}$ |
|---|---|---|---|---|
| 1 | $(A, A)$ | 1 | $\{1, 2, 3, 4\}$ | 1 |
| 1 | $(B, B)$ | 1 | $\{2, 4\}$ | 1 |
| 2 | $(B, A)$ | 1 | $\{1, 4\}$ | 2 |
| 2 | $(A, B)$ | 2 | $\{1, 2, 3, 4\}$ | 2 |

The $x_{pq}/|I^{pq}|$ ratios of the shift combinations are $\frac{1}{4}$, $\frac{1}{2}$, $\frac{1}{2}$, and $\frac{2}{4}$, respectively, hence shift combinations 2, 3, and 4, are the 'least flexible' (Step 1a). We have $x_{BB} = 1, x_{BA} = 1$, and $x_{AB} = 2$, so $(B, B)$ and $(B, A)$ have to be assigned the least number of times (Step 1b). Since $(B, B)$ is a shift combination of the first weekend, and $(B, A)$ of the second, $(B, B)$ is earliest in time (Step 1c), so $(B, B)$ is to be assigned.

Since there are no shift combinations assigned yet, employees 2 and 4 (the employees available for $(B, B)$) are equally busy (Step 2a). For both these employees $S^i_{BB} = 1$, so Step 2b gives no conclusion on which employee to select. However, since employee 2 has only 3 remaining shift combinations (including $(B, B)$) and employee 4 has 4, Step 2c assigns shift combination $(B, B)$ to employee 2.

The other shift combinations are assigned to employees in an analogous way. $(B, A)$ is assigned randomly to 1 or 4, say it is assigned to 1. Then $(A, B)$ is assigned randomly to 3 or 4, say it is assigned to 3. Next, the other $(A, B)$ combination is assigned to 4, and, finally, $(A, A)$ is assigned randomly to 1, 3 or

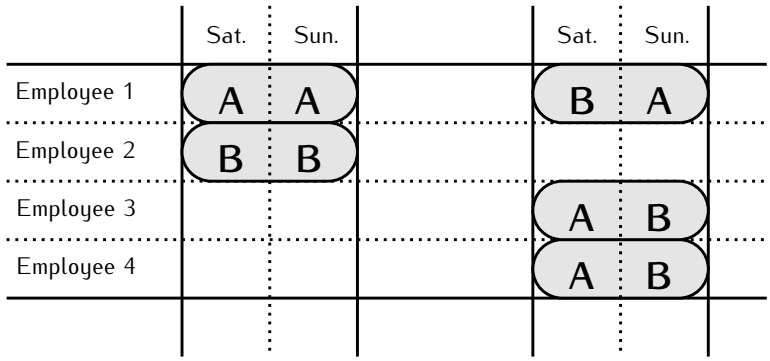



Figure 7.1: Example - Resulting work schedule

4, say it is assigned to 1. We then get the work schedule as in Figure 7.1.

### Local search

The initial solution, created via the heuristics described in Section 7.3.3 and Section 7.3.3, has a cost implied by violations of the soft constraints. We try to improve this solution using two local search techniques: subsequently a cyclic exchange method and a 2-opt search are applied.

Strictly speaking, our cyclic exchange method is a Very Large-scale Neighborhood Search (VLNS) as introduced in [3], but with a limit on the cycle length. Our cyclic exchange method applies *cyclical* swaps of shift combinations, comparable with the *cyclic exchange neighborhood* applied in [3]. Possible swaps are: reassigning a shift combination from one employee to another employee, swapping two shift combinations, or cyclically swapping three up to five shift combinations. The first two options are visualized in Figure 7.2. Swaps can be applied both to shift combinations in the same weekend and to shift combinations in different weekends. The algorithm consists of finding resource constrained negative cycles in an improvement graph, see [213].

Each iteration of 2-opt search calculates per shift whether the total penalty cost improves if this shift is swapped with another shift. That is, if employee 1 is assigned to shift A, and employee 2 to shift B, 2-opt calculates whether the total penalty cost decreases when employee 1 is assigned to shift B, and employee 2 to shift A, see Figure 7.3. The swap is accepted if the total penalty cost decreases. Note that 2-opt also considers swapping not assigned shifts with assigned shifts.

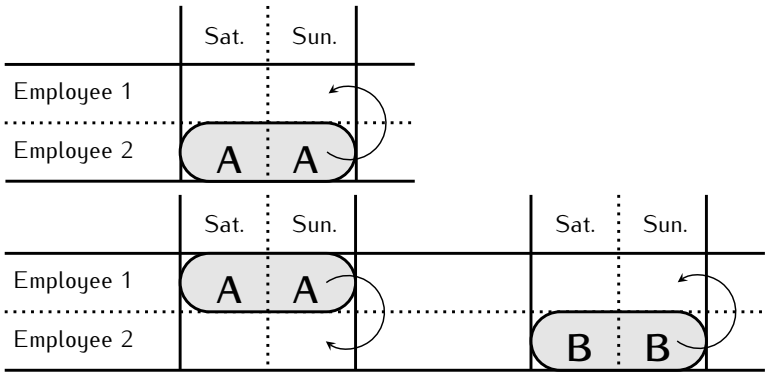



Figure 7.2: Local search: cyclic exchanges. Assigning a shift combination to another employee (top), and swapping two shift combinations (bottom)

The 2-opt search is executed after the cyclic exchange method. The cyclic exchange method optimizes the assignment of the working weekends whereas the 2-opt search optimizes the shift assignment within the working weekends. Since the shift assignment strongly depends on preferences of individual employees, for, e.g., specific shift sequences or shift preferences, it is logical to first apply the cyclic exchange method before the 2-opt search.

The local search techniques are applied as in Algorithm 7.1:

**Algorithm 7.1** Local search scheme

1. Apply cyclic exchanges, until no improvements found.
2. Apply 2-opt, until no improvements found.

### 7.3.4 Weekday shift assignment

The first phase considered the weekend shifts and assigned these as good as possible. In the second phase we will assign the weekday shifts, while keeping the weekend shifts assigned as decided in the first phase. To assign the weekday shifts, we use a commercial algorithm (CA) that is implemented in ORTEC's workforce scheduling software [201]. Mathematical details of this algorithm are found in [66,

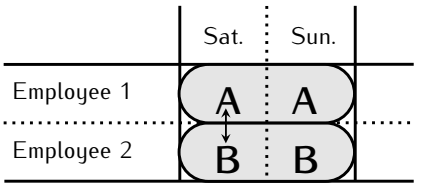



Figure 7.3: Local search: 2-opt search

214]. In short, this algorithm first employs a hybrid heuristic ordering method to construct multiple initial shift schedules. After that, these initial schedules are improved by a genetic algorithm. Next, the best schedule found by the genetic algorithm is improved using a variable neighborhood search that applies various neighborhood operators, such as the 2-opt operator that is also included in our WRP approach. The schedule produced by the variable neighborhood search is improved using an iterated local search, that iteratively unassigns all shifts in a part of the schedule and then uses the variable neighborhood search of the previous stage to reassign these shifts. The iterated local search is continued until a set computation time expires, or if the algorithm concludes that an optimal solution is reached.

## 7.4 Results

This section discusses experimental results. First, Section 7.4.1 describes two case studies in which we tested the weekend rostering algorithm. Next, Section 7.4.2 evaluates the local search components of the algorithm, as described in Section 7.3.3, and illustrates that they have a significant impact on the produced results. After that, Section 7.4.3 benchmarks the weekend rostering algorithm against the commercial algorithm described in Section 7.3.4. Section 7.4.4 performs the same benchmark on public nurse rostering instances from the PATAT 2010 Nurse Rostering Competition [146].

### 7.4.1 Case studies

Two case studies are used in our experiments. These are provided by a Belgian Police department and a Dutch care provider, which are described in Section 7.4.1 and Section 7.4.1, respectively.

### Case 1: Belgian Police

The first case is provided by a Belgian Police department. This department consists of 60 employees, and it uses two-month rosters. In each roster, approximately 2000 shifts of the following four types need to be assigned: morning shifts (7h-13h), afternoon shifts (13h-22h), night shifts (22h-7h), and days off. Shifts are assigned on both week and weekend days.

For the weekend work schedules, all constraints defined in Section 7.2 are implied. For the complete roster, additionally hard constraints are implied on the number of consecutive shifts and the number of consecutive night shifts, and soft constraints are implied on the minimum and maximum number of shifts of a given type, the minimum total working hours during the scheduling horizon, and on combinations of different shift types. Finally, the main objective is to assign as many shifts as possible.

Before the Belgian Police department started to use the WRP+CA approach, as proposed in this chapter, they manually assigned the weekend shifts before constructing the complete roster. Note that WRP+CA uses exactly the same decomposition.

### Case 2: Dutch care provider

The second case is provided by a Dutch care provider for visually impaired people that offers intramural and extramural care. We test the weekend planner on two departments called C1 and C2. These have 12 and 42 employees, respectively, and both use monthly rosters. C1 has two shift types: morning shifts (8h-15h) and afternoon shifts (15h-22h), and shifts must be scheduled on week and weekend days. In every month, approximately 135 shifts need to be assigned. C2 has three shift types: morning shifts (8h-15h), day shifts (10h-14h), and afternoon shifts (15h-22h), and again shifts must be assigned on week and weekend days. For C2 approximately 450 shifts must be assigned every month.

For the weekend work schedules, all hard constraints defined in Section 7.2 are implied on the rosters, however from the soft constraints only soft constraint S1 is implied for both departments. For department C2, additionally soft constraint S6 regarding forbidden combinations of shifts is defined. For the complete roster, soft constraints are implied on minimal and maximal shift series lengths, minimal number of consecutive days off, and the minimal and maximal number of shifts (of a specific type) in a one week and two week period. In addition, preferably stand-alone shifts should not be scheduled.

### 7.4.2 Preliminary results

This section performs a preliminary analysis of the cyclic exchange and the 2-opt heuristics, as described in Section 7.3.3, that are used in our weekend rostering heuristic. We apply the weekend rostering heuristic to the 12 scheduling instances provided by the case studies, 6 instances (P1-P6) were provided by the Belgian Police and the other 6 instances, 3 for each department (C1.1-C2.3), were provided by the Dutch care provider. For all instances, we registered computation times and solution quality. The solution quality is expressed as a penalty cost implied by violations of soft constraints expressed in the scheduling instances.

Each instance is solved twice. Both times the first step is to construct an initial weekend shift schedule by creating and assigning weekend shift combinations as described in Section 7.3.3 and Section 7.3.3, respectively. After that, in the first run, we subsequently execute the cyclic exchange and the 2-opt method, and in the second run, we only apply the 2-opt method. We do not analyze the effect of first applying the 2-opt and subsequently the cyclic exchange method, since, as outlined in Section 7.3.3, it makes no sense to run the local search method in that order. Table 7.3 summarizes results.

Table 7.3: Analysis of local search methods

| | Exch. & 2-opt | | | | | Only 2-opt | | |
|---|---|---|---|---|---|---|---|---|
| | Time (sec) | | | Penalty | | Time (sec) | | Penalty |
| Instance | Constr. | Exch. | 2-opt | Exch. | 2-opt | Constr. | 2-opt | 2-opt |
| P1 | 714 | 3440 | 26 | -11.7% | -18.1% | 727 | 33 | -16.4% |
| P2 | 1954 | 12582 | 61 | -34.4% | -3.2% | 1891 | 80 | -4.9% |
| P3 | 2042 | 10744 | 61 | -32.1% | -2.0% | 2080 | 63 | -2.9% |
| P4 | 2227 | 12616 | 56 | -40.0% | -0.8% | 2142 | 77 | -11.2% |
| P5 | 1954 | 12582 | 61 | -34.4% | -3.2% | 2088 | 67 | -4.0% |
| P6 | 1793 | 10857 | 45 | -34.6% | -5.8% | 1851 | 51 | -3.4% |
| C1.1 | 24 | 9 | 0 | -47.7% | 0.0% | 25 | 0 | 0.0% |
| C1.2 | 18 | 8 | 0 | -96.3% | 0.0% | 17 | 0 | 0.0% |
| C1.3 | 19 | 10 | 0 | -65.7% | 0.0% | 18 | 0 | 0.0% |
| C2.1 | 40 | 21 | 0 | 0.0% | 0.0% | 36 | 0 | -0.1% |
| C2.2 | 28 | 9 | 0 | -83.4% | 0.0% | 26 | 0 | 0.0% |
| C2.3 | 25 | 9 | 0 | 0.0% | 0.0% | 59 | 0 | 0.0% |

Although not directly relevant for our analysis, we observe from Table 7.3 that the computation times of the Police instances are much larger than those of the

Care instances. A major factor in this is that these instances have more employees and a longer scheduling horizon, however there were also some database issues that we were unable to resolve. Although this resulted in longer computation times, see Table 7.3, this had no effect on the relative computation times of the individual planning stages.

From the fifth column of Table 7.3, we observe that in general the exchange method significantly reduces the penalty function value of both the Police and the Care instances. In addition, for the Police instances, the 2-opt method also significantly reduces the penalty function value, independent if the exchange method is used or not (sixth and ninth column of Table 7.3). For the Care instances, we observe that the 2-opt method only affects the result for instance C2.1, since only for department C2 there is a soft constraint defined for which the 2-opt method has a positive effect.

The computation times of the exchange and 2-opt methods are listed in the third, fourth, and eighth column of Table 7.3. Note that, for the Care instances, the computation times for the 2-opt method are only a couple of milliseconds, hence appearing as zeros when rounded to seconds. Considering the computation times of the exchange and 2-opt methods, we observe that the exchange method requires more time than the 2-opt method. However, the exchange method achieves a larger reduction in penalty value. Therefore, we choose to include both the exchange method and the 2-opt method in the experiments of Section 7.4.3 and Section 7.4.4.

### 7.4.3 Case study results

This section presents computational results on the case studies. We start with a description of the experimental setup and after that we discuss the results.

#### Experimental setup

We want to study the effect weekend rostering has on the complete roster, i.e., the work schedule of the entire week. We compare two approaches, called WRP+CA and CA. WRP+CA first assigns the weekend shifts using the weekend rostering heuristic. Then, it 'fixes' these shifts, and assigns the weekday shift using the commercial algorithm, i.e., the CA considers the weekend shift schedule as given in the WRP+CA approach. This is done to resemble how in practice schedules are created by human planners. The CA approach assigns all shifts (weekday and weekend) using the same commercial algorithm. Both approaches use the same set of hard and soft constraints.

Using this set-up, we can study the effect of first generating the weekend shift schedule, and fixing this, before the weekday shift schedule is generated, which is one of the motivations for this research.

#### Experimental Results

This section presents our experimental results for the cases presented in Section 7.4.1 and Section 7.4.1. For both approaches a time limit of 1 hour was set. Table 7.4 summarizes the results.

In Table 7.4 we observe for all instances of the Belgian Police case that both the fraction of complete working weekends (column 'On'), and the fraction of complete off weekends (column 'Off') is larger when the WRP+CA approach is applied. This implies that, for all instances, the fraction of half weekends (column 'Half') is smaller if WRP+CA is applied. We say that an employee works a half weekend if the employee works either on Saturday or on Sunday, but not on both days. In fact, on average WRP+CA assigns 57.8% weekends On, 40.2% weekends Off, and 2.1% half weekends. CA assigns on average 44.4% weekends Complete On, 26.7% weekends Complete Off, and 28.9% half weekends. Furthermore, the number of not assigned weekend shifts (seventh column) and the total number of not assigned shifts (eighth column) are approximately the same for WRP+CA and CA. Hence, WRP+CA outperforms CA on the weekend shift assignment, while the number of assigned shifts is approximately equal.

For department C1 of the Dutch Care Provider we observe no difference between both approaches. We believe this is caused by the relatively small size of this department. For C2 we observe that WRP+CA outperforms CA on the weekend shift assignment. On average WRP+CA assigns 27.8% weekends On, 67.3% weekends Off, and 5.0% half weekends. CA assigns on average 24.2% weekends Complete On, 64.3% weekends Complete Off, and 11.6% half weekends. Again the number of weekend shifts not assigned and the total number of shifts not assigned are about equal. For case C2.2, note that CA assigns only one shift more than WRP+CA does.

### 7.4.4 Benchmark results

In addition to the practical case studies, we also study the effect of our two approaches, WRP+CA and CA (see Section 7.4.3, on public benchmark instances. We used the benchmark instances of the Nurse Rostering Competition of the PATAT 2010 Conference, see [146]. The instances are available online at [208]. The instances have a scheduling horizon of 28 days, contain employee requests for shifts

Table 7.4: Experimental results

| Instance | Approach | Weekend (%) On | Off | Half | Not assigned shifts Weekend | Total |
|---|---|---|---|---|---|---|
| P1 | WRP+CA | 73.8 | 23.8 | 2.5 | 0.3% | 1.4% |
| | CA | 66.9 | 14.8 | 18.3 | 0.9% | 2.0% |
| P2 | WRP+CA | 60.7 | 36.9 | 2.4 | 0.3% | 0.6% |
| | CA | 48.5 | 24.6 | 26.9 | 0.3% | 0.4% |
| P3 | WRP+CA | 52.8 | 46.0 | 1.3 | 0.1% | 0.1% |
| | CA | 38.8 | 27.3 | 33.9 | 0.2% | 0.3% |
| P4 | WRP+CA | 54.2 | 40.9 | 4.9 | 0.0% | 0.3% |
| | CA | 38.8 | 35.8 | 25.4 | 0.0% | 0.0% |
| P5 | WRP+CA | 52.4 | 46.5 | 1.1 | 0.2% | 0.2% |
| | CA | 34.8 | 29.6 | 35.6 | 0.3% | 0.5% |
| P6 | WRP+CA | 52.7 | 47.1 | 0.2 | 0.4% | 1.9% |
| | CA | 38.6 | 27.9 | 33.5 | 0.4% | 0.9% |
| C1.1 | WRP+CA | 33.3 | 66.7 | 0.0 | 0.0% | 0.0% |
| | CA | 33.3 | 66.7 | 0.0 | 0.0% | 0.0% |
| C1.2 | WRP+CA | 33.3 | 66.7 | 0.0 | 0.0% | 0.0% |
| | CA | 33.3 | 66.7 | 0.0 | 0.0% | 0.0% |
| C1.3 | WRP+CA | 26.7 | 66.7 | 6.7 | 0.0% | 0.0% |
| | CA | 26.7 | 66.7 | 6.7 | 0.0% | 0.0% |
| C2.1 | WRP+CA | 27.9 | 67.4 | 4.7 | 0.0% | 0.0% |
| | CA | 23.7 | 63.3 | 13.0 | 0.0% | 0.0% |
| C2.2 | WRP+CA | 28.6 | 66.7 | 4.8 | 0.0% | 0.0% |
| | CA | 25.0 | 63.1 | 11.9 | 0.0% | 0.0% |
| C2.3 | WRP+CA | 26.8 | 67.7 | 5.5 | 2.1% | 5.8% |
| | CA | 23.8 | 66.5 | 9.8 | 2.0% | 7.0% |

and days off, and numerous scheduling constraints, such as a minimum and maximum number of shifts, the maximum number of consecutive working weekends, and the complete weekends constraint. The instances are divided in three categories: 'sprint', 'medium', and 'long', having around 10, 30, and 50 employees, respectively.

The weekend shift scheduling constraints that are implied on these instances are:

- Minimum and maximum number of consecutive weekends

- Working complete weekends
- Work identical shifts during weekends
- No night shift before free weekends

where a complete weekend is defined as working Saturday and Sunday, and instance-dependent also Friday. Most constraints in these instances are soft constraints. The exceptions are that employees can work at most one shift per day, and if an instance includes multiple skill categories, a hard constraint may imply that employees must be skilled sufficiently.

From our experimental results, we excluded instances with a three day weekend definition (Friday, Saturday, Sunday), since this is not included in the CA. The instances that are excluded are the 'Long hidden', 'Long hint' and 'Long late' instances. The CA also does not include the pattern constraint 'no D-E-D'. This constraint penalizes the consecutive assignment of a 'Day' shift, an 'Early' shift, and again a 'Day' shift. We have ignored this constraint in the modeling of the instances, since it is contained in almost all instances.

The WRP definition, see Section 7.2, does not include the no night shift before free weekend and the minimum number of consecutive weekends constraint. However, we did include instances containing these constraints, to keep a sufficient number of test instances.

We used calculation times of 1 hour, 30 minutes, 10 minutes for the 'long', 'medium', and 'sprint' instances, respectively. Results are summarized in Table 7.5. For each instance category, e.g., 'Sprint hidden' or 'Medium late', we included the average results in the table.

For the benchmark instances, we looked at the same performance indicators as for the instances from practice. For all instances all shifts were assigned by both approaches. Hence, the number of not assigned shifts is not reported in Table 7.5.

The columns 'On', 'Off', and 'Half', again report the number of complete on, off and half weekends, respectively. For the benchmark instances, the weekend is considered to include Friday, Saturday and Sunday if the instance includes the constraint that it is not allowed to schedule a shift on Friday before an off day on both Saturday and Sunday. For the other instances, the weekend is defined as Saturday and Sunday. For all instances, we see that the WRP+CA approach schedules more complete weekends on, more complete weekends off, and less half weekends. On average, the WRP+CA approach schedules 6.3% more complete weekends on, 5.1% more complete weekends off, and 11.4% half weekends less.

The sixth, seventh and eighth column of Table 7.5 list violations of constraints related to the weekend schedule. The column 'Same shifts' indicates the number

Table 7.5: Results benchmark instances PATAT 2010 NRP competition

| Instance | Approach | Weekend (%) On | Off | Half | Same shifts | Cons. wknds max 2 | max 3 | Penalty |
|---|---|---|---|---|---|---|---|---|
| Sprint | WRP+CA | 40.0 | 40.0 | 20.0 | 0.00 | 0.00 | 0.00 | 9.9% |
| | CA | 38.0 | 38.8 | 23.3 | 0.60 | 3.40 | 0.40 | 2.0% |
| Sprint | WRP+CA | 46.3 | 42.9 | 10.8 | 0.00 | 3.33 | 0.33 | 42.7% |
| hidden | CA | 35.4 | 32.5 | 32.1 | 1.67 | 9.50 | 3.50 | 25.8% |
| Sprint | WRP+CA | 40.0 | 46.7 | 13.3 | 0.00 | 0.00 | 0.00 | – |
| hint | CA | 34.2 | 41.7 | 24.2 | 0.67 | 4.67 | 0.00 | – |
| Sprint | WRP+CA | 38.9 | 49.7 | 11.4 | 0.44 | 1.44 | 1.33 | 69.0% |
| late | CA | 35.6 | 46.9 | 17.5 | 2.00 | 2.67 | 0.22 | 26.6% |
| Medium | WRP+CA | 51.6 | 48.4 | 0.0 | 0.00 | 0.00 | 0.00 | 12.9% |
| | CA | 50.0 | 46.8 | 3.2 | 4.40 | 0.00 | 0.00 | 6.8% |
| Medium | WRP+CA | 44.0 | 41.3 | 14.7 | 0.00 | 4.40 | 0.00 | 171.2% |
| hidden | CA | 31.0 | 31.0 | 38.0 | 10.00 | 17.60 | 2.80 | 71.3% |
| Medium | WRP+CA | 36.7 | 43.3 | 20.0 | 0.00 | 2.00 | 0.00 | – |
| hint | CA | 31.4 | 40.3 | 28.3 | 9.33 | 12.67 | 1.67 | – |
| Medium | WRP+CA | 37.3 | 50.7 | 12.0 | 0.00 | 0.80 | 0.00 | 343.6% |
| late | CA | 26.3 | 42.2 | 31.5 | 13.60 | 15.40 | 3.60 | 159.5% |
| Long | WRP+CA | 40.8 | 39.6 | 19.6 | 0.00 | 0.00 | 0.00 | 15.3% |
| | CA | 32.8 | 33.3 | 34.0 | 16.00 | 15.60 | 1.00 | 10.6% |
| Average | WRP+CA | 41.7 | 44.6 | 13.7 | 0.08 | 1.41 | 0.30 | 82.0% |
| | CA | 35.5 | 39.5 | 25.0 | 5.57 | 8.91 | 1.50 | 36.8% |

of violations of the soft constraint that employees prefer to work identical shift types during the weekend. The columns 'Cons. wknds Max 2' and 'Cons. wknds Max 3' indicate the number of violations of the soft constraints that employees prefer to work at most 2 or at most 3 consecutive weekends, respectively. From Table 7.5 it is clear that WRP+CA outperforms CA on the weekend scheduling constraints.

The final column of Table 7.5 lists a comparison of the objective values found by the respective approaches compared to the best known solutions as retrieved from [208]. The objective values include the violations of both the shift scheduling constraints related to the weekend shift schedule as well as constraints related to the overall schedule. As expected the overall quality of the CA approach dominates the WRP+CA approach. This is expected since, in the WRP+CA approach,

the weekend shift schedule that is optimized in the first stage is considered as given for the second stage that assigns the weekday shifts. This implies less room for optimization in the second stage. For the 'Sprint hint' and the 'Medium hint' instances no percentages are listed since for these instances the best known solutions cannot be retrieved from [208].

Note that in comparing the objective values to the best known solutions, we included the penalties incurred for violating the no night shift before free weekend and the minimum number of consecutive weekends constraint. Since the definition of the WRP does not include these constraints, the results of the WRP+CA approach might be improved by including these constraints in the algorithm.

The solution quality of the WRP+CA approach is on average 19% worse than the solution quality of the CA approach, if we express the solution quality of the WRP+CA approach as a percentage of the solution quality of the CA approach, which is the 'cost' one has to pay for the achieved improved in the weekend shift schedule.

## 7.5 Conclusions

In this chapter, we introduce the Weekend Rostering Problem (WRP), a rostering problem focused on weekend shift assignment. It is motivated by our experience that employee preferences predominantly focus on the weekends, due to many social activities happening during the weekend. Despite of its practical relevance, the WRP is underexposed in both the literature and decision support software.

In this chapter, we introduce a two-phase heuristic to solve the WRP. The first phase assigns weekend shifts, the second phase assigns the remaining, weekday, shifts. For the first phase we design a special-purpose heuristic, whereas for the second phase we use an algorithm that is implemented in commercial software. This decomposition approach is inspired by our experience of how work schedules are created in practice. An algorithm specifically designed to create weekend work schedules supports the natural planning process: planners often start assigning the 'hard' shifts, such as the weekend shifts. The decomposition algorithm allows the planner to adjust the automatically generated weekend work schedules, before the rest of the work schedule is constructed. When the complete work schedule is already generated, it is harder for the planner to improve the weekend work schedule manually, since in a complete work schedule the reassigning of weekend shifts is constrained by shifts assigned to weekdays [160].

The main motivation for this research is to analyze whether such a decomposition is also useful from a computational point of view. We encourage research

to improve the algorithm proposed in this chapter, since it is the first algorithm we know of that solves the WRP. Furthermore, we encourage further research into alternative decomposition approaches, such as decomposition on night shifts, days off or skills.

Experiment results show that the heuristic designed for the weekend shift assignment performs well on instances from practice as well as a set of public benchmark instances. When we use this heuristic as a first phase in the shift rostering problem, results obtained via this decomposition approach look promising. Our decomposition outperforms the commercial algorithm on all of our weekend work schedule quality defining performance indicators. This proves that our decomposition is valuable when weekend related performance indicators are key determinants of the quality of rosters.

We incorporated the proposed algorithm in the commercial workforce scheduling software of ORTEC and the algorithm is currently used to create rosters for the Belgian Police department case discussed in this chapter.

CHAPTER 8

# Shift Rostering from Staffing Levels: a Branch-and-Price Approach

## 8.1 Introduction

In many personnel scheduling applications, shift scheduling and shift rostering are separate planning decisions. In Chapter 3, we have defined these planning decisions. In short, shift scheduling, defines a set of shifts that are supposed to cover the staffing levels as efficiently as possible, and shift rostering, assigns personnel to the created shifts.

The main disadvantage with separating shift scheduling and shift rostering is that personnel preferences cannot, or only limitedly, be taken into account in shift scheduling. This implies that it might be hard or even impossible to account for these personnel preferences in assigning personnel to shifts in shift rostering. As stressed in Chapter 2 it is important to carefully consider personnel preferences in personnel planning and scheduling. In order to better account for personnel preferences in shift scheduling and shift rostering, we propose a method that integrates these planning decisions, as is graphically illustrated by Figure 8.1.

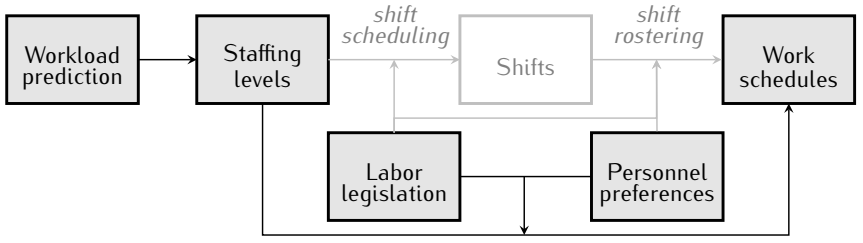



Figure 8.1: Integrated shift scheduling and rostering: a one-step approach.

This integrated approach for shift scheduling and rostering creates work schedules directly from staffing levels, and, while doing this, it takes both labor legislation and personnel preferences into account.

The main objective of this research is to develop a method that creates work schedules directly from staffing levels. The method assigns employees to staffing levels (time slots and skills to work on), and, while doing this, it takes employee specific constraints into account. Employee specific constraints consist of employee specific labor rules and employee preferences. Note that such a method may be applied to other personnel scheduling problems as well. For example, it also applies to single day workstation planning on a radiology department where positions have to be staffed during opening hours, and, at the same time, personnel must have rest periods during their shifts.

This chapter is structured as follows. Section 8.2 discusses the related literature. In Section 8.3, the modeling approach is outlined. Section 8.4 discusses the experimental results. Section 8.5 presents conclusions and discussion.

## 8.2 Related literature

There exists some literature that integrates shift scheduling and shift rostering. The research in [163] outlines a method that creates work schedules using 'shift templates'. With this method, only shifts from a predefined set of (template) shifts may be used in creating work schedules. However, this method does not keep track of employee preferences, because the cost of assigning employees to shifts is not employee specific. In [115], a straightforward extension to the model of [163] is outlined that makes the assignment of employees to shifts employee specific. The method proposed in [135] starts by generating a set of shifts and tries to assign employees to them. If necessary, a tabu search method is applied to redesign the shifts such that they better match the employee preferences. A mathematical program that integrates shift scheduling and shift rostering for a single skill setting is proposed in [58]. They include quite a number of hard constraints considering minimum and maximum shift length, breaks and overtime, and some soft constraints.

In [77], a method is introduced that solves the shift scheduling and shift rostering phase iteratively thereby creating work schedules directly from staffing levels. Given a very large set of shifts and corresponding weights, the shift scheduling phase selects a subset of shifts that covers a given demand and minimizes the total weight. Subsequently, shift rostering assigns the employees to these shifts. If it turns out that the resulting work schedules do not match the employee preferences, the weights of all shifts are updated, and the shift rostering is resolved.

The updating is based on changes in lower bounds of the optimal solution. By increasing weights of 'bad' shifts, these shifts are forced out of the optimal solution. This updating and resolving of the shift scheduling and shift rostering problem is repeated until a specified stopping criterion is met. Unfortunately, the method of [77] cannot directly be applied to our problem, since they apply it to a task scheduling problem, with the objective of minimizing the total time needed to work a complete schedule. This is different from our problem where the time horizon is fixed and all staffing levels need to be matched within the given time frame. However, the major problem when iteratively solving the shift scheduling and shift rostering phase is that it is unclear what information should be provided to the shift scheduling phase when shift rostering cannot find a solution. Moreover, when the shift rostering phase is unable to find a solution, it is also not clear whether this is inherent from the set of created shifts or whether there is no solution at all.

In Section 8.3, we present a Branch-and-Price approach to create work schedules directly from staffing levels. For a detailed introduction to the concept of Branch-and-Price the reader is referred to [143]. Branch-and-Price techniques have been applied to healthcare related optimization problems by others before us. For example, the generation of cyclic schedules using Branch-and-Price under some general labor restrictions with nurse preferences is discussed in [217]. In [46], a Branch-and-Price method is proposed for a model that integrates shift rostering and surgery scheduling. A Branch-and-Price approach for a multi-skill shift rostering problem is considered in [182], which differs from our model in that [182] uses a given set of shifts, whereas our implementation creates shifts based on staffing levels.

## 8.3 Modeling

This chapter addresses the creation of work schedules directly from staffing levels. The problem studied in the remainder of this chapter considers multiple skills, but restricts each employee to have a single period of working time, without interruptions, that is possibly preceded or succeeded by rest time. This single period of working time is subject to employee specific constraints on the minimum and maximum duration of this period. Hence, it is assumed that employees prefer their working time to be without pre-emptions. Of course, this is a restrictive assumption which is used within our modeling. However, as we will argue, it is not too hard to extend this model to incorporate additional scheduling constraints. Furthermore, this class of problems is interesting from a mathematical point of view, since it is NP-complete. This is a consequence of the fact that we are dealing with staffing

levels for multiple skills, see [257].

We study two ways of modeling this problem. The first one is integer linear programming (ILP), in which shifts are modeled implicitly, i.e., we assign a binary decision variable to every combination of employee, time slot and skill. By including constraints on these decision variables we ensure that valid shifts are created. For the details of this ILP model the reader is referred to [257]. A major drawback of the implicit ILP model is that it has a weak LP relaxation. One way to overcome this drawback is to reformulate the ILP model. This reformulated model has a stronger LP relaxation, but involves a larger number of variables.

For the reformulation, let $J$ be the set of skills, indexed by $j$, and $T$ the set of time slots, indexed by $t$, (of, e.g., an hour each). Let $d_{jt}$ denote the staffing level for skill $j$ at time slot $t$. Furthermore, let $I$ be the set of employees, and, for each employee $i \in I$, let the set $K_i$ denote the set of shifts for employee $i$. These shifts respect the shift constraints and shift preferences of employee $i$. The generation of these shift sets $K_i$ is part of our solution approach. The cost corresponding to this shift is denoted by $c_k$, which may be used to represent the shift preferences of employee $i$. In our implementation, we define $c_k$ as the total number of time slots that employee $i$ is expected to work according to shift $k$. Now, let the binary variable $x_k$ equal 1 if shift $k \in K_i$ is performed by employee $i$ and 0 otherwise. Let $a_{jt}^k$ denote whether skill $j$ is provided during time slot $t$ in shift $k$ ($a_{jt}^k = 1$), or not ($a_{jt}^k = 0$). Then the formulation we propose is:

$$\min \quad \sum_{i \in I} \sum_{k \in K_i} c_k x_k \tag{8.3.1a}$$

$$\text{s.t.} \quad \sum_{i \in I} \sum_{k \in K_i} a_{jt}^k x_k \geq d_{jt} \quad j \in J, t \in T \tag{8.3.1b}$$

$$\sum_{k \in K_i} x_k \leq 1 \quad i \in I \tag{8.3.1c}$$

$$x_k \in \{0, 1\} \quad i \in I, k \in K_i \tag{8.3.1d}$$

The objective function (8.3.1a) minimizes the total prefernce cost. The set of constraints (8.3.1b) imply that the staffing levels are met, and constraints (8.3.1b) ensure that an employee works at most one shift. Constraints (8.3.1d) restrict the $x_k$ to be binary.

Note that model (8.3.1) every shift is explicitly represented by a variable. This formulation requires a large number of variables. For realistic problems, the number of variables is likely to be too large to be practical. Since the model also has integrality constraints on the decision variables, we choose to apply a

Branch-and-Bound procedure combined with column generation in the nodes of the branching tree. This approach is commonly referred to as Branch-and-Price (B&P). The column generation component is an implicit generation of shifts $k$ and shift sets $K_i$.

Problem (8.3.1) is used as the master problem in the (B&P) method. Note that, the master problem is sometimes referred to as the *restricted* master problem, since it contains only a subset of the variables. An easy way to initialize the master problem is to set all $K_i$ such that they contain exactly one 'super' shift $k$ for which $a_{jt}^k = d_{jt}$. If one of the employees works such a shift, the total demand is covered. Note that such a 'super' shift is not valid in practice, since we assume an employee cannot work on two skills at the same time. However, by setting the $c_k = M$ for these initial shifts, where $M$ is a significantly large number, we ensure that after generating a number of additional (valid) shifts, these initial (invalid) shifts disappear from the solution. Hence, all information about the validity of shifts is assumed to be incorporated in the column generation.

To generate additional columns, we have to solve the column generation problem, which is commonly referred to as the pricing problem. To solve the pricing problem, it is necessary to calculate the reduced costs. However, to calculate the reduced costs, the dual of the master problem must be solved first. In doing so, we relax the previously binary variables $x_k$ to be non-negative and real valued. This makes the dual problem tractable, and because of Constraints (8.3.1c) the feasible region of the master problem remains unchanged. Let the dual variables corresponding to Constraints (8.3.1b) and (8.3.1c) be denoted by $\pi_{jt}$ and $v_i$, respectively. Then, the dual of the relaxed master problem is:

$$\max \quad \sum_{j\in J}\sum_{t\in T} d_{jt}\pi_{jt} + \sum_{i\in I} v_i \tag{8.3.2a}$$

$$\text{s.t.} \quad \sum_{j\in J}\sum_{t\in T} a_{jt}^k \pi_{jt} + v_i \leq c_k \quad i \in I, k \in K_i \tag{8.3.2b}$$

$$\pi_{jt} \geq 0 \quad j \in J, t \in T \tag{8.3.2c}$$

$$v_i \leq 0 \quad i \in I. \tag{8.3.2d}$$

The reduced cost of a shift $k'$ of employee $i$ is then:

$$c_{k'} - \sum_{j\in J}\sum_{t\in T} a_{jt}^{k'} \pi_{jt}^* - v_i^*, \tag{8.3.3}$$

where $\pi_{jt}^*$ and $v_i^*$ denote the optimal values of $\pi_{jt}$ and $v_i$, respectively. Next, we have to determine the shift with the lowest reduced cost, since we have a

minimization problem. To determine this shift, it is possible, given the size of our problem, to efficiently enumerate all possible shifts, calculate their reduced costs, and select the one with the lowest reduced cost. Let $J_i$ denote the set of skills employee $i$ has, then we define $\bar{\pi}_t$ as follows:

$$\bar{\pi}_t = \max_{j \in J_i} \{\pi^*_{jt}\} \quad t \in T. \tag{8.3.4}$$

Define a working pattern $w$ as a sequence of 0s and 1s indicating in which time slots an employee is working (1) or not working (0). Note that, the cost of a shift with this working pattern equals $\sum_{t \in T} w_t$. Next, we enumerate all working patterns $w$ that employee $i$ is allowed to work. These are the working patterns that consist of a single period of working time that satisfy the constraints on the minimum and maximum duration of a working time period. After this, for each working pattern $w$ the reduced cost $\sum_{t \in T} w_t \pi_t$ is calculated and thus we can find a working pattern $w^*$ for employee $i$ with the lowest reduced cost. Given this working pattern $w^*$ and $\bar{\pi}$, we determine for every period the skill the employee is working. As a result, the corresponding employee shift is determined. If it holds that:

$$\sum_{t \in T} w^*_t \bar{\pi}_t > \sum_{t \in T} w^*_t - v^*_i, \tag{8.3.5}$$

the reduced cost of this shift is negative, and it is added to the master problem.

Let the minimum and maximum number of consecutive time slots employee $i$ needs to work (when called to work) be denoted by $l_i^{\min}$ and $l_i^{\max}$, respectively. Then the number of working patterns is bounded by:

$$\sum_{s=|T|-l_i^{\max}+1}^{|T|-l_i^{\min}+1} s \leq \sum_{s=1,\ldots,|T|} s = \frac{1}{2}|T|(|T|+1) \leq |T|^2. \tag{8.3.6}$$

Determining $\max_{j \in J}\{\pi^*_{jt}\}$ for $t \in T$ has time complexity $\mathcal{O}(|J| \cdot |T|)$, enumerating all possible working patterns has time complexity $\mathcal{O}(|T|^2)$, and for every working pattern we have to calculate the reduced cost, which gives a total time complexity of the pricing problem (for each employee) of $\mathcal{O}(|J| \cdot |T| + |T|^3)$, which is polynomial, since $|T|$ is constant.

Employees are evaluated according to index, i.e., we first solve the pricing problem for employee 1 and determine whether a shift with negative reduced cost exists for employee 1. If it exists, it is added to the master problem, and the master problem is resolved. Otherwise, employee 2 is selected and the procedure repeats.

After generating a, possibly large, number of columns, the algorithm arrives at a point where no columns with negative reduced cost are found. If the solution

obtained to the linear relaxation of the master problem is integer, we have found an optimal solution. However, when the solution is fractional, branching needs to be applied. The branching needs to result in an integer solution. It is important that branching decisions can be incorporated in the pricing problem, i.e., it must be possible to adjust the pricing problem in such a way that the generated columns respect the branching decisions. Otherwise the pricing problem could generate invalid columns, which might slow down the solution process considerably.

In our model branching is, for each employee, applied to the time slots where an employee starts and stops working. All shifts $k \in K_i$ that violate the branching condition are removed from the current branch of the branching tree. Note that the 'super' shifts are never removed to ensure that the master problem has a solution. The branching decisions are easily incorporated in the pricing problem; we simply exclude shifts that have working periods before the start period or after the stop period from the enumeration. The branching scheme leads to an integer solution, as it should. However, even if the branching tree has specified a start and stop period for each employee we might not end up with a feasible solution. Example 8.1 illustrates this.

**Example 8.1.** *Full-branching, i.e., specifying a start and stop period for each employee, does not guarantee a feasible integer solution.*

Given a rostering problem with 2 employees, 2 time slots and 2 skills. Demand $(d_{jt})$ is given by:

$$D = \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}. \tag{8.3.7}$$

Assume the generated shifts are $k_1 = (1,1)$ and $k_2 = (2,2)$ for employee 1; and $k_3 = (1,2)$ and $k_4 = (2,1)$ for employee 2, where the first number indicates the skill that is provided during time slot 1, and the second number indicates the skill that is provided during time slot 2. Furthermore, assume the cost of the shifts equals the number of working periods in it, hence $c_{k_1} = c_{k_2} = c_{k_3} = c_{k_4} = 2$. For both employees 1 and 2 assume time slot 1 is fixed as start period and time slot 2 is fixed as stop period. The optimal solution then equals $x_{k_1} = x_{k_2} = x_{k_3} = x_{k_4} = \frac{1}{2}$. This is the unique optimal solution, and there does not exist a feasible integer solution among the generated shifts. Furthermore, there are no shifts with negative reduced cost, since this solution is optimal given the demand $D$. Since the start and stop periods are fixed for both employees, the example shows that after full branching it is possible that we end up with a fractional solution.

Fortunately, an integer solution can be easily constructed if start and stop periods are fixed for all employees. We do this by formulating and solving a number

of $b$-matching problems. A $b$-matching problem is similar to regular matching problems except that in a $b$-matching problem for every vertex $v$ and a given nonnegative integer $b_v$ related to that vertex, the matching has to contain exactly $b_v$ edges that are connected to $v$ [232].

For every time slot $t$, define a $b$-matching problem as follows. First, create vertices for all employees that are allowed to work during $t$. Furthermore, for every skill $j$, where $d_{jt} > 0$, create a demand vertex $v$ with $b_v = d_{jt}$. Now, connect an employee vertex to a demand vertex if the employee has the corresponding skill. After each of these $b$-matching problems is solved, shifts are created from their solutions and this offers a solution to the rostering problem for the current node of the branching tree. Note that $b$-matching problems are solvable in polynomial time [232]. We know that there are feasible solutions to these $b$-matching problems since the fractional solutions for the rostering problem offer fractional solutions for the $b$-matching problems. Since $b$-matching has a totally unimodular technology matrix, such a fractional solution is a convex combination of integral solutions. Solving the $b$-matching problems directly returns integral solutions, and from these the shifts are created. Hence, the (B&P) method, of which the branching scheme and these $b$-matching problems are an important part, leads to an integral solution. Of course, the $b$-matching problem only needs to be solved if, after complete branching, the solution in the current node of the branching tree is fractional and its objective function value is less than the best integer solution found so far.

Although we engineered our (B&P) implementation such that it solves our rostering problem, the method offers flexibility to solve other rostering problems as well. In fact, the pricing problem determines the type of constraints that are implied on shifts, since the master problem is nothing but a constrained set covering problem, which only uses shifts generated by the pricing problem. Even if there are many constraints on shifts, this 'difficulty' is isolated in the pricing problem part of the solution approach. Of course, the branching strategy also has to be adjusted if other rostering problems are modeled, but we believe that this is not too difficult. Furthermore, heuristics may be combined with exact approaches to solve the pricing problems, to reduce the solving time used by the pricing problem. Finally, we note the potential of (B&P) to incorporate 'practical' constraints. For example, in practice it is often preferred to have employees work only shifts from a predefined set of 'template' shifts. This is easily incorporated in the (B&P) model by simply letting the pricing problem check which, if any, of the template shifts offers the best reduced cost.

## 8.4 Experimental results

To assess the performance of our Branch-and-Price (B&P) method, we compare its performance with that of the integer linear program (ILP) described in [257]. Note that, ILP works here, due to our restriction that employees are allowed to work only one shift that consists of a single time slot without interruptions. However, the ILP model, requires artificial binary variables for every employee and every time slot to model the rostering problem correctly. When employees are allowed to work multiple shifts, or when more elaborate constraints are implied, the number of artificial variables increases further, making the rostering problem both hard to model and hard to solve in an ILP setting.

The ILP model is solved via the ILOG Cplex 11.0 callable library. The B&P model is implemented in SCIP. The SCIP acronym stands for Solving Constraint Integer Programs. The non-commercial software SCIP offers a framework for constraint integer programming, Branch-and-Cut, and Branch-and-Price. For more information on SCIP, the reader is referred to [2]. The B&P approach uses Cplex 11.0 to solve the relaxed master problem.

To assess the quality of both solution methods 7920 randomized datasets are used. For each skill category $j$ the time horizon $1, \ldots, |T|$ is divided into a random number of subsets of consecutive time slots:

$$\{1, \ldots, t_1\}, \{t_1 + 1, \ldots, t_2\}, \ldots, \{t_x + 1, \ldots, |T|\} \tag{8.4.1}$$

Then, for each subset of time slots a demand for skill $j$ is generated; all time slots within a set have the same demand. For example, for a given skill category $j$, we have a demand for 2 skilled workers during time slots 1 up till $t_1$, a demand for 4 skilled workers during time slots $t_1 + 1$ up till $t_2$, and so on.

In order to be reasonably sure that demand can be met in every time slot $t$ and for each skill category $j$ we add the restriction:

$$d_{jt} \leq \left\lfloor \frac{|I|}{|J|} \right\rfloor. \tag{8.4.2}$$

Using this restriction, it is likely that sufficient employees are available to meet demand.

We think this kind of demand, where the demand for a skill category is constant during a series of consecutive time slots, fits reality better than a uniform random demand for time slot $t$ and skill $j$. The latter implies very strong fluctuations in demand, whereas the former implies a more 'controlled' demand pattern, which, we believe, is a better simulation of practice.

In addition to random demand, we created 'random' skill sets for the employees. For every employee first a uniform random number $m'$ is generated from $1, \ldots, |J|$. After that, uniform random numbers are drawn from $1, \ldots, |J|$ until $m'$ different numbers are obtained, indicating the skills the employee has. With these randomized demand and skills there is no assurance that there are enough available (skilled) employees to cover demand. However, in [257] it is argued that the number of employees that have skill $j$ is *in expectation* larger than the demand for skill $j$ in any time slot $t$. Hence, it is likely that demand can be met.

In Table 8.1, the test instances used in our experimental results are listed. In this chapter, we provide average results for groups of instances. For a full and detailed discussion of all results, we refer to [257]. The main focus in this chapter is the comparison of solving times of the B&P approach and the ILP approach. There is no need to evaluate the solution quality, since both implementations produce optimal solutions.

Table 8.1: Test instances

| Instance | $\|I\|$ | $\|J\|$ | $\|T\|$ | $l^{\min}$ | $l^{\max}$ |
|---|---|---|---|---|---|
| 1 | 20 | 2 | 24 | $11, 12, \ldots, 20$ | $16, 17, \ldots, 24$ |
| 2 | 20 | 5 | 24 | $11, 12, \ldots, 20$ | $16, 17, \ldots, 24$ |
| 3 | 20 | 2 | $4, 8, \ldots, 48$ | $1/2 \cdot \|T\|$ | $3/4 \cdot \|T\|$ |
| 4 | 20 | 5 | $4, 8, \ldots, 48$ | $1/2 \cdot \|T\|$ | $3/4 \cdot \|T\|$ |
| 5 | $5, 10, \ldots, 50$ | 2 | 24 | 12 | 18 |
| 6 | $5, 10, \ldots, 50$ | 5 | 24 | 12 | 18 |

For each class of test instances, Table 8.1 lists the number of employees $|I|$, the number of skills $|J|$, and the number of time slots $|T|$ in the second, third and fourth column, respectively. The columns $l^{\min}$ and $l^{\max}$ indicate the minimum number of time slots an employee should work and the maximum number of time slots an employee is allowed to work, respectively. Note that for each parameter sets (set of fixed values of $|I|$, $|J|$, $|T|$, $l^{\min}$ and $l^{\max}$), we generate and solve 20 randomized test instances, implying that the number of unsolved instances may be large. Note that for test instances 1 and 2 the values of $l^{\min}$ and $l^{\max}$ are varied. For test instances 3 and 4 the value of $|T|$ is varied, and for test instances 5 and 6 the value of $|I|$ is varied. Test instances for which $l^{\min} > l^{\max}$ are ignored, since they are infeasible.

Table 8.2 presents the experimental results. The columns 'avg. solving time (sec)' and 'number of unsolved instances' indicate the average time needed to find a solution and the number of unsolved instances, respectively. An instance is

regarded as 'unsolved' if no solution is found within 3600 seconds (i.e., one hour).

Table 8.2: Experimental results

| | Branch-and-Price | | ILP | |
|---|---|---|---|---|
| Instance | avg time (sec.) | unsolved instances | avg time (sec.) | unsolved instances |
| 1 | 5.3 | 0 | 3.6 | 2 |
| 2 | 68.6 | 24 | 9.4 | 23 |
| 3 | 103.3 | 4 | 7.2 | 0 |
| 4 | 430.8 | 30 | 25.6 | 3 |
| 5 | 6.5 | 0 | 13.4 | 0 |
| 6 | 65.3 | 5 | 44.6 | 9 |

From Table 8.2, we observe that B&P performs relatively poorly for instances with large values of $|T|$. Furthermore, we observe that B&P has a stronger dependence on the value of $|J|$ than ILP has. The relatively poor performance of B&P for larger $T$ is caused by the fact that the time needed to solve the pricing problem depends cubically on the value of $T$, see Section 8.3. Furthermore, for larger $|T|$ (and $|J|$) the number of columns that needs to be generated is also larger, which implies that the total time consumed by the pricing problem increases. On the positive side, although not directly apparent from Table 8.2, B&P outperforms ILP as $|I|$ increases.

For most istances we see that ILP outperforms B&P. Even though, the performance of B&P is quite acceptable in most intances, and there exist several possibilities to improve the B&P method. We list some good possibilities for improving the B&P implementation. As outlined above, the time needed by the pricing problem increases a lot as $T$ gets larger. There also is a slight dependence on the value of $m$. Decreasing these dependencies will improve both the time needed by the pricing problem as well as the overall required solving time. For instance, instead of looking for the column with the best reduced cost, an alternative might be to stop after the first column with negative reduced cost is found. Moreover, in the current implementation we only generate one column at a time when the pricing problem is called. Generating multiple columns at once might also improve the solving speed of the B&P implementation.

Solving the LP relaxations of the master problem is, next to solving the pricing problem, the most time consuming part of the B&P algorithm. For larger $|J|$ and $|T|$ the master problem becomes very large during the solving process, and hence solving the LP relaxations consumes more and more time. Column management

strategies, that control the number of variables in the master problem, might further decrease the total solving time. Moreover, one could also accept a non-optimal solution to the master problem that is within a certain range from the best LP bound.

Perhaps most importantly, we note that we studied a scheduling problem that ignores some complex constraints from practice. We expect that for such constraints it may be hard to include them in an extended mathematical programming formulation, while keeping it solvable in a reasonable time. Hence, for more complex and larger problems, we prefer B&P over mathematical programming as solution methodology.

## 8.5 Conclusions

Many personnel scheduling methods first create shifts based on staffing levels, and then create work schedules from this set of created shifts. In this chapter, we outline why these rostering methods often have a hard time accounting for employee specific preferences and characteristics. To create rosters directly from staffing levels, which allows accounting for employee preferences when creating shifts, we investigate a Branch-and-Price implementation. Due to the structure of the Branch-and-Price method, we are able to isolate labor legislation in the column generation sub-problem, which makes it flexible and powerful. The flexibility implies that it is not too difficult to extend the Branch-and-Price model to include complex constraints derived from actual practice. Moreover, due to the structure of Branch-and-Price methods, there is flexibility to deal with these 'difficult' constraints efficiently. Furthermore, Branch-and-Price methods also allow the use of so-called 'template shifts'; for the creation of work schedules based on shifts from a predefined set of templates, which is an intuitive way for planners to control the types of shifts that can be part of the work schedules.

CHAPTER 9

# A Flexible Iterative Improvement Heuristic to Support Creation of Feasible Work Schedules in Self-Scheduling

## 9.1 Introduction

In service industries, such as healthcare and security services, shifts have to be staffed around the clock. Considering the many employee preferences and labor legislation that are implied on schedules from such circumstances, it is often hard to come up with good or fair shift schedules [54]. Self-scheduling is a way to better cope with employee preferences leading also to an increased job satisfaction and an improved employee commitment and cooperation [154, 164, 222].

Several self-scheduling processes exist in practice. The basic structure of these processes is that employees propose a schedule by indicating for each day in the schedule the shift they prefer to work, or whether they would like to have a day off. These proposed schedules have to comply with labor legislation and meet contract hours. The organization now evaluates the proposed schedules and has to ensure that sufficient employees are assigned to each shift. If the joint schedules of the employees do not meet these bounds, feedback information is provided to the employees. Based on this information, employees may choose to update their proposed schedules, for example by trading shifts with other employees. This process leads to updated schedules that hopefully fulfill the demands. However, it may happen that some shifts are still insufficiently staffed. In this case, the department manager or a human shift planner has to decide which shifts to reassign.

The method proposed in this chapter is concerned with the final planning phase. Our method operates independently of the actual self-scheduling process applied by the organization. The input used by our method consists of a set of proposed work schedules and a shift demand, indicating for each shift the minimum number of

employees that have to be assigned to this shift. Given this input, understaffed and overstaffed shifts are identified, whereby we refer to a shift as *understaffed* if less employees have signed up for this shift than specified by the staffing demand and *overstaffed* if the opposite holds. The method now has to resolve this unbalance by reassigning shifts, taking into account several constraints and criteria.

The criteria used within our method are derived from case studies from practice. The goal of our method is to minimize the total understaffing, while satisfying labor legislation. This is accomplished using 'shift swaps'. Shift swaps are defined by an unassignment of a shift of some employee and an assignment of another shift to the same employee. Essentially, our method iteratively selects combinations of shift swaps in order to reduce the total understaffing as much as possible. By including only shift swaps in the method that satisfy labor legislation we ensure that our method does not violate labor legislation. Next to minimizing the total understaffing and satisfying labor legislation, the considered case studies indicate that the method has to be transparent, meaning that the planner should be able to understand the method's decisions. Besides this, a specified fraction of each schedule proposed by the employees has to be retained, since otherwise employees get discouraged to participate in the self-scheduling process. These requirements are accomplished in our method by proper choices of parameter values.

Shift rostering instances as provided by the case studies are used to evaluate the proposed method. In total, 72 instances were used. We analyze the influence of various parameters on the work schedules produced by the method. The main outcome is the observed trade-off between the total understaffing on the one hand and the number of schedule changes on the other hand.

Our contribution is twofold. First, the designed method is flexible and easily extendable, since the checking of labor legislation is an isolated component in our approach. Labor legislation is checked in the method that defines the allowed shift swaps, but not in the optimization method that selects the swaps to be applied. Second, the iterative nature of our method helps planners to understand the decisions taken by the algorithm in each iteration, since only a limited number of shift reassignments are performed in each iteration. This is especially the case if in each iteration only one or only a few employees are allowed to get a changed schedule.

This chapter is organized as follows. Section 9.2 discusses the related literature. Then, Section 9.3 provides a problem description and outlines our principal solution approach. Next, Section 9.4 presents the mathematical implementation of our solution approach and Section 9.5 discusses results on the instances derived from the case studies. The chapter closes with conclusions and discussion in Section 9.6.

## 9.2 Literature review

Employee preferences, such as requests for a day off or to work some specific shift on some specific day, are often quantified using soft constraints, see Chapter 3. Optimizing the work schedules of individual employees, while satisfying hard scheduling rules is referred to as *preference scheduling*. Examples are presented in, e.g., [32] and [217]. In [110] an auction model is proposed, where employees bid on shifts and rest days using 'points'. Shifts are awarded to employees based on their bids. A mathematical program is used to check for feasibility and awarding shifts.

Another way of handling employee preferences is self-scheduling. Qualitative aspects of self-scheduling are discussed in, e.g., [26] and [242]. Self-scheduling starts with the organization specifying the staffing demand, i.e., specifying per day how many employees have to be staffed on each shift. After that, employees propose their preferred schedules. For each day in the planning horizon, the employees choose the shift they prefer, or whether they like to have a day off. This proposed schedule has to satisfy labor legislation and other scheduling constraints defined by the organization. The staffing resulting from the proposed schedules of all employees is compared to the staffing demand identifying understaffed and overstaffed shifts. In most self-scheduling processes, feedback information is provided to the employees. Based on some incentive, such as 'scores' for shifts, employees may choose to change some of the shifts in their schedules, leading hopefully to a decrease in the total understaffing. In some self-scheduling processes, employees can negotiate about changing shifts, which [264] models as a multi-agent model in which employees are represented by agents. For the remaining understaffed shifts, mostly a human planner decides who works these shifts, since the planner has the final responsibility to create work schedules without understaffed shifts.

The main difference between preference scheduling and self-scheduling is that the proposed schedules of the employees in self-scheduling have to satisfy labor legislation, while in preference scheduling a schedule that satisfies all preferences specified by the employees does not necessarily imply that this schedule is allowed by labor legislation.

Algorithms that assist the human planner to decide which employee works which understaffed shift are proposed in [18, 210, 225]. In [18], a method is proposed that first unassigns overstaffed shifts. This is done in an iterative way by first selecting an overstaffed shift, based on some priority rule, and then selecting an employee to be unassigned from this shift, again based on some priority rule. When this process is finished, understaffed shifts are assigned to employees in an

analogous fashion. Next, several shift swapping and shift reassignment steps are applied, which are similar to the shift reassignment steps considered in an analogous approach of [210]. Whereas the algorithms proposed in [18] and [210] select at most two shifts at a time, the method proposed [225] considers the complete schedule. In addition to proposing schedules, employees prioritize between shifts by specifying either a weak request, a strong request, or a veto for each shift. Based on the proposed schedules and the requests, a mathematical program is used to create a feasible schedules. The mathematical program reduces the total understaffing while balancing the number of preferences honored per employee.

The objective in rescheduling applications is also to reassign shifts in order to find a schedule without understaffed shifts [93, 183, 192, 193]. Rescheduling is required if shifts get understaffed due to unexpected absences, for example due to illness. The heuristics proposed in [192] and the genetic algorithm proposed in [193] try to find a schedule without understaffed shifts while minimizing the number of shift swaps. The research in [93] proposes a mathematical programming approach that minimizes a weighted sum of the total overstaffing and the total understaffing. The research in [183] proposes a genetic algorithm that also considers shift preferences and distributes the workload after reconstructing the work schedule.

This chapter proposes an iterative method to assist the planner in reducing the total understaffing. We propose an iterative method, as opposed to [225], such that planners can evaluate intermediate results. However, in our method, multiple shift reassignments are allowed in a single iteration, which leads to a more globalized optimization, as opposed to the iterative method proposed in [18] that assigns or unassigns a single shift in every iteration. Moreover, as opposed to other self-scheduling and rescheduling literature, the method proposed in this chapter guarantees that some given fraction of each schedule that is proposed by an employee is preserved by the algorithm.

## 9.3 Problem description and principal approach

In this section, we give a detailed problem description and discuss the principal ideas underlying our approach.

### 9.3.1 Problem description

We are given a schedule period with a set of shifts $K$ and a set of employees $I$. For each shift $k \in K$, a demand $d_k$ is specified that indicates the minimum number

of employees that have to be assigned to that shift. Note that a specific shift (e.g., a morning shift) on different days is represented by separate shifts in the set $K$.

For every employee $i \in I$, we are given a proposed schedule $S_i$. This schedule specifies, for each day in the planning horizon, either the shift the employee prefers to work or that the employee prefers to have a day off. We assume that the proposed schedules satisfy all hard constraints implied on the schedules, e.g., labor legislation, and that the overall contract hours of the employee over the planning horizon are fulfilled.

Based on the proposed schedules, the difference between the specified staffing demand $d_k$ and the actual staffing can be calculated for each shift $k \in K$. The differences are denoted by a difference parameter $v_k$, which indicates how far the proposed schedules are away from the preferred situation. We define $v_k$ to be positive for understaffed shifts and negative for overstaffed shifts. We introduce scores $\sigma_k$ for each shift $k \in K$, which are given as a function of $v_k$, i.e., $\sigma_k = f(v_k)$. We define $f$ to be increasing in the value of $v_k$. So, for overstaffed shifts, $\sigma_k$ is smaller than for understaffed shifts. Using the scores $\sigma_k$, we calculate for each employee $i \in I$ a score $s_i$ by summing up the scores of all shifts the employee has in his schedule $S_i$. An employee with a relatively high score $s_i$ has chosen relatively many understaffed shifts and is thus doing well for the organization. For employees with relatively low scores $s_i$, the opposite holds.

The goal now is to change the schedules $S_i$ of the employees such that the total understaffing and the total overstaffing, as expressed by $v_k$, is reduced. Hereby, we have to ensure that the new schedules satisfy labor legislation and contract hours. Moreover, an employee's schedule should not change too much, since otherwise the employee might get discouraged to propose schedules in the future. In addition, we have to ensure that the burden of shift reassignments is divided fairly throughout the employees.

First, in the next section, we present the principal ideas underlying our approach. A formal specifications of our method is given in Section 9.4.

### 9.3.2 Principal approach

To reduce the total understaffing and the total overstaffing, we apply 'swaps' in the proposed schedules. A swap changes the schedule of an employee by unassigning one of his shifts and assigning this employee to some other shift, whereby these shifts do not have to be on the same day. To reduce the total understaffing and the total overstaffing, a swap unassigns an overstaffed shift and assigns an understaffed shift. This means, that we do not allow that an overstaffed shift becomes understaffed or that an understaffed shift becomes overstaffed. The reason for this

is, that such changes lead to smaller fractions of the proposed schedules of the employees to be retained, but do not contribute directly to the objective of reducing the total understaffing and the total overstaffing. For the same reason we also exclude swaps from understaffed shifts to understaffed shifts and from overstaffed shifts to overstaffed shifts. To make sure that the final schedule satisfies labor legislation, we only allow swaps that comply with labor legislation. Moreover, strict unavailabilities of employees are incorporated in the model by excluding swaps that assign shifts to a day where an employee is strictly unavailable.

The proposed solution approach iteratively selects combinations of swaps, where in each iteration at most one swap per employee is selected. Since each swap satisfies labor legislation and labor legislations are defined per employee and not for groups of employees, this implies that all employee schedules satisfy labor legislation after applying swaps. The reason that we restrict to at most one swap per employee per iteration is that a combination of two swaps that individually are allowed by labor legislation, may not necessarily lead to an allowed schedule. For example, if it is allowed to work at most 5 shifts consecutively, additional checks are required to make sure that a combination of two swaps satisfies this constraint. Since, in general, labor legislation does not imply constraints on the relation between schedules of different employees, we may apply swaps at multiple employees simultaneously without having to include labor legislation checks in the swap selection method. Therefore, we consider a subset of employees for whom we simultaneously apply swaps in an iteration. The maximum size of this subset is an input parameter of the approach. Moreover, employees can explicitly be excluded from this subset. For example, employees that already have received a certain number of shift swaps may be excluded from receiving additional shift swaps.

This approach has two advantages. First, note that labor legislation only influences the selection of possible swaps, but does not interfere with finding a good combination of swaps. In this way, labor legislation is an isolated component in our approach. This component may be changed without the need to adapt the other components. Second, the size of the subset of employees that is selected in an iteration defines a trade-off between transparency of the approach on the one hand and larger improvements in the schedule on the other hand. Applying a single swap for one employee makes the decisions taken by the model easier to understand, whereas applying swaps for multiple employees simultaneously leads, in general, to larger improvements in the schedules.

Applying multiple swaps at one employee in one iteration makes decisions taken by the model harder to understand for the planner and implies that we have to include labor legislation checks in the mathematical model. As both of these are

undesirable consequences, in our model we apply at most one swap per employee in a single iteration.

## 9.4 Realization of the approach

To reduce the total understaffing and the total overstaffing, we apply an iterative method, in which in every iteration a mathematical program is solved to select a combination of swaps to apply. First, we discuss the mathematical programming formulation, after which we present the details of our iterative approach.

### 9.4.1 Swap selection model

Given is a set of schedules $\{\bar{S}_1, \ldots, \bar{S}_{|I|}\}$ at the start of the current iteration and a vector of shift demands $\left(d_1, \ldots, d_{|K|}\right)$. For each employee $i \in I$, a score $\bar{s}_i$ is calculated by summing the scores $\sigma_k$ of the shifts $k$ that are in the current schedule $\bar{S}_i$ of employee $i$. Note that the scores $\sigma_k$ only depend on the value of $v_k$ and are not updated during the iterative method.

For the swap selection, we introduce sets $R_i$, which contain all possible swaps for the employees $i \in I$. Each swap $r \in R_i$ is allowed by labor legislation and unassigns an overstaffed shift and assigns an understaffed shift. For each swap $r \in R_i$, a decision variable $x_r$ is introduced which denotes whether this swap $r$ is selected for employee $i$ ($x_r = 1$) or not ($x_r = 0$), i.e.:

$$x_r \in \{0, 1\} \quad r \in R_i, \ i \in I. \tag{9.4.1a}$$

As outlined in Section 9.3.2, in every iteration, at most one swap is selected per employee. This is enforced by the constraint:

$$\sum_{r \in R_i} x_r \leq 1 \quad i \in I. \tag{9.4.1b}$$

Also, we do not allow that an overstaffed shift becomes understaffed or the other way around. For this, we introduce the set $K^O \subseteq K$ that denotes the set of overstaffed shifts, i.e.:

$$K^O = \{k \in K | v_k \leq 0\}$$

and the set $K^U \subseteq K$ that denotes the set of understaffed shifts, i.e.:

$$K^U = \{k \in K | v_k \geq 0\}.$$

Next to the sets $K^O$ and $K^U$, we introduce a variable $n_k$, which denotes the difference between the staffing demand and the actual staffing *after* swaps have been applied. Hence, $n_k$ is positive if shift $k$ is understaffed after swaps have been applied and negative if $k$ is overstaffed after swaps have been applied. Let the parameter $\bar{v}_k$ denote the difference between the staffing demand and the actual staffing at the start of the current iteration. Let $U_i^k \subseteq R_i$ contain all swaps that unassign shift $k$ from employee $i$, and let $A_i^k \subseteq R_i$ contain the swaps that assign shift $k$ to employee $i$. Using the sets $U_i^k$ and $A_i^k$, we can calculate $n_k$ as follows:

$$n_k = \bar{v}_k + \sum_{i \in I} \left( \sum_{r \in U_i^k} x_r - \sum_{r \in A_i^k} x_r \right) \quad k \in K. \tag{9.4.1c}$$

Now, the following constraints ensure that overstaffed shifts do not become understaffed:

$$n_k \leq 0 \quad k \in K^O \tag{9.4.1d}$$

and that understaffed shifts do not become overstaffed:

$$n_k \geq 0 \quad k \in K^U. \tag{9.4.1e}$$

To evaluate the quality of a solution, we consider an objective function that has two components. The first component is responsible for minimizing the total understaffing. The second component specifies that swaps should be applied in the schedules of employees that have a low score $\bar{s}_i$. The employees that have low scores $\bar{s}_i$ at the start of the iteration are the employees that have relatively many overstaffed shifts in their current schedule. Combining these two components leads to the following objective:

$$\min \lambda_1 \sum_{k \in K^U} n_k + \lambda_2 \sum_{i \in I} \bar{s}_i \sum_{r \in R_i} x_r. \tag{9.4.1f}$$

Note that the second component makes the swap selection more fair, since $\bar{s}_i$ influences the swap selection, meaning that employees with a low score a more likely to get a shift reassignment. Also note that if the total overstaffing is smaller than the total understaffing the first component cannot achieve the value 0. The parameters $\lambda_1$ and $\lambda_2$ are used to define the relative importance of the components.

The resulting model (9.4.1a)-(9.4.1f) is an integer linear program that can be solved using standard solvers.

### 9.4.2 Solution approach

To reduce the total understaffing and the total overstaffing, we apply an iterative solution approach. Each iteration consists of three phases. First, a subset of employees $I' \subseteq I$ is selected. Second, for the employees in $I'$, we determine the set of possible swaps $R_i$, and third, we select a combination of swaps to be applied using the ILP model presented in Section 9.4.1.

Shift swaps are applied only to the schedules of the employees $i \in I'$. As outlined in Section 9.3.2, the number of employees to include in the set $I'$ is a model parameter. In $I'$, we include the employees with the smallest scores $\bar{s}_i$. The scores $\bar{s}_i$ are calculated as explained in Section 9.4.1. Hence, this selection is influenced by the scores $\sigma_k$ that are determined by the function $f$. If an employee's score $\bar{s}_i$ is low, this employee has relatively many overstaffed shifts, which is undesirable from an organizational point of view. Note that the scores $\bar{s}_i$ are updated after swaps are applied.

Next, for each employee $i \in I'$ the set of allowed swaps $R_i$ is determined. As outlined in Section 9.3.2, we only allow swaps from overstaffed to understaffed shifts. This assumption is relaxed later on in Section 9.4.3.

Given $I'$ and the corresponding sets $R_i$, the mathematical program (9.4.1) is used to determine the best combination of swaps. Note that, in every iteration, the sets $K^O$ and $K^U$ are based on the *current* schedules.

### 9.4.3 Extensions and discussion

In addition to swaps that unassign overstaffed shifts and assign understaffed shifts, to which we refer as *primary swaps*, we optionally extend the approach with *secondary swaps*. A secondary swap is performed at two employees. The first employee is unassigned from an overstaffed shift and assigned to a shift for which staffing demand is *exactly* matched; we refer to such a shift as a *matching* shift. The second employee is unassigned from the same matching shift and assigned to an understaffed shift. Note that secondary swaps explicitly assign from and to matching shifts, since otherwise we could have applied a primary swap that leads to the same reduction in overstaffed shifts, but retains a larger fraction of the employees' proposed schedules. Secondary swaps make it possible to reduce the total overstaffing and understaffing in situations where primary swaps are not able to do so.

To include secondary swaps in the method, swaps from overstaffed shifts to matching shifts and swaps from matching shifts to understaffed shifts are included in the sets $R_i$. Let the set $K^M \subseteq K$ be the set of matching shifts, hence $K^M =$

$K^O \cap K^U$. Constraints (9.4.1d) and (9.4.1e) ensure that $n_k$ remains 0 for $k \in K^M$. Note that secondary swaps are a kind of ejection chains (with a length of 2). If we would also include swaps from matching shifts to matching shifts in the sets $R_i$, we would allow the model to select even larger 'swap-chains'.

Finally, it is interesting to include shift pattern preferences. For example, employees often do not prefer to have isolated working days, i.e., a single shift with days off on either side. Violations of these preferences and the corresponding penalty weights can be calculated when creating swaps. In the objective function, a trade-off between the minimization of understaffed shifts and violated preferences can then be made. However, how this trade-off should be done is not straightforward and we leave this for future research.

## 9.5 Case studies and results

This section studies the effects of various input parameters on the work schedules produced by our approach. For this, we apply the developed approach to scheduling instances based on data from the practice of several organizations. In Section 9.5.1, we describe some important criteria for the solution approach that were stated by these organizations. After that, Section 9.5.2 describes some characteristics of the provided work schedules and Section 9.5.3 analyzes the performance of our approach on these schedules.

### 9.5.1 Criteria from practice

We have asked a general hospital, a center for forensic psychiatry, a transportation company, and a service company to indicate criteria that a solution approach has to meet. This resulted in the following set of criteria:

1. Minimize the total understaffing.
2. Do not violate labor legislation.
3. At least 80% of each proposed work schedule must be preserved.
4. Decisions made by the approach have to be transparent.
5. The computation time has to be short, i.e., at most a couple of minutes.

Criterion 1, 2, and 5, have straightforward explanations. Criterion 3 might imply that understaffed shifts remain that could have been solved if a smaller percentage than 80% would have been permitted. However, the organizations indicate

that if an insufficient part of the proposed schedules is preserved, employees are discouraged to propose their preferred schedules in the future. The organizations indicate that 80% is a suitable threshold. Note that, in our experimental results we evaluate the effect of decreasing this percentage to 70%. Criterion 4 implies that the algorithm should be such that planners are able to understand the decisions taken by the algorithm and if necessary to explain them to managers or employees. The first helps to stimulate the implementation success; the organizations expect that planners will more easily accept a system that they understand. The second may help to convince the employees to accept the system.

All these criteria can be handled in the proposed approach. Criterion 1 coincides with the objective (9.4.1f) of the mathematical program, which is used in the iterations, if we set the values of $\lambda_1$ and $\lambda_2$ such that the first component of objective (9.4.1f) dominates the second component. Furthermore, Criterion 2 can be satisfied by only including swaps that do not violate labor legislation. Criterion 3 can be realized by excluding employees from $I'$ for which an additional swap would imply that less than 80% of their schedules is preserved. Next to this, Criterion 4 may be handled by including only one or a few employees in $I'$ and Criterion 5 asks for setting a time limit on the computation time.

### 9.5.2 Experimental setup

We are provided with final work schedules as executed by two departments of the general hospital and by a department of the center for forensic psychiatry. In total, these organizations provided 72 work schedules with a one-month planning horizon and between 14 and 79 employees. The resulting shift occupancy determines the shift demand used in the computational experiments. Since the executed work schedules obviously match this demand exactly, we apply randomizations to the executed work schedules to create work schedules that can be used to evaluate our approach. Randomization is applied in two ways: to the executed work schedules, and to the shift demand. Details of this generation process can be found in [252].

Looking at the overstaffed and understaffed shifts, the randomization leads to two classes of work schedules:

1. *Demand randomization.* Weekly alternating overstaffed and understaffed shifts. In these schedules, the following pattern is repeated every two weeks: the first week contains overstaffed shifts, but no understaffed shifts. The second week contains understaffed shifts, but no overstaffed shifts. On average, these schedules contain a total understaffing of 30.1 shifts.

2. *Schedule randomization.* Overstaffed and understaffed shifts occur in each

week of the schedule. On average, these schedules contain a total understaffing of 92.0 shifts.

Hence, these randomization imply two totally different ways of distributing the understaffed and overstaffed shifts throughout the work schedules, which enables us to evaluate whether our method is able to effectively handle these different classes of work schedules. To analyze the results and assess the quality of the work schedules produced by our approach, we consider two performance indicators: the remaining total understaffing and the average fraction of the schedules proposed by the employees that is retained.

For the organization, it is important that the total understaffing is minimized, since understaffed shifts imply that an insufficient number of employees are staffed on these shifts. If $\sum_{k \in K^U} n_k > \sum_{k \in K^O} |n_k|$, we correct the remaining total understaffing by the difference, since in this case this difference is a lower bound to the minimum $\sum_{k \in K^U} n_k$ and thereby, after this correction, the natural lower bound on the total understaffing is 0. For the two classes of work schedules that we consider, the mentioned total understaffing is the corrected total understaffing that, in theory, can thus be reduced to 0.

For the employees, it is important that the fraction of their schedules that is retained is as high as possible, since this implies that they get to work a large fraction of the schedule they proposed. We calculate this fraction as the number of days in the schedule of the employee where the assignment is unchanged (including days off) divided by the length of the planning horizon (in days).

### 9.5.3 Experimental results

We investigate the influence of the following five parameters on the results of the proposed method: swap strategy, constraints on the minimum remaining fraction, the size of $I'$, the values of $\lambda_1$ and $\lambda_2$, and the function $f(v_k)$. For a detailed results analysis we refer the reader to [252]. These results indicate that the latter three parameters have no noticeable influence on the performance indicators for reasonable choices of the parameters. The values we use for $|I'|$ are $|I'| = 1$ and $|I'| = 3$. For $(\lambda_1, \lambda_2)$, we compare $(\lambda_1, \lambda_2) = (1, 0)$, which implies that the objective only considers minimizing the total understaffing, with $(\lambda_1, \lambda_2) = (1, \varepsilon)$ for some small $\varepsilon > 0$, which implies that the minimization of the total understaffing dominates the objective function, but that this minimization is guided by the scores

$\bar{s}_i$ of the employees. For the score function $f(v_k)$ we compared:

$$f(v_k) = \begin{cases} 3 & \text{if } v_k > 0 \\ 2 & \text{if } v_k = 0 \\ 1 & \text{if } v_k < 0 \end{cases}, \tag{9.5.1}$$

with:

$$f(v_k) = \frac{v_k + d_k}{d_k} = \frac{\text{'employees that have chosen } k\text{'}}{\text{staffing demand shift } k}. \tag{9.5.2}$$

In this section, we focus our analysis on the effect of the minimum remaining percentage and the swap strategy. For this, each of the 72 randomized schedules is solved for each combination of parameter values. Hence, each schedule is solved $2^5 = 32$ times. We have made these instances available for others to challenge our results, see [234].

We start by investigating the influence of secondary swaps. For this, we apply our approach with and without secondary swaps. Next, we check the influence of the constraint on the fraction of the schedule that has to be retained. The organizations indicated that this fraction equals 0.8, which we are going to compare with a fraction of 0.7.

First, we look at results for the first class of work schedules, where overstaffed and understaffed shifts occur in alternating weeks. Table 9.1 shows the effects of the bound on the minimum remaining fraction and the swap strategy on the average remaining total understaffing (ARU) and the average remaining fraction (ARF) of the schedules. Moreover, average and maximum computation times are shown in Table 9.1. These averages were calculated from the various instances that were solved using the various parameter values.

Table 9.1: Results for first class of work schedules

| | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|
| | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| Swaps | | | avg | max | | | avg | max |
| prim. | 0.85 | 94.0% | 1.6 | 4.7 | 0.72 | 93.9% | 1.6 | 4.8 |
| prim. & sec. | 0.25 | 93.6% | 1.7 | 5.2 | 0.14 | 93.5% | 1.7 | 5.7 |

From Table 9.1 we observe that in all cases the average remaining total understaffing is reduced from 30.1 shifts to less than 1 shift. Furthermore, the influence of secondary swaps on the achieved results is larger than that of the bound on the minimum remaining fraction: using secondary swaps reduces the ARU by almost

0.6 shifts compared to only using primary swaps, whereas reducing the bound from 80% to 70% leads only to an additional reduction of around 0.1 shifts. Furthermore, note that if the constraint on the remaining fraction is relaxed to 70%, the resulting ARF is still far above 80% (around 93.5%). However, there are a few employees that now retain less than 80% of their schedules; on average this holds for less than 2% of the employees. We see that the average and maximum computation times are small and thus satisfy Criterion 5. In the appendix in Section 9.7, Tables 9.3-9.5 present detailed results per instance.

For the second class of work schedules, where overstaffed and understaffed shifts occur in the same week, we summarize the corresponding results in Table 9.2.

Table 9.2: Results for second class of work schedules

| | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|
| | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| Swaps | | | avg | max | | | avg | max |
| prim. | 1.49 | 89.7% | 3.4 | 9.7 | 0.30 | 89.1% | 3.3 | 11.5 |
| prim. & sec. | 1.31 | 89.6% | 3.4 | 10.2 | 0.19 | 89.0% | 3.3 | 11.9 |

Table 9.2 shows that the total understaffing is again reduced heavily: from on average 92 shifts to less than 1.5 shifts. For this class of work schedules, the bound on the minimum remaining fraction has a larger influence on the achieved results than secondary swaps have: the ARU is reduced by over 1.1 shift if the bound is relaxed from 0.8 to 0.7, whereas using secondary swaps in addition to primary swaps leads only to an additional reduction of around 0.15 shifts. Furthermore, if the constraint on the remaining fraction is set to 0.7, the ARF is again still far above 0.8 (around 89.1%). However, in that case some employees retain less than 80% of their schedule; on average this was the case for about 9.5% of the employees.

In general, we observe that the method performs well. For both classes of schedules, the remaining total understaffing is small, while on average 89% or more of each employee proposed schedule is retained. For the first class of work schedules, secondary swaps have the largest influence on the key performance indicators: the remaining total understaffing and the average fraction of each proposed schedule that is retained. We think the effect of secondary swaps is larger here, since overstaffing and understaffing do not occur in the same week. Then, swaps have to be applied that unassign a shift in one week, and assign a shift in another. Since labor legislation implies restrictions on, for example,

the number of shifts in a week and the number of consecutive shifts, secondary swaps offer a source of flexibility that turns out to be useful. To the contrary, for the second class of work schedules, the effect of decreasing the constraint on the minimum remaining fraction has the largest influence, which is caused by the larger understaffing in these schedules. Also swaps that swap shifts on the same day are very often allowed by labor legislation, which implies a smaller need for secondary swaps.

## 9.6 Conclusions and discussion

In this chapter, we have considered a problem occurring within self-scheduling processes. In particular, we have studied how to reduce the total understaffing and overstaffing resulting from work schedules proposed by employees. For this, we have designed an iterative solution approach. In each iteration, the total understaffing is reduced by selecting shift swaps, i.e., an unassignment of an overstaffed shift and an assignment of an understaffed shift to one and the same employee. The swaps are selected using mathematical programming. Our method is flexible in that labor legislation is an isolated component that may be adapted without having to change the mathematical program. Moreover, the decisions taken in our approach can be made very easy to trace and understand by the planner and the employees. We interviewed four organizations to determine important criteria for our self-scheduling method. These organizations have stated that the method has to incorporate labor legislation, be understandable, and enforce that at least 80% of an employee proposed schedule is preserved.

We have applied our approach to 72 work schedules with a one-month planning horizon, which are based on work schedules provided by two organizations. These work schedules, which we subdivided into two classes, contain on average a total understaffing of 30.1 and 92 shifts, respectively. On average, our method reduces this understaffing to less than 0.5 and 0.9 shifts, respectively. The remaining total understaffing is mainly affected by two model parameters: the swap strategy and the constraint on the minimum fraction of each schedule proposed by an employee that must be retained. If overstaffed shifts and understaffed shifts occur in alternating weeks, we observe that the total understaffing decreases by more than 1 shift if we include so-called secondary swaps in our swap strategy. However, if both overstaffed shifts and understaffed shifts occur in each week of the schedule, the minimum remaining fraction has the largest influence on the remaining total understaffing.

For further research, we suggest to study metaheuristic approaches, since the

greedy local search character of our solution approach may cause that we block possible future swaps. Note, however, that this influences the transparency of the method, which was an important criterion in the design. From a practical point of view, it is interesting to incorporate specific employee preferences that are not necessarily honored by the proposed approach, such as 'work 8 hour shifts'.

# 9.7 Appendix. Detailed results

Table 9.3: Demand randomization - Case Forensic Psychiatry

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | prim. | 2.50 | 92.6 | 0.8 | 0.9 | 2.75 | 92.7 | 0.7 | 0.9 |
| | prim. & sec. | 1.63 | 91.7 | 0.8 | 1.2 | 1.75 | 91.7 | 0.8 | 0.9 |
| 2 | prim. | 2.25 | 90.2 | 1.0 | 1.6 | 2.25 | 90.2 | 0.9 | 1.1 |
| | prim. & sec. | 0.50 | 88.5 | 1.2 | 1.6 | 0.38 | 88.4 | 1.1 | 1.5 |
| 3 | prim. | 1.75 | 90.0 | 0.8 | 1.1 | 1.75 | 90.0 | 0.8 | 0.9 |
| | prim. & sec. | 0.38 | 88.5 | 1.0 | 1.2 | 0.38 | 88.6 | 1.0 | 1.3 |
| 4 | prim. | 1.00 | 89.0 | 0.9 | 1.3 | 0.75 | 88.9 | 0.9 | 1.2 |
| | prim. & sec. | 0.00 | 88.0 | 0.9 | 1.1 | 0.00 | 88.1 | 0.9 | 1.1 |
| 5 | prim. | 0.13 | 90.4 | 0.7 | 1.0 | 0.13 | 90.4 | 0.6 | 0.8 |
| | prim. & sec. | 0.00 | 90.2 | 0.6 | 0.9 | 0.00 | 90.2 | 0.6 | 0.8 |
| 6 | prim. | 0.38 | 89.3 | 0.9 | 1.1 | 0.25 | 89.1 | 0.8 | 1.1 |
| | prim. & sec. | 0.00 | 88.9 | 0.9 | 1.5 | 0.00 | 88.8 | 1.0 | 1.2 |
| 7 | prim. | 2.13 | 90.2 | 0.7 | 1.1 | 1.13 | 89.6 | 0.8 | 1.2 |
| | prim. & sec. | 1.63 | 89.7 | 0.9 | 1.2 | 0.38 | 88.9 | 1.0 | 1.2 |
| 8 | prim. | 3.25 | 88.9 | 0.9 | 1.1 | 2.63 | 88.6 | 1.0 | 1.6 |
| | prim. & sec. | 1.38 | 87.1 | 1.1 | 1.4 | 0.75 | 87.0 | 1.1 | 1.5 |
| 9 | prim. | 0.13 | 92.4 | 0.6 | 1.1 | 0.13 | 92.4 | 0.7 | 1.1 |
| | prim. & sec. | 0.00 | 92.4 | 0.7 | 1.0 | 0.00 | 92.3 | 0.8 | 1.4 |
| 10 | prim. | 2.13 | 90.2 | 0.8 | 0.9 | 1.00 | 89.6 | 0.9 | 1.0 |
| | prim. & sec. | 0.25 | 88.3 | 1.0 | 1.3 | 0.00 | 88.6 | 1.0 | 1.2 |
| 11 | prim. | 3.50 | 90.6 | 0.9 | 1.1 | 2.75 | 90.2 | 0.9 | 1.0 |
| | prim. & sec. | 1.63 | 88.9 | 1.1 | 1.4 | 0.63 | 88.2 | 1.2 | 1.5 |
| 12 | prim. | 2.63 | 89.6 | 1.0 | 1.5 | 2.38 | 89.5 | 0.9 | 1.2 |
| | prim. & sec. | 0.75 | 88.1 | 1.2 | 1.3 | 0.50 | 87.9 | 1.1 | 1.3 |

Table 9.4: Demand randomization - General Hospital Department 1

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | prim. | 0.13 | 95.8 | 0.9 | 1.5 | 0.13 | 95.8 | 1.1 | 2.4 |
| | prim. & sec. | 0.00 | 95.8 | 0.9 | 2.0 | 0.00 | 95.8 | 1.0 | 2.1 |
| 2 | prim. | 0.13 | 95.4 | 1.2 | 1.9 | 0.13 | 95.4 | 1.1 | 1.7 |
| | prim. & sec. | 0.00 | 95.4 | 1.2 | 1.9 | 0.00 | 95.4 | 1.2 | 2.1 |
| 3 | prim. | 0.63 | 94.5 | 1.8 | 2.2 | 0.25 | 94.4 | 1.7 | 2.3 |
| | prim. & sec. | 0.00 | 94.3 | 1.9 | 2.5 | 0.00 | 94.4 | 1.9 | 2.7 |
| 4 | prim. | 0.38 | 94.3 | 1.7 | 2.5 | 0.38 | 94.3 | 1.7 | 2.7 |
| | prim. & sec. | 0.00 | 94.2 | 1.7 | 2.7 | 0.00 | 94.3 | 1.8 | 2.7 |
| 5 | prim. | 0.25 | 94.4 | 2.0 | 2.5 | 0.25 | 94.4 | 2.0 | 2.4 |
| | prim. & sec. | 0.00 | 94.3 | 2.1 | 2.9 | 0.00 | 94.3 | 2.1 | 2.7 |
| 6 | prim. | 1.88 | 95.3 | 2.4 | 2.7 | 1.63 | 95.2 | 2.4 | 2.7 |
| | prim. & sec. | 0.25 | 94.8 | 2.7 | 3.1 | 0.13 | 94.8 | 2.7 | 3.1 |
| 7 | prim. | 0.00 | 96.0 | 0.9 | 1.2 | 0.00 | 96.0 | 1.0 | 1.2 |
| | prim. & sec. | 0.00 | 96.0 | 0.9 | 1.2 | 0.00 | 96.0 | 0.9 | 1.3 |
| 8 | prim. | 1.13 | 95.5 | 2.7 | 3.8 | 1.13 | 95.5 | 2.6 | 3.4 |
| | prim. & sec. | 0.00 | 95.3 | 3.0 | 4.2 | 0.00 | 95.3 | 2.9 | 3.8 |
| 9 | prim. | 0.00 | 96.3 | 1.0 | 1.3 | 0.00 | 96.3 | 1.0 | 1.3 |
| | prim. & sec. | 0.00 | 96.3 | 1.0 | 1.3 | 0.00 | 96.3 | 0.9 | 1.3 |
| 10 | prim. | 1.00 | 96.0 | 2.3 | 2.5 | 0.88 | 96.0 | 2.2 | 2.4 |
| | prim. & sec. | 0.13 | 95.7 | 2.5 | 3.1 | 0.00 | 95.7 | 2.5 | 2.8 |
| 11 | prim. | 0.63 | 95.8 | 2.4 | 2.9 | 0.63 | 95.8 | 2.4 | 3.1 |
| | prim. & sec. | 0.00 | 95.6 | 2.7 | 3.5 | 0.00 | 95.6 | 2.6 | 3.7 |
| 12 | prim. | 0.00 | 95.1 | 2.2 | 2.7 | 0.00 | 95.1 | 2.1 | 2.6 |
| | prim. & sec. | 0.00 | 95.1 | 2.1 | 2.7 | 0.00 | 95.1 | 2.1 | 2.7 |

Table 9.5: Demand randomization - General Hospital Department 2

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | prim. | 0.13 | 95.6 | 1.9 | 3.6 | 0.13 | 95.6 | 2.3 | 4.1 |
| | prim. & sec. | 0.00 | 95.6 | 2.0 | 4.0 | 0.00 | 95.6 | 2.2 | 4.3 |
| 2 | prim. | 0.00 | 96.6 | 1.2 | 1.6 | 0.00 | 96.6 | 1.1 | 1.4 |
| | prim. & sec. | 0.00 | 96.6 | 1.2 | 1.5 | 0.00 | 96.6 | 1.1 | 1.4 |
| 3 | prim. | 0.00 | 96.4 | 2.2 | 2.7 | 0.00 | 96.4 | 2.2 | 2.6 |
| | prim. & sec. | 0.00 | 96.4 | 2.2 | 2.7 | 0.00 | 96.4 | 2.3 | 2.6 |
| 4 | prim. | 0.13 | 96.5 | 1.3 | 2.8 | 0.13 | 96.5 | 1.4 | 2.8 |
| | prim. & sec. | 0.00 | 96.5 | 1.4 | 3.6 | 0.00 | 96.5 | 1.5 | 3.8 |
| 5 | prim. | 0.00 | 95.4 | 2.3 | 2.9 | 0.00 | 95.4 | 2.3 | 3.0 |
| | prim. & sec. | 0.00 | 95.4 | 2.2 | 3.0 | 0.00 | 95.4 | 2.2 | 2.9 |
| 6 | prim. | 0.00 | 96.6 | 1.8 | 2.4 | 0.00 | 96.6 | 1.9 | 2.4 |
| | prim. & sec. | 0.00 | 96.6 | 1.7 | 2.2 | 0.00 | 96.6 | 1.8 | 2.2 |
| 7 | prim. | 0.00 | 96.8 | 1.1 | 1.6 | 0.00 | 96.8 | 1.0 | 1.5 |
| | prim. & sec. | 0.00 | 96.8 | 1.1 | 1.8 | 0.00 | 96.8 | 1.1 | 1.7 |
| 8 | prim. | 0.63 | 96.6 | 2.9 | 3.8 | 0.38 | 96.6 | 2.8 | 3.8 |
| | prim. & sec. | 0.13 | 96.5 | 3.1 | 4.1 | 0.00 | 96.5 | 2.9 | 4.3 |
| 9 | prim. | 0.00 | 96.4 | 1.3 | 1.7 | 0.00 | 96.4 | 1.2 | 1.7 |
| | prim. & sec. | 0.00 | 96.4 | 1.4 | 1.8 | 0.00 | 96.4 | 1.3 | 1.8 |
| 10 | prim. | 0.25 | 96.2 | 3.1 | 4.5 | 0.25 | 96.2 | 3.1 | 4.8 |
| | prim. & sec. | 0.00 | 96.1 | 3.3 | 4.9 | 0.00 | 96.1 | 3.3 | 5.1 |
| 11 | prim. | 1.00 | 96.4 | 4.0 | 4.7 | 1.00 | 96.4 | 3.9 | 4.7 |
| | prim. & sec. | 0.13 | 96.3 | 4.5 | 5.2 | 0.13 | 96.3 | 4.5 | 5.7 |
| 12 | prim. | 0.75 | 96.8 | 3.5 | 4.5 | 0.75 | 96.8 | 3.6 | 4.4 |
| | prim. & sec. | 0.13 | 96.7 | 3.8 | 5.0 | 0.13 | 96.7 | 3.7 | 4.7 |

Table 9.6: Schedule randomization - Forensic Psychiatry

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | Prim | 1.88 | 82.5 | 1.3 | 1.6 | 1.25 | 82.2 | 1.4 | 1.7 |
| | prim. & sec. | 1.63 | 82.2 | 1.4 | 1.9 | 1.00 | 82.0 | 1.7 | 2.0 |
| 2 | prim. | 10.25 | 81.0 | 1.5 | 2.0 | 1.38 | 76.3 | 1.8 | 2.6 |
| | prim. & sec. | 10.13 | 80.9 | 1.5 | 2.0 | 0.63 | 75.5 | 1.9 | 2.3 |
| 3 | prim. | 1.75 | 83.8 | 1.2 | 1.5 | 0.00 | 82.7 | 1.0 | 1.4 |
| | prim. & sec. | 1.38 | 83.4 | 1.2 | 1.6 | 0.00 | 82.7 | 1.0 | 1.4 |
| 4 | prim. | 0.00 | 90.4 | 0.5 | 0.7 | 0.00 | 90.4 | 0.5 | 0.7 |
| | prim. & sec. | 0.00 | 90.4 | 0.5 | 0.7 | 0.00 | 90.4 | 0.6 | 0.8 |
| 5 | prim. | 0.50 | 84.9 | 1.1 | 1.3 | 0.00 | 84.6 | 0.9 | 1.2 |
| | prim. & sec. | 0.13 | 84.6 | 1.2 | 1.5 | 0.00 | 84.6 | 0.9 | 1.2 |
| 6 | prim. | 5.13 | 83.7 | 1.3 | 1.7 | 0.38 | 80.8 | 1.5 | 1.9 |
| | prim. & sec. | 4.00 | 82.8 | 1.4 | 1.9 | 0.00 | 80.5 | 1.7 | 2.3 |
| 7 | prim. | 1.50 | 85.0 | 1.1 | 1.5 | 0.00 | 84.3 | 1.0 | 1.4 |
| | prim. & sec. | 1.38 | 84.9 | 1.2 | 1.6 | 0.00 | 84.3 | 1.1 | 1.5 |
| 8 | prim. | 0.88 | 84.1 | 1.3 | 1.8 | 0.00 | 83.6 | 1.0 | 1.4 |
| | prim. & sec. | 0.75 | 84.1 | 1.2 | 1.4 | 0.00 | 83.6 | 1.1 | 1.5 |
| 9 | prim. | 5.88 | 83.7 | 1.8 | 2.3 | 0.13 | 80.8 | 2.0 | 2.5 |
| | prim. & sec. | 5.38 | 83.3 | 1.8 | 2.1 | 0.00 | 80.7 | 2.0 | 2.5 |
| 10 | prim. | 11.38 | 82.9 | 1.7 | 2.0 | 4.75 | 78.9 | 1.9 | 2.5 |
| | prim. & sec. | 11.38 | 82.9 | 1.7 | 2.2 | 4.13 | 78.3 | 2.0 | 2.4 |
| 11 | prim. | 0.50 | 85.0 | 1.4 | 1.8 | 0.00 | 84.4 | 1.2 | 2.0 |
| | prim. & sec. | 0.50 | 85.0 | 1.4 | 1.6 | 0.00 | 84.4 | 1.1 | 1.4 |
| 12 | prim. | 1.00 | 84.2 | 1.3 | 1.7 | 0.13 | 83.7 | 1.2 | 1.7 |
| | prim. & sec. | 0.50 | 83.8 | 1.4 | 1.9 | 0.00 | 83.5 | 1.2 | 1.7 |

Table 9.7: Schedule randomization - Case General Hospital Department 1

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | prim. | 5.25 | 87.5 | 5.2 | 6.4 | 1.75 | 86.8 | 7.1 | 8.0 |
| | prim. & sec. | 4.00 | 87.1 | 5.5 | 7.3 | 0.38 | 86.4 | 7.2 | 8.1 |
| 2 | prim. | 0.00 | 97.9 | 0.8 | 1.0 | 0.00 | 97.9 | 0.8 | 1.1 |
| | prim. & sec. | 0.00 | 97.9 | 0.7 | 1.0 | 0.00 | 97.9 | 0.8 | 1.2 |
| 3 | prim. | 0.25 | 90.8 | 3.5 | 3.9 | 0.00 | 90.7 | 3.4 | 4.1 |
| | prim. & sec. | 0.25 | 90.8 | 3.5 | 4.0 | 0.00 | 90.7 | 3.5 | 4.1 |
| 4 | prim. | 0.00 | 92.4 | 3.8 | 4.9 | 0.00 | 92.4 | 4.0 | 5.6 |
| | prim. & sec. | 0.00 | 92.4 | 3.8 | 4.9 | 0.00 | 92.4 | 3.9 | 5.1 |
| 5 | prim. | 3.50 | 89.1 | 8.4 | 9.7 | 0.75 | 88.6 | 9.9 | 11.5 |
| | prim. & sec. | 3.50 | 89.1 | 8.5 | 10.2 | 0.75 | 88.6 | 10.3 | 11.9 |
| 6 | prim. | 0.00 | 90.3 | 3.7 | 4.2 | 0.00 | 90.2 | 3.7 | 4.2 |
| | prim. & sec. | 0.00 | 90.3 | 3.8 | 4.4 | 0.00 | 90.2 | 3.6 | 4.2 |
| 7 | prim. | 0.00 | 92.3 | 3.7 | 4.7 | 0.00 | 92.2 | 3.7 | 4.7 |
| | prim. & sec. | 0.00 | 92.3 | 3.8 | 4.6 | 0.00 | 92.2 | 3.6 | 4.6 |
| 8 | prim. | 0.00 | 91.2 | 4.9 | 5.2 | 0.00 | 91.2 | 4.2 | 4.8 |
| | prim. & sec. | 0.00 | 91.2 | 5.0 | 5.4 | 0.00 | 91.2 | 4.1 | 4.5 |
| 9 | prim. | 0.00 | 94.5 | 2.2 | 2.9 | 0.00 | 94.5 | 2.2 | 3.1 |
| | prim. & sec. | 0.00 | 94.5 | 2.2 | 3.1 | 0.00 | 94.5 | 2.3 | 3.0 |
| 10 | prim. | 0.00 | 92.2 | 3.9 | 5.4 | 0.00 | 92.2 | 3.7 | 4.9 |
| | prim. & sec. | 0.00 | 92.2 | 3.8 | 5.2 | 0.00 | 92.2 | 3.6 | 5.0 |
| 11 | prim. | 0.00 | 92.8 | 4.0 | 5.2 | 0.00 | 92.8 | 3.6 | 4.6 |
| | prim. & sec. | 0.00 | 92.8 | 3.9 | 5.2 | 0.00 | 92.8 | 3.5 | 4.6 |
| 12 | prim. | 0.00 | 91.9 | 3.5 | 4.6 | 0.00 | 91.8 | 3.4 | 4.5 |
| | prim. & sec. | 0.00 | 91.9 | 3.6 | 4.7 | 0.00 | 91.8 | 3.3 | 4.4 |

Table 9.8: Schedule randomization - General Hospital Department 2

| | | min. remaining = 80% | | | | min. remaining = 70% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ARU | ARF | Time (sec.) | | ARU | ARF | Time (sec.) | |
| # | Swaps | | | avg | max | | | avg | max |
| 1 | prim. | 0.00 | 97.1 | 1.6 | 1.8 | 0.00 | 97.1 | 1.6 | 2.0 |
| | prim. & sec. | 0.00 | 97.1 | 1.6 | 2.1 | 0.00 | 97.1 | 1.6 | 2.0 |
| 2 | prim. | 0.00 | 93.5 | 4.2 | 5.3 | 0.00 | 93.4 | 4.3 | 5.4 |
| | prim. & sec. | 0.00 | 93.5 | 4.2 | 5.1 | 0.00 | 93.4 | 4.3 | 5.6 |
| 3 | prim. | 0.00 | 97.3 | 1.4 | 1.8 | 0.00 | 97.3 | 1.5 | 1.8 |
| | prim. & sec. | 0.00 | 97.3 | 1.5 | 1.7 | 0.00 | 97.3 | 1.5 | 1.9 |
| 4 | prim. | 0.00 | 90.3 | 6.1 | 7.4 | 0.00 | 90.3 | 5.4 | 6.6 |
| | prim. & sec. | 0.00 | 90.3 | 6.2 | 7.7 | 0.00 | 90.3 | 5.3 | 6.1 |
| 5 | prim. | 0.75 | 90.8 | 6.8 | 9.1 | 0.00 | 90.7 | 5.7 | 7.0 |
| | prim. & sec. | 0.25 | 90.7 | 7.1 | 10.0 | 0.00 | 90.7 | 5.6 | 7.0 |
| 6 | prim. | 0.00 | 92.9 | 4.1 | 4.7 | 0.00 | 92.8 | 3.8 | 4.8 |
| | prim. & sec. | 0.00 | 92.9 | 4.1 | 4.9 | 0.00 | 92.8 | 3.9 | 4.8 |
| 7 | prim. | 3.38 | 91.1 | 8.1 | 8.5 | 0.25 | 90.5 | 7.5 | 9.6 |
| | prim. & sec. | 2.13 | 90.8 | 8.7 | 9.2 | 0.00 | 90.5 | 7.5 | 10.2 |
| 8 | prim. | 0.00 | 91.8 | 5.1 | 6.1 | 0.00 | 91.7 | 4.3 | 5.8 |
| | prim. & sec. | 0.00 | 91.8 | 5.0 | 6.0 | 0.00 | 91.7 | 4.4 | 6.3 |
| 9 | prim. | 0.00 | 91.6 | 5.6 | 6.3 | 0.00 | 91.5 | 5.5 | 6.9 |
| | prim. & sec. | 0.00 | 91.6 | 5.6 | 6.5 | 0.00 | 91.5 | 5.5 | 6.6 |
| 10 | prim. | 0.00 | 90.0 | 6.9 | 8.0 | 0.00 | 90.0 | 6.1 | 7.4 |
| | prim. & sec. | 0.00 | 90.0 | 7.1 | 8.1 | 0.00 | 90.0 | 6.2 | 7.5 |
| 11 | prim. | 0.00 | 94.0 | 4.9 | 6.1 | 0.00 | 94.0 | 4.7 | 5.3 |
| | prim. & sec. | 0.00 | 94.0 | 4.9 | 6.2 | 0.00 | 94.0 | 4.7 | 5.8 |
| 12 | prim. | 0.00 | 94.0 | 3.5 | 4.7 | 0.00 | 94.0 | 3.6 | 4.7 |
| | prim. & sec. | 0.00 | 94.0 | 3.5 | 4.7 | 0.00 | 94.0 | 3.6 | 4.8 |

CHAPTER 10

# Epilogue

Efficient personnel schedules are important considering that personal wages form a major part of the operational expenses of many organizations. As indicated by the recent literature review by Bergh et al. [255]: "there are still some great opportunities in finding algorithms that efficiently cope with employee preferences". The research in this dissertation is an exploration in that direction. We develop and analyze algorithms that are able to efficiently cope with specific requests and preferences of individual employees in various personnel planning and scheduling decisions. In addition, they help to have the right professional available at the right time. Moreover, most of the developed algorithms are applied or implemented in practice. Here, we discuss how personnel preferences are incorporated in the algorithms and implementations discussed in this dissertation.

First, we have shown that decomposition techniques are able to effectively handle preferences of individual employees in personnel scheduling. The idea behind the studied decompositions is to address the important preferences of employees first, such that they are matched as good as possible. We are the first to propose a method that focuses on the scheduling of weekend shifts in the first phase of a decomposition approach, resulting in high-quality weekend work schedules. Creating schedules by first assigning weekend shifts also reflects how schedules are often created manually.

Furthermore, we show that successful applications of annualized hours potentially permit significant savings. Unfortunately, in the Netherlands, most hospitals use annualized hours as a 'correction mechanism', i.e., if an employee works too many hours in one month, this is compensated in the next month. We illustrate that a prospective application of annualized hours offers a potential savings of 5.2% for a department of a Dutch hospital. For employers, to successfully implement annualized hours, it is important to make good mutual agreements with individual employees. Hence, mathematical methods should, and as we show can, specifically support incorporation of individualized agreements.

In addition, we discuss a self-scheduling application. Increasingly, organizations start to use some form of self-scheduling and this topic has also gained interest in the literature. With regard to employee preferences, self-scheduling has huge potential, since employee preferences are explicitly considered in self-scheduling. Key success factor for self-scheduling is the involvement of employees, who have to get actively involved to propose schedules and negotiate about required shift reassignments. Reassignments are needed if the schedules proposed by the employee do not match with the staffing requirements of the organization. Mathematical methods that assist with rescheduling decisions, such as the one proposed in this dissertation, contribute to the success of self-scheduling.

Next to carefully considering personnel preferences, we believe that another important aspect in algorithmic designs is that the business user should be able to understand and steer the outcomes of the algorithms. The mentioned decomposition approaches contribute to this since the different parts of the decomposition focus on specific parts of the optimization problem, making it better tractable for the business user. Furthermore, the decompositions resemble how schedules are often created manually. The iterative approach applied to the discussed self-scheduling application allows to evaluate intermediate results and adjust these where required, which makes this approach tractable from a user perspective.

Although this dissertation reveals several mathematical challenges open for further research, we believe that the most important challenge lies with implementations in practice. The research in this dissertation is based on practical studies or applications, and we encourage additional research in this direction. Many of the algorithm designs explicitly consider that the algorithms should enable business users to understand and effectively steer the outcomes of these methods. We believe that this aspect, combined with the incorporation of employee specific preferences, will be instrumental in the implementation success of personnel planning and scheduling algorithms.

# Bibliography


[1] Abdennadher S. and Schlenker H., (1999). Nurse scheduling using constraint logic programming. In *Proceedings of the 1998 Winter Simulation Conference Proceedings*, pp. 838–843.

[2] Achterberg T., (2007). *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, Germany.

[3] Ahuja R., Orlin J., and Sharma D., (2000). Very large-scale neighborhood search. *International Transactions in Operational Research*, 7(4-5):301–317.

[4] Aickelin U., Burke E.K., and Li J.P., (2009). An evolutionary squeaky wheel optimization approach to personnel scheduling. *IEEE Transactions on Evolutionary Computation*, 13(2):433–443.

[5] 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.

[6] Aickelin U. and Dowsland K.A., (2004). An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761–778.

[7] Aickelin U. and White P., (2004). Building better nurse scheduling algorithms. *Annals of Operations Research*, 128(1-4):159–177.

[8] Akbari M., Zandieh M., and Dorri B., (2012). Scheduling part-time and mixed-skilled workers to maximize employee satisfaction. *International Journal of Advanced Manufacturing Technology*, 64(5-8):1017–1027.

[9] Al-Yakoob S.M. and Sherali H.D., (2007). Mixed-integer programming models for an employee scheduling problem with multiple shifts and work locations. *Annals of Operations Research*, 155(1):119–142.

[10] Al-Yakoob S.M. and Sherali H.D., (2007). Multiple shift scheduling of hierarchical workforce with multiple work centers. *Informatica*, 18(3):325–342.

[11] Alfares H.K., (1998). An efficient two-phase algorithm for cyclic days-off scheduling. *Computers & Operations Research*, 25(11):913–923.

[12] Alfares H.K., (2001). Efficient optimization of cyclic labor days-off scheduling. *OR Spectrum*, 23(2):283–294.

[13] Alfares H.K., (2002). Optimum workforce scheduling under the (14, 21) days-off timetable. *Journal of Applied Mathematics and Desicision Sciences*, 6(3):191–199.

[14] Alfares H.K., (2003). Four-day workweek scheduling with two or three consecutive days off. *Journal of Mathematical Modelling and Algorithms*, 2(1):67–80.

[15] Alfares H.K., (2004). Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1-4):145–175.

[16] Alfares H.K., (2007). A simulation approach for stochastic employee days-off scheduling. *International Journal of Modelling and Simulation*, 27(1):9–15.

[17] Altiparmak F. and Dengiz B., (2009). A cross entropy approach to design of reliable networks. *European Journal of Operational Research*, 199(2):542–552.

[18] Ásgeirsson E., (2012). Bridging the gap between self schedules and feasible schedules in staff scheduling. *Annals of Operations Research*. In press.

[19] Awadallah M., Khader A., Al-Betar M., and Bolaji A., (2011). Nurse rostering using modified harmony search algorithm. In *Swarm, Evolutionary, and Memetic Computing*, vol. 7077 of *Lecture Notes in Computer Science*, pp. 27–37. Springer Berlin Heidelberg.

[20] Azaiez M.N. and Al Sharif S.S., (2005). A 0-1 goal programming model for nurse scheduling. *Computers & Operations Research*, 32(3):491–507.

[21] Azmat C.S., Hürlimann T., and Widmer M., (2004). Mixed integer programming to schedule a single-shift workforce under annualized hours. *Annals of Operations Research*, 128(1):199–215.

[22] Azmat C.S. and Widmer M., (2004). A case study of single shift planning and scheduling under annualized hours: A simple three-step approach. *European Journal of Operational Research*, 153(1):148–175.

[23] Bagatourova O. and Mallya S.K., (2004). Coupled heuristic and simulation scheduling in a highly variable environment. In *Proceedings of the 2004 Winter Simulation Conference*, vol. 1-2, pp. 1856–1860.

[24] Bai R., Burke E.K., Kendall G., Li J., and McCollum B., (2010). A hybrid evolutionary approach to the nurse rostering problem. *IEEE Transactions on Evolutionary Computation*, 14(4):580–590.

[25] Bailey J., (1985). Integrated days off and shift personnel scheduling. *Computers & Industrial Engineering*, 9(4):395–404.

[26] Bailyn L., Collins R., and Song Y., (2007). Self-scheduling for hospital nurses: an attempt and its difficulties. *Journal of Nursing Management*, 15(1):72–77.

[27] Baker K.R., (1974). Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Science*, 20(12):1561–1568.

[28] Baker K.R., (1974). Scheduling full-time and part-time staff to meet cyclic requirements. *Operational Research Quarterly (1970-1977)*, 25(1):65–76.

[29] Baker K.R., (1976). Workforce allocation in cyclical scheduling problems: A survey. *Operational Research Quarterly*, 27(1):155–167.

[30] Baker K.R., Burns R.N., and Carter M., (1979). Staff scheduling with day-off and workstretch constraints. *AIIE Transactions*, 11(4):286–292.

[31] Baker K.R. and Magazine M.J., (1977). Workforce scheduling with cyclic demands and day-off constraints. *Management Science*, 24(2):161–167.

[32] Bard J. and Purnomo H., (2005). Short-term nurse scheduling in response to daily fluctuations in supply and demand. *Health Care Management Science*, 8(4):315–324.

[33] Bard J.F., (2004). Staff scheduling in high volume service facilities with downgrading. *IIE Transactions*, 36(10):985–997.

[34] Bard J.F. and Purnomo H.W., (2005). Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510 – 534.

[35] Bard J.F. and Purnomo H.W., (2007). Cyclic preference scheduling of nurses using a lagrangian-based heuristic. *Journal of Scheduling*, 10(1):5–23.

[36] Bartholdi J.J., (1981). A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operations Research*, 29(3):501–510.

[37] Bartholdi J.J., Orlin J.B., and Ratliff H.D., (1980). Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28(5):1074–1085.

[38] Baxter J. and Mosby M., (1988). Generating acceptable shift-working schedules. *The Journal of the Operational Research Society*, 39(6):537–542.

[39] Beasley, J.E.cen, (2004). Multidimensional knapsack problem instances. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html. Accessed: October 2013.

[40] Beaumont N., (1997). Using mixed integer programming to design employee rosters. *Journal of the Operational Research Society*, 48(6).

[41] Bechtold S.E., (1981). Work force scheduling for arbitrary cyclic demands. *Journal of Operations Management*, 1(4):205–214.

[42] Bechtold S.E. and Brusco M.J., (1994). A microcomputer-based heuristic for tour scheduling of a mixed workforce. *Computers & Operations Research*, 21(9):1001–1009.

[43] Beddoe G., Petrovic S., and Li J., (2009). A hybrid metaheuristic case-based reasoning system for nurse rostering. *Journal of Scheduling*, 12(2):99–119.

[44] Beddoe G.R. and Petrovic S., (2006). Selecting and weighting features using a genetic algorithm in a case-based reasoning approach to personnel rostering. *European Journal of Operational Research*, 175(2):649–671.

[45] Bekker J. and Aldrich C., (2011). The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, 211(1):112–121.

[46] Beliën J. and Demeulemeester E., (2008). A branch-and-price approach for integrating nurse and surgery scheduling. *European Journal of Operational Research*, 189(3):652–668.

[47] Bellanti F., Carello G., Della Croce F., and Tadei R., (2004). A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research*, 153(1):28–40.

[48] Berrada I., Ferland J.A., and Michelon P., (1996). A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-Economic Planning Sciences*, 30(3):183–193.

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

[50] Bilgin B., Demeester P., Misir M., Vancroonenburg W., Vanden Berghe G., and Wauters T., (2010). A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. In *Proceedings of the 8th International Conference on Practice and Theory of Automated Timetabling*.

[51] Bish E.K. and Wang Q., (2004). Optimal investment strategies for flexible resources, considering pricing and correlated demands. *Operations Research*, 52(6):954–964.

[52] Bogaert S. and Vloeberghs D., (2005). Differentiated and individualized personnel management: Diversity management in belgium. *European Management Journal*, 23(4):483–493.

[53] Bowey A., (1977). Corporate manpower planning. *Management Decision*, 15(5):421–447.

[54] Brooks I. and Swailes S., (2002). Analysis of the relationship between nurse influences over flexible working and commitment to nursing. *Journal of Advanced Nursing*, 38(2):117–126.

[55] Brownell W.S. and Lowerre J.M., (1976). Scheduling of work forces required in continuous operations under alternative labor policies. *Management Science*, 22(5):597–605.

[56] Brucker P., Burke E., Curtois T., Qu R., and Vanden Berghe G., (2010). A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16(4):559–573.

[57] Brunner J.O. and Bard J.F., (2013). Flexible weekly tour scheduling for postal service workers using a branch and price. *Journal of Scheduling*, 6(1):129–149.

[58] Brunner J.O., Bard J.F., and Kolisch R., (2009). Flexible shift scheduling of physicians. *Health Care Management Science*, 12(3):285–305.

[59] Brunner J.O., Bard J.F., and Kolisch R., (2011). Midterm scheduling of physicians with flexible shifts using branch and price. *IIE Transactions*, 43(2):84–109.

[60] Brusco M.J., (2008). An exact algorithm for a workforce allocation problem with application to an analysis of cross-training policies. *IIE Transactions*, 40(5):495–508.

[61] Burke E., Cowling P., De Causmaecker P., and Vanden Berghe G., (2001). A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214.

[62] Burke E., Curtois T., Hyde M., Kendall G., Ochoa G., Petrovic S., Vázquez-Rodríguez J.A., and Gendreau M., (2010). Iterated local search vs. hyper-heuristics: towards general-purpose search algorithms. In *2010 IEEE Congress on Evolutionary Computation*, pp. 1–8.

[63] Burke E., De Causmaecker P., Petrovic S., and Berghe G.V.. *Variable neighborhood search for nurse rostering problemschoi*, pp. 153–172. Kluwer Academic Publishers, Norwell, MA, USA, (2004).

[64] Burke E.K. and Curtois T., (2010). An ejection chain method and a branch and price algorithm applied to the instances of the first international nurse rostering competition, 2010. In *Proceedings of the 8th International Conference on Practice and Theory of Automated Timetabling*.

[65] Burke E.K. and Curtois T., (2012). New computational results for nurse rostering benchmark instances. *Unpublished work*.

[66] Burke E.K., Curtois T., Post G., Qu R., and Veltman B., (2008). A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341.

[67] Burke E.K., Curtois T., Qu R., and Vanden Berghe G., (2010). A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society*, 61(11):1667–1679.

[68] Burke E.K., Curtois T., van Draat L.F., van Ommeren J.K., and Post G., (2011). Progress control in iterated local search for nurse rostering. *Journal of the Operational Research Society*, 62(2):360–367.

[69] Burke E.K., De Causmaecker P., Petrovic S., and Vanden Berghe G., (2006). Metaheuristics for handling time interval coverage constraints in nurse scheduling. *Applied Artificial Intelligence*, 20(9):743–766.

[70] Burke E.K., De Causmaecker P., and Vanden Berghe G., (2004). Novel meta-heuristic approaches to nurse rostering problems in Belgian hospitals. In *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, pp. 44.1–44.18.

[71] 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.

[72] Burke E.K., Li J.P., and Qu R., (2010). A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484–493.

[73] Burns R.N. and Carter M.W., (1985). Work force size and single shift schedules with variable demands. *Management Science*, 31(5):599–607.

[74] Burns R.N. and Koop G.J., (1987). A modular approach to optimal multiple-shift manpower scheduling. *Operations Research*, 35(1):100–110.

[75] Burns R.N., Narasimhan R., and Smith L.D., (1998). A set-processing algorithm for scheduling staff on 4-day or 3-day work weeks. *Naval Research Logistics (NRL)*, 45(8):839–853.

[76] Campbell G.M., (2011). A two-stage stochastic program for scheduling and allocating cross-trained workers. *Journal of the Operational Research Society*, 62(6):1038–1047.

[77] Caprara A., Monaci M., and Toth P., (2001). A global method for crew planning in railway applications. In *Voß, S., Daduna, J.R. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505*, pp. 17–36.

[78] Carrasco R.C., (2010). Long-term staff scheduling with regular temporal distribution. *Computer Methods and Programs in Biomedicine*, 100(2):191–199.

[79] Caserta M. and Nodar M., (2009). A cross entropy based algorithm for reliability problems. *Journal of Heuristics*, 15(5):479–501.

[80] Caserta M. and Rico E.Q., (2009). A cross entropy-based metaheuristic algorithm for large-scale capacitated facility location problems. *Journal of the Operational Research Society*, 60(10):1439–1448.

[81] Caserta M. and Rico E.Q., (2009). A cross entropy-lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers & Operations Research*, 36(2):530–548.

[82] Caserta M., Rico E.Q., and Uribe A.M., (2008). A cross entropy algorithm for the knapsack problem with setups. *Computers & Operations Research*, 35(1):241–252.

[83] Centraal Bureau voor de Statistiek [Statistics Netherlands], (2013). Population pyramid. http://www.cbs.nl/en-GB/menu/themas/bevolking/cijfers/extra/piramide-fx.htm?Languageswitch=on. Accessed: October 2013.

[84] Centraal Bureau voor de Statistiek [Statistics Netherlands], (2013). Uitgaven aan zorg met 3,7 procent gestegen (healthcare expenses risen by 3.7 percent). http://www.cbs.nl/nl-NL/menu/themas/gezondheid-welzijn/publicaties/artikelen/archief/2013/2013-037-pb.htm. Accessed: October 2013.

[85] Centraal Planbureau [Netherlands Bureau for Economic Policy Analysis], (2011). Trends in gezondheid en zorg (trends in health and healthcare). http://www.cpb.nl/publicatie/trends-in-gezondheid-en-zorg. Accessed: October 2013.

[86] 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.

[87] Chiaramonte M.V. and Chiaramonte L.M., (2008). An agent-based nurse rostering system under minimal staffing conditions. *International Journal of Production Economics*, 114(2):697–713.

[88] Chod J. and Rudi N., (2005). Resource flexibility with responsive pricing. *Operations Research*, 53(3):532–548.

[89] CHOIR - Center for Healthcare Operations Improvement and Research, (2013). http://www.utwente.nl/choir/. Accessed: October 2013.

[90] Chu P.C. and Beasley J.E., (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86.

[91] Chvátal V., (1983). *Linear Programming*. W.H. Freeman and Company, New York.

[92] Cipriano R., Di Gaspero L., and Dovier A., (2006). Hybrid approaches for rostering: A case study in the integration of constraint programming and local search. In *Hybrid Metaheuristics*, vol. 4030 of *Lecture Notes in Computer Science*, pp. 110–123. Springer Berlin Heidelberg.

[93] Clark A.R. and Walker H., (2011). Nurse rescheduling with shift preferences and minimal disruption. *Journal of Applied Operational Research*, 3(3):148–162.

[94] Corominas A., Lusa A., and Olivella J., (2012). A detailed workforce planning model including non-linear dependence of capacity on the size of the staff and cash management. *European Journal of Operational Research*, 216(2):445 – 458.

[95] Corominas A., Lusa A., and Pastor R., (2002). Using MILP to plan annualised working hours. *The Journal of the Operational Research Society*, 53(10):1101–1108.

[96] Corominas A., Lusa A., and Pastor R., (2004). Planning annualised hours with a finite set of weekly working hours and joint holidays. *Annals of Operations Research*, 128(1-4):217–233.

[97] Corominas A., Lusa A., and Pastor R., (2005). Annualised hours a real flexibility tool. *OR Insight*, 18(1):10–14.

[98] Corominas A., Lusa A., and Pastor R., (2005). Characteristics and classification of the annualised working hours planning problems. *International Journal of Services Technology and Management*, 5(5-6):435–447.

[99] Corominas A., Lusa A., and Pastor R., (2007). Planning production and working time within an annualised hours scheme framework. *Annals of Operations Research*, 155(1):5–23.

[100] Corominas A., Lusa A., and Pastor R., (2007). Using a MILP model to establish a framework for an annualised hours agreement. *European Journal of Operational Research*, 177(3):1495–1506.

[101] Corominas A., Olivella J., and Pastor R., (2010). Capacity planning with working time accounts in services. *Journal of the Operational Research Society*, 61(2):321–331.

[102] Corominas A. and Pastor R., (2010). Replanning working time under annualised working hours. *International Journal of Production Research*, 48(5):1493–1515.

[103] Costa A., Jones O.D., and Kroese D., (2007). Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573 – 580.

[104] Costa M.C., Jarray F., and Picouleau C., (2006). An acyclic days-off scheduling problem. *4OR: A Quarterly Journal of Operations Research*, 4(1):73–85.

[105] Curtois, T, (). Employee scheduling benchmark data sets. http://www.cs.nott.ac.uk/~tec/NRP/. Accessed: October 2013.

[106] Dantzig G.B., (1954). Letter to the editor - a comment on Edie's "traffic delays at toll booths". *Operations Research*, 2(3):339–341.

[107] Day P.R. and Ryan D.M., (1997). Flight attendant rostering for short-haul airline operations. *Operations Research*, 45(5):649–661.

[108] De Boer P.T., Kroese D., Mannor S., and Rubinstein R.Y., (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

[109] De Causmaecker P. and Vanden Berghe G., (2011). A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16.

[110] 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):228–242.

[111] De Matta R. and Peters E., (2009). Developing work schedules for an intercity transit system with multiple driver types and fleet types. *European Journal of Operational Research*, 192(3):852–865.

[112] Deloitte, (2011). Benchmark 2011 VVT- en GHZ-instellingen. http://www.deloitte.com/assets/Dcom-Netherlands/LocalAssets/Documents/NL/Pers/nl_nl_sectorraportage_vvt_ghz_2011.pdf. Accessed: October 2013.

[113] Deng L., Qiao L., and Peng X., (2009). A directed quantile cross-entropy method for 0/1 knapsack problems. In *Proceedings of the 9th International Conference on Electronic Measurement Instruments*, pp. 4–319 – 4–322.

[114] Devlin K.J., (2003). *The millennium problems: the seven greatest unsolved mathematical puzzles of our timedel*. Basic Books, New York.

[115] Dowsland K. and Thompson J., (2000). Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51(7):825–833.

[116] Edie L.C., (1954). Traffic delays at toll booths. *Operations Research*, 2(2):107–138.

[117] Edwards J.S., (1983). A survey of manpower planning models and their application. *The Journal of the Operational Research Society*, 34(11):1031–1040.

[118] Eitzen G., Panton D., and Mills G., (2004). Multi-skilled workforce optimisation. *Annals of Operations Research*, 127(1-4):359–372.

[119] Elshafei M. and Alfares H.K., (2008). A dynamic programming algorithm for days-off scheduling with sequence dependent labor costs. *Journal of Scheduling*, 11(2):85–93.

[120] Emmons H., (1985). Work-force scheduling with cyclic requirements and constraints on days off, weekends off, and work stretch. *IIE Transactions*, 17(1):8–16.

[121] Emmons H. and Burns R.N., (1991). Off-day scheduling with hierarchical worker categories. *Operations Research*, 39(3):484–495.

[122] Emmons H. and Fuh D.S., (1997). Sizing and scheduling a full-time and part-time workforce with off-day and off-weekend constraints. *Annals of Operations Research*, 70(0):473–492.

[123] Ernst A., Jiang H., Krishnamoorthy M., Owens B., and Sier D., (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1):21–144.

[124] Ernst A., 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.

[125] Fathabadi H.S. and Ghiyasvand M., (2007). A new algorithm for solving the feasibility problem of a network flow. *Applied Mathematics and Computation*, 192(2):429–438.

[126] Felici G. and Gentile C., (2004). A polyhedral approach for the staff rostering problem. *Management Science*, 50(3):381–393.

[127] Fowler J.W., Wirojanagud P., and Gel E.S., (2008). Heuristics for workforce planning with worker differences. *European Journal of Operational Research*, 190(3):724–740.

[128] Fréville A., (2004). The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*, 155(1):1–21.

[129] Fréville A. and Hanafi S., (2005). The multidimensional 0-1 knapsack problem-bounds and computational aspects. *Ann. Oper. Res.*, 139(1):195–227.

[130] Frey L., Hanne T., and Dornberger R., (2009). Optimizing staff rosters for emergency shifts for doctors. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 2540–2546.

[131] Gärtner J., Musliu N., and Slany W., (2001). Rota: a research project on algorithms for workforce scheduling and shift design optimization. *AI Communications*, 14(2):83–92.

[132] Gendreau M., Ferland J., Gendron B., Hail N., Jaumard B., Lapierre S., Pesant G., and Soriano P., (2007). Physician scheduling in emergency rooms. In Burke E.K. and Rudová H., editors, *Practice and Theory of Automated Timetabling VI*, vol. 3867 of *Lecture Notes in Computer Science*, pp. 53–66. Springer Berlin Heidelberg.

[133] Ghiyasvand M., (2006). A new approach for computing a most positive cut using the minimum flow algorithms. *Applied Mathematics and Computation*, 176(1):27–36.

[134] Glass C.A. and Knight R.A., (2010). The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2):379–389.

[135] Glover F. and McMillan C., (1986). The general employee scheduling problem. an integration of ms and ai. *Computers & Operations Research*, 13(5):563–573.

[136] Goodman M.D., Dowsland K.A., and Thompson J.M., (2009). A grasp-knapsack hybrid for a nurse-scheduling problem. *Journal of Heuristics*, 15(4):351–379.

[137] Gordon L. and Erkut E., (2004). Improving volunteer scheduling for the Edmonton folk festival. *Interfaces*, 34(5):367–376.

[138] Grabot B. and Letouzey A., (2000). Short-term manpower management in manufacturing systems: new requirements and DSS prototyping. *Computers in Industry*, 43(1):11–29.

[139] Evrim Didem Güneş, (1999). Workforce scheduling. Technical Report, Department of Industrial Engineering, Bilkent University, Turkey.

[140] Günther M. and Nissen V., (2010). Combined working time model generation and personnel scheduling. In Dangelmaier W., Blecken A., Delius R., and Klöpfer S., editors, *Advanced Manufacturing and Sustainable Logistics*, vol. 46 of *Lecture Notes in Business Information Processing*, pp. 210–221. Springer Berlin Heidelberg.

[141] Gutjahr W.J. and Rauner M.S., (2007). An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. *Computers & Operations Research*, 34(3):642–666.

[142] Hadwan M. and Ayob M.B., (2009). An exploration study of nurse rostering practice at hospital universiti kebangsaan malaysia. In *Proceedings of the 2009 Conference on Data Mining and Optimization*, pp. 107–114.

[143] Hans E.W., (2001). *Resource loading by branch-and-price techniques*. PhD thesis, University of Twente, the Netherlands, Enschede.

[144] Hao G., Lai K.K., and Tan M., (2004). A neural network application in personnel scheduling. *Annals of Operations Research*, 128(1-4):65–90.

[145] Harper P.R., Powell N.H., and Williams J.E., (2010). Modelling the size and skill-mix of hospital nursing teams. *Journal of the Operational Research Society*, 61(5):768–779.

[146] Haspeslagh S., De Causmaecker P., Schaerf A., and Stølevik M., (2012). The first international nurse rostering competition 2010. *Annals of Operations Research*. In press.

[147] Helvik B.E. and Wittner O., (2001). Using the cross-entropy method to guide/govern mobile agent's path finding in networks. In *Proceedings of the Third International Workshop on Mobile Agents for Telecommunication Applications*, pp. 255–268.

[148] Hertz A., Lahrichi N., and Widmer M., (2010). A flexible MILP model for multiple-shift workforce planning under annualized hours. *European Journal of Operational Research*, 200(3):860–873.

[149] Hojati M., (2010). Near-optimal solution to an employee assignment problem with seniority. *Annals of Operations Research*, 181(1):539–557.

[150] Hopp W. and Spearman M., (2000). *Factory Physics, 2nd Edition*. McGraw-Hill/Irwin, New York.

[151] Hui K.P., Bean N., Kraetzl M., and Kroese D.P., (2005). The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134(1):101–118.

[152] Hulshof P.J.H., Kortbeek N., Boucherie R.J., Hans E.W., and Bakker P.J.M., (2012). Taxonomic classification of planning decisions in health care: a structured review of the state of the art in OR/MS. *Health Systems*, 1(2):129–175.

[153] Hung R., (1991). Single-shift workforce scheduling under a compressed workweek. *Omega*, 19(5):494–497.

[154] Hung R., (1992). Improving productivity and quality through workforce scheduling. *Industrial Management*, 34(6):4.

[155] Hung R., (1994). Multiple-shift workforce scheduling under the 3-4 workweek with different weekday and weekend labor requirements. *Management Science*, 40(2):280–284.

[156] Hung R., (1994). Single-shift off-day scheduling of a hierarchical workforce with variable demands. *European Journal of Operational Research*, 78(1):49–57.

[157] Hung R., (1995). Hospital nurse scheduling. *Journal of Nursing Administration*, 25(7-8):21–23.

[158] Hung R., (1999). A multiple-shift workforce scheduling model under annualized hours. *Naval Research Logistics (NRL)*, 46(6):726–736.

[159] Hung R., (1999). Scheduling a workforce under annualized hours. *International Journal of Production Research*, 37(11):2419–2427.

[160] Ikegami A. and Niwa A., (2003). A subproblem-centric model and approach to the nurse scheduling problem. *Mathematical Programming*, 97(3):517–541.

[161] Jarray F., (2009). A 4-day or 3-day workweeks scheduling problem with a given workforce size. *Asia-Pacific Journal of Operational Research*, 26(5):685–696.

[162] 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.

[163] Keith E.G., (1979). Operator scheduling. *AIIE Transactions*, 11(1):37–41.

[164] Kellogg D. and Walczak S., (2007). Nurse Scheduling: From Academia to Implementation or Not? *Interfaces*, 37(4):355–369.

[165] Knust S. and Schumacher E., (2011). Shift scheduling for tank trucks. *Omega-International Journal of Management Science*, 39(5):513–521.

[166] Koop G.J., (1986). Cyclic scheduling of offweekends. *Operations Research Letters*, 4(6):259–263.

[167] Laporte G. and Pesant G., (2004). A general multi-shift scheduling system. *Journal of the Operational Research Society*, 55(11):1208–1217.

[168] Lezaun M., Pérez G., and Sáinz de la Maza E., (2006). Crew rostering problem in a public transport company. *Journal of the Operational Research Society*, 57(10):1173–1179.

[169] Lezaun M., Pérez G., and Sáinz de la Maza E., (2007). Rostering in a rail passenger carrier. *Journal of Scheduling*, 10(4-5):245–254.

[170] Lezaun M., Pérez G., and Sáinz de la Maza E., (2010). Staff rostering for the station personnel of a railway company. *Journal of the Operational Research Society*, 61(7):1104–1111.

[171] Li J., Burke E.K., Curtois T., Petrovic S., and Rong Q., (2012). The falling tide algorithm: a new multi-objective approach for complex workforce scheduling. *Omega*, 40(3):283–293.

[172] Li J., Burke E.K., and Qu R., (2012). A pattern recognition based intelligent search method and two assignment problem case studies. *Applied Intelligence*, 36(2):442–453.

[173] Li Y., Chen J., and Cai X., (2007). An integrated staff-sizing approach considering feasibility of scheduling decision. *Annals of Operations Research*, 155(1):361–390.

[174] Lowerre J.M., (1977). Work stretch properties for the scheduling of continuous operations under alternative labor policies. *Management Science*, 23(9):963–971.

[175] Lu Z. and Hao J.K., (2012). Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research*, 218(3):865–876.

[176] Lusa A., Corominas A., and Muñoz N., (2008). A multistage scenario optimisation procedure to plan annualised working hours under demand uncertainty. *International Journal of Production Economics*, 113(2):957–968.

[177] Lusa A., Corominas A., Olivella J., and Pastor R., (2009). Production planning under a working time accounts scheme. *International Journal of Production Research*, 47(13):3435–3451.

[178] Lusa A., Corominas A., and Pastor R., (2008). An exact procedure for planning holidays and working time under annualized hours considering cross-trained workers with different efficiencies. *International Journal of Production Research*, 46(8):2123–2142.

[179] Lusa A. and Pastor R., (2011). Planning working time accounts under demand uncertainty. *Computers & Operations Research*, 38(2):517–524.

[180] Maenhout B. and Vanhoucke M., (2006). New computational results for the nurse scheduling problem: A scatter search algorithm. In *6th European Conference on Evolutionary Computation in Combinatorial Optimization*, vol. 3906, pp. 159–170. Lecture Notes in Computer Science.

[181] Maenhout B. and Vanhoucke M., (2008). Comparison and hybridization of crossover operators for the nurse scheduling problem. *Annals of Operations Research*, 159(1):333–353.

[182] 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.

[183] Maenhout B. and Vanhoucke M., (2011). An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*, 38(10):1400–1411.

[184] Maenhout B. and Vanhoucke M., (2013). An integrated nurse staffing and scheduling analysis for longer-term nursing staff allocation problems. *Omega*, 41(2):485–499.

[185] Maier-Rothe C. and Wolfe H.B., (1973). Cyclical scheduling and allocation of nursing staff. *Socio-Economic Planning Sciences*, 7(5):471–487.

[186] Margolin L., (2005). On the convergence of the cross-entropy method. *Annals of Operations Research*, 134(1):201–214.

[187] Metivier J.P., Boizumault P., and Loudni S., (2009). Solving nurse rostering problems using soft global constraints. In *15th International Conference on Principles and Practice of Constraint Programming*, vol. 5732, pp. 73–87. Lecture Notes in Computer Science.

[188] Miller H.E., Pierskalla W.P., and Rath G.J., (1976). Nurse scheduling using mathematical programming. *Operations Research*, 24(5):857–870.

[189] Mirrazavi S.K. and Beringer H., (2007). A web-based workforce management system for Sainsburys Supermarkets Ltd. *Annals of Operations Research*, 155(1):437–457.

[190] Mohan S., (2008). Scheduling part-time personnel with availability restrictions and preferences to maximize employee satisfaction. *Mathematical and Computer Modelling*, 48(11-12):1806–1813.

[191] Morris J.G. and Showalter M.J., (1983). Simple approaches to shift, days-off and tour scheduling problems. *Management Science*, 29(8):942–950.

[192] 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):179–197.

[193] Moz M. and Pato M.V., (2007). A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, 34(3):667–691.

[194] Musliu N., (2005). Combination of local search strategies for rotating workforce scheduling problem. In *Proocedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 1529–1530.

[195] Musliu N., Gärtner J., and Slany W., (2002). Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 118(1-2):85–98.

[196] Narasimhan R., (1997). An algorithm for single shift scheduling of hierarchical workforce. *European Journal of Operational Research*, 96(1):113–121.

[197] Nonobe K., (2010). INRC2010: an approach using a general constraint optimization solver. In *Proceedings of the 8th International Conference on Practice and Theory of Automated Timetabling*.

[198] Nurmi K., Kyngäs J., and Post G., (2011). Driver rostering for bus transit companies. *Engineering Letters*, 19(2):125–132.

[199] Okada M., (1992). An approach to the generalized nurse scheduling problem - generation of a declarative program to represent institution-specific knowledge. *Computers and Biomedical Research*, 25(5):417–434.

[200] ORTEC, (2013). http://www.ortec.com. Accessed: October 2013.

[201] ORTEC Workforce Scheduling, (2013). http://www.ortec.com/products/ortec_workforce_scheduling.aspx. Accessed: October 2013.

[202] 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.

[203] Özcan E., (2005). Memetic algorithms for nurse rostering. In *Computer and Information Sciences - ISCIS 2005*, vol. 3733, pp. 482–492. Lecture Notes in Computer Science.

[204] Paris, S., (2008). Multi-knapsack solver. http://www.mathworks.com/matlabcentral/fileexchange/20436-multi-knapsack-solver. Accessed: October 2013.

[205] Parr D. and Thompson J., (2007). Solving the multi-objective nurse scheduling problem with a weighted cost function. *Annals of Operations Research*, 155(1):279–288.

[206] Pastor R. and Corominas A., (2010). A bicriteria integer programming model for the hierarchical workforce scheduling problem. *Journal of Modelling in Management*, 5(1):54–62.

[207] Pastor R. and Olivella J., (2008). Selecting and adapting weekly work schedules with working time accounts: A case of a retail clothing chain. *European Journal of Operational Research*, 184(1):1–12.

[208] PATAT 2010 Nurse Rostering Competition, (2010). http://www.kuleuven-kulak.be/nrpcompetition. Accessed: October 2013.

[209] Pedrosa D. and Constantino M., (2001). Days-off scheduling in public transport companies. In Voß S. and Daduna J.R., editors, *Computer-Aided Scheduling of Public Transport*, vol. 505, pp. 215–232. Lecture Notes in Economics and Mathematical Systems.

[210] Petrovic S., Beddoe G., and Vanden Berghe G., (2003). Storing and adapting repair experiences in employee rostering. In Burke E. and Causmaecker P., editors, *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling*, vol. 2740, pp. 148–165. Lecture Notes in Computer Science.

[211] Petrovic S. and Vanden Berghe G., (2012). A comparison of two approaches to nurse rostering problems. *Annals of Operations Research*, 194(1):365–384.

[212] Pinedo M.L., (2009). *Planning and Scheduling in Manufacturing and Services, 2nd Edition*. Springer, New York.

[213] Post G., Ahmadi S., and Geertsema F., (2012). Cyclic transfers in school timetabling. *OR Spectrum*, 34(1):133–154.

[214] Post G. and Veltman B., (2004). Harmonious personnel scheduling. In *Proceedings of the 5th international conference on the Practice and Theory of Automated Timetabling*, pp. 557–559.

[215] Price W., Martel A., and Lewis K., (1980). A review of mathematical models in human resource planning. *Omega*, 8(6):639–645.

[216] Purkiss C., (1981). Corporate manpower planning: a review of models. *European Journal of Operational Research*, 8(4):315–323.

[217] Purnomo H.W. and Bard J.F., (2007). Cyclic preference scheduling for nurses using branch and price. *Naval Research Logistics (NRL)*, 54(2):200–220.

[218] Qi X.T. and Bard J.F., (2006). Generating labor requirements and rosters for mail handlers using simulation and optimization. *Computers & Operations Research*, 33(9):2645–2666.

[219] Qu R. and He F., (2009). A hybrid constraint programming approach for nurse rostering problems. In *28th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 211–224.

[220] Raff S., (1983). Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211.

[221] Rekik M., Cordeau J.F., and Soumis F., (2004). Using Benders decomposition to implicitly model tour scheduling. *Annals of Operations Research*, 128(1-4):111–133.

[222] Robb E.A., Determan A.C., Lampat L.R., Scherbring M.J., Slifka R.M., and Smith N.A., (2003). Strategies at work: Self-scheduling: Satisfaction guaranteed? *Nursing Management*, 34(7):16–18.

[223] Rodriguez M., (2003). Flexible working patterns using annualised hours. *Work Study*, 52(3):145–149.

[224] Rong A.Y., (2010). Monthly tour scheduling models with mixed skills considering weekend off requirements. *Computers & Industrial Engineering*, 59(2):334–343.

[225] Rönnberg E. and Larsson T., (2010). Automating the self-scheduling process of nurses in Swedish healthcare: a pilot study. *Health Care Management Science*, 13(1):35–53.

[226] Rosenbloom E. and Goertzen N., (1987). Cyclic nurse scheduling. *European Journal of Operational Research*, 31(1):19–23.

[227] Rothstein M., (1973). Hospital manpower shift scheduling by mathematical programming. *Health Services Research*, 8(1):60–66.

[228] Rubinstein R.Y., (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190.

[229] Rubinstein R.Y., (2002). Cross-entropy and rare events for maximal cut and partition problems. *ACM Transactions on Modeling and Computer Simulation*, 12(1):27–53.

[230] Rubinstein R.Y., (2008). Semi-iterative minimum cross-entropy algorithms for rare-events, counting, combinatorial and integer programming. *Methodology and Computing in Applied Probability*, 10(2):121–178.

[231] Rubinstein R.Y. and Kroese D.P., (2004). *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag, New York.

[232] Schrijver A., (2003). *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin.

[233] Seckiner S.U., Gokcen H., and Kurt M., (2007). An integer programming model for hierarchical workforce scheduling problem. *European Journal of Operational Research*, 183(2):694–699.

[234] Self-Rostering Instances, (2013). http://www.utwente.nl/mb/iebis/staff/schutten/SelfRostering/Instances.zip. Accessed: October 2013.

[235] Shahnazari-Shahrezaei P., Tavakkoli-Moghaddam R., and Kazemipoor H., (2013). Solving a new fuzzy multi-objective model for a multi-skilled manpower scheduling problem by particle swarm optimization and elite tabu search. *International Journal of Advanced Manufacturing Technology*, 64(9-12):1517–1540.

[236] Siferd S.P. and Benton W., (1992). Workforce staffing and scheduling: Hospital nursing specific models. *European Journal of Operational Research*, 60(3):233–246.

[237] Smet P., Bilgin B., De Causmaecker P., and Vanden Berghe G., (2012). Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*. In press.

[238] Sodhi M.S. and Norris S., (2004). A flexible, fast, and optimal modeling approach applied to crew rostering at London underground. *Annals of Operations Research*, 127(1):259–281.

[239] Spyropoulos C.D., (2000). AI planning and scheduling in the medical hospital environment. *Artificial Intelligence in Medicine*, 20(2):101–111.

[240] Stolletz R., (2010). Operational workforce planning for check-in counters at airports. *Transportation Research Part E-Logistics and Transportation Review*, 46(3):414–425.

[241] Syslo M.M., Deo N., and Kowalik J.S., (1983). *Discrete optimization algorithms: with Pascal programs*. Prentice-Hall, Englewood Cliffs, New Jersey.

[242] Teahan B., (1998). Implementation of a self-scheduling system: a solution to more than just schedules! *Journal of Nursing Management*, 6(6):361–368.

[243] Thiel M.P., (2008). Team-oriented airline crew rostering for cockpit personnel. In Hickman M., Mirchandani P., and Voß S., editors, *Computer-aided Systems in Public Transport*, vol. 600, pp. 91–114. Lecture Notes in Economics and Mathematical Systems.

[244] Thompson G.M. and Goodale J.C., (2006). Variable employee productivity in workforce scheduling. *European Journal of Operational Research*, 170(2):376–390.

[245] Thompson G.M. and Pullman M.E., (2007). Scheduling workforce relief breaks in advance versus in real-time. *European Journal of Operational Research*, 181(1):139–155.

[246] Tibrewala R., Philippe D., and Browne J., (1972). Optimal scheduling of two consecutive idle periods. *Management Science*, 19(1):71–75.

[247] Tien J.M. and Kamiyama A., (1982). On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287.

[248] Topaloglu S., (2006). A multi-objective programming model for scheduling emergency medicine residents. *Computers & Industrial Engineering*, 51(3):375–388.

[249] Topaloglu S., (2009). A shift scheduling model for employees with different seniority levels and an application in healthcare. *European Journal of Operational Research*, 198(3):943–957.

[250] Trilling L., Guinet A., and Le Magny D., (2006). Nurse scheduling using integer linear programming and constraint programming. In *Proceedings of the 12th IFAC International Symposium*, vol. 3, pp. 651–656.

[251] Tsai C.C. and Li S.H.A., (2009). A two-stage modeling with genetic algorithms for the nurse scheduling problem. *Expert Systems with Applications*, 36(5):9506–9512.

[252] Uijland S., (2012). Creating feasible schedules in the last step of the self rostering process. Master's thesis, University of Twente, the Netherlands.

[253] 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.

[254] Valouxis C. and Housos E., (2000). Hybrid optimization techniques for the work-shift and rest assignment of nursing personnel. *Artificial Intelligence in Medicine*, 20(2):155–175.

[255] 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.

[256] Van den Hurk H., (2007). *Arbeidsvoorwaarden in 100 vragen: voor OR'en en HR-medewerkers (in Dutch)*. Kluwer, Alphen aan den Rijn, The Netherlands.

[257] Van der Veen E., (2009). Rostering from staffing levels: a branch-and-price approach. Master's thesis, Rijksuniversiteit Groningen, the Netherlands.

[258] Van Mieghem J.A. and Dada M., (1999). Price versus production postponement: Capacity and competition. *Management Science*, 45(12):1639–1649.

[259] Van Veldhoven S., (2011). Days off personnel scheduling. Master's thesis, University of Twente, the Netherlands.

[260] Vanhoucke M. and Maenhout B., (2009). On the characterization and generation of nurse scheduling problem instances. *European Journal of Operational Research*, 196(2):457–467.

[261] Veldman B., Post G., Winkelhuijzen W., and Fijn van Draat L., (2006). Harmonious personnel scheduling. *Medium Econometrische Toepassingen*, 14(1):4–7.

[262] Versteegh F., (2009). Let the weekend begin! A solution for solving the weekend scheduling problem for ORTEC Harmony. Master's thesis, University of Twente, the Netherlands.

[263] Wan L. and Bard J.F., (2007). Weekly staff scheduling with workstation group restrictions. *Journal of the Operational Research Society*, 58(8):1030–1046.

[264] Wang Z. and Wang C., (2009). Automating nurse self-rostering: A multiagent systems model. In *IEEE International Conference on Systems, Man and Cybernetics, 2009*, pp. 4422 –4425.

[265] Warner D.M., (1976). Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, 24(5):842–856.

[266] White C.A., Nano E., Nguyen-Ngoc D.H., and White G.M., (2007). An evaluation of certain heuristic optimization algorithms in scheduling medical doctors and medical students. In *Proceedings of the 6th international conference on the Practice and Theory of Automated Timetabling*, vol. 3867, pp. 105–115.

[267] Wright P.D. and Bretthauer K.M., (2010). Strategies for addressing the nursing shortage: Coordinated decision making and workforce flexibility. *Decision Sciences*, 41(2):373–401.

[268] Wright P.D., Bretthauer K.M., and Côté M.J., (2006). Reexamining the nurse scheduling problem: Staffing ratios and nursing shortages. *Decision Sciences*, 37(1):39–70.

[269] Yilmaz E., (2012). A mathematical programming model for scheduling of nurses' labor shifts. *Journal of Medical Systems*, 36(2):491–496.

[270] Yunes T.H., Moura A.V., and de Souza C.C., (2005). Hybrid column generation approaches for urban transit crew management problems. *Transportation Science*, 39(2):273–288.

# CHAPTER 11

# Summary

**The personnel of an organization often has two conflicting goals. Individual employees like to have a good work-life balance, by having personal preferences taken into account, whereas there is also the common goal to work efficiently.**

**By applying techniques and methods from Operations Research, a subfield of applied mathematics, we show that operational efficiency can be achieved while taking personnel preferences into account. In the design of optimization methods, we explicitly consider that these methods should enable the business users to understand and effectively steer the outcomes of these methods.**

**Designing such methods, and applying these to personnel scheduling methods is at the core of the research in this dissertation.**

The content of this dissertation is summarized in this chapter.

*Chapter 2: Research Relevance and Outline*

In Chapter 2, we motivate that employee preferences should be carefully considered in personnel planning and scheduling. In service industries, especially in healthcare, personnel wages are a major part of the operational expenses. Hence, efficient personnel schedules help to control operational expenses. In addition, aging populations imply that on the one hand demand is increasing, and on the other hand that the relative size of the 'working population' becomes smaller, which stresses a need for efficient personnel scheduling. In this dissertation, we discuss various operations research methods and practice implementations that address requests and preferences of individual employees on different levels of planning and scheduling. In addition, Chapter 2 gives a short introduction into Operations Research, and provides a brief description of the research environment.

*Chapter 3: Terminology and Literature Survey*

Chapter 3 discusses how the literature considers preferences and characteristics of individual employees in personnel planning and scheduling decisions. Furthermore, it introduces a terminology for personnel planning and scheduling decisions, and provides an overview

of the various personnel preferences and characteristics that are considered in the literature. Next to this, Chapter 3 outlines how mathematical optimization methods incorporate these preferences and characteristics. Finally, in Chapter 3, we point to some interesting research directions and discuss how the research in this dissertation provides steps in those directions.

*Chapter 4: Cost-Efficient Staffing under Annualized Hours*

Chapter 4 studies how flexibility in workforce capacity can be used to efficiently match workforce capacity and demand. Flexibility in workforce capacity is introduced by the annualized hours regime. Annualized hours allow organizations to measure working time per year, instead of per month or per week, thereby allowing organizations to let employees work more hours in one week and less in another. An additional source of flexibility is hiring employees with different contract types, such as full-time, part-time, and min-max, and by hiring subcontractors.

In Chapter 4, we propose a mathematical programming formulation that incorporates annualized hours and shows to be very flexible with regard to modeling various contract types. The objective of the model is to minimize salary cost, thereby covering workforce demand, and using annualized hours. The model is able to address various business questions regarding tactical workforce planning problems, e.g., with regard to annualized hours, subcontracting, and vacation planning. In a case study for a Dutch hospital, we demonstrate that applying annualized hours potentially saves up to 5.2% in personnel wages annually.

*Chapter 5: Staffing under Annualized Hours Using Cross-Entropy Optimization*

In Chapter 5, a Cross-Entropy optimization implementation is proposed to solve an annualized hours model that is strongly related to the model of Chapter 4. The goal is to select a cost-efficient set of employees that is supposed to cover a given workforce demand, under the annualized hours regime.

Our experimental results show that Cross-Entropy optimization is efficient across a broad range of instances, where real-life sized instances are solved in seconds, which significantly outperforms a mathematical programming formulation that is solved with Cplex. In addition, the solution quality of Cross-Entropy closely approaches the optimal solutions obtained by Cplex. Thereby, our Cross-Entropy implementation offers an outstanding method for real-time decision making, for example in response to unexpected staff illnesses, and scenario analysis.

*Chapter 6: Shift Rostering Using Decomposition: Assign Days Off First*

Chapter 6 studies a two-phase decomposition approach to solve the shift rostering problem. The first phase creates a days off schedule, indicating working days and days off for each employee. The second phase assigns shifts to the working days in the days off schedule. This decomposition is motivated by the fact that personnel scheduling constraints are often divided in two categories: one that specifies constraints on working days and days off,

while the other specifies constraints on shift assignments. To assess the performance of the decomposition approach, we apply it to public benchmark instances, and compare this to an approach that solves the personnel scheduling problem directly. We use mathematical programming to solve the various shift rostering formulations. We also study the extension that includes night shifts, in addition to days off, in the first phase of the decomposition.

Chapter 6 presents a detailed results analysis, and analyzes the effect of various instance parameters on the decompositions' results. In general, the decompositions significantly reduce the computation time and produce good solutions for most instances.

*Chapter 7: Shift Rostering Using Decomposition: Assign Weekend Shifts First*

Chapter 7 introduces a shift rostering problem that surprisingly has not been studied in the literature: the weekend shift rostering problem. It is motivated by our experience that employees' shift preferences predominantly focus on the weekends, since many social activities happen during weekends. The weekend shift rostering problem addresses the rostering of weekend shifts, for which we have designed a problem specific heuristic. In this chapter, we consider the weekend shift rostering problem as the first phase of the shift rostering problem. To complete the schedule, the second phase assigns the weekday shifts using an existing algorithm. We discuss effects of this two-phase approach both on the weekend work schedule and on the schedule as a whole. We demonstrate that our method is effective both on real-life instances and on public benchmark instances. For situations where the weekend work schedule is one of the key determinants of the quality of the complete schedule, our two-phase approach shows to be effective when incorporated in a commercially implemented algorithm.

*Chapter 8: Shift Rostering from Staffing Levels: a Branch-and-Price Approach*

In Chapter 8, we outline an approach that creates work schedules directly from staffing levels. This in contrast with many scheduling methods that first create shifts based on staffing levels, and afterwards create work schedules from the set of created shifts. Our proposed approach offers flexibility with respect to incorporating employee preferences in the creation of work schedules. When creating work schedules directly from staffing levels, employee preferences can be considered effictively when defining shifts. To solve the underlying combinatorial optimization model, we compare a Branch-and-Price (B&P) formulation with a mathematical programming formulation. The mathematical programming approach outperforms B&P in most cases, but we believe B&P to be better able to handle extra scheduling and employee preference constraints.

*Chapter 9: An Iterative Improvement Heuristic to Support Self-Scheduling*

Chapter 9 studies a self-scheduling application. Self-scheduling is receiving more and more attention in the literature and in practice. With self-scheduling, employees propose their personal work schedule they prefer to work during a given planning horizon. However, these schedules often do not match with the staffing demand as specified by the organization.

Chapter 9, presents an approach to support creating feasible work schedules that uses the work schedules proposed by the employees as input and that aims to divide the burden of shift reassignments 'fair' among the employees. Computational results are discussed and we indicate how performance indicators of the resulting schedules can be influenced through various model parameters. The presented approach is flexible and easily extendable, since labor rule checks are isolated from the actual algorithm, which makes it easy to include additional labor rules in the approach. Moreover, our approach enables the user to make a trade-off between the quality of the resulting roster and the extent to which the planner is able to track the decisions of the algorithm.

*Conclusions*

This dissertation discusses various personnel planning and scheduling applications. Personnel preferences are an important topic throughout the dissertation. Although we reveal several mathematical challenges open for further research, we believe that the most important challenge lies with implementing these methods in practice. We think it is important that the design of these methods enable the business users to understand and effectively steer the outcomes of these methods. The research in this dissertation is based on practical studies or applications, and we encourage additional research in this direction. Many of the algorithm designs explicitly consider that the algorithms should enable business users to understand and effectively steer the outcomes of these methods. We believe that this aspect, combined with the incorporation of employee specific preferences, will be instrumental in the implementation success of personnel planning and scheduling algorithms.

CHAPTER 12

# Samenvatting (Dutch Summary)

**Het personeel van een organisatie heeft vaak twee tegenstrijdige belangen. Aan de ene kant wil ieder individu graag een goede balans tussen werk en privé door roosters en planningen te hebben waarin zijn persoonlijke wensen en voorkeuren meegenomen worden. Aan de andere kant is er een algemeen belang om efficiënt te werken.**

**Door technieken en methoden toe te passen uit de *Operations Research* (Nederlands: "Besliskunde"), een vakgebied binnen de toegepaste wiskunde, laten we zien dat rekening houden met voorkeuren van individuen niet ten koste hoeft te gaan van de efficiency. In het ontwerp van deze methoden nemen we expliciet mee dat gebruikers van deze methoden in staat moeten zijn de methoden te begrijpen en de uitkomsten effectief te kunnen sturen.**

**Het ontwerpen van dit soort methoden en het toepassen ervan op personeelsplanning en personeelsroostering vormt de kern van het onderzoek in dit proefschrift.**

De inhoud van dit proefschrift is samengevat in dit hoofdstuk.

*Hoofdstuk 2: Motivatie en overzicht*

In hoofdstuk 2 beschrijven we waarom het belangrijk is medewerkervoorkeuren mee te nemen in personeelsplanning. In de dienstverleningssector, met name in de gezondheidszorg, wordt het grootste deel van de operationele uitgaven gevormd door personeelskosten. Een efficiënte personeelsplanning draagt daarom bij aan het beheersen van operationele kosten. Door de vergrijzing wordt het belang van een efficiënte personeelsplanning verder onderschreven, omdat dit aan de ene kant zorgt voor een toenemende zorgvraag, maar aan de andere kant ook voor een relatief kleiner wordende 'werkende populatie'. Zoals we in hoofdstuk 1 beargumenteren is het voor een efficiënte personeelsplanning belangrijk voorkeuren van medewerkers expliciet mee te nemen in personeelsplanning. In dit proefschrift komen verschillende besliskundige modellen en praktijkimplementaties aan de orde die focussen op het meenemen van medewerkervoorkeuren in personeelsplanning.

Naast het onderzoeksonderwerp geeft hoofdstuk 2 een beschrijving van *Operations Research*. Daarnaast beschrijft hoofdstuk 2 de organisaties die betrokken zijn bij dit onderzoek.

*Hoofdstuk 3: Terminologie en literatuuroverzicht*

Hoofdstuk 3 geeft een overzicht van de literatuur op het gebied van personeelsplanning en personeelsroostering en beschrijft hoe deze literatuur voorkeuren en eigenschappen van individuele medewerkers meeneemt in personeelsplanning. We geven in dit hoofdstuk aan hoe en welke medewerkervoorkeuren en eigenschappen in de literatuur gemodelleerd worden. Daarnaast wordt in hoofdstuk 3 een terminologie voor personeelsplanningsbeslissingen geïntroduceerd. Ten slotte identificeren we interessante onderzoeksrichtingen en geven we aan hoe dit proefschrift bijdraagt in die onderzoeksrichtingen.

*Hoofdstuk 4: Kostenefficiënte capaciteitsplanning door een planmatige toepassing van jaarurensystematiek*

Hoofdstuk 4 onderzoekt hoe flexibiliteit in personeelsinzet gebruikt kan worden om vraag en aanbod van personeel op elkaar af te stemmen. De jaarurensystematiek biedt hierbij een vorm van flexibiliteit. De jaarurensystematiek laat toe dat de gewerkte uren van een medewerker gemeten worden op jaarniveau in plaats van op week- of maandniveau. Dit staat organisaties toe om medewerkers meer uren te laten werken in de ene week dan in de andere. Daarnaast bieden verschillende contractvormen, zoals fulltime, parttime, min-max en nuluren, naast uitzendkrachten, nog een andere bron van flexibiliteit.

In hoofdstuk 4 formuleren we een mathematisch programma dat zowel de jaarurensystematiek als verschillende contractvormen modelleert. De doelstellingsfunctie van het mathematisch programma minimaliseert salariskosten, onder de beperking dat de vraag naar personeel gewaarborgd moet zijn. Zoals we in het hoofdstuk laten zien is het model in staat ondersteuning te bieden voor het beantwoorden van verschillende tactische personeelsplanningsvraagstukken met betrekking tot jaarurensystematiek, inhuur van uitzendkrachten en vakantieplanning. In een casus voor een Nederlands ziekenhuis laten we zien dat het planmatig toepassen van jaarurensystematiek een potentiële jaarlijkse besparing van 5,2% op personeelskosten oplevert.

*Hoofdstuk 5: Kostenefficiënte capaciteitsplanning door een planmatige toepassing van jaarurensystematiek met behulp van Cross-Entropy optimalisatie*

In hoofdstuk 5 wordt een Cross-Entropy optimalisatie model geformuleerd dat net als hoofdstuk 4 een planmatige toepassing van jaarurensystematiek modelleert. Het doel van dit model is een kostenefficiënte selectie van medewerkers te maken, die samen de vraag naar personeel afdekt, waarbij gebruik gemaakt wordt van de flexibiliteit van de jaarurensystematiek.

Rekenresultaten laten zien dat Cross-Entropy optimalisatie op een brede set van testinstanties goede oplossingen geeft, waarbij instanties met een realistische omvang in enkele seconden worden opgelost, wat beduidend sneller is dan een mathematisch programma dat opgelost wordt met Cplex. Daarnaast ligt de oplossingskwaliteit van Cross-Entropy optimalisatie dichtbij de optimale oplossing van Cplex. Onze Cross-Entropy implementatie is daarmee een uitstekend instrument voor real-time beslissingsondersteuning, bijvoorbeeld bij onverwachte absenties, en voor scenario-analyses.

*Hoofdstuk 6: Dienstroostering via decompositie: plan vrije dagen eerst*

In hoofdstuk 6 introduceren en analyseren we een twee-fasen decompositiemodel om dienstroosters op te stellen. De eerste fase richt zich op het opstellen van een vrije dagen rooster, welke aangeeft op welke dagen medewerkers moeten werken en op welke dagen ze vrij zijn. De tweede fase wijst diensten toe aan de dagen waarop de medewerkers werken. Deze decompositie wordt gemotiveerd door het feit dat dienstroosterbeperkingen zich vaak laten splitsen in twee groepen: een groep die beperkingen specificeert ten aanzien van werkdagen en vrije dagen en een groep die beperkingen specificeert ten aanzien van specifieke dienst-toewijzingen. Om de kwaliteit van deze decompositie aanpak te toetsen passen wij deze toe op een aantal publieke benchmark instanties en vergelijken wij dit met een modelaanpak die een rooster creëert zonder decompositie. We gebruiken mathematisch programmeren om de verschillende dienstroosterproblemen op te lossen. Verder beschouwen we ook een decompositie variant die naast vrije dagen ook de nachtdiensten roostert in de eerste fase van de decompositie aanpak.

Hoofdstuk 6 analyseert de resultaten in detail en analyseert het effect van verschillende karakteristieken van de roosterinstanties op de resultaten van de decompositie aanpak. In het algemeen concluderen we dat de decompositie de oplossingstijd sterk verkort en dat voor de meeste instanties goede kwaliteit roosters worden gevonden.

*Hoofdstuk 7: Dienstroostering via decompositie: plan weekenden eerst*

Hoofdstuk 7 introduceert een roosterprobleem dat verrassend genoeg nooit eerder beschreven is in de literatuur: het weekenddienstrooster probleem. Het weekenddienstrooster probleem komt voort uit onze ervaring dat medewerkervoorkeuren voor een groot deel focussen op weekenden, aangezien veel sociale activiteiten in het weekend plaatsvinden. Het weekenddienstrooster probleem richt zich op het roosteren van weekenddiensten, waarvoor wij een probleem specifieke heuristiek hebben ontwikkeld. In dit hoofdstuk, beschouwen wij het weekenddienstrooster probleem als de eerste fase van het opstellen van dienstroosters. Om een compleet dienstrooster op te stellen, worden de doordeweekse diensten in een tweede fase geroosterd. Voor die stap gebruiken we een bestaand algoritme. We analyseren het effect van deze tweestapsmethode op het weekendrooster en op het rooster als geheel. We laten zien dat onze weekendrooster heuristiek goede resultaten behaald op praktijk instanties en publieke benchmark instanties. We hebben deze heuristiek geïmplementeerd in een commercieel algoritme, wat effectief blijkt te zijn wanneer de kwaliteit van het weekendrooster één van de belangrijkste graadmeters van het dienstrooster als geheel is.

*Hoofdstuk 8: Roosteren direct vanuit bezettingseisen: een Branch-and-Price toepassing*

In hoofdstuk 8 ontwikkelen we een roostermethode die dienstroosters creëert direct vanuit bezettingseisen. Dit in tegenstelling tot vele andere methoden die eerst diensten maken op basis van bezettingseisen en vervolgens roosters maken op basis van de set van gecreëerde diensten. Onze aanpak kan flexibel omgaan met specifieke medewerkervoorkeuren.

Wanneer roosters direct vanuit bezettingseisen worden opgesteld, kunnen medewerkervoorkeuren goed meegenomen worden bij het definiëren van diensten. Om het onderliggende combinatorische optimalisatieprobleem op te lossen vergelijken we een zogenaamde Branch-and-Price (B&P) formulering met een mathematisch programmeerformulering. De studie hier betreft met name een onderzoek of deze modellen werken voor dit type probleem. We zien dat mathematisch programmeren in veel gevallen beter presteert dan B&P, maar we verwachten dat B&P beter in staat is om te gaan met additionele roosteringsbeperkingen en medewerkervoorkeuren.

*Hoofdstuk 9: Een flexibele iteratieve verbeterheuristiek om dienstroosters op te stellen in zelfroosteren*

Hoofdstuk 9 behandelt een zelfroostertoepassing. Zelfroosteren is een concept dat meer en meer aandacht krijgt vanuit zowel de literatuur als de praktijk. Met zelfroosteren stellen medewerkers zelf hun persoonlijke roosters voor die zij willen werken. Echter alle persoonlijke roosters bij elkaar voldoen niet noodzakelijkerwijs aan de dienstbezetting die is gespecificeerd door de organisatie. Hoofdstuk 9 stelt een methode voor om roosters the maken die voldoen aan de gespecificeerde dienstbezetting op basis van door de medewerkers voorgestelde roosters. Deze methode heeft als doel diensten 'eerlijk' te herverdelen over de medewerkers. Aan de hand van rekenresultaten laten we zien hoe verschillende modelparameters effect hebben op de kwaliteit van het uiteindelijke dienstrooster. De voorgestelde aanpak is flexibel en makkelijk uit te breiden, omdat het controleren van arbeidstijdenregels geïsoleerd is van het wiskundige algoritme, wat het makkelijk maakt extra arbeidstijdenregels toe te voegen in ons model. Verder stelt het model de gebruiker in staat een afweging te maken tussen de kwaliteit van het resulterende rooster en de mate waarin de planner in staat is de beslissingen die genomen worden in het algoritme te volgen.

*Conclusies*

Dit proefschrift behandelt verschillende personeelplannings- en personeelroosteringstoepassingen. Medewerkervoorkeuren zijn hierin een centraal onderwerp. In dit proefschrift worden verschillende wiskundige uitdagingen geïdentificeerd. De belangrijkste uitdaging zal echter liggen bij het in de praktijk implementeren van besliskundige modellen. Bij het ontwerp van deze methoden vinden wij het belangrijk dat zij eindgebruikers in staat stellen deze methoden effectief te gebruiken en de uitkomsten ervan te kunnen sturen. Het onderzoek in dit proefschrift is gebaseerd op praktijkcasussen en wij moedigen anderen aan verder onderzoek in deze richting aan. De mate waarin wiskundige algoritmen om kunnen gaan met voorkeuren van individuele medewerkers zal bepalend zijn voor het implementatiesucces van deze algoritmen.

CHAPTER 13

# About the author

Egbert van der Veen was born in Leeuwarden, the Netherlands on November 20th, 1984. Egbert obtained his VWO diploma at CSG Liudger in Drachten in 2003. In 2007, he earned a Bachelor's of Science degree in Business Mathematics from the University of Groningen, and for his Bachelor's thesis he developed a method to determine an all-time speed skaters ranking. In 2009, he earned a cum laude Master's of Science degree in Business mathematics and a cum laude Master's of Science degree in Econometrics, Operations Research & Actuarial Studies, both at the University of Groningen. His Master's thesis was the basis for the research in Chapter 8 of this dissertation, in which an integration of two often separately addressed scheduling decisions is proposed.

In 2009, Egbert started his job as a personnel planning and scheduling consultant at ORTEC, and as a Ph.D. candidate at the University of Twente (UT), under the supervision of Richard Boucherie, Erwin Hans, and Bart Veltman. As a consultant at ORTEC, he is involved in the design, development and implementation of personnel planning and scheduling software and algorithms. Customer cases and practice applications initiated the various research topics in his Ph.D. research, which culminates with this dissertation.

CHAPTER 14

# List of publications


[1] Van der Veen E., Hurink J.L., Schutten J.M.J., and Uijland S.T., (2013). A flexible iterative improvement heuristic to support creation of feasible shift rosters in self-rostering. Memorandum 2016, Department of Applied Mathematics, University of Twente, Enschede
*(basis for Chapter 9)*

[2] Van der Veen E., and Hans E.W., (2013). Eerst weekend! Wiskunde in dienst van een sociaal leven. *STAtOR*, 14(1):16-19.

[3] Van Veldhoven S., Post G.F., Van der Veen E., and Curtois T., (2013). An assessment of a days off decomposition approach to personnel scheduling. Memorandum 2005, Department of Applied Mathematics, University of Twente, Enschede.
*(basis for Chapter 6)*

[4] Van der Veen E., Boucherie R.J., and Van Ommeren J.C.W., (2012). Optimal staffing under an annualized hours regime using cross-entropy optimization. Memorandum 1982, Department of Applied Mathematics, University of Twente, Enschede.
*(basis for Chapter 5)*

[5] Van der Veen E., Hans E.W., Post G.F., and Veltman B., (2012). Shift rostering using decomposition: assign weekend shifts first. Memorandum 1974, Department of Applied Mathematics, University of Twente, Enschede.
*(basis for Chapter 7)*

[6] Van der Veen E., Hans E.W., Veltman B., Berrevoets L.M., and Berden H.J.J.M., (2012). Cost-efficient staffing under annualized hours. Memorandum 1995, Department of Applied Mathematics, University of Twente, Enschede.
*(basis for Chapter 4)*

[7] Hulshof P.J.H., Boucherie R.J., Van Essen J.T., Hans E.W., Hurink J.L., Kortbeek N., Litvak N., Vanberkel P.T., Van der Veen E., Veltman B., Vliegen I.M.H., and Zonderland M.E. 2011. ORchestra: an online reference database of OR/MS literature in health care. *Health Care Management Science*, 14(4):383-384.

[8] Van der Veen E., and Veltman B., (2011). Rostering from staffing levels: a branch-and-price approach. *International Journal of Health Management and Information*, 2(1): 41-52.
*(basis for Chapter 8)*

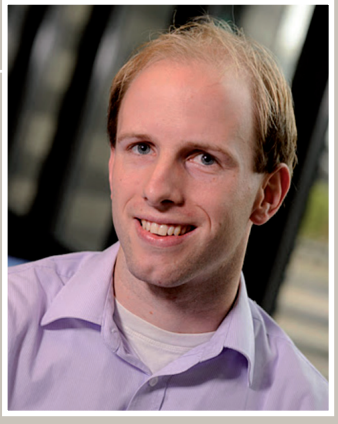
# Personnel Preferences in Personnel Planning and Scheduling

The personnel of an organization often has two conflicting goals. Individual employees like to have a good work-life balance, by having personal preferences taken into account, whereas there is also the common goal to work efficiently.

By applying techniques and methods from Operations Research, a subfield of applied mathematics, we show that operational efficiency can be achieved while taking personnel preferences into account. In the design of optimization methods, we explicitly consider that these methods should enable the business users to understand and effectively steer the outcomes of these methods.

Designing such methods, and applying these to personnel scheduling methods is at the core of the research in this dissertation.

## About Egbert van der Veen



During the time of this research, Egbert has been employed as a personnel planning and scheduling consultant at ORTEC, and as a Ph.D. candidate at the University of Twente. As such, he has been involved in the design, development and implementation of personnel scheduling algorithms and software.

Egbert holds a BSc degree in Business Mathematics, a MSc degree in Business mathematics, and a MSc Degree in Econometrics, Operations Research & Actuarial Studies.

UNIVERSITY OF TWENTE.

Department of Applied Mathematics, Stochastic Operations Research Group,
Center for Healthcare Operations Improvement and Research

ORTEC
OPTIMIZE YOUR WORLD