UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

KU LEUVEN
ARENBERG DOCTORAL SCHOOL
FACULTY OF ENGINEERING TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

TONI ISMAEL WICKERT

Personnel rostering: models and algorithms for scheduling, rescheduling and ensuring robustness

Thesis presented in partial fulfillment of the requirements for the degree of Doctor of Computer Science by UFRGS and the degree of Doctor of Engineering Technology (PhD) by KU Leuven.

Supervisors: Prof. Dr. Luciana S. Buriol
Prof. Dr. Greet Vanden Berghe
Co-supervisor: Dr. Pieter Smet

Porto Alegre - Brazil
November 2019

CIP — CATALOGING-IN-PUBLICATION

Wickert, Toni Ismael

Personnel rostering: models and algorithms for scheduling, rescheduling and ensuring robustness / Toni Ismael Wickert. – 2019.

143 f.

Supervisors: Luciana S. Buriol, Greet Vanden Berghe.

Co-supervisor: Pieter Smet.

Dual Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS and KU Leuven, Arenberg Doctoral School, Faculty of Engineering Technology, Department of Computer Science, Gent, Belgium, 2019.

1. Personnel Rostering. 2. Physician Rostering. 3. Cyclic Rostering. 4. Nurse Rerostering. 5. Robust Rostering. I. Buriol, Luciana S., supervisor. II. Vanden Berghe, Greet, supervisor. III. Smet, Pieter, co-supervisor. IV. Título.

ACKNOWLEDGMENT

UFRGS and KU Leuven are ranked among the top universities in the world, developing a Dual PhD in both universities was a great experience to me. Firstly, I would like to thank the supervisors Luciana S. Buriol, Greet Vanden Berghe and Pieter Smet for giving me the opportunity and for the guidance to the development of this thesis.

I was also lucky to have fantastic colleagues at UFRGS and the time that I have stayed in Gent - Belgium. Tulio, Fleur and Everton thank you for helping me when I arrived in Gent, you made me feel at home even being thousands of kilometer far away. In addition, thanks to Carlo, Thor, Alberto, Olinto, Federico, Thomas, Wim, Evert-Jan, Jan, Manos, Yihang, Reshma, Annelies, Farzaneh, Michiel and Luke for the friendship, talks and nice lunchtimes.

Luke thank you for your revisions and editorial consulting. You have the ability to make computer scientists writing texts such that they are much easier to understand.

Thank you to my colleagues at Hospital de Clínicas de Porto Alegre (HCPA) and my coworkers Márcio Boniatti, José Miguel Dora which significantly contributed such that the present research could be applied in a real-world scenario.

I also would like to thank the examination committee composed by Brecht Cardoen, Haroldo Gambini Santos, Karl Hendrickx, Marcus Ritt, Tony Wauters and Mark Versteyhe for their valuable feedback which helped to improve this thesis.

Finally, thank you to my family, wife Jana, brother Lucas, parents Rudi and Solange which always encouraged me to study.
Davi, you'll be always my baby!
Daddy loves you!

November 2019 – Toni I. Wickert

ABSTRACT

Existing academic literature contains a significant number of publications which address personnel rostering problems. Providing a variety of combinatorial optimization techniques such as metaheuristics, integer linear programming and hybrid algorithms developed to approach such problems. Despite this progress in recent decades, a considerable number of institutions continue to prepare and organize their rosters manually.

There are many advantages in automating the generation of rosters using these techniques. These include *(i)* cost-saving: reduction of both the overtime and time needed to prepare and organize rosters, thereby enabling planners to work on other tasks, *(ii)* fairness: decisions follow rules based on some predefined parameters, improving employees satisfaction concerning their working schedule with a better balance of unpopular shifts, *(iii)* possibility of rerostering: disruptions are very hard for humans to solve due to time pressure and the constraints incurred by both the initial scheduling and rescheduling problem.

The present thesis addresses four primary shortcomings in academic literature by providing: *(i)* an integer programming model based on a real-world scenario and a matheuristic to generate results in short computation times to replace manual rostering, *(ii)* an effective integer programming model for cyclic rostering considering both academic and real-world scenarios which generates state-of-the-art results, *(iii)* new rerostering strategies for repairing disruptions in complex multi-skilled employee scenarios, and *(iv)* a metric for quantifying the robustness of rosters.

In addition to these contributions, this thesis also focuses on additional issues which should be considered by researches in the future development of solving methods to maximize the chances of their application in practice.

Keywords: Personnel rostering, physician rostering, cyclic rostering, nurse rerostering, robust rostering, integer programming, matheuristic.

Escalas de funcionários: modelos e algoritmos para escalonamento, reescalonamento e garantia de robustez

RESUMO

A literatura acadêmica possui um número significativo de publicações que abordam problemas de escala de pessoal. Além disso, uma variedade de técnicas de otimização combinatória, como metaheurísticas, programação linear inteira e algoritmos híbridos foram desenvolvidas para abordar tais problemas. Apesar deste progresso nas últimas décadas, um número considerável de instituições continua preparando e organizando suas escalas manualmente.

Existem diversas vantagens em automatizar a geração de escalas usando essas técnicas. Isso inclui *(i)* redução de custos: diminuição tanto de horas-extras quanto do tempo necessário para preparar e organizar as escalas, permitindo que os escalonadores trabalhem em outras tarefas, *(ii)* justiça: as decisões seguem regras baseadas em alguns parâmetros pré-definidos, melhorando a satisfação dos funcionários em relação ao seu horário de trabalho, com um melhor equilíbrio entre os turnos impopulares, *(iii)* reescalonamento: as infactibilidades das escalas são muito difíceis para os humanos resolverem devido à pressão do tempo e às restrições incorridas tanto pelo problema de escalonamento inicial quanto pelo de reescalonamento.

Esta tese aborda quatro ausências primárias na literatura acadêmica, fornecendo: *(i)* um modelo de programação inteira baseado em um cenário do mundo real e uma matheurística para gerar resultados em tempos computacionais curtos para substituir o escalonamento manual, *(ii)* um modelo de programação inteira eficaz para escalonamento cíclico, considerando cenários acadêmicos e do mundo real que geram resultados estado da arte, *(iii)* novas estratégias de reescalonamento para reparar infactibilidade em escalas considerando cenários complexos de funcionários multi-qualificados, *(iv)* uma métrica para quantificar a robustez das escalas.

Além dessas contribuições, esta tese também foca em questões adicionais que devem ser consideradas no futuro desenvolvimento de métodos de solução para maximizar as chances de sua aplicação na prática.

Palavras-chave: Escalonamento de funcionários, escalonamento de médicos, escalonamento cíclico, reescalonamento de enfermagem, escalonamento robusto, programação

inteira, matheurística.

LIST OF ABBREVIATIONS AND ACRONYMS

CP	Constraint Programming
CPU	Central Processing Unit
F&O	Fix-and-Optimize
GA	Genetic Algorithm
GPU	Graphics Processing Unit
HCPA	Hospital de Clínicas de Porto Alegre
INRC-I	International Nurse Rostering Competition I
INRC-II	International Nurse Rostering Competition II
IP	Integer Programming
MIP	Mixed Integer Programming
NRP	Nurse Rostering Problem
NRRP	Nurse Rerostering Problem
OFV	Objective Function Value
PRP	Physician Rostering Problem
SMT	Satisfiability Modulo Theories
STL	Subproblem Time Limit
TL	Time Limit
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

According to a study conducted by the United Nations Department of Economic and Social Affairs, the number of people aged over 60 years will grow by 56% between 2015 and 2030. In absolute numbers, this constitutes an increase from 901 million to 1.4 billion. Twenty years later, by 2050, the elderly population is expected to double its size compared to 2015 and reach a total of 2.1 billion (UNITED NATIONS, 2015). This aging population poses governments around the world with significant challenges, particularly in relation to retirement and investment in healthcare programs.

Despite this demographic shift, the number of healthcare staff – such as nurses, physicians, dentists, pharmacists and psychologists – who provide care for the aging population, is not projected to grow at the same rate. The increased demand associated with such professionals can end up severely compromising both patient care and employee job satisfaction (DONNELLY, 2017; BBC, 2017). Efficient use of these limited resources is therefore essential for both patients and employees.

Existing academic literature contains a large body of work regarding personnel rostering (ERNST et al., 2004). Burke et al. (2004) and Erhard et al. (2018) provided a general analysis of nurse and physician rostering literature, respectively. Advanced solving techniques were developed including a rotation-based branch-and-price method (LEGRAIN; OMER; ROSAT, 2019), Variable Neighborhood Search (VNS) to accelerate the column generation (GOMES; TOFFOLO; SANTOS, 2017) and matheuristics (SANTOS et al., 2014; CROCE; SALASSA, 2014).

Despite these significant publications and the existence of solving methods which generate excellent results in limited computation time, many institutions continue to organize their rosters manually. This not only demands considerable effort on the part of human planners but in many cases the rosters generated are of poor quality. This thesis investigates several rostering scenarios considering a range of problems to arrive at a suite of usable methods in practice. These include rostering (normal and cyclic), rerostering and robust rostering. In addition, this thesis aims to answer the following research question: which elements in terms of constraints, solving technique and type of rosters are important to consider when aiming to deploy automated decision support systems in practice? To answer this question this thesis will study four complementary staff rostering problems, namely: physician rostering, cyclic rostering, nurse rerostering and robust rostering.

Hospital de Clínicas de Porto Alegre (HCPA) is located in the south of Brazil and motivated the study regarding physician rostering. The hospital's managers reported a series of problems occurring with their manual scheduling organization. These include the long time required to organize and check the accuracy of the rosters, elevated overtime as well as an inefficient balance of working hours both in terms of overtime and working hours during non-business days between physicians. In addition, improper rostering decisions due to not consulting the data from previous rostering periods were incurred, examples of which include invalid shift successions and maximum consecutive working days.

Such a scenario motivates the investigation of existing literature concerning physician rostering problems as well as the nurse rostering problem, due to its similarities. The inexistence of a mathematical formulation that covers all the constraints provided by the hospital managers motivated the development of a new integer programming (IP) formulation such that the roster resulting from solving the IP model is acceptable and can be used in practice at the hospital. However, can this IP formulation be addressed by standalone solvers and generate good results within acceptable computational runtimes or is it necessary to develop heuristic methods? Moreover, paid solvers require additional investment in licenses, which raises the question of how an open-source solver would fare.

Some healthcare and industrial problems involve a type of roster where the scheduled pattern repeats after a certain time period, thereby resulting in cycles. These cycles means that such rosters are naturally fair between employees. However, while previously introduced cyclic models could not include complex constraints due to their model variables or the resulting lengthy computation times, the present research proposes a general model which is capable of capturing all the complex constraints required in practice. Nevertheless, how do solvers perform when addressing such complex multi-skilled cyclic rostering problems? In terms of solving time, is this general model still competitive with other methods proposed in existing literature?

Uncertainty concerning employee availability is inherent in every personnel rostering problem. Common causes of employee absenteeism include heavy workloads, childcare obligations and illness (FORBES, 2013). Such absences may render an existing roster infeasible and difficult for human planners to remedy with last-minute changes. The fact that planners are under stress in such situations motivates the need to develop automated rerostering approaches which not only provide updated rosters very quickly but which also solve the problem more efficiently than humans are capable to do under such stressful conditions. Several novel strategies for rerostering based on relaxations of different problem parameters including soft constraints and the rescheduling horizon are investigated. For example, the difference in terms of solving time and solution quality is evaluated when the full scheduling horizon is considered and when only a limited part is taken into account. Existing literature has previously only investigated simplified scenarios considering single-skilled employees, mostly focused on metaheuristics approaches. The question then arises as to whether it would still be possible to address such problems using standalone solvers for large instances when considering multi-skilled employees?

Despite the proposed rerostering method being capable of repairing disrupted rosters, negative effects are unavoidable when using such a reactive method. This includes last-minute changes concerning employees' personal organization and high associated costs, for example, with paid overtime required to remedy employee shortages. The generation of robust rosters is therefore crucial to minimize the occurrence of these negative factors. Academic literature, however, currently lacks a metric for measuring the robustness of both single- and multi-skilled staff rosters. Therefore, would it be possible to develop a general metric for quantifying and enforcing the robustness level of personnel rosters in advance? If so, what are the operational costs variations associated with enforcing robustness in a roster?

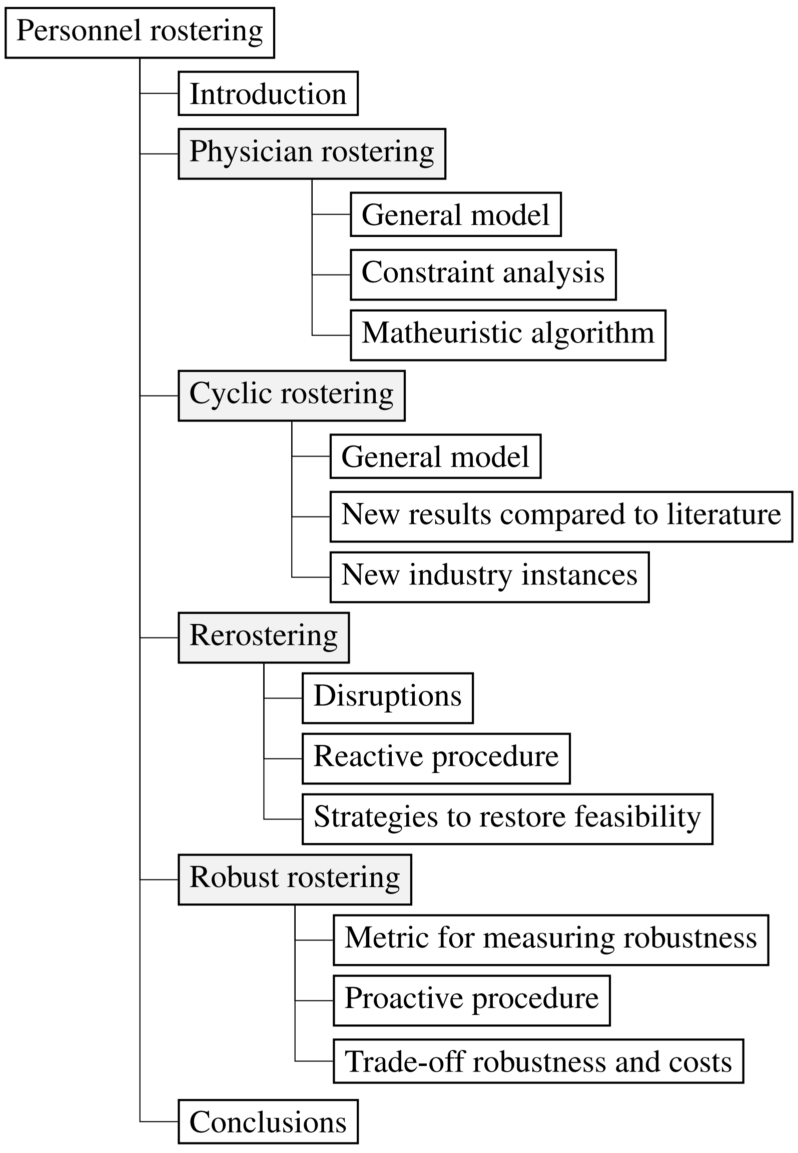
All the approached problems include at least a subset of series constraints: invalid shift successions, the minimum and maximum number of consecutive working days, days off or working days at the same shift. These constraints have a significant influence on the computational complexity of rostering problems (SMET, 2018). Moreover, problems such as the rerostering feasibility and cyclic staff rostering were proven NP-complete by Moz and Pato (2007) and Bartholdi (1981), respectively.

The present thesis provides scientific contributions by proposing and combining a set of methods for optimizing staff allocations. These include: *(i)* rostering (normal and cyclic): where general models are proposed and solutions still can be obtained in acceptable computational time, *(ii)* reactive procedures: exploring different strategies for solving disrupted rosters and *(iii)* proactive procedures: by proposing a metric for quantifying robustness and determining the best robustness levels which should be enforced in order to arrive at sufficiently robust rosters. In addition to these theoretical contributions, applying the proposed methods in real-world scenarios demonstrated the benefits of approaching the essential problem of reduced available resources both in terms of budget and healthcare staff.

1.1 Thesis structure and organization

Figure 1.1 provides a diagrammatic overview of the general thesis structure and the primary elements addressed in each problem. Note that each problem type, highlighted in gray, represents an independent chapter.

Figure 1.1: Thesis structure and organization.



The remainder of the thesis is organized as follows. Chapter 2 introduces the studied physician rostering problem, detailing the similarities and differences between the constraints found in both physician and nurse rostering. An integer programming model is proposed and results using real-world instances provided by Hospital de Clínicas de Porto Alegre (HCPA), Brazil, are presented. Chapter 3 details the cyclic rostering problem and a general integer programming formulation employed to address both literature and industry instances. Chapter 4 presents the nurse rerostering problem, including a general IP model as well as strategies for solving the rerostering problem using the proposed integer programming formulation and a Variable Neighborhood Descent (VND) algorithm. Chapter 5 provides a metric for quantifying the robustness of staff rosters in addition to the costs associated with ensuring certain robustness levels in rosters. Finally, chapter 6 presents the conclusions and outlines directions for future research.

2 PHYSICIAN ROSTERING

2.1 Introduction

The Physician Rostering Problem (PRP) which aims to schedule physicians qualified in one or more specialties has obtained increased academic attention. Erhard et al. (2018) provide a thorough overview of the existing literature. Problems were classified as staffing problems, focusing on how the required size of the workforce is defined; rostering problems, where the main objective is the generation of the rosters; and re-planning problems, addressing short-term adjustments to already scheduled physicians.

Similarly to the PRP the Nurse Rostering Problem (NRP) is among the most common problems found related to personnel scheduling. There is a large number of publications approaching problems related to specific hospitals such as Burke, Li and Qu (2010), Petrovic and Vanden Berghe (2012) and Burke et al. (2006). Supplementary to such publications, two competitions, the First and Second Nurse Rostering Competition (INRC-I) (HASPESLAGH et al., 2014) and (INRC-II) (CESCHIA et al., 2019), have been organized. For these competitions, a common set of instances was proposed, making it easier to compare solving techniques and stimulating the research on algorithms for solving the NRP. The INRC-I addressed a single-skilled static problem, where the entire planning horizon must be solved at once. By contrast, the INRC-II approached a multi-skilled problem and each week of the planning horizon must be solved separately. Meaning that the final roster is a combination of separate solutions. Each solving method was free to decide how to deal with historical data and the uncertainty of future data.

Clearly, the INRC-II has made significant progress compared to the INRC-I, by trying to specify a problem-solving environment that is closer to real-world conditions by way of including multi-skilled employees and considering the history of the previous roster. Nevertheless, these models are still lacking generality. For example, there is no possibility to specify the balance of working hours between the scheduled employees or a preferred skill if an employee has more than one. In addition, only a fixed problem was proposed without the option of turning off certain constraints or changing them from hard to soft (or vice versa). Such changes would render solving methods more flexible and, therefore, heighten the chances of them actually being used in real-world scenarios.

This work approaches the PRP based on the data provided by Hospital de Clínicas de Porto Alegre (HCPA), Brazil. The objective is to answer the following research question: Is recent academic progress relevant such that there is a possibility to generalize an IP formulation to solve a hospital's specific problem? The methodology used to answer this includes a basic integer programming model developed based on initial requirements provided by the hospital. Afterward, managers provided feedback and report back with improvements that should be implemented to obtain a better balance of working hours between the scheduled physicians. The combination of both basic and extended model resulted in a general model that attended all the hospital requirements. Another contribution lies in the fix-and-optimize (F&O) matheuristic that generates good results in acceptable computation time limits. The F&O matheuristic was necessary to address the most challenging instances when standalone solvers were incapable of generating good results within the imposed time limit.

Computational experiments are conducted using real-world instances provided by HCPA. Detailed insights are presented regarding the constraints violation that improved the balance of working hours between the physicians. In addition, to enable an academic validation, both the IP formulation and the F&O matheuristic were compared to the heuristic Late Acceptance Hill Climbing (LAHC) developed by Sanchotene and Buriol (2018). Logs of the experiments, instances and results are available on-line[1].

The remaining chapter is organized as follows. Section 2.2 presents the literature review. Section 2.3 details the definition of the studied PRP and compares its constraints against those in the existing literature. Section 2.4 provides the basic and extended integer programming formulation for the PRP. Section 2.5 introduces the F&O matheuristic applied to solve the PRP instances. Section 2.6 details the computational results, while Section 2.7 discusses the conclusions.

2.2 Literature Review

Due to similarities between the PRP and the NRP, this section details related literature regarding both problems. Moreover, solving methods that were successfully applied to solve instances proposed for the INRC-I and INRC-II are also detailed. Since there is a large body of publications related to the NRP, we limited the scope of the literature to only those solving methods which address the instances from INRC-I and INRC-II.

Sanchotene and Buriol (2018) proposed a heuristic for the physician rostering problem. The method is divided into two phases. In the first phase, a constructive heuris-

[1]<http://www.inf.ufrgs.br/~tiwickert/download/2017/physician>

tic is employed to generate a feasible solution. Afterward, the LAHC heuristic is executed to improve the solution. Computational experiments were conducted utilizing the same instances used in this research. Results demonstrated that the LAHC heuristic obtained slightly better results for instances with 150 physicians, while the present research achieved better results for instances with up to 100 physicians.

An Integer Programming (IP) model was developed by Bruni and Detti (2014) to address a real-world physician rostering problem of an Italian university hospital. The IP model satisfies all service requirements and contractual agreements (including rest periods and annual leave) while trying to respect, as much as possible, employee preferences. Particular attention is paid to workload balancing. Stolletz and Brunner (2012) addressed a PRP with flexible shifts, which have a minimum duration and can begin at any time during the day. In addition, fair distribution of working hours is addressed. The problem was solved using a decomposition heuristic where the entire problem is broken down into weekly subproblems. Computational experiments demonstrated improved results when compared to previous research (BRUNNER; BARD; KOLISCH, 2009). Brunner and Edenharter (2011) developed a column generation method to tackle the PRP of an anesthesia department in an 1100-bed hospital. This procedure was necessary because a standalone MIP solver was incapable of solving weekly subproblems to optimality within several hours. A real-world problem in the surgery department of a large government hospital in Singapore was approached by Gunawan and Lau (2013) using a heuristic. Instead of assigning physicians to shifts, they are assigned to a set of tasks incorporating a large number of constraints and complex physician preferences. Constraint programming combined with local search and genetic algorithms was proposed by Rousseau, Pesant and Gendreau (2002) using data provided by two Canadian hospitals. A genetic algorithm to schedule physicians for emergency rooms was proposed by Puente et al. (2009), while a combined emergency and surgery scheduling problem was addressed by Huele and Vanhoucke (2014).

Compared to existing literature, the primary differences of this study are the shift regime which does not follow the same pattern on weekends. Furthermore, a larger number of workload balancing constraints are required to equilibrate overtime, worked day and night shifts as well as worked hours on non-business days.

The following literature addresses relevant solving methods applied to the NRP. The INRC-I winner method proposed by Valouxis et al. (2012) used a two-stage approach to decompose the problem in manageable parts which can be solved to optimality using a mathematical solver. The second-placed solving method (BURKE; CURTOIS, 2011) and third-placed (BILGIN et al., 2010) developed methods based on branch-and-price and hyperheuristics, respectively. After the end of the competition, Santos et al. (2014) developed a MIP model and employed heuristic techniques to decompose the problem. These subproblems are generated by fixing a subset of days or shifts and the resulting subproblem is solved using a MIP solver. Computational results demonstrated that several best-known solutions were improved.

The INRC-II winner method formulated the problem as a network flow model (RÖMER; MELLOULI, 2016). The second-placed team modeled the problem utilizing integer programming. Each column of the IP corresponds to a rotation, that is, a sequence of consecutive worked days for a nurse and not a complete individual roster. This procedure is called rotation-based branch and price (LEGRAIN; OMER; ROSAT, 2019).

In both competitions, the best-ranked methods include at least one component based on mathematical models outperforming approaches based only on metaheuristics. After the end of the competition, many methods were applied to the static version of the INRC-II instances. This means that the entire planning horizon (4 or 8 weeks depending on the instance) is solved at once. During the competition it was mandatory to solve each week separately, that is, the solving method must deal with the uncertainty of future data solving sequential weeks and having the complete result only after the last week had been solved. These methods include a Variable Neighborhood Search (VNS) to accelerate the column generation procedure proposed by Gomes, Toffolo and Santos (2017), and the same procedure developed by the second-placed (LEGRAIN; OMER; ROSAT, 2019) but now applied to solve the entire planning horizon at once.

Nurse rostering problems usually include a combination of constraints minimum and maximum number of consecutive working days, days off and working days on the same shift. This set of consecutiveness constraints render the problem particularly challenging to solve (SMET, 2018). This is not the case for the present PRP which only has consecutiveness constraints maximum number of consecutive working days and working days on the same shift.

In terms of solving technique, the present fix-and-optimize matheuristic is similar to the method proposed by Santos et al. (2014) and Croce and Salassa (2014). The main difference with respect to Santos et al. (2014) is that we decompose the problem into subsets of physicians, days and shifts similar to Croce and Salassa (2014) which in our preliminary experiments generated better results compared against using only days and shifts decompositions. The reason for choosing this solving technique in contrast to those proposed by Gomes, Toffolo and Santos (2017) and Legrain, Omer and Rosat (2019) is due to good results on the instances proposed for the INRC-I. Moreover, as the problem size will increase in the near-future, heuristic methods are more appropriate for these large instances.

2.3 Problem definition

The general physician rostering problem aims to assign physicians to shifts for each day during a scheduling horizon. The objective is to minimize the cost associated with the violation of the soft constraints such as the maximum number of consecutive working days, overtime and physician's preferences. The case-specific PRP addressed in the present study also includes the concept of locations which means that physicians may be allowed to work at specific locations and not at others within the hospital unit. In addition to the minimization of the overall cost and violation of physician preferences, this study also introduces constraints to generate a fair distribution of working hours between the physicians.

2.3.1 Basic model

Throughout the model definition, non-business days refer to Saturdays, Sundays and holidays. Working days on weekdays, meanwhile, refer to weekdays which are not holidays. Constraints are either hard (H) or soft (S):

H1. A physician can be assigned to at most one shift per day during weekdays;

H2. Minimum number of physicians per day/shift/location;

H3. Maximum number of physicians per day/shift/location;

H4. A physician must be assigned to both Early and Late shifts, or a Night shift, or have a day off on non-business days;

H5. Invalid shift succession;

H6. A physician can be unavailable for some shifts or days;

H7. When working both Early and Late shifts, they must be worked at the same location;

H8. A physician must be qualified to work at specific locations;

S1. Maximum number of consecutive assignments to the same shift;

S2. Maximum number of consecutive assignments;

S3. Undesired working day or shift;

S4. Complete weekend, that is, a physician ideally works both Saturday and Sunday;

S5. Minimum number of assignments over the planning horizon (according to the working contract);

S6. Maximum number of assignments over the planning horizon (according to the working contract);

S7. Maximum number of working weekends.

Constraints included in the basic model are commonly found in the existing literature. For example, Constraints (H2), (H5) and (S1)–(S7) are present in the instances proposed for the INRC-I and INRC-II. Constraint (H8) is present in the INRC-II, but in a slightly different manner. The difference is that for the INRC-II nurses have specific skills, for example, a nurse can have skills caretaker and trainee, while here physicians are qualified to be assigned to particular locations within a working unit. Another difference is that in the case of the NRP a head nurse can assume the work of less qualified nurses, while this is not the case in the PRP. Constraint (H1) is commonly found in the literature for all business and non-business days (BEAULIEU et al., 2000), (HASPESLAGH et al., 2014) and (CESCHIA et al., 2019). However, the present PRP has an exception because this constraint is only valid for business days. During weekends and holidays, working both Early and Late shifts is mandatory. Moreover, these shifts must be worked at the same location.

In comparison with existing PRP literature, Bruni and Detti (2014) described heterogeneous working contracts as well as similar 12-hours working shifts. However, these 12-hours shifts are valid for the entire week and not only for non-business days. Brunner, Bard and Kolisch (2009) and Brunner and Edenharter (2011) contrasting the presented constraints where each shift has a fixed start and end time, provided models where shifts can start at any time during a working day and have an arbitrary duration. Another observation is the restricted number of consecutiveness constraints of this model compared for example, to those proposed for the INRC-I and INRC-II. While this model has only two consecutiveness constraints (maximum number of consecutive working days and worked days on the same shift), INRC-II problem has six consecutiveness constraints.

2.3.2 Extended model

The primary objective of adding this set of constraints is to improve the balance of working hours between physicians. For example, an egalitarian distribution of overtime and working hours on non-business days is desired. The following constraints were incorporated into the basic model:

H9. Minimum number of assignments on non-business days;

H10. Maximum number of assignments on non-business days;

H11. Maximum number of monthly assignments on weekdays;

S8. Preferred number of assignments on non-business days;

S9. Preference for a location;

S10. Equal day and night working hours during weekends;

S11. Maximum weekly assignments during day (Early and Late) or Night shifts;

S12. Assign physicians to the minimum possible number of locations.

Compared to the literature, Salassa and Vanden Berghe (2012) proposed a solving method where the basic idea is to have a long term balance in terms of workload between employees. Similarly to constraints (H9)–(H11), (S8) and (S10), Stolletz and Brunner (2012) addressed fairness concerning the distribution of working hours, both in terms of work less-than-contracted hours and overtime by penalizing deviation from a pre-established minimum and maximum working times. The same way, the model introduced by Stolletz and Brunner (2012) has the possibility of modeling employee-specific preferences or restrictions. This is possible in the presented model through constraints (S9) and (S3) from the basic model. Constraint (S11) concerns a particular case where some physicians have a small number of contracted hours per month. This constraint ensures that such physicians do not work all their contracted hours during the two first weeks of the month.

2.3.3 Shifts organization and example of a roster

Table 2.1 provides a simple roster presented in the physician-day view, where rows represent the physicians and columns the days. The example has three physicians (Physician1, Physician2 and Physician3), three shifts (Early [E], Late [L] and Night [N]), and three locations (In-patient Units [1], [2] and [3]). Day shifts (Early and Late) are 6 hours, while Night shifts are 12 hours long. The shifts are organized as follows:

- Early (6h): 08-14h;
- Late (6h): 14-20h;
- Night (12h): 20-08h.

When a physician works a Night shift, it is considered as two worked shifts. This procedure is necessary to calculate the total number of working shifts during the planning horizon. As an example, Physician1 works on Monday the Late shift at In-patient Unit1, on Tuesday on Late shift at In-patient Unit2, and on Saturday/Sunday on both day shifts (Early and Late) at In-patient Unit1. Dashes represent days off. Day shifts (Early and Late) on non-business days that must be worked together (enforced by H4) and Nights shifts with 12 working hours, which need to be considered twice in order to calculate the total number of worked shifts per roster, are highlighted in gray.

Table 2.1: Example of a roster with seven days and three physicians.

Physician	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Physician1	L[1]	L[2]	N[1]	–	–	E/L[1]	E/L[1]
Physician2	N[2]	N[3]	–	–	–	N[3]	N[2]
Physician3	–	E[1]	–	L[2]	N[1]	–	–

2.4 Integer Programming Formulation

This section introduces an integer programming formulation considering both hard and soft constraints for the PRP. Table 2.2 presents the indices (first column) used to identify the variables associated with their respective constraints. The second column describes the constraints, the third column (Id) is the identifier, while the last column presents the weight per unit of violation of each constraint. For example, in the objective function in Eq. 2.1 the first term $c^3_{nd}\omega^3$ refers to the variables related to the maximum number of consecutive assignments to the same shift (identified by index 3 in Table 2.2) multiplied by its respective weight ω^3 (15).

Table 2.3 presents the sets, decision and auxiliary variables employed in the formulation.

Table 2.2: Indices used to associate each soft or hard constraint with their respective variable or input data in the formulation.

Index	Constraint description	Id	Weight (ω^i)
Basic model			
1	Minimum number of physician per day/shift/location	H2	-
2	Maximum number of physician per day/shift/location	H3	-
3	Maximum number of consecutive assignments to the same shift	S1	15
4	Maximum number of consecutive assignments (worked days)	S2	30
5	Physician undesired working day/shift	S3	10
6	Complete weekend	S4	30
7	Minimum number of assignments over the planning horizon	S5	20
8	Maximum number of assignments over the planning horizon	S6	20
9	Maximum number of working weekends	S7	30
Extended model			
10	Minimum number of assignments on non-business days	H9	-
11	Maximum number of assignments on non-business days	H10	-
12	Maximum number of assignments on working days	H11	-
13	Number of assignments below the ideal on non-business days	S8	100
14	Number of assignments above the ideal on non-business days	S8	100
15	Priority per location (physicians may express priority for one or more location)	S9	15
16,17	Equilibrium between day and night working hours during weekends	S10	10
18	Maximum weekly working assignments day (Early and Late) or Night shifts	S11	10
19	Assign physicians to the minimum possible number of locations	S12	50

Table 2.3: Indices, sets and variables used in the mathematical formulation for both the basic and extended models.

Symbol	Definition
Input Data	
$n \in N$	n is the index of the physician, and N is the set of all physician indices;
$d \in D$	d is the index of the day, and D is the set of all day indices;
$d \in \tilde{D}$	d is the index of the non-business day, and $\tilde{D}$ is the set of all non-business day indices;
$s \in S$	s is the index of the shift, and S is the set of all shift indices;
$k \in K$	k is the index of the location, and K is the set of all location indices;
$(n,k) \in L$	set containing the pairs of forbbiden locations of physician n at location k;
$(n,k) \in P$	set containing the pairs where the physician n has a preference not to work at location k;
$(n,d,s) \in R$	set containing triples with unavailable physician n on day d, and shift s;
$(n,d,s) \in U$	set containing triples with the undesired working day d, and shift s of physician n;

$(s', s'') \in \hat{S}$	set containing the pairs of invalid shift successions;
$w \in W$	w is a Saturday index and W the set of all Saturday indices not including the last Saturday if it is the last day of the month;
$w \in \tilde{W}$	w is the week index, and $\tilde{W}$ the set of all week indices;
α^i_{dsk}	$i \in \{1, 2\}$, that is, the minimum and maximum number of physicians per day d, shift s, and location k;
β^i_n	limits of constraints with indices $3, 4, 7, 8, 9, 10, 11, 12$ and 13 in Table 2.2. That is, the maximum number of consecutive assignments to the same shift (3), maximum number of consecutive working days (4), minimum/maximum number of assignments over the planning horizon (7, 8), maximum number of working weekends (9), minimum/maximum number of assignments on non-business days (10, 11), maximum number of assignments on working days (12), and ideal number of assignments on non-business days (13);
ω^i_n	weight for violating the lower and upper bounds of soft constraint i for physician n;
s_e	Early shift index;
s_l	Late shift index;
s_n	Night shift index.

Decision Variables

$x_{ndsk} \in \{0, 1\}$	1 if physician n is allocated to shift s, day d, and location k, and 0 otherwise;
$y_{nw} \in \{0, 1\}$	1 if physician n works weekend w, and 0 otherwise;
$z_{nd} \in \{0, 1\}$	1 if physician n works both the Early and Late shifts on day d, and 0 otherwise;
$o_{nd} \in \{0, 1\}$	1 if physician n is allocated to work on day d, and 0 otherwise;
$q_{nk} \in \{0, 1\}$	1 if physician n works at location k, and 0 otherwise.

Auxiliary Variables

$c^i_{nd} \in \mathbb{N}$	number of violations of the soft constraint with indices $i \in \{3, 4\}$ in Table 2.2, for physician n on day d;
$g^5_{nds} \in \mathbb{N}$	number of violations of the soft constraint with index 5 in Table 2.2, for physician n on day d, shift s;

$h^{6}_{nw} \in \mathbb{N}$	number of violations of the soft constraint with index 6 in Table 2.2, for physician n on weekend w;
$j^{i}_{n} \in \mathbb{N}$	number of violations of the soft constraint with indices $i \in \{7,\dots,9,13,14,16,17,19\}$ in Table 2.2, for physician n;
$m^{15}_{nk} \in \mathbb{N}$	number of violations of the soft constraint with indices 15 in Table 2.2, for physician n at location k;
$l^{18}_{nws} \in \mathbb{N}$	number of violations of the soft constraint with index 18 in Table 2.2, for physician n on week w;

Although rosters are typically organized with a planning horizon of one month, data from the previous month is important to avoid infeasible solutions. For example, if physicians work night shifts on the last day of the previous month, they cannot work early or late shifts on the first day of the current month. To avoid such situations, before the solving method starts the border data from the previous month is read, that is, the total number of assignments, last assigned shift type, number of consecutive assignments of the last shift type, and number of consecutive worked days. These data is necessary to ensure or calculate the violation of constraints: invalid shift type succession (H5), maximum number of consecutive assignments to the same shift (S1), maximum number of consecutive working days (S2) and complete weekend (S4).

2.4.1 Basic model

In this section we describe the basic model, that is, which includes those constraints which are most likely to be found in many applications.

$$\textbf{Minimize:} \quad \sum_{n\in N}\sum_{d\in D}\sum_{i\in\{3,4\}} c^{i}_{nd}\omega^{i} + \sum_{n\in N}\sum_{d\in D}\sum_{s\in S} g^{5}_{nds}\omega^{5} + \sum_{n\in N}\sum_{w\in W} h^{6}_{nw}\omega^{6} + \sum_{n\in N}\sum_{i\in\{7,8,9\}} j^{i}_{n}\omega^{i} \tag{2.1}$$

Subject to:

$$\sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 1 \qquad \forall n\in N, d\in D\setminus\tilde{D} \tag{2.2}$$

$$\sum_{k\in K}(x_{nds_ek} + x_{nds_lk}) = 2z_{nd} \qquad \forall n\in N, d\in\tilde{D} \tag{2.3}$$

$$\sum_{k\in K} x_{nds_nk} + z_{nd} \leq 1 \qquad \forall n \in N, d \in \tilde{D} \tag{2.4}$$

$$\sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 2o_{nd} \qquad \forall n \in N, d \in D \tag{2.5}$$

$$\sum_{k\in K} (x_{nds'k} + x_{n(d+1)s''k}) \leq 1 \qquad \forall n \in N, d \in \{1,\ldots,|D|-1\}, (s',s'') \in \hat{S} \tag{2.6}$$

$$\sum_{d\in D}\sum_{s\in S} x_{ndsk} = 0 \qquad \forall (n,k) \in L \tag{2.7}$$

$$\sum_{k\in K} x_{ndsk} = 0 \qquad \forall (n,d,s) \in R \tag{2.8}$$

$$x_{nds_ek} - x_{nds_lk} = 0 \qquad \forall n \in N, d \in \tilde{D}, k \in K \tag{2.9}$$

$$\sum_{n\in N} x_{ndsk} \geq \alpha^1_{dsk} \qquad \forall d \in D, s \in S, k \in K \tag{2.10}$$

$$\sum_{n\in N} x_{ndsk} \leq \alpha^2_{dsk} \qquad \forall d \in D, s \in S, k \in K \tag{2.11}$$

$$\sum_{d'=d}^{\beta^3_n+d}\sum_{k\in K} x_{nd's_nk} - c^3_{nd} \leq \beta^3_n \qquad \forall n \in N, d \in \{1,\ldots,|D|-\beta^3_n\} \tag{2.12}$$

$$\sum_{d'=d}^{\beta^4_n+d} o_{nd'} - c^4_{nd} \leq \beta^4_n \qquad \forall n \in N, d \in \{1,\ldots,|D|-\beta^4_n\} \tag{2.13}$$

$$\sum_{k\in K} x_{ndsk} \leq g^5_{nds} \qquad \forall (n,d,s) \in U \tag{2.14}$$

$$o_{nw} + o_{n(w+1)} + h^6_{nw} = 2y_{nw} \qquad \forall n \in N, w \in W \tag{2.15}$$

$$\sum_{d\in D}\sum_{s\in\{s_e,s_l\}}\sum_{k\in K} x_{ndsk} + \sum_{d\in D}\sum_{k\in K} 2x_{nds_nk} + j^7_n \geq \beta^7_n \qquad \forall n \in N \tag{2.16}$$

$$\sum_{d\in D}\sum_{s\in\{s_e,s_l\}}\sum_{k\in K} x_{ndsk} + \sum_{d\in D}\sum_{k\in K} 2x_{nds_nk} - j^8_n \leq \beta^8_n \qquad \forall n \in N \tag{2.17}$$

$$\sum_{w\in W} y_{nw} - j^9_n \leq \beta^9_n \qquad \forall n \in N \tag{2.18}$$

Constraints (2.2) ensure a physician is assigned to at most one shift per day on business days. Constraints (2.3) and (2.4) ensure that a physician must be assigned to both day shifts, or one Night shift, or no shift on non-business days. Constraints (2.5) set auxiliary variable o_{nd} to one if physician n works on day d, and zero otherwise. Constraints (2.6) ensure a shift type succession must be valid (for example, if physicians work Night shifts they cannot be followed by Early or Late shifts on the next day). Constraints (2.7) ensure a physician is assigned to a location only if allowed. Constraints (2.8) ensure a physician is scheduled only if he/she is available. Constraints (2.9) ensure a physician works both shifts (Early and Late) at the same location if they are worked on non-business days. This constraint is to avoid that a physician splits a 12-hour working shift across two different locations. Constraints (2.10) and (2.11) ensure the minimum and maximum number of physicians per day/shift/location, respectively.

Constraints (2.12) calculate the maximum number of consecutive assignments to Night shifts violations. Constraints (2.13) calculate the maximum number of consecutive assignments (worked days) violations. Constraints (2.14) calculate the undesired worked day or shift violations. Constraints (2.15) calculate the complete weekend violations. Constraints (2.16) and (2.17) calculate the minimum and maximum number of worked shifts violations over the scheduling period, respectively. Constraints (2.18) calculate the maximum number of working weekends violations.

2.4.2 Extended model

In this section the extended model is described, that is, constraints that aim a fair distribution of the working hours between the physicians.

$$\textbf{Minimize:} \quad \sum_{n\in N}\sum_{w\in\tilde{W}}\sum_{s\in S} l^{18}_{nws}\omega^{18} + \sum_{n\in N}\sum_{k\in K} m^{15}_{nk} + \sum_{n\in N}\sum_{i\in\{13,14,16,17,19\}} j^i_n\omega^i + (2.1) \tag{2.19}$$

Subject to:

$$\beta^{10}_n \leq \sum_{d\in\tilde{D}} o_{nd} \leq \beta^{11}_n \qquad \forall n\in N \tag{2.20}$$

$$\sum_{d\in D\backslash\tilde{D}} o_{nd} \leq \beta^{12}_n \qquad \forall n\in N \tag{2.21}$$

$$\sum_{d\in\tilde{D}} o_{nd} + j^{13}_n - j^{14}_n = \beta^{13}_n \qquad \forall n\in N \tag{2.22}$$

$$\sum_{d\in D}\sum_{s\in S} x_{ndsk} - m^{15}_{nk} = 0 \qquad \forall (n,k)\in P \tag{2.23}$$

$$\begin{aligned}&\sum_{d\in\tilde{D}}\sum_{k\in K}\left(x_{nds_ek} + x_{n(d+1)s_ek}\right) - \\ &\sum_{d\in\tilde{D}}\sum_{k\in K}\left(x_{nds_nk} + x_{n(d+1)s_nk}\right) - j^{16}_n + j^{17}_n = 0\end{aligned} \qquad \forall n\in N \tag{2.24}$$

$$\sum_{d\in\{1,\ldots,7\}}\sum_{k\in K} x_{n(d\times w)sk} - l^{18}_{nws} \leq \gamma^{18}_{nws} \qquad \forall n\in N, w\in\tilde{W}, s\in S \tag{2.25}$$

$$x_{ndsk} \leq q_{nk} \qquad \forall n\in N, d\in D, s\in S, k\in K \tag{2.26}$$

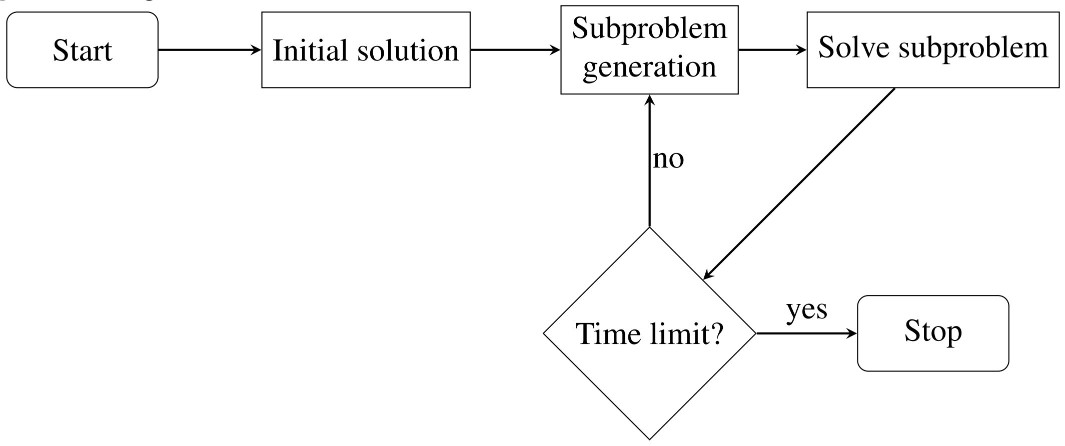
$$\sum_{k \in K} q_{nk} - j_n^{19} \leq 1 \qquad \forall n \in N \tag{2.27}$$

Constraints (2.20) ensure the minimum and maximum number of assignments on non-business days for each physician. Constraints (2.21) ensure the maximum number of assignments on business days for each physician. Constraints (2.22) penalize the difference between the ideal and actual number of assignments on non-business days. Constraints (2.23) penalize physicians working out of the preferred location. Constraints (2.24) penalize the difference between the number of assignments of day and night shifts on non-business days. Constraints (2.25) penalize weekly allocations in excess of the maximum. Constraints (2.26) and (2.27) calculate the number of distinct locations that a physician works and penalize if this value is greater than one.

2.5 Fix-and-optimize matheuristic

This section provides the F&O matheuristic developed to approach the PRP. The proposed algorithm was adapted from a previous version to address the NRP (WICKERT; SARTORI; BURIOL, 2016). Figure 2.1 provides an overview of the algorithm execution flow. The algorithm begins generating a feasible solution using a MIP solver only considering the hard constraints. Afterward, a subset of variables is iteratively fixed to their current values, decomposing the problem into subproblems, which are then successively solved using a MIP solver until the computation time limit is reached. All hard and soft constraints are considered when the subproblem is solved. The algorithm returns the best solution found.

Figure 2.1: Algorithm execution flowchart.



Before fully explaining the algorithm, the following general terminology is introduced. Variables or physicians are denoted as *free* when the associated decision variable has the lower bound set to zero and the upper bound to one. This consequently implies that the solver can decide on setting the value either to zero or to one. On the other hand, *fixing* a day, physician or shift indicates that the decision variable is set to the corresponding value in the incumbent solution. In this case, the solver cannot change the variable's value. *PhysicianFreeCombinationSet* is a set of combinations without repetition.

For example, when there are $n = 5$ physicians and the parameter *kPhysician=2*, the combinations without repetitions is 10 unique possibilities, resulting in the set *PhysicianFreeCombinationSet([1,2], [1,3], [1,4], [1,5], [2,3], [2,4], [2,5], [3,4], [3,5], [4,5])*. If *kLimitPhysician*=5, only 5 out of 10 items will be randomly added to the set *PhysicianFreeCombinationSet*. If the *kLimitPhysician* $\geq$ *10* all possible combinations will be added to the set *PhysicianFreeCombinationSet*. During the algorithm's execution, the decision variable will have physicians 1 and 2 with lower bound zero and upper bound one, while physicians 3–10 will have their lower and upper bounds fixed to their current values. As such, the solver can only change the value of the decision variables of physicians 1 and 2.

Algorithm 1 provides the pseudo-code. Input parameters are passed to the algorithm, where *kMaxDay*, *kMaxPhysician*, *kMaxWeek* and *kMaxShift* represent the maximum number of free variables of each type. Meanwhile, *kLimitDay*, *kLimitPhysician*, *kLimitWeek* and *kLimitShift* are the limits of combinations generated for each type of neighborhood. Algorithm 1 begins by generating an initial feasible solution x (line 2) considering only the hard constraints using a MIP solver. The number of free variables to optimize is initialized with one (line 3), and the loop (lines 4 to 17) is iterated until the time limit (TL) is reached.

```
1  FixAndOptimize(kMaxWeek, kLimitWeek, kMaxShift, kLimitShift, kMaxDay, kLimitDay, kMaxPhysician,
     kLimitPhysician, TL, STL)
2  x = generateInitialSolution()
3  kWeek = kShift = kDay = kPhysician = 1
4  do
5      x = FixPerDay(x, kDay, kLimitDay, STL)
6      kDay = kDay+1
7      x = FixPerPhysician(x, kPhysician, kLimitPhysician, STL)
8      kPhysician=kPhysician+1
9      x = FixPerWeek(x, kWeek, kLimitWeek, STL)
10     kWeek=kWeek+1
11     x = FixPerShift(x, kShift, kLimitShift, STL)
12     kShift=kShift+1
13     if (kDay > kMaxDay) kDay = 1
14     if (kPhysician > kMaxPhysician) kPhysician = 1
15     if (kWeek > kMaxWeek) kWeek = 1
16     if (kShift > kMaxShift) kShift = 1
17 while TL not reached
18 return x
```

Algorithm 1: Fix-and-optimize matheuristic algorithm.

Inside the loop (lines 5 to 16), the algorithm executes different methods representing the neighborhoods. Each neighborhood is explored either a local minimum is found or it reaches the STL (subproblem time limit). For each step, the value of *kDay*, *kPhysician*, *kWeek* and *kShift* is incremented. If the limit of each type is exceeded, variables are reset to 1 (lines 13 to 16).

Algorithm 2 begins generating the combination of physicians that will be free to be optimized, until the *kLimitPhysician* is reached (the *combination* function at line 2). The loop (lines 4 to 16) is iterated until no improvement of 16.6% is found. The loop begins by storing the current solution value and the best neighbor value (function *OFV* lines 5 and 6). The nested loop (lines 7 to 14) explores the neighborhood by fixing the entire problem (line 8), and unfixing only the free variables that will be optimized (line 9). The MIP solver is called and executed until either the optimal solution is found or STL is reached (line 10). If the Objective Function Value (OFV) of subproblem *x* is lower than the OFV of the best neighbor (line 11), then the *bestNeighborValue* variable is updated accordingly (line 12).

```
1  FixPerPhysician(x, kPhysician, kLimitPhysician, STL)
2  physicianFreeCombinationSet = combination(kPhysician, kLimitPhysician)
3  improved = false
4  do
5      currentSolutionValue = OFV(x)
6      bestNeighborValue = OFV(x)
7      foreach Integer free : physicianFreeCombinationSet do
8          fixAll(x)
9          unFix(free, x)
10         solve(x, STL)
11         if OFV(x) < bestNeighborValue then
12             bestNeighborValue = OFV(x)
13         end
14     end
15     improved = bestNeighborValue*1.2 < currentSolutionValue
16 while improved
17 return x
```

Algorithm 2: Fix per physician.

Figures 2.2 and 2.3 detail an iteration of a *fix per physician* neighborhood with *kPhysician*=1 and *kPhysician*=2, respectively. Rows with a gray background are available to be optimized by the solver, meaning that the decision variables have lower bound set to zero and upper bounds of one. By contrast, rows with a white background have the associated decision variable bounds fixed to the current incumbent value, meaning that the solver must not change these values. Observe that, when a decision variable has both upper and lower bounds set to the same value, they are ignored by the MIP solver. Since the *fix per week*, *fix per shift* and *fix per day* decompositions follow the same idea, the pseudo-code of these algorithms has been omitted for textual economy.

Figure 2.2: Fix per physician (*kPhysician*=1).

Physician	Mon	Tue	Wed
P1	L[1]	L[2]	N[2]
P2	N[1]	N[1]	N[3]
P3	–	E[3]	–

↓

P1	L[1]	N[2]	N[2]
P2	N[1]	N[1]	N[3]
P3	–	E[3]	–

↓

P1	L[1]	N[2]	N[2]
P2	N[2]	N[1]	N[3]
P3	–	E[3]	–

Figure 2.3: Fix per physician (*kPhysician*=2).

Physician	Mon	Tue	Wed
P1	L[1]	N[2]	N[2]
P2	N[2]	N[1]	N[3]
P3	–	E[3]	–

↓

P1	L[1]	N[1]	N[2]
P2	N[2]	L[1]	N[3]
P3	–	E[3]	–

↓

P1	L[1]	N[1]	N[2]
P2	E[2]	L[1]	N[3]
P3	E[3]	E[3]	–

2.6 Computational experiments

This section analyzes a series of computational experiments to investigate whether the proposed IP formulation can be solved using commercial and open-source MIP solvers for both small and large instances.

2.6.1 Data sets and experimental setup

The source code was written in Java and compiled with OpenJDK 1.8. The experiments were conducted on an Intel Core i5-2410M CPU 2.30GHz (2 cores) with 6GB of RAM memory running Linux Mint 17.2 64-bits. The solvers employed were CPLEX version 12.6.2 and Coin-OR CBC version 2.9.9. Both solvers were run with default parameters. The *gap* is calculated using the equation $gap = 100 \times \frac{OFV-LB}{LB}$, where OFV is the objective function value and LB is the lower bound. All parameters of the F&O matheuristic were tuned using irace (LÓPEZ-IBÁÑEZ et al., 2016).

Table 2.4 presents the parameter names, tested ranges and the chosen parameter values computed by irace, respectively. Irace reported the values *STL = 8s, kMaxWeek = 2, kLimitWeek = 4, kMaxShift = 3, kLimitShift = 3, kMaxPhysician = 20, kLimitPhysician = 30, kMaxDay = 8, and kLimitDay = 8* as the best parameter values to be used by the proposed F&O matheuristic. The default irace parameters were used for the experiments, that is, the confidence level is 0.95.

The dataset employed in the experiments was generated based on the information

Table 2.4: Parameter ranges.

Name	Range of Values	Irace
STL	[5, 8, 12]	8
kMaxWeek	[1,...,4]	2
kLimitWeek	[1,...,8]	4
kMaxShift	[1,...,3]	3
kLimitShift	[1,...,7]	3
kMaxPhysician	[1,5,10,15,20,25,30]	20
kLimitPhysician	[5,10,15,30,35,40,45,50]	30
kMaxDay	[1,8,16]	8
kLimitDay	[4,8,16,32]	8

provided by HCPA. The algorithm was tested in 30 generated instances. Currently, the real number of physicians to schedule is 50. However, the number of physicians will increase in the near future, and so this is the reasoning behind generating larger instances. The objective is to analyze whether these larger and more demanding instances can still be solved using the proposed methods. The following instances were generated:

- 10 instances with 50 physicians and four weeks;
- 10 instances with 100 physicians and four weeks;
- 10 instances with 150 physicians and four weeks.

The computation time limit for each experiment was fixed according to the instance size and algorithm. Note that the F&O matheuristic has half of the computational time limit compared against the MIP solvers. This solving method is used when quick results are required in reduced computational time. The following experimental setup was proposed:

- One single execution using CPLEX solver and a computation time limit of 12h;
- One single execution using CPLEX and Coin-OR CBC with computation time limits of 20, 40 and 60 minutes for the instances with 50, 100 and 150 physicians, respectively;
- 10 executions using the F&O matheuristic with computation time limits of 10, 20 and 30 minutes for the instances with 50, 100 and 150 physicians, respectively.

In addition, two instances using data from April and May of 2019 provided by HCPA were employed for the computational experiments using the extended model. The reported results and statistics constitute the real roster used in the hospital.

2.6.2 MIP solver results

Table 2.5 provides the results when the basic IP formulation is solved using the standalone MIP solvers CPLEX and Coin-OR CBC. The column labels LB, OFV and Gap correspond to the lower bound (provided by CPLEX), objective function value and the gap calculated according to the equation introduced in the previous section. Experiments are split into three blocks. The first block provides the results when CPLEX was executed for a long computation time (12h). The objective of this experiment was to generate good LBs to use them for comparing the other solving methods. In practice, this time limit is not considered acceptable therefore other experiments have shorter computation time limits.

Table 2.5: MIP solver results.

	CPLEX						Coin-OR CBC		
	Time Limit - 12h			Time Limit - 20, 40, 60 min[1]			Time Limit - 20, 40, 60 min[1]		
Instance	LB	OFV	Gap(%)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
p050_inst_01	30,305	30,305	0.00	30,305	0.00	438	30,305	0.00	754
p050_inst_02	30,460	30,460	0.00	30,460	0.00	352	30,460	0.00	520
p050_inst_03	30,505	30,505	0.00	30,505	0.00	428	30,505	0.00	1,185
p050_inst_04	30,965	30,965	0.00	30,965	0.00	511	30,965	0.00	283
p050_inst_05	30,685	30,685	0.00	30,685	0.00	374	30,685	0.00	435
p050_inst_06	31,705	31,705	0.00	31,705	0.00	415	31,705	0.00	301
p050_inst_07	30,015	30,015	0.00	30,015	0.00	401	30,015	0.00	315
p050_inst_08	30,215	30,215	0.00	30,215	0.00	403	30,225	0.03	1,200
p050_inst_09	31,670	31,670	0.00	31,670	0.00	447	31,670	0.00	484
p050_inst_10	30,765	30,765	0.00	30,765	0.00	408	30,765	0.00	1,182
Average			0.00		0.00			0.00	
p100_inst_01	24,429	25,525	4.49	26,320	7.74	2,400	-	-	2,400
p100_inst_02	26,720	27,945	4.58	29,940	12.05	2,400	-	-	2,400
p100_inst_03	25,082	26,300	4.86	-	-	2,400	-	-	2,400
p100_inst_04	24,280	25,285	4.14	-	-	2,400	-	-	2,400
p100_inst_05	24,633	25,775	4.64	28,615	16.17	2,400	-	-	2,400
p100_inst_06	25,660	26,920	4.91	29,490	14.93	2,400	-	-	2,400
p100_inst_07	23,205	24,505	5.60	26,180	12.82	2,400	-	-	2,400
p100_inst_08	25,282	26,445	4.60	-	-	2,400	-	-	2,400
p100_inst_09	25,946	27,130	4.56	-	-	2,400	-	-	2,400
p100_inst_10	23,775	25,030	5.28	28,185	18.55	2,400	-	-	2,400
Average			4.77		-				
p150_inst_01	55,742	60,030	7.69	-	-	3,600	-	-	3,600
p150_inst_02	54,011	58,320	7.98	-	-	3,600	-	-	3,600
p150_inst_03	54,135	58,070	7.27	-	-	3,600	-	-	3,600
p150_inst_04	53,457	57,595	7.74	-	-	3,600	-	-	3,600
p150_inst_05	54,786	59,740	9.04	-	-	3,600	-	-	3,600
p150_inst_06	54,170	58,720	8.40	-	-	3,600	-	-	3,600
p150_inst_07	53,959	57,570	6.69	-	-	3,600	-	-	3,600
p150_inst_08	53,199	56,750	6.67	-	-	3,600	-	-	3,600
p150_inst_09	53,396	57,580	7.84	-	-	3,600	-	-	3,600
p150_inst_10	51,782	55,810	7.78	-	-	3,600	-	-	3,600
Average			7.71		-				

[1] Computation time limits for instances with 50, 100 and 150 physicians, respectively.

The second and third blocks present the results when CPLEX and Coin-OR CBC were employed to solve the IP formulation using computation time limits of 20, 40 and 60 minutes for instances with 50, 100 and 150 physicians, respectively. Instances containing 50 physicians were solved to optimality employing CPLEX and near-optimality using Coin-OR CBC. For larger instances, with 100 and 150 physicians, Coin-OR was not capable to generate feasible solutions within the time limit, while CPLEX generated 6 out of 10 feasible solutions when instances with 100 physicians are tackled. Both solvers could not find feasible solutions for instances with 150 physicians within 1h.

These results show that MIP solvers are a good option for solving instances up to 50 physicians which is the hospital's current situation. Both CPLEX and the open-source solver Coin-OR CBC generated good results. Optimal and near-optimal results were obtained within acceptable computation time limits. However, for large instances with 100 and 150 physicians, improvements in the IP formulation or other solving methods are necessary to generate better results.

2.6.3 Fix-and-optimize results

Table 2.6 presents the results using the fix-and-optimize (F&O) matheuristic. The three last columns provide CPLEX results. However, a direct comparison with heuristics is not possible since MIP solvers address problems improving both upper and lower bounds aiming to prove optimality. The gap is calculated relative to the LB (first column), obtained by CPLEX when executed with a time limit of 12h.

As explained in Section 2.5, the F&O matheuristic uses CPLEX to solve the subproblems and results are compared to the Late Acceptance Hill Climbing (LAHC) developed by Sanchotene and Buriol (2018). Computational results demonstrated that for instances with 50 physicians, results were very close to optimality with an average relative gap of 0.03%. Instances with 100 and 150 physicians have average relative gaps of 6.00% and 8.75%, respectively.

In general, the F&O matheuristic generated similar results as the LAHC heuristic developed by Sanchotene and Buriol (2018). These computational experiments show that both the LAHC heuristic and the F&O matheuristic are good alternatives to address large problems with 100 and 150 physicians. Both methods generated good results within short computation times. By contrast, when solving problems with up to 50 physicians the standalone MIP solvers are the most effective alternative. Note that the proposed F&O matheuristic is solver-independent and general, being capable of addressing both the basic and extended models without compromising the quality of the results compared

Table 2.6: Heuristic LAHC and F&O results.

		LAHC-10,20,30 min[1]		F&O - 10, 20, 30 min[1]			CPLEX - 20, 40, 60 min[1]		
Instance	LB	OFV	Gap(%)	OFV	Std. Dev.	Gap(%)	OFV	Gap(%)	Time(s)
p050_inst_01	30,305	30,329	0.08	30,318	± 7	0.04	30,305	**0.00**	438
p050_inst_02	30,460	30,480	0.07	30,468	± 8	0.03	30,460	**0.00**	352
p050_inst_03	30,505	30,517	0.04	30,516	± 12	0.04	30,505	**0.00**	428
p050_inst_04	30,965	30,977	0.04	30,975	± 8	0.03	30,965	**0.00**	511
p050_inst_05	30,685	30,695	0.03	30,692	± 11	0.02	30,685	**0.00**	374
p050_inst_06	31,705	31,737	0.10	31,719	± 7	0.04	31,705	**0.00**	415
p050_inst_07	30,015	30,051	0.12	30,022	± 7	0.02	30,015	**0.00**	401
p050_inst_08	30,215	30,236	0.07	30,226	± 10	0.04	30,215	**0.00**	403
p050_inst_09	31,670	31,714	0.14	31,685	± 12	0.05	31,670	**0.00**	447
p050_inst_10	30,765	30,783	0.06	30,772	± 8	0.02	30,765	**0.00**	408
Average			0.07			0.03		**0.00**	
p100_inst_01	24,429	25,979	6.34	25,835	± 66	**5.76**	26,320	7.74	2,400
p100_inst_02	26,720	28,479	6.58	28,324	± 55	**6.00**	29,940	12.05	2,400
p100_inst_03	25,082	26,659	6.29	26,608	± 95	**6.08**	-	-	2,400
p100_inst_04	24,280	25,764	6.11	25,558	± 78	**5.26**	-	-	2,400
p100_inst_05	24,633	26,144	6.13	26,034	± 50	**5.69**	28,615	16.17	2,400
p100_inst_06	25,660	27,391	6.75	27,253	± 45	**6.21**	29,490	14.93	2,400
p100_inst_07	23,205	25,008	7.77	24,801	± 66	**6.88**	26,180	12.82	2,400
p100_inst_08	25,282	26,867	6.27	26,754	± 89	**5.82**	-	-	2,400
p100_inst_09	25,946	27,592	6.34	27,333	± 58	**5.35**	-	-	2,400
p100_inst_10	23,775	25,534	7.40	25,438	± 89	**6.99**	28,185	18.55	2,400
Average			6.60			**6.00**		-	
p150_inst_01	55,742	60,207	**8.01**	60,517	± 123	8.57	-	-	3,600
p150_inst_02	54,011	58,691	**8.66**	58,950	± 154	9.14	-	-	3,600
p150_inst_03	54,135	58,539	**8.14**	58,875	± 95	8.75	-	-	3,600
p150_inst_04	53,457	57,842	**8.20**	58,133	± 139	8.75	-	-	3,600
p150_inst_05	54,786	59,375	**8.38**	59,640	± 122	8.86	-	-	3,600
p150_inst_06	54,170	58,973	**8.87**	59,141	± 56	9.18	-	-	3,600
p150_inst_07	53,959	58,119	**7.71**	58,443	± 115	8.31	-	-	3,600
p150_inst_08	53,199	57,378	**7.86**	57,733	± 199	8.52	-	-	3,600
p150_inst_09	53,396	57,673	**8.01**	58,029	± 126	8.68	-	-	3,600
p150_inst_10	51,782	55,990	**8.13**	56,311	± 100	8.75	-	-	3,600
Average			**8.20**			8.75		-	

[1] Computation time limits for instances with 50, 100 and 150 physicians, respectively.

to the case-specific LAHC heuristic.

Table 2.7: F&O results for 2h of computation running time.

Instance	LB CPLEX	OFV	Std. Dev.	Gap (%)
p150_inst_01	55,742	60,028	± 136	7.69
p150_inst_02	54,011	58,496	± 78	8.30
p150_inst_03	54,135	58,309	± 76	7.71
p150_inst_04	53,457	57,606	± 147	7.76
p150_inst_05	54,786	59,127	± 94	7.92
p150_inst_06	54,170	58,609	± 63	8.19
p150_inst_07	53,959	57,847	± 79	7.21
p150_inst_08	53,199	57,204	± 122	7.53
p150_inst_09	53,396	57,447	± 102	7.59
p150_inst_10	51,782	55,790	± 100	7.74
Average				7.76

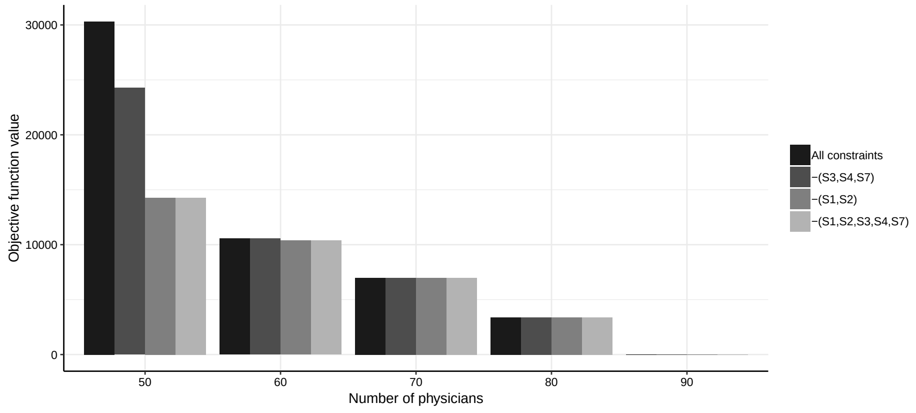
Table 2.7 details the results obtained with a computation time limit of two hours using instances with 150 physicians. On average, the relative gap was reduced from 8.20% when the time limit was 30 minutes to 7.76%. These results are almost the same as those obtained by CPLEX when the running time was limited to 12h which generated a

relative gap of 7.71%. Such results indicate that the proposed method is a good alternative to CPLEX for large instances when limited time is available.

2.6.4 Objective function analysis

Figure 2.4 provides an analysis of the OFV when varying the number of physicians and removing different subsets of constraints. The objective is to evaluate the influence of the different sets of constraints upon the OFV. Black bars indicate the results when all constraints are considered when varying the number of physicians from 50 to 90. With 60 physicians the OFV reduced approximately one third and eventually reached an ideal of zero when the number of available physicians is 90. Moreover, results indicate a notable reduction in the OFV when S1 and S2 are removed considering 50 physicians. Experiments show that these two constraints represent a minor influence on the OFV when the number of physicians are 60 or more. Such results indicate that the majority of the violations concern overtime constraints.

Figure 2.4: OFV impact when varying the number of physicians and removing constraints.



2.6.5 Extended model results

This section provides the results when using the extended model for the generated roster of April and May of 2019. Since a CPLEX license is required for commercial use, only Coin-OR CBC was used for the computational experiments in this section. Appendices A and B present the complete roster of April and May of 2019, respectively. Table 2.8 provides the results employing Coin-OR CBC standalone solver (second column) and using the F&O matheuristic (third column). The objective of this experiment is not a direct comparison between the exact and heuristic method. Instead, they are complementary, being the F&O matheuristic employed mainly when fast results must be available in reduced time. Results show that the standalone solver was capable of reaching near-optimum results with a relative gap of less than 0.1% within 20 minutes. Another experiment, using the F&O matheuristic with a 10 minutes time limit, demonstrated that the algorithm is a good alternative when results are needed quickly. The gaps (last column), calculated relative to the lower bound provided by the MIP solver, were 1.83% and 2.46% for the months of April and May, respectively.

Table 2.8: Results using Coin-OR CBC standalone solver and the F&O matheuristic.

	20 min	10min
	Coin-OR CBC	F&O
	Gap(%)	Gap(%)
April 2019	0.05	1.83
May 2019	0.09	2.46

Note that both the F&O matheuristic and the standalone MIP solver are used in practice at HCPA. The F&O matheuristic method is employed to generate fast solutions when the roster of a new month is going to be organized. During this process managers may change the input data including days off requests, vacations, constraints violation weight, and recompute the roster several times. When this process is more stable, the exact method is executed for longer runtimes to generate near-optimum final rosters.

Tables 2.9 and 2.10 provide the most relevant constraint violation analysis concerning the rosters generated for April and May of 2019. The first column represents the physician identification, while the S1 column provides the maximum of two consecutive night shift violations. Column S4 details the number of incomplete worked weekends, column S5/S6 provides the contracted hours (in parentheses), where positive and negative values indicate whether the respective physician worked more or less than their contracted hours. Observe that Early and Late shifts have six hours and Night shifts twelve hours. However, the majority of the physicians' contracts are not a multiple of six, and therefore it is technically impossible for most physicians to work their precise contractual hours. Column S7 presents the maximum number of worked weekends in parentheses and the number effectively worked. Column S8 provides the ideal number of worked hours on non-business days in parentheses, where positive and negative values indicate if the physician worked more or less than the ideal. These ideal hours vary from one physician to another depending on the total number of working hours and the seniority of the contract. Column S10 presents the difference between worked day and night shifts (day - night) for which the result should, ideally, be zero. Finally, column S12 details the number of times a physician was allocated for each area. For example, Physician1 was scheduled to work 10 times at Area2 and zero times in other areas (in the example there are six areas).

Table 2.9: April 2019 - Roster analysis.

Name	S1	S4	S5/S6	S7	S8	S10	S12
P1	0	0	(100) +8	(2) 1	(24) 0	0	[0, 10, 0, 0, 0, 0]
P2	0	1	(100) +2	(2) 2	(36) 0	12	[12, 0, 0, 0, 0, 0]
P3	0	2	(100) +2	(2) 2	(36) 0	0	[15, 0, 0, 0, 0, 0]
P4	0	1	(100) +2	(2) 2	(36) 0	12	[12, 0, 0, 0, 0, 0]
P5	0	2	(100) +2	(2) 2	(36) 0	0	[0, 12, 0, 0, 0, 0]
P6	0	1	(115) +5	(2) 2	(36) 0	-12	[0, 14, 0, 0, 0, 0]
P7	0	0	(72) 0	(2) 1	(24) 0	0	[0, 8, 0, 0, 0, 0]
P8	0	0	(60) 0	(2) 1	(24) 0	0	[0, 0, 0, 0, 6, 0]
P9	0	0	(83) +1	(2) 1	(24) 0	0	[0, 10, 0, 0, 0, 0]
P10	0	2	(100) +8	(2) 2	(36) 0	0	[0, 11, 0, 0, 0, 0]
P11	0	1	(100) +2	(2) 2	(36) 0	-12	[11, 0, 0, 0, 0, 0]
P12	0	1	(100) +2	(2) 2	(36) 0	12	[12, 0, 0, 0, 0, 0]
P13	0	1	(100) +2	(2) 2	(36) 0	12	[0, 12, 0, 0, 0, 0]
P14	0	0	(48) 0	(2) 1	(24) 0	0	[0, 0, 0, 0, 5, 0]
P15	0	2	(100) +2	(2) 2	(36) 0	0	[12, 0, 0, 0, 0, 0]
P16	0	0	(125) +1	(2) 2	(48) 0	0	[0, 0, 14, 0, 0, 0]
P17	1	1	(68) +4	(2) 2	(36) 0	-12	[7, 1, 0, 0, 0, 0]
P18	0	2	(100) +2	(2) 2	(36) 0	0	[11, 0, 0, 0, 0, 0]
P19	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 12, 0, 0, 0]
P20	0	2	(100) +8	(2) 2	(36) 0	0	[0, 0, 13, 0, 0, 0]
P21	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 0, 13, 0, 0]
P22	0	2	(150) 0	(2) 2	(36) 0	0	[7, 0, 0, 13, 0, 0]
P23	0	1	(100) +2	(2) 2	(36) 0	-12	[0, 0, 0, 12, 0, 0]
P24	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 12, 0, 0, 0]
P25	0	0	(125) -11	(2) 2	(48) 0	0	[13, 0, 0, 0, 0, 0]
P26	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 0, 0, 12, 0]
P27	0	2	(100) +2	(2) 2	(36) 0	0	[0, 0, 0, 12, 0, 0]
P28	0	2	(100) +2	(2) 2	(36) -12	0	[0, 0, 0, 14, 0, 0]
P29	0	1	(100) +8	(2) 2	(36) 0	-12	[10, 0, 0, 0, 0, 0]
P30	0	2	(100) -4	(2) 2	(36) 0	0	[0, 11, 0, 0, 0, 0]
P31	0	1	(100) +2	(2) 2	(36) 0	-12	[0, 7, 5, 0, 0, 0]
P32	0	2	(100) -4	(2) 2	(36) 0	0	[0, 0, 11, 0, 0, 0]
P33	0	1	(65) +1	(2) 1	(24) 0	12	[0, 0, 1, 0, 7, 0]
P34	0	2	(100) -4	(2) 2	(36) 0	0	[0, 0, 0, 0, 12, 0]
P35	0	2	(125) -5	(2) 2	(36) 0	0	[0, 0, 0, 0, 0, 16]
P36	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 0, 12, 0, 0]
P37	0	1	(125) +7	(2) 2	(36) 0	-12	[0, 0, 0, 0, 11, 5]
P38	1	1	(100) -4	(2) 2	(36) 0	-12	[0, 0, 0, 0, 11, 0]
P39	0	2	(125) -11	(3) 2	(36) 0	0	[0, 0, 0, 0, 0, 12]
P40	0	0	(100) +2	(2) 1	(36) -12	0	[0, 0, 0, 14, 0, 0]
P41	0	1	(100) +2	(2) 2	(36) 0	-12	[0, 7, 0, 0, 5, 0]
P42	0	1	(100) +2	(2) 2	(36) 0	12	[3, 10, 0, 0, 0, 0]
P43	0	1	(100) +2	(2) 2	(36) 0	-12	[0, 5, 7, 0, 0, 0]
P44	0	1	(100) +2	(2) 2	(36) 0	12	[0, 0, 0, 0, 0, 13]
P45	0	1	(100) +2	(2) 2	(36) 0	-12	[3, 9, 0, 0, 0, 0]
P46	0	1	(125) -5	(2) 2	(36) 0	-12	[0, 0, 0, 0, 0, 15]
P47	0	1	(125) -5	(2) 2	(36) 0	-12	[0, 0, 0, 0, 0, 15]
P48	0	1	(125) -11	(2) 2	(36) 0	12	[0, 0, 0, 0, 0, 14]

Overtime: 99h; Debt: 64h; Difference: +35h

From Table 2.9 it can be observed that there are a low number of violations concerning constraint maximum number of consecutive worked night shifts (S1). The maximum number of worked weekends (S7), which for the majority of physicians was limited to two, had no violations. The same tendency is observed for ideal working hours on non-business days (S8), which was always equal or below the stipulated value. Although incomplete worked weekends (S4) presented a few violations they were still within an acceptable range. Constraint (S10) only presented multiples of 12 violations because this is the minimum number of working hours on non-business days. This constraint presented more violations, but it is acceptable given the reduced number of assigned areas (S12), which is desirable.

Table 2.9, which represents April of 2019, was a month where almost all physicians were available the entire month contributing to less overtime. Because of that, some physicians worked fewer hours than their contract stipulates. Before the solver was implemented, manual rosters were deployed to the physicians with an average of 400 overtime hours.

Table 2.10: May 2019 - Roster analysis.

Name	S1	S4	S5/S6	S7	S8	S10	S12
P1	0	1	(104) +4	(2) 2	(36) 0	-12	[0, 10, 0, 0, 0, 0]
P2	0	2	(52) +8	(2) 2	(24) 0	0	[6, 0, 0, 0, 0, 0]
P3	0	2	(104) +4	(2) 2	(36) 0	0	[7, 0, 0, 0, 9, 0]
P4	0	1	(72) 0	(2) 2	(24) +12	-12	[7, 0, 0, 0, 0, 0]
P5	0	1	(52) +2	(2) 1	(24) 0	12	[0, 7, 0, 0, 0, 0]
P6	0	0	(130) +8	(2) 1	(36) 0	24	[0, 13, 0, 6, 0, 0]
P7	0	2	(104) +4	(2) 2	(36) 0	0	[0, 10, 0, 2, 0, 0]
P8	0	0	(60) 0	(2) 1	(24) 0	0	[0, 0, 0, 0, 6, 0]
P9	0	0	(86) +10	(2) 1	(24) 0	0	[0, 10, 0, 0, 0, 0]
P10	0	1	(104) +4	(2) 2	(36) 0	12	[0, 11, 0, 0, 0, 0]
P11	0	0	(52) +2	(2) 1	(24) 0	0	[6, 0, 0, 0, 0, 0]
P12	0	2	(104) +4	(2) 2	(36) 0	0	[9, 0, 4, 0, 0, 0]
P13	0	1	(104) +4	(2) 2	(36) 0	-12	[0, 9, 0, 3, 0, 0]
P14	0	1	(30) +6	(2) 1	(12) 0	12	[0, 0, 0, 0, 4, 0]
P15	0	0	(104) +4	(2) 1	(36) 0	0	[10, 3, 0, 0, 0, 0]
P16	0	0	(130) +2	(2) 2	(48) 0	0	[0, 0, 16, 0, 0, 0]
P17	0	1	(104) +4	(2) 2	(36) 0	-12	[10, 0, 0, 0, 0, 0]
P18	0	1	(104) +10	(2) 2	(36) 0	-12	[9, 0, 0, 0, 3, 0]
P19	0	2	(116) +10	(2) 2	(36) 0	0	[0, 0, 14, 0, 0, 0]
P20	0	0	(104) +4	(2) 2	(36) +12	0	[0, 0, 12, 0, 0, 0]
P21	0	0	(104) +4	(2) 1	(36) 0	0	[0, 0, 0, 12, 0, 0]
P22	0	2	(156) +6	(2) 2	(36) 0	0	[5, 0, 0, 13, 3, 0]
P23	0	2	(88) +2	(2) 2	(24) 0	0	[0, 0, 0, 10, 0, 0]
P24	0	0	(104) +4	(2) 2	(36) +12	0	[0, 0, 12, 0, 0, 0]
P25	0	1	(95) +1	(2) 2	(36) 0	12	[8, 0, 0, 0, 4, 0]
P26	0	2	(104) +4	(2) 2	(36) 0	0	[0, 0, 0, 0, 12, 0]
P27	0	2	(104) +4	(2) 2	(36) 0	0	[0, 0, 0, 14, 0, 0]
P28	0	2	(52) +2	(2) 2	(24) 0	0	[0, 0, 0, 6, 0, 0]
P29	0	0	(104) +4	(2) 2	(36) +12	0	[11, 0, 0, 0, 0, 0]
P30	0	2	(104) +4	(2) 2	(36) 0	0	[0, 11, 0, 0, 0, 0]
P31	0	0	(104) +4	(2) 2	(36) +12	0	[7, 4, 0, 0, 0, 0]
P32	0	2	(104) +10	(2) 2	(36) 0	0	[0, 0, 13, 0, 0, 0]
P33	0	1	(104) +10	(2) 2	(36) 0	-12	[0, 0, 0, 0, 14, 0]
P34	0	2	(52) +2	(2) 2	(24) 0	0	[0, 0, 0, 0, 6, 0]
P35	0	1	(130) +8	(2) 2	(36) 0	-12	[0, 0, 0, 0, 0, 18]
P36	0	0	(104) +4	(2) 2	(36) +12	0	[0, 0, 0, 13, 0, 0]
P37	0	0	(130) +8	(2) 2	(36) +12	0	[0, 0, 0, 0, 11, 6]
P38	0	2	(104) +4	(2) 2	(36) 0	0	[0, 0, 0, 0, 13, 0]
P39	0	2	(130) +2	(3) 3	(36) +12	0	[0, 3, 0, 0, 0, 12]
P40	0	0	(104) +4	(2) 2	(36) +12	0	[0, 0, 0, 14, 0, 0]
P41	0	0	(104) +4	(2) 2	(36) +12	0	[0, 11, 0, 0, 1, 0]
P42	0	0	(104) +10	(2) 2	(36) +12	0	[2, 10, 0, 0, 0, 0]
P43	0	1	(104) +4	(2) 2	(36) +12	12	[0, 0, 0, 0, 0, 13]
P44	0	0	(104) +10	(2) 2	(36) +12	0	[7, 3, 0, 0, 0, 4]
P45	0	2	(52) +8	(2) 2	(24) 0	-24	[0, 0, 0, 0, 0, 6]
P46	0	1	(130) +8	(2) 2	(36) 0	-12	[0, 0, 0, 0, 0, 18]
P47	1	1	(130) +2	(2) 2	(36) +12	36	[0, 0, 0, 0, 0, 16]

Overtime: 231h; Debt: 0h; Difference: +231h

Table 2.10, which corresponds to May of 2019, had more physicians on vacation compared to Table 2.9 (April of 2019). Overtime was also more prevalent (+231), however, still far below the average (+400) when the roster was organized manually. Another consequence is the number of working hours on non-business days (S8) and the day and night shift balance (S10), which presented more violations compared to the previous month.

Managers considered the presented results much better than those generated manually. The primary reasons being the reduction of overtime, better distribution of overtime and working hours on non-business days between physicians. Another important remark is the reduction of mistakes in the roster. For example, scheduling a physician for a Night shift on the last day of the previous month and an Early or Late shift the first day of the next month, was a common error when the rosters were organized manually. Moreover, the development of the overview tables improved transparency of how the rosters are prepared and organized.

2.7 Conclusions

The present chapter proposed an integer programming formulation and a fix-and-optimize matheuristic for the PRP. Moreover, a comparison between constraints present in the NRP and the studied PRP demonstrate similarities and differences between these problems. A basic model was developed addressing the most common constraints to generate a physician roster. The extended model aims to improve the balance of overtime, working hours during non-business days and working hours during day and night shifts. Such constraints are important to be considered when real-world solving methods are developed because the resulting roster is fair between the scheduled physicians.

Computational experiments indicate that both MIP solvers, CPLEX and Coin-OR CBC, were capable to generate optimal and near-optimal results, respectively, when solving small instances with up to 50 physicians. For these small instances, the computation times of Coin-OR CBC were similar to CPLEX. Open-source solver such as Coin-OR CBC, therefore, is a suitable alternative to commercial solvers when the number of physicians to schedule is limited. For larger instances, with 100 and 150 physicians, both solvers were unable to find feasible solutions in most of the instances within an acceptable computation time limit.

A fix-and-optimize matheuristic was proposed to improve these results employing the standalone solvers. Results near-optimum were generated for instances with 50 physicians in 10 minutes. Moreover, utilizing instances with 100 and 150 physicians, even with short computation time limits (20 and 30 minutes), the F&O matheuristic generated good results. For the larger instances (150 physicians) the LAHC proposed by Sanchotene and Buriol (2018) obtained slightly better results when compared to the F&O matheuristic.

Manual rosters had an average of 400 hours of overtime, this number was reduced to 35 and 231 using the proposed IP formulation in April and May of 2019. The primary reason for the significant reduction is due to the capability of physician reallocation in different locations. It was not possible with manual scheduling because the number of possibilities for a human to organize the roster rendered the problem very difficult to solve manually. Optimality for instances with 100 and 150 physicians was not proved. Methods such as branch-and-price or network flow models are perspectives for future research.

3 CYCLIC ROSTERING

3.1 Introduction

Cyclic workforce rostering, also referred to as rotating workforce rostering in the academic literature, generates, for each team, a fixed roster that repeats after a certain period (MUSLIU, 2006). Table 3.1 presents an example of a roster with nine teams and a planning horizon of one week. Shifts are categorized as Early (E), Late (L) or Night (N), while dashes represent days off. The same roster might be considered cyclic if in the second week Team1 works the schedule of Team2, Team2 works the schedule of Team3 and so on. In effect the work schedule shifts by one position each week, with the pattern repeating every nine weeks. The primary advantage associated with cyclic rostering is fairness since all employees are subject to the same composition of shifts over each nine-week period. Another advantage is that employees know their roster long beforehand. However, a major disadvantage lies in the fact that individual roster preferences are difficult to accommodate.

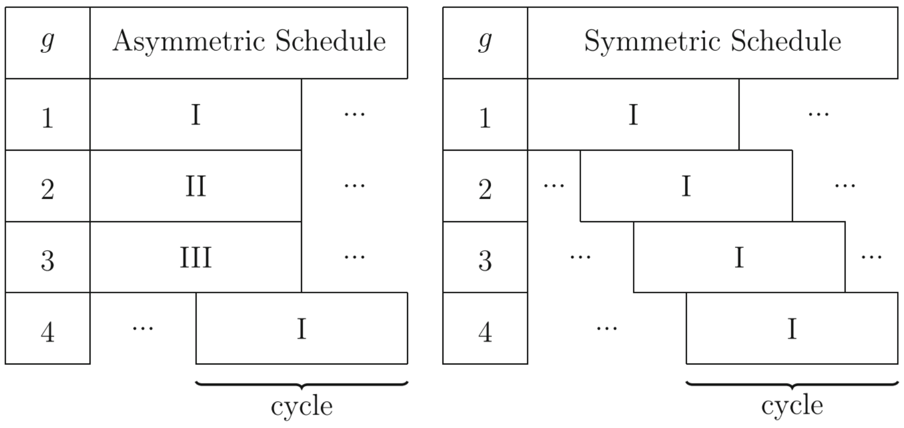
Table 3.1: Example of a roster with nine teams over a seven day period.

Team	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Team1	-	E	E	E	E	E	N
Team2	N	-	-	E	E	E	E
Team3	E	N	N	-	-	L	L
Team4	L	L	L	N	N	-	-
Team5	-	L	L	E	E	N	N
Team6	N	-	-	L	L	E	E
Team7	-	-	-	L	L	L	L
Team8	L	N	N	-	-	-	-
Team9	E	E	E	N	N	N	-

Cyclic schedules have a property which defines their type of cyclicality as being either symmetric or asymmetric (BECKER; STEENWEG; WERNERS, 2018). When the cyclic problem is symmetric all teams work the same schedule lagged in time, whereas in the asymmetric version teams work different schedules. Figure 3.1 presents an example of this terminology. Table 3.1 presented an example of a symmetric cyclic roster because teams work the same schedule lagged in time. However, if at least one team has its own unique pattern which is cycled through every nine weeks, then this roster would be asymmetric.

Integer programming models have been successfully employed for general workforce rostering problems and generate excellent results in reduced computational time.

Figure 3.1: Asymmetric and symmetric rosters with four teams (BECKER; STEENWEG; WERNERS, 2018)



The present study therefore investigates the following gaps in the existing literature, namely: *(i)* can existing models for general workforce rostering be adapted to cyclic rostering? *(ii)* what is the impact upon the performance? and *(iii)* considering multi-skilled teams in the cyclic rostering model would it be possible to minimize understaffing? *(iv)* How do multi-skilled employees in cyclic rostering models affect the performance?

In comparison to general workforce rostering problems, cyclic rostering has received far less attention throughout the academic literature. Musliu, Gärtner and Slany (2002) developed an algorithm focused on interaction with decision-makers during the generation of partial solutions such as choosing the length of working blocks and setting particular sequences of working and days off while respecting the minimum staff requirements. The algorithm was included in a commercial software package called First Class Scheduler (FCS). Musliu (2006) introduced 20 real-life cyclic workforce instances, collected from various areas of the industry and made them publicly available. In a follow-up study, Musliu (2006) proposed three methods for cyclic workforce rostering: tabu search, a min-conflict heuristic and a hybrid of both. Computational experiments demonstrated that the proposed heuristics generated good results in reduced computational time.

Rocha, Oliveira and Carravilla (2013) formulated cyclic rostering as an integer programming model for a case study focusing on glass manufacturing. Some predefined shift patterns must be respected. A rest shift (B) is mandatory between each working shift change (M - Morning, N - Night, A - Afternoon) resulting in a pattern M-B-N-B-A-B. Results demonstrate that the integer programming model was capable of solving the particular cyclic workforce problem for this glass manufacturing context. However, when applied to the benchmark instances of Musliu (2006), the solving method was unable to find feasible solutions on 8 out of 20 instances.

Triska and Musliu (2011) used constraint programming and intended to improve the FCS method (MUSLIU, 2006). The advantages of their new method concern its easy maintenance, portability across programming languages and its declarative approach which makes extensions easier. Computational experiments demonstrate that the proposed method generates competitive results compared against FCS.

Erkinger and Musliu (2017) proposed a new approach by formulating the cyclic rostering problem as a *Satisfiability Modulo Theories* (SMT) problem. This enables the use of SMT-solvers to generate cyclic rostering solutions. In comparison to previous literature using exact methods, the proposed approach resulted in an increased number of feasible solutions when the benchmark instances were employed.

Musliu, Schutt and Stuckey (2018) implemented two solver-independent models for the cyclic rostering problem using constraint programming and mixed integer programming. Using these models, different solvers could be used to optimize different objectives such as speed to solution and robustness of solving. Computational results indicate that the solver-independent was capable of solving all the benchmark instances (MUSLIU, 2006).

Becker, Steenweg and Werners (2018) proposed an integer programming model for cyclic medical emergency service rostering considering unexpected employee absences. To minimize these disruptions caused by unforeseen events, a limited number of employees are assigned to on-call shifts where employees are expected to be available at all times. This methodology is integrated into an integer programming model and, in the same way as regular shifts, on-call shifts rotate after each cycle to ensure fairness. Computational experiments, based on data provided by a German medical emergency service demonstrated that large instances were solved to optimality.

Table 3.2 presents a classification of existing literature according to the type of cyclic rostering problem addressed and the solving technique proposed. Teams are composed of one or more employees. The majority of these publications applied exact methods such as IP, CP or SAT solvers. Only Musliu (2006) introduced heuristic methods. The industrial instances used are case-specific problem variants approached in each cited publication. The benchmark instances proposed by Musliu (2006) were used for experimentation by all the approaches except Becker, Steenweg and Werners (2018), who applied the solving method in a German medical emergency service. Moreover, except for the present research, all other studies addressed problems with single-skilled teams. Note that all the constraints present in the benchmark instances can be derived and adapted from the general workforce rostering such as nurse and physician rostering. The objective function was only present in industrial problems. Rocha, Oliveira and Carravilla (2013) minimize the maximum number of days that a team works in each shift, while Becker, Steenweg and Werners (2018) employ a weighted sum to maximize the number of free weekends and minimize the number of night-to-day rotations.

Table 3.2: Cyclic rostering literature classification.

References	Single-skill	Multi-skills	Number of teams	Scheduling horizon (days)	Solving method
Musliu (2006)					
Benchmark instances	✓		7-163	63-1141	Heuristic
Triska and Musliu (2011)					
Benchmark instances	✓		7-163	63-1141	CP
Rocha, Oliveira and Carravilla (2013)					
Benchmark instances	✓		7-163	63-1141	IP
Industry instances*	✓		5-49	25-30	IP
Erkinger and Musliu (2017)					
Benchmark instances	✓		7-163	63-1141	SAT
Musliu, Schutt and Stuckey (2018)					
Benchmark instances	✓		7-163	63-1141	CP/IP
Becker, Steenweg and Werners (2018)					
Industry instances	✓		80	28-35	IP
This research					
Benchmark instances	✓		7-163	63-1141	IP
Industry instances		✓	4	28	IP

IP = Integer programming, CP = Constraint programming, SAT = Satisfiability solver, (*) Not publically available.

In contrast to existing approaches which primarily focus on finding feasible solutions, the present research addresses the problem by proposing an objective function to minimize understaffing. Another difference is a complex multi-skill team structure which is required to perform the tasks. The data was provided by a research institute based in Leuven, Belgium. Since this problem is new and there is no basis for comparison with existing literature, the proposed model was generalized to also solve the 20 benchmark instances proposed by Musliu (2006). The objective of this experiment is to prove the effectiveness of the proposed IP model compared against existing solving methods.

The remainder of this chapter is organized as follows. Section 3.2 introduces the cyclic rostering problem. Section 3.3 details the solution representation used in the IP formulation. Section 3.4 presents the integer programming formulation. Section 3.5 describes the computational experiments, while Section 3.6 presents conclusions and outlines future research directions.

3.2 Cyclic rostering definition

General cyclic rostering problems can be defined by a set of teams $N = \{1, \dots, |N|\}$ (a team is composed by one or more employees), a set of days $D = \{0, \dots, |D|\}$, a set of shifts $S = \{1, \dots, |S|\}$, and a set of skills $K = \{1, \dots, |K|\}$. The goal is to either find a feasible solution for the problem or to minimize an objective function which aggregates the soft constraints violations. The present problem considers multi-skilled teams and requires an objective function to minimize understaffing and overstaffing per day and shift in relation to specific skills.

The benchmark instances used to compare the IP model with results available in the literature, conform to a particular variant of the general definition because only single-skilled employees are scheduled. Observe that for this variant a team can be viewed as an employee because it is composed by a single employee. The goal is to find a feasible solution by assigning shifts to employees subject to a set of hard constraints.

It is important to highlight that when the single-skilled version of the problem is addressed, the scheduling horizon can be reduced to only a single week. This is possible because all employees work the same schedule with one week offset and there are no constraints which require evaluation over longer scheduling horizons such as the maximum number of worked days over the entire period or the maximum number of worked weekends per month.

3.3 Solution representation

3.3.1 Single-skilled cyclic rostering

Table 3.3 provides an example of a roster for the single-skilled variant of cyclic rostering. It is represented by a single row where teams work with seven days of offset. The roster presented in Table 3.3 reproduces from a different perspective, the same first two rows of Table 3.1. Instead of only showing the first week per line, it has a consecutive view without breaks. Using this notation, weeks are naturally connected and the numerical row represents the days. Moreover, the last week *(Team n)* is connected to the first week *(Team 1)*. This notation was introduced by Felici and Gentile (2004) to simplify the formulation of cyclic problems.

Table 3.3: Single-skilled cyclic rostering problem.

Team 1							Team 2							Team n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	…
M	T	W	T	F	S	S	M	T	W	T	F	S	S	…
-	E	E	E	E	E	N	N	-	-	E	E	E	E	…

3.3.2 Multi-skilled cyclic rostering

Multi-skilled cyclic rostering is motivated by a problem provided by a semiconductor institute. The problem consists of teams with multiple skills and there must be at least one employee working per skill at all times. Note that existing literature does not address such a problem and therefore the present research proposes a new IP formulation. Teams must perform tasks which require specific skills during their working shifts.

Table 3.4 provides an example of the multi-skilled variant of the cyclic rostering problem. The solution representation is not a single row anymore but multiple rows representing teams. Such a structure is necessary due to the heterogeneous skill structure. Teams T1 and T2 both work the same schedule, but with seven days offset. The set of teams $|N|$ and days $|D|$, now have the same representation as a classic rostering problem, the only difference are the constraints at the end of the schedule which must be connected to those at the beginning.

Table 3.4: Multi-skilled cyclic rostering problem.

	Week 1							Week 2							Week n
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	...
	M	T	W	T	F	S	S	M	T	W	T	F	S	S	...
T1	E	E	E	E	E	-	-	N	N	N	-	-	E	E	...
T2	N	N	N	-	-	E	E	E	E	E	E	E	-	-	...

3.4 Integer programming formulation

Table 3.5 details the sets, decision and auxiliary variables utilized throughout the proposed IP formulation for the multi-skilled cyclic rostering problem. This formulation includes the constraints provided by the semiconductor research institute, which generalizes the cyclic rostering problem proposed by Musliu (2006). Note that to have the notation which respects the cyclicality of the problem, variable d is always considered cyclical when receiving values from the set of days D. In other words, d = $(d \ mod \ |D|)$ throughout the formulation. Such a notation is necessary, for example, to check invalid shift successions between the last and first days of the scheduling horizon.

Table 3.5: Indices, sets and variables used in the formulation.

Symbol	Definition
Input Data	
$n \in N$	n is the team index, and N is the set of team indices;
$d \in D$	d is the day index, and D is the set of day indices (representing the scheduling horizon), note that d is always considered as $d \ mod \ \lvert D \rvert$ throughout the formulation;
$\hat{d} \in \hat{D}$	$\hat{d}$ is the day index, and $\hat{D} = \{0, \ldots, 6\}$ is the set of day indices of a week;
$\tilde{d} \in \tilde{D}$	$\tilde{d}$ is the day index, and $\tilde{D} = \{x_1, x_2, x_3, \ldots, x_n\}$ is the set of day indices with an offset of seven, $0 \leq x_n \leq \lvert N \rvert$, that is, $\tilde{D} = \{0, 7, 14, \ldots, x_n\}$;
$s \in S$	s is the shift index, and S is the set of shift indices;
$k \in K$	k is the skill index, and K is the set of skill indices;
$l_{nk} \in \{0, 1\}$	a binary parameter equal to one if team n has skill k, and zero otherwise;
$r_{ds} \in \mathbb{N}_{\geq 0}$	number of required teams on day d and shift s;
$(s1, s2) \in \hat{S}$	$\hat{S}$ contains the pairs of invalid shift sequences;
$(s1, s2) \in \tilde{S}$	$\tilde{S}$ contains the pairs of invalid shift sequences with a day off between them;
T^w	set of patterns $T^w = \{T^w_t : t \in \{1, 2, \ldots, p^w\}\}$, where p^w is the minimum number of consecutive working days minus one. T^w_t is a binary vector of dimension $t+2$, with a zero at both the first and the last position, with t being the number of ones that appear in it. For example, considering four as the minimum number of consecutive working days, the patterns to search are $T^w = \{T^w_1 = (0, 1, 0), T^w_2 = (0, 1, 1, 0), T^w_3 = (0, 1, 1, 1, 0)\}$. None of these patterns are allowed because they represent fewer than four consecutive working days;
T^r	follows the same idea as T^w, and represents a set of patterns $T^r = \{T^r_t : t \in \{1, 2, \ldots, p^r\}\}$, where p^r is the minimum number of consecutive days off minus one;
T^s	follows the same idea as T^w, and represents a set of patterns $T^s = \{T^s_{t_s} : t_s \in \{1, 2, \ldots, p^s\}\}$, where p^s is the minimum number of consecutive working days minus one for shift s;

$\underline{D}^{on}, \overline{D}^{on}$	the minimum and maximum number of consecutive working days, respectively;
$\underline{D}^{off}, \overline{D}^{off}$	the minimum and maximum number of consecutive days off, respectively;
$\underline{S}_s, \overline{S}_s$	the minimum and maximum number of consecutive assignments to shift s, respectively;
$\overline{W}$	maximum number of worked weekends over the scheduling horizon;
$w \in W$	w is a Saturday index and W the set of all Saturday indices;
$\tilde{w} \in \tilde{W}$	$\tilde{w}$ is a Saturday or Sunday index and $\tilde{W}$ is the set of all Saturday and Sunday indices;
ρ	penalty for a shift with understaffing during a weekend.

Decision Variables

$x_{nds} \in \{0,1\}$	1 if team n is allocated to shift s on day d, 0 otherwise;
$y_{nw} \in \{0,1\}$	1 if team n works on weekend w, 0 otherwise.

Auxiliary Variables

$u_{dsk} \in \mathbb{N}_{\geq 0}$	number of violations of the minimum number of teams per day, shift and skill.
$o_{dsk} \in \mathbb{N}_{\geq 0}$	number of violations of the maximum number of teams per day, shift and skill.

$$\textbf{Minimize:} \sum_{d \in D \setminus \tilde{W}} \sum_{s \in S} \sum_{k \in K} (u_{dsk} + o_{dsk}) + \rho \sum_{\tilde{w} \in \tilde{W}} \sum_{s \in S} \sum_{k \in K} (u_{\tilde{w}sk} + o_{\tilde{w}sk}) \tag{3.1}$$

Subject to

$$\sum_{s \in S} x_{nds} \leq 1 \qquad \forall n \in N, d \in D \tag{3.2}$$

$$x_{nds1} + x_{n(d+1)s2} \leq 1 \qquad \forall n \in N, d \in D, (s1,s2) \in \hat{S} \tag{3.3}$$

$$x_{nds1} - \sum_{s \in S} x_{n(d+1)s} + x_{n(d+2)s2} \leq 1 \qquad \forall n \in N, d \in D, (s1,s2) \in \tilde{S} \tag{3.4}$$

$$\sum_{d'=d}^{t+d+1} \sum_{s \in S} x_{nd's} + \sum_{d' \in \{d,\, t+d+1\}} \sum_{s \in S} x_{nd's} \underline{D}^{on} + \sum_{d'=d+1}^{t+d} (1 - \sum_{s \in S} x_{nd's}) \underline{D}^{on} \geq \underline{D}^{on}$$
$$\forall n \in N, t \in \{1, \dots p^w\}, d \in \{0, \dots, |D|-1\} \tag{3.5}$$

$$\sum_{d'=d}^{\overline{D}^{on}+d} \sum_{s \in S} x_{nd's} \leq \overline{D}^{on} \qquad \forall n \in N, d \in \{0,\ldots,|D|-1\} \tag{3.6}$$

$$\sum_{d'=d}^{t+d+1} (1-\sum_{s \in S} x_{nd's}) + \sum_{d' \in \{d,\, t+d+1\}} (1-\sum_{s \in S} x_{nd's})\underline{D}^{off} + \sum_{d'=d+1}^{t+d} \sum_{s \in S} x_{nd's}\underline{D}^{off} \geq \underline{D}^{off}$$

$$\forall n \in N, t \in \{1,\ldots p^r\}, d \in \{0,\ldots,|D|-1\} \tag{3.7}$$

$$\sum_{d'=d}^{\overline{D}^{off}+d} (1-\sum_{s \in S} x_{nd's}) \leq \overline{D}^{off} \qquad \forall n \in N, d \in \{0,\ldots,|D|-1\} \tag{3.8}$$

$$\sum_{d'=d}^{t_s+d+1} x_{nd's} + \sum_{d' \in \{d,\, t_s+d+1\}} x_{nd's}\underline{S}_s + \sum_{d'=d+1}^{t_s+d} (1-x_{nd's})\underline{S}_s \geq \underline{S}_s$$

$$\forall n \in N, s \in S, t_s \in \{1,\ldots p^s\}, d \in \{0,\ldots,|D|-1\} \tag{3.9}$$

$$\sum_{d'=d}^{\overline{S}_s+d} x_{nd's} \leq \overline{S}_s \qquad \forall n \in N, s \in S, d \in \{0,\ldots,|D|-1\} \tag{3.10}$$

$$\sum_{n \in N} \sum_{\tilde{d} \in \tilde{D}} x_{n(\tilde{d}+\hat{d})s} = r_{\hat{d}s} \qquad \forall \hat{d} \in \hat{D}, s \in S \tag{3.11}$$

$$\sum_{n \in N} l_{nk} x_{nds} + u_{dsk} - o_{dsk} = 1 \qquad \forall d \in D, s \in S, k \in K \tag{3.12}$$

$$x_{nds} = x_{(n+1)(d+7)s} \qquad \forall n \in \{1,\ldots,|N|-1\}, d \in D, s \in S \tag{3.13}$$

$$\sum_{s \in S} (x_{nws} + x_{n(w+1)s}) = 2y_{nw} \qquad \forall n \in N, w \in W \tag{3.14}$$

$$\sum_{w \in W} y_{nw} \leq \overline{W} \qquad \forall n \in N \tag{3.15}$$

$$\sum_{w \in W} \sum_{s \in S} (x_{n(w-1)s} + x_{n(w+2)s} + y_{nw}) \leq 2 \qquad \forall n \in N \tag{3.16}$$

Constraints (3.2) ensure a single worked shift per day. Constraints (3.3) ensure that a shift succession is valid. Constraints (3.4) ensure that a sequence of working shift, day off, working shift is a valid succession. However, this restriction is not included in the model if the minimum number of consecutive days off is greater than one. Constraints (3.5) ensure the minimum number of consecutive working days, calculated as the *(sum of the working days)* + *(two border bits* $\times$ $\underline{D}^{on}$*)* + *(complement of middle bits* $\times$ $\underline{D}^{on}$*)*. Constraints (3.6) ensure the maximum number of consecutive working days. Constraints (3.7) ensure the minimum number of consecutive days off. These constraints are evaluated similarly to Equation (3.5), the only difference lies in the fact that bits are inverted and the sum is relative to days off rather than working days. Constraints (3.8) ensure the maximum number of consecutive days off. Constraints (3.9) ensure the minimum number of consecutive working days with the same shift. These constraints are also evaluated similarly to Equation (3.5). Constraints (3.10) ensure the maximum number of consecutive working days with the same shift. Constraints (3.11) ensure the required number of teams per day and shift.

Constraints (3.12) calculate the minimum and maximum demand violations. Constraints (3.13) ensure that teams work the same schedules with seven days offset. Constraints (3.14) ensure that teams work complete weekends, which means that they either work both Saturday and Sunday or they have both days off. Constraints (3.15) ensure that teams do not work more than the maximum number of weekends over the planning horizon. Constraints (3.16) ensure that a team working a weekend must have either the Friday preceding or Monday succeeding off.

Neither a objective function nor constraints (3.12)–(3.16) are necessary for the cyclic rostering problem proposed by Musliu (2006). A feasible solution to the problem is sufficient.

3.5 Computational experiments

Computational experiments are conducted using instances from both industry as well as benchmarks available in the literature. The objective is to evaluate the application of the IP formulation to real-world problems and check the performance compared to other solving methods. All models were implemented in Java and compiled with OpenJDK 1.8. The experiments were conducted on an Intel Core i5-2410M CPU 2.30GHz (2 cores) with 6GB of RAM memory running Linux Mint 17.2 64-bits. CPLEX version 12.7.1 with default parameters was used to solve the integer programming formulation.

3.5.1 Research institute instances

This section details a real-world problem provided by a semiconductor research institute. In contrast to published approaches concerning cyclic rostering problems, this one includes teams with multiple skills. An objective function is necessary to minimize the demand deviation, which in effect means that the complete set of required skills should be available at any moment during each 24-hour schedule. This institute groups employees into teams which are composed to cover various skills. These teams are given as input data. During weekdays teams are assigned to 8-hours shifts that are Early, Late, Night or they have the day off. Weekends have a special shift regime with only two 12-hour shifts classified as Day and Night, with a team either working both weekend days or neither. Each team is allowed to work a maximum two weekends out of four. Moreover, the minimum and maximum number of consecutive working days are two and five, respectively. The minimum and maximum number of consecutive days off are two and three, respectively.

3.5.1.1 Experimental setup and data analysis

Table 3.6 details the original skill structure provided by the research institute. Each instance (Instance01 and Instance02) corresponds to a different production hall. Instance01 has 183 unique skills while Instance02 has 181. The proportion of skills that are covered is represented by the percentage columns. These skills are required to operate the machinery and to perform various production-related tasks. Currently, not all employees are trained to use all the machines. The objective of this experiment is to investigate the most appropriate skill structure for the teams. Note that this experiment helps direct managerial decision-making since it will be possible to determine whether investing in training will actually result in a reduction of understaffing.

Table 3.6: Original skill structure of the four teams.

	Instance01		Instance02	
	#Skills	Percentage	#Skills	Percentage
Team1	137	74.86	167	92.27
Team2	177	96.72	160	88.40
Team3	166	90.71	174	96.13
Team4	150	81.97	158	87.29
Average	-	86.07	-	91.02

After analyzing the data provided by the research institute it was observed that if a team has a sequence of working days, then they must all be worked on the same shift. Such shift successions are very restrictive and raised the following question: what would be the effect of relaxing this constraint by allowing more shift successions? Another question is: what would be the result if all employees were qualified to perform any task? The left-hand side of Table 3.7 corresponds to the original shift succession, while the right-hand side is when these constraints are relaxed.

Table 3.7: Original and relaxed shift succession.

Original Shift Succession				Relaxed Shift Succession			
Shift	Invalid Succession			Shift	Invalid Succession		
Early	Late	Night	Day	Early	-	-	-
Late	Early	Night	Day	Late	Early	Day	-
Night	Early	Late	Day	Night	Early	Late	Day
Day	Early	Late	Night	Day	-	-	-

3.5.1.2 Computational results

Tables 3.8 and 3.9 provide the results using the institute's current skill structure and the relaxed shift succession to estimate the impact of when all teams have been trained to handle all machines, respectively. The first and fifth columns provide the instance identification, while OFV represents the Objective Function Value, Gap (%) is the relative gap to the optimum solution and the Time column represents the time in seconds required to achieve the optimum solution.

Table 3.8: Computational results employing research institute instances with original skill structure.

Original skill structure and shift succession				Original skill structure and relaxed shift succession			
	OFV	Gap(%)	Time(s)		OFV	Gap(%)	Time(s)
Instance01	1836	0.00	0.23	Instance01	1836	0.00	0.23
Instance02	1170	0.00	0.19	Instance02	1170	0.00	0.13

Computational results indicate that relaxing invalid shift succession did not reduce understaffing. Such a result can be explained by the fact that the number of assignments could not be increased by only relaxing the shift succession constraints. In addition, the OFV of Instance02 is lower than Instance01. Note that this is due to Instance01's teams cover 86.07% of the possible skills, while Instance02's teams cover 91.02% of the skills. This clearly indicates a gain, in terms of coverage demand, when teams have more qualified employees.

Table 3.9 provides the results supposing the teams are trained to cover all skills required in the research institute. This experiment could also be interpreted as a skill-less environment since every team is qualified to perform all tasks. Computational experiments demonstrate that in both cases, using the original shift succession and the relaxed shift succession generated zero as OFV. These results prove the entirely intuitive assumption that investing in training would provide a significant reduction in the amount of skills not covered during shifts.

Table 3.9: Computational results employing research institute instances with full-skill structure.

Full-skill structure and original shift succession				Full-skill structure and relaxed shift succession			
	OFV	Gap(%)	Time(s)		OFV	Gap(%)	Time(s)
Instance01	0	0.00	0.11	Instance01	0	0.00	0.12
Instance02	0	0.00	0.11	Instance02	0	0.00	0.13

It can thus be concluded that investment in training is the best choice for the research institute to avoid skilled employee shortage. Moreover, a reduction concerning on-call employees, currently used to cover this understaffing, could be reduced which would also provide a reduction in the overall cost, although there is the initial investment in employees' training which requires consideration. Computational execution times remain very low when changing the problem characteristics concerning teams skills and shift successions.

3.5.2 Benchmark instances

In addition to the computational experiments on research institute instances, experiments were also conducted using benchmark instances[1]. The primary objective is to validate the proposed IP formulation in an academic context. Table 3.10 provides the characteristics of the instances. The first column presents the instance Id, while others provide the number of days, employees and shifts, respectively. Employees are all single-skilled.

Table 3.10: Benchmark instances characteristics.

Instance Id	#Days	#Employees	#Shifts
1	63	9	3
2	63	9	3
3	119	17	3
4	91	13	3
5	77	11	3
6	49	7	3
7	203	29	3
8	112	16	3
9	329	47	3
10	189	27	3
11	210	30	3
12	140	20	2
13	49	7	3
14	91	13	3
15	448	64	3
16	203	29	3
17	231	33	2
18	371	53	3
19	840	120	3
20	1141	163	3

Table 3.11 presents the computational results employing the benchmark instances. 20 instances were evaluated and compared to published results (columns 2-6 in Table 3.11). MC-T[1] and FCS[1] are the results obtained by Musliu (2006), where MC-T is a min-conflicts heuristic with tabu list and FCS is a commercial software package called

[1]Available at <http://www.dbai.tuwien.ac.at/staff/musliu/benchmarks/>

First Class Scheduler. Columns 4-6 provide the results of CP-Rota[2] (TRISKA; MUSLIU, 2011), MIP[3] (ROCHA; OLIVEIRA; CARRAVILLA, 2013) and MathSAT5[4] (ERKINGER; MUSLIU, 2017), which are solving methods based on constraint programming, mixed integer programming and a SAT solver, respectively.

Since the previously published methods were executed on different computers, the values were standardized using a benchmark website[2]. The last column presents the results obtained by the MIP model proposed in this work. Results are presented in seconds with the symbol *greater* meaning that no feasible solution was found within the given time limit.

As can be observed in the last column, all instances except one were solved in less than seven seconds. When compared to MC-T the computation time was similar in the majority of the instances. The proposed approach always found feasible solutions in less than seven seconds, with exception of Instance7, while FCS was unable to find feasible solutions for instances 9, 12, 13, 15, 17, 19 and 20.

Table 3.11: Computation time in seconds for benchmark instances.

Instance	MC-T[1]	FCS[1]	CP-Rota[2]	MIP[3]	MathSAT5[4]	MIP[5]
1	<1.00	<1.00	<1.00	<1.00	<1.00	<1.00
2	<1.00	<1.00	<1.00	<1.00	<1.00	<1.00
3	<1.00	<1.00	<1.00	79.00	<1.00	<1.00
4	<1.00	<1.00	<1.00	<1.00	<1.00	<1.00
5	<1.00	<1.00	<1.00	<1.00	<1.00	<1.00
6	<1.00	<1.00	<1.00	<1.00	<1.00	<1.00
7	2.68	<1.00	<1.00	>1741.32	>1000	187.19
8	<1.00	6.30	48.96	<1.00	<1.00	<1.00
9	<1.00	>50.79	<1.00	>1741.32	1.01	<1.00
10	<1.00	<1.00	>50.79	>1741.32	<1.00	<1.00
11	<1.00	18.64	>50.79	>1741.32	3.37	6.89
12	<1.00	>50.79	>50.79	26.99	1.56	2.98
13	<1.00	>50.79	5.79	22.28	<1.00	<1.00
14	<1.00	<1.00	47.74	6.37	<1.00	<1.00
15	8.08	>50.79	>50.79	>1741.32	>1000	6.46
16	<1.00	<1.00	10.97	167.43	<1.00	<1.00
17	<1.00	>50.79	<1.00	2.58	<1.00	<1.00
18	<1.00	<1.00	>50.79	>1741.32	6.23	<1.00
19	3.85	>50.79	>50.79	>1741.32	489.32	<1.00
20	3.63	>50.79	>50.79	>1741.32	355.21	<1.00

[1]Musliu (2006), [2]Triska and Musliu (2011), [3]Rocha, Oliveira and Carravilla (2013), [4]Erkinger and Musliu (2017), [5]this work.

Rocha, Oliveira and Carravilla (2013)'s method was incapable of finding feasible solutions within 1741.32 seconds for instances 7, 9, 10, 11, 15, 18, 19 and 20 whereas the proposed MIP formulation solved these instances in less than seven seconds, except

[2]https://www.cpubenchmark.net/

Instance7 where the solver required 187.19 seconds to find a feasible solution. Although there is a significant improvement in terms of computation time and feasibility when compared against the MIP model proposed by Rocha, Oliveira and Carravilla (2013), this conclusion must be considered with caution since the CPLEX version is not the same. To find a feasible solution for Instance7, it was necessary to add a slack variable relaxing the maximum demand. Results demonstrate a clear advantage of using an efficient IP formulation combined with a MIP solver over other methods based on constraint programming and SAT solvers.

3.6 Conclusions

The present research contributes addressing industrial requirements concerning cyclic personnel rostering in a multi-skilled environment. It provides recommendations for improving understaffing during the scheduling horizon. To achieve these results three primary issues were investigated: *(i)* designing an objective function to minimize understaffing, *(ii)* examining the impact of shift succession relaxations, and *(iii)* training the employees enabling them to assume more tasks.

The effectiveness was demonstrated throughout the computational experiments employing real-world data provided by a semiconductor research institute based in Leuven, Belgium. Computational results demonstrated a reduction of understaffing indicating that cross-training employees is a worthwhile investment. Relaxing the shift succession constraints, however, showed that they did not contribute to a reduction of understaffing. This was primarily due to the fact that the number of assignments could not be increased. Therefore, results demonstrated that shift succession constraints may be maintained as the current definition.

Experiments were also conducted employing benchmark instances. Computational results demonstrated that previously unsolved instances using mathematical formulations were solved to optimality in a few seconds. Moreover, the proposed approach significantly outperformed other methods such as constraint programming and methods based on SAT solvers. Future research should consider the possibility of generalizing workforce MIP models to also address cyclic workforce rostering problems. Finding a trade-off training cost and improved coverage is another possibility to be explored in future research.

4 NURSE REROSTERING

4.1 Introduction

Work absences occur due to a variety of reasons. Depending on the working area, unscheduled absence rates typically range from 5% to 10%. Emergency services and healthcare in hospitals have the highest absenteeism rates (10.7%) when compared against other sectors such as, for example, utilities (8.7%), transportation (8.5%), customer services (7.7%) or manufacturing (6.4%) (AGUIRRE; KERIN, 2014). Forbes (2013) estimates the annual cost of lost productivity in the United States to be \$3.6 billion for nurses and \$0.25 billion for physicians.

Effectively managing disruptions caused by high absenteeism rates is thus clearly of vital importance but at the same time presents a difficult task to manual planners. A solving method based on optimization techniques facilitates the decision-maker whenever a quick solution for handling disruptions is required. The Nurse Rerostering Problem (NRRP), as defined by Moz and Pato (2003), occurs when one or more nurses cannot work in the shifts previously assigned to. If no pool of reserve nurses exists to replace those absent, the current roster must be rebuilt using a rerostering method.

This work revisits the NRRP by exploring several strategies based on relaxations of specific problem parameters which may be applied when addressing the disruptions. First, a general integer programming formulation is developed which includes both the constraints from the Nurse Rostering Problem (NRP) and the additional NRRP restrictions. Second, two types of relaxation strategies are proposed and evaluated. The first strategy determines which part of the scheduling horizon to consider when rerostering: either the complete period or only a restricted part. The second strategy concerns which constraints are included when solving the NRRP. An approach which relaxes some of the NRP constraints and only includes the NRRP constraints is evaluated. Finally, a Variable Neighborhood Descent (VND) heuristic is developed to address the problem without the use of a Mixed Integer Programming (MIP) solver. The primary objective of the VND heuristic is to provide hospitals with a solver free from third-party dependencies and which can be implemented without any additional cost. Computational experiments are conducted on adapted instances from the Second International Nurse Rostering Competition (INRC-II) and real-world instances from a Lisbon hospital.

The present research addresses four primary research questions, namely:

- Is it necessary to include all original hard and soft constraints from the NRP when attempting to generate good quality solutions for the NRRP?
- What is the difference, in terms of computation time and solution quality, between solving the full NRRP model and a surrogate model which only considers disruption minimization objectives?
- What is the impact of the considered scheduling horizon when rerostering? Should only those days where nurses are absent be considered? The complete scheduling horizon? From the first absent day until the last absent day? Should this restricted period be extended with some days before and after?
- Is it possible to generate competitive results using a simple heuristic, compared against an integer programming formulation using a state-of-the-art commercial solver?

The remainder of the chapter is organized as follows. Section 4.2 reviews the relevant literature related to NRP and NRRP. Section 4.3 specifies the NRRP and discusses how it differs from the NRP. Section 4.4 presents the general integer programming formulation, including the NRRP and NRP constraints. Following this, Section 4.5 details the Variable Neighborhood Descent (VND) algorithm. Section 4.6 describes the computational experiments, while Section 4.7 concludes the chapter and indicates directions for future research.

4.2 Literature review

Despite the NRRP representing a common problem in hospitals, Clark et al. (2015) identified only eight relevant papers in the academic literature. The solution approaches proposed in these studies are typically based on heuristic search and integer programming. Moz and Pato (2003) were the first authors to formally define the NRRP. In addition to a multi-commodity network flow formulation, they also introduced an aggregated integer programming model which decreased the model size enabling the problem to be solved faster (MOZ; PATO, 2004). To evaluate their formulations, 16 real instances from a Lisbon hospital were utilized. Using CPLEX, 15 out of these 16 instances were solved to optimality within a time limit of two hours. More recently, Moz and Pato (2007) developed a Genetic Algorithm (GA) and performed tests on the same set of real-world instances. The GA outperformed the constructive heuristic of Moz and Pato (2003) in terms of solution quality within an acceptable time limit.

Maenhout and Vanhoucke (2011) investigated whether it is necessary to re-optimize the complete scheduling horizon or if restricting rerostering possibilities represents a feasible strategy. The best results were obtained when only a very limited fraction of the roster, typically between 10% and 30%, was re-optimized. In a follow-up study, Maenhout and Vanhoucke (2013) further explored which parts of a disrupted roster to re-optimize when rerostering. An empirical study confirmed previous results insofar as they determined that it is unnecessary to consider the complete scheduling horizon to obtain good solutions. Instead, only a period before and after the disruptions should be considered, including the period of disruptions themselves. The total number of absent nurses had little impact on the length of the rerostering period that should be considered. However, the more clustered the disruptions, the more important the days before and after the disruptions become. Regarding which resources to consider when rerostering, they concluded that it is unnecessary to consider all nurses but instead only those whose roster is disrupted along with an additional subset of nurses, specially selected for any particular reason or at random. The fewer absent nurses, the smaller are the additional number of required nurses.

Bäumelt et al. (2016) developed two parallel algorithms executed on a Graphics Processing Unit (GPU) to solve the NRRP, using the instances of Moz and Pato (2007). Two models of parallelization were compared: a homogeneous model in which the entire algorithm runs on a GPU, and a heterogeneous model where the algorithm is partially solved on a CPU and partially on a GPU. The homogeneous model resulted in solutions being generated between 12.6 and 17.7 times faster for instances with 19 and 32 nurses, respectively. By contrast, the heterogeneous model provided average speedups of 2.3 and 2.4 for the same datasets. The results demonstrated that the parallel algorithm achieves the same quality of results in significantly shorter computation time compared against the sequential algorithm.

Table 4.1 details the scope of this work compared to the existing literature. The second column classifies each variant of the NRRP according to the $\alpha|\beta|\gamma$ notation. Category α refers to the description of the personnel environment with information about the number of staff, their skills and their availabilities. Category β refers to work characteristics describing the actual services to be delivered and the time structure. Category γ refers to optimization objectives and allows the distinction between various modes of decision support (DE CAUSMAECKER; VANDEN BERGHE, 2011). In general, the problem considered in the present research generalizes previously published models by including more constraints from practice such as multi-skilled nurses. The third, fourth and fifth columns compare which rerostering strategies have been applied with respect to existing rerostering models. The comparison shows that most previous studies have not considered any specific strategy when rerostering. Only Maenhout and Vanhoucke (2011), Maenhout and Vanhoucke (2013) have investigated different relaxations of the available nurses and the scheduling horizon, the latter of which is also explored in the present research. Finally, the last two columns detail which techniques have been employed to solve the NRRP. This comparison reveals that in the last decade the focus has shifted towards heuristic methods.

Table 4.1: Existing approaches to the NRRP.

		Strategies			Solving technique	
Reference	Problem classification	Constraint relaxation	Scheduling horizon relaxation	Staffing size	Exact method	Heuristic approach
This research	*ASN/VN/PLR*	✓	✓		✓	✓
Moz and Pato (2003)	*AS/RN/PR*				✓	✓
Moz and Pato (2004)	*AS/RN/PR*				✓	
Moz and Pato (2007)	*AS/RN/PR**					✓
Pato and Moz (2008)	*AS/RN/PRM**					✓
Maenhout and Vanhoucke (2011)	*ASB/RN/PLRM**		✓	✓		✓
Maenhout and Vanhoucke (2013)	*ASB/V3/PLR**		✓	✓		✓
Bäumelt et al. (2016)	*AS/RN/PR**					✓

(*) Papers without a mathematical model, but, the problems considered in their studies are classified as described.

4.3 The nurse rerostering problem

The NRP assigns nurses to shifts during a scheduling horizon. These assignments are subject to a set of hard and soft constraints. Hard constraints must be respected, while violations of the soft constraints are penalized in the objective function. Table 4.2 shows an example of a feasible NRP solution with five nurses and a scheduling horizon of seven days. The shifts are Early (E), Late (L) and Night (N), while dashes indicate a day off. At least one nurse is required to work during each shift on each day.

Table 4.2: Example of a roster with seven days and five nurses.

Nurse	Mon	Tue	Wed	Thu	Fri	Sat	Sun
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	N	N	N	–	–	–
N4	–	–	–	–	N	N	N
N5	E	E	L	L	–	E	E

The NRRP occurs when a nurse who is scheduled to work, cannot be present due to unforeseen events. This disruption may make the solution infeasible, thereby requiring

Table 4.3: Roster with two absent nurses in gray (left table) and a reroster solution (right table).

Nurse	M	T	W	T	F	S	S
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	~~N~~	N	N	–	–	–
N4	–	–	–	–	~~N~~	N	N
N5	E	E	L	L	–	E	E

Nurse	M	T	W	T	F	S	S
N1	N	–	E	E	E	–	–
N2	L	L	–	–	L	L	L
N3	N	~~N~~	N	N	N	–	–
N4	–	N	–	–	~~N~~	N	N
N5	E	E	L	L	–	E	E

another nurse to be reallocated to cover the absence. However, this new solution must still comply with labor rules and institutional constraints as per the original rostering problem. Moreover, it should be as similar as possible to the original roster given that unpredictable schedule changes can negatively impact workers' child-care arrangements, school classes and other personal responsibilities (WILLIAMS; LAMBERT; KESAVAN, 2017).

Table 4.3 on the left side shows a disruption in which two absent nurses are highlighted in gray. Considering the minimum of one nurse per day/shift, these two absences render the solution infeasible, as now nobody is working the Night shifts on Tuesday and Friday. Table 4.3 on the right side presents one possible new solution after rerostering, wherein Nurse 4, who before had a free day on Tuesday, is now working a Night shift. Nurse 3, who had a free day on Friday, is now also working the Night shift.

In this trivial example, the solution's feasibility may be restored by simply swapping two nurses. This operation has minimal impact on the existing solutions and maintains the same number of working shifts as in the original roster. Generally, however, the situation is more complex as various time-related constraints impose additional restrictions on the solution, such as minimum/maximum number of consecutive days worked or minimum rest time between two consecutive working days. Moreover, if they are modeled as soft constraints, their violations should be minimized.

4.4 General integer programming formulation for the NRRP

This section presents a general integer programming formulation for the NRRP. Moz and Pato (2003), Moz and Pato (2004) were, so far, the only authors to address the nurse rerostering problem using integer programming. Moz and Pato (2003) formulated the problem as an integer multi-commodity flow problem with side constraints in a multi-level acyclical network. This formulation was further improved in Moz and Pato (2004) by including node aggregation in the network. In contrast to these flow models, the formulation proposed in the present research is based on an assignment problem, thereby enabling more general problem characteristics to be included such as multi-skilled nurses and various hard and soft constraints typically found in hospitals (CESCHIA et al., 2019). Appendix C details the full integer programming formulation containing all original NRP constraints. In what follows, the presentation of the integer programming formulation is restricted to constraints and objectives relevant to the rerostering problem. Table 4.4 presents the problem's parameters in addition to the main and auxiliary decision variables employed in the NRRP formulation.

Table 4.4: Sets and variables employed in the formulation.

Symbol	Definition
Parameters	
$n \in N$	*n* is the index of the nurse, where *N* is the set of all nurse indices;
$\hat{N} \subseteq N$	set of absent nurses indices;
$d \in D$	*d* is the index of the day, where *D* is the set of all day indices;
$s \in S$	*s* is the index of the shift, where *S* is the set of all shift indices;
$k \in K$	*k* is the index of the skill, where *K* is the set of all skill indices;
ω^i	weight for violating the lower and upper limits of soft constraint *i*;
δ_n	original number of assignments associated with nurse *n*;
$c_{ndsk} \in \{0,1\}$	1 if nurse *n* is allocated to shift *s* and day *d* with skill *k* in the original roster, and 0 otherwise;
$\hat{c}_{nd} \in \{0,1\}$	parameter modeling the disruptions which is 1 if nurse *n* is absent on day *d*, and 0 otherwise;
Decision Variables	
$x_{ndsk} \in \{0,1\}$	1 if nurse *n* is allocated to shift *s* and day *d* with skill *k*, and 0 otherwise;
Auxiliary Variables	
$y'_{nds} \in \{0,1\}$	1 if nurse *n* works in the original schedule or in the new roster on day *d* and shift *s*, and 0 otherwise;
$y''_{nds} \in \{0,1\}$	auxiliary variable to calculate the number of changes compared to the original roster;
$v^{13}_{nd} \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of changes compared to the original roster;
$\hat{v}^{14}_n \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of working days less than the original roster;
$\hat{v}^{15}_n \in \mathbb{N}_0$	auxiliary variable to calculate the violations of the number of working days more than the original roster

$$\textbf{Minimize:} \quad \sum_{n\in N}\sum_{d\in D} v^{13}_{nd}\omega^{13} + \sum_{n\in N}\sum_{i\in\{14,15\}} \hat{v}^i_n\omega^i + C.1 \tag{4.1}$$

Subject to:

$$\hat{c}_{nd} + \sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 1 \qquad \forall n \in N, d \in D \tag{4.2}$$

$$\sum_{k\in K}(c_{ndsk} + x_{ndsk}) \leq 2y'_{nds} \qquad \forall n \in N \setminus \hat{N}, d \in D, s \in S \tag{4.3}$$

$$\sum_{k\in K}(c_{ndsk}+x_{ndsk})+y''_{nds}\geq 2y'_{nds} \qquad \forall n\in N\setminus\hat{N}, d\in D, s\in S \qquad (4.4)$$

$$\sum_{s\in S}y''_{nds}-2v^{13}_{nd}\leq 0 \qquad \forall n\in N\setminus\hat{N}, d\in D \qquad (4.5)$$

$$\sum_{d\in D}\sum_{s\in S}\sum_{k\in K}x_{ndsk}+\hat{v}^{14}_{n}\geq \delta_n \qquad \forall n\in N \qquad (4.6)$$

$$\sum_{d\in D}\sum_{s\in S}\sum_{k\in K}x_{ndsk}-\hat{v}^{15}_{n}\leq \delta_n \qquad \forall n\in N \qquad (4.7)$$

Objective function (4.1) minimizes a weighted sum of different terms related to the NRRP and NRP. The first term penalizes changes made in the rerostering solution with respect to the original roster. The second term minimizes the difference in number of working days in the roster before and after rerostering. The last part of the objective function consists of the soft constraint violations of the NRP as defined in Equation (C.1) (Appendix C). Constraints (4.2) ensure that an absent nurse is not scheduled to work. Constraints (4.3), (4.4) and (4.5) determine the number of changes in the new roster compared to the original roster. Constraints (4.6) and (4.7) calculate the change in number of working days.

An example of how Constraints (4.3), (4.4) and (4.5) count the number of changes for a single nurse is provided in Tables 4.5 and 4.6. The formulation penalizes whenever a working day is changed to a day off (or vice versa) and whenever a shift which was assigned is modified. Changes regarding assigned skills are not penalized, as these cases are not considered to have a significant impact on the nurses. The example in Table 4.5 considers three days and three shifts: Early (E), Late (L) and Night (N). The first row of integer values represents the current solution. In this case, the nurse works a Late shift on the first day, the second day is free, and a Night shift on the third day. The second row represents the solution after rerostering. The nurse now works a Night shift on the first day, an Early shift on the second day and another Night shift on the third day. The third row is the sum of the current solution and the newly rerostered solution. Finally, the fourth row shows the values variables y'_{nds} assume. In this example there are two changes, one is a shift change from Late to Night on the first day, and the second change concerns the second day where the nurse previously had a day off, but for which she is now scheduled to work an Early shift. There are no changes on the third day as the nurse continues working on the already-scheduled Night shift. Table 4.6 presents the penalization results stored in variable v^{13}_{nd} (last column of the right table). On the first and second day, the variable assumes a value of 1, meaning that there is one change (one violation), and on the third day the variable assumes a value of 0, denoting zero violations.

Table 4.5: Example of a reroster solution with two changes.

Day 1 E	Day 1 L	Day 1 N	Day 2 E	Day 2 L	Day 2 N	Day 3 E	Day 3 L	Day 3 N	Description
0	1	0	0	0	0	0	0	1	$\sum_{k\in K} c_{ndsk}$ (current solution)
0	0	1	1	0	0	0	0	1	$\sum_{k\in K} x_{ndsk}$ (newly rerostered solution)
0	1	1	1	0	0	0	0	2	$\sum_{k\in K}(c_{ndsk}+x_{ndsk})$
0	1	1	1	0	0	0	0	1	$\sum_{k\in K}(c_{ndsk}+x_{ndsk}) \leq 2y'_{nds}$ (y'_{nds} values)

Table 4.6: Example of how Constraints (4.4) and (4.5) are evaluated.

$\sum_{k\in K}(c_{ndsk}+x_{ndsk})+y''_{nds} \geq 2y'_{nds}$			
Day 1	E	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$
	L	$1+0+y''_{nds} \geq 2$	$y''_{nds}=1$
	N	$0+1+y''_{nds} \geq 2$	$y''_{nds}=1$
Day 2	E	$0+1+y''_{nds} \geq 2$	$y''_{nds}=1$
	L	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$
	N	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$
Day 3	E	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$
	L	$0+0+y''_{nds} \geq 0$	$y''_{nds}=0$
	N	$1+1+y''_{nds} \geq 2$	$y''_{nds}=0$

$\sum_{s\in S} y''_{nds} - 2v^{13}_{nd} \leq 0$	
2 - $2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=1$
1 - $2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=1$
0 - $2v^{13}_{nd} \leq 0$	$v^{13}_{nd}=0$

4.5 Variable neighborhood descent

This section presents a Variable Neighborhood Descent (VND) heuristic specifically designed to address the NRRP based on the general concepts first introduced by Mladenović and Hansen (1997). The VND heuristic is selected primarily due to its simplicity, its ability to integrate several neighborhood structures, and the successful application of this algorithm and its variants for the NRP (BURKE et al., 2008; ZHENG; LIU; GONG, 2017; GOMES; TOFFOLO; SANTOS, 2017). The proposed method seeks to generate results quickly and without the dependence on third-party software packages.

4.5.1 Main method - VND

Algorithm 3 outlines the main method which takes as input parameters the current solution, the maximum number of top-level loop iterations and the number of iterations after which an intensification/diversification procedure is called. Function *OFV(cs)* returns the objective function value (OFV) of the solution *cs*. Note that it is only when the objective value is calculated for the first time that all nurses, days, shifts and skills are considered. When neighboring solutions are evaluated, a relative recalculation of the objective value is executed in order to reduce the computation time.

In each iteration of the top-level loop (lines 5-17), a sequence of procedures which explore different neighborhoods is executed until either no improvement is found or a feasible solution is reached. All neighborhoods, except for *changeShift*, *assignMissingShift-DeleteNext*, *intensDiverLS* , consider only the days on which absences occur. However, it is not always possible to remedy infeasibilities by exploring only these restricted neighborhoods. For this reason, *changeShift*, *assignMissingShiftDeleteNext* and *intensDiverLS* are also included in the VND heuristic as they explore a larger proportion of the search space, thereby increasing the likelihood that infeasibilities will be solved should the deterministic neighborhoods fail in doing so, albeit at the expense of longer computational runtimes. The algorithm terminates by returning the best solution found. Input parameters are passed to each subroutine, however, in the main pseudo-code they are omitted for presentation reasons.

```
   Input   : cs current solution, maxTrials maximum number of iterations, maxTrialsIntDiv number of iterations
             after which the intensification/diversification procedure is called
   Output  : solution
1  cs ← assignMissingShift
2  cs ← changeAssignMissingShift
3  bestOFV ← ∞
4  iterations ← 0
5  while bestOFV > OFV(cs) or (hasHardViolation(cs) and iterations < maxTrials) do
6      bestOFV ← OFV(cs)             // returns the OFV from the current solution cs
7      cs ← assignDeleteShift
8      cs ← changeShift
9      cs ← swapShift
10     cs ← assignMissingShift
11     cs ← changeAssignMissingShift
12     cs ← assignMissingShiftDeleteNext
13     if hasHardViolation(cs) and iterations > maxTrialsIntDiv then
14         cs ← intensDiverLS
15     end
16     iterations ← iterations + 1
17 end
18 return cs                                    // returns the best solution found
```

Algorithm 3: Variable Neighborhood Descent (VND).

4.5.2 Assign and delete shift neighborhood

Algorithm 4 moves an assigned shift from one nurse to another. Lines 4-18 iterate over each day with insufficient coverage. Line 7 iterates over each working nurse w and each idle nurse f on day d. Line 8 generates a neighboring solution in which the shift and skill assignments of nurse w are reassigned to nurse f. If the neighboring solution does not violate any hard constraints and is the best neighbor found (line 9), the variables are updated (lines 10-11). The neighborhood's size is $O(|D_v||W_d||F_d|)$. The procedure terminates by returning the best solution found.

```
     Input   : $D_v$ set of days with insufficient coverage, $W_d$ set nurses working on day $d$, $F_d$ set of idle nurses on day
               $d$, $cs$ current solution
     Output  : Solution
 1   improved ← true
 2   while improved do
 3       improved ← false
 4       foreach $d \in D_v$ do
 5           bestNeighbor ← null
 6           bestNeighborOFV ← OFV(cs)
 7           foreach $w \in W_d, f \in F_d$ do
 8               $cs'$ ← assignDelete(cs, d, w, f)          // returns null if infeasible
 9               if $cs' \neq null$ and $OFV(cs') < bestNeighborOFV$ then
10                   bestNeighbor ← $cs'$
11                   bestNeighborOFV ← OFV($cs'$)
12               end
13           end
14           if $bestNeighbor \neq null$ then               // if an improved neighbor is found
15               cs ← bestNeighbor
16               improved ← true
17           end
18       end
19   end
20   return cs
```

Algorithm 4: Assign and delete shift.

4.5.3 Change shift neighborhood

Algorithm 5 changes nurses' assignments on consecutive days. The method *assignShift* (line 8) generates a neighboring solution in which shift s is assigned to nurse n on day $d+d'$. If there is a feasible solution and the objective value of the neighboring solution is lower than that of the current solution, the variables are updated accordingly (lines 9-12). If shift s is a working shift (line 14), the nurses' skills are iterated over (loop 15-22). Method *assignSkill* (line 16) changes the assigned skill of nurse n on day $d+d'$ in shift s to k. If there is a feasible solution and the new objective value is lower than the current objective value (line 17), the current solution and variables are updated (lines 18-20). The neighborhood's size is $O(|D||S||w||K_n|)$. The goal of the variable d' is to change the assignments in a sequence of days, thereby aiming to reduce the number of violations of constraints concerning the minimum/maximum number of consecutive working days and similar constraints. Preliminary experiments demonstrate that the most suitable value of the parameter w is four. The procedure terminates by returning the best solution found.

```
Input  : $D$ set of all days, $N$ set of nurses, $S$ set of shifts, $S'$ set of working shifts,
         $K_n$ set of skills of nurse $n$, $cs$ current solution, $w$ maximum consecutive days window size
Output : Solution
improvedLS ← true
while improvedLS do
    improvedLS ← false
    foreach $d \in D, n \in N, s \in S, d' \leftarrow 1$ to $w$ do
        improved ← false
        csBackup ← cs
        if $((d + d') < |D|)$ then
            $cs' \leftarrow$ assignShift(cs, n, $(d + d')$, s)        // returns null if infeasible
            if $cs' \neq null$ and $OFV(cs') < OFV(cs)$ then
                improved ← true
                improvedLS ← true
            end
            cs ← $cs'$
            if $s \in S'$ then
                foreach $k \in K_n$ do
                    $cs' \leftarrow$ assignSkill(cs, n, $(d + d')$, s, k)
                    if $cs' \neq null$ and $OFV(cs') < OFV(cs)$ then
                        cs ← $cs'$
                        improved ← true
                        improvedLS ← true
                    end
                end
            end
        end
        if not improved then
            cs ← csBackup
        end
    end
end
return cs
```

Algorithm 5: Change shift.

4.5.4 Swap shift neighborhood

Algorithm 6 swaps the shift and skill assignments of two nurses. The loops (lines 7-8) iterate over each pair of nurses *n1* and *n2*. Line 9 generates a neighboring solution by swapping nurse *n1*'s shift and skill with nurse *n2*'s, and vice versa. If the new solution does not violate any hard constraints and is the best neighbor found (line 10), the variables are updated accordingly (lines 11-12). The neighborhood's size is $O(|D_v||N|^2)$. The procedure terminates by returning the best solution found.

4.5.5 Assign missing shift neighborhood

Algorithm 7 assigns the shifts and skills associated with absences which have occurred to idle nurses in a greedy manner such that the largest decrease in objective value is obtained. Line 4 iterates over the days and shifts for which the number of nurses is below the minimum. For each idle nurse n (lines 7-13), a neighboring solution is generated by assigning the missing shift s and skill k on day d (line 8). If the neighboring

```
    Input   : $D_v$ set of days with insufficient coverage, $N$ set of nurses, $cs$ current solution
    Output  : Solution
 1  improved ← true
 2  while improved do
 3      improved ← false
 4      foreach $d \in D_v$ do
 5          bestNeighbor ← null
 6          bestNeighborOFV ← OFV(cs)
 7          foreach n1 ← 1 to |N|-1 do
 8              foreach n2 ← n1+1 to |N| do
 9                  $cs'$ ← swapAssignments(cs, d, n1, n2)        // returns null if infeasible
10                  if $cs' \neq null$ and OFV($cs'$) < bestNeighborOFV then
11                      bestNeighbor ← $cs'$
12                      bestNeighborOFV ← OFV($cs'$)
13                  end
14              end
15          end
16          if $bestNeighbor \neq null$ then
17              cs ← bestNeighbor
18              improved ← true
19          end
20      end
21  end
22  return cs
```

Algorithm 6: Swap shift.

solution does not violate any hard constraint and is the best neighbor found (line 9), the variables are updated (lines 10-11). The neighborhood's size is $O(|D_v||S_d||K_d||F_d|)$. The procedure terminates by returning the best solution found.

```
    Input   : $D_v$ set of days with insufficient coverage ordered by most violated days,
              $S_d$ set of shifts with insufficient coverage on day d, $K_d$ set of skills with insufficient coverage on day d,
    $F_d$ set of idle nurses on day d, $cs$ current solution
    Output  : Solution
 1  improved ← true
 2  while improved do
 3      improved ← false
 4      foreach $d \in D_v, s \in S_d, k \in K_d$ do
 5          bestNeighbor ← null
 6          bestNeighborOFV ← OFV(cs)
 7          foreach $n \in F_d$ do
 8              $cs'$ ← assignShiftSkill(cs, n, d, s, k)        // returns null if infeasible
 9              if $cs' \neq null$ and OFV($cs'$) < bestNeighborOFV then
10                  bestNeighbor ← $cs'$
11                  bestNeighborOFV ← OFV($cs'$)
12              end
13          end
14          if $bestNeighbor \neq null$ then        // if an improved neighbor is found
15              cs ← bestNeighbor
16              improved ← true
17          end
18      end
19  end
20  return cs
```

Algorithm 7: Assign missing shift.

4.5.6 Change and assign missing shift neighborhood

Algorithm 8 first moves a working shift to an idle nurse and then assigns the missing shift and skill. Line 4 iterates over the days where the number of nurses is below the minimum. For each working nurse w and each idle nurse f (lines 7-18), the currently assigned shift and skill are saved (lines 8-9) and the missing shift and skill are assigned to nurse w (line 10). If the resulting solution does not violate any hard constraints, the algorithm assigns an idle nurse f the shift and skill previously assigned to nurse w (line 12). If the resulting solution does not violate any hard constraints and is the best neighbor found (line 14), the variables are updated (lines 15-16). The neighborhood's size is $O(|D_v||S_d||K_d||W_d||F_d|)$. The procedure terminates by returning the best solution found.

```
Input   : $D_v$ set of days with insufficient coverage ordered by most violated days,
          $S_d$ set of shifts with insufficient coverage on day d, $K_d$ set of skills with insufficient coverage on day d,
          $F_d$ set of idle nurses on day d, $W_d$ set of working nurses on day d, cs current solution
Output  : Solution
1  improved ← true
2  while improved do
3      improved ← false
4      foreach $d \in D_v, s \in S_d, k \in K_d$ do
5          bestNeighbor ← null
6          bestNeighborOFV ← OFV(cs)
7          foreach $w \in W_d, f \in F_d$ do
8              backupShift ← getShift(cs, w, d)            // backup current shift
9              backupSkill ← getSkill(cs, w, d, s)         // backup current skill
10             $cs'$ ← assignShiftSkill(cs, w, d, s, k)      // returns null if infeasible
11             if $cs' \neq null$ then
12                 $cs''$ ← assignShiftSkill($cs'$, f, d, backupShift, backupSkill)
13             end
14             if $cs'' \neq null$ and $OFV(cs'') < bestNeighborOFV$ then
15                 bestNeighbor ← $cs''$
16                 bestNeighborOFV ← OFV($cs''$)
17             end
18         end
19         if $bestNeighbor \neq null$ then              // if an improved neighbor is found
20             cs ← bestNeighbor
21             improved ← true
22         end
23     end
24 end
25 return cs
```

Algorithm 8: Change and assign missing shift.

4.5.7 Assign missing shift and delete next shift neighborhood

Algorithm 9 attempts to fix disruptions by inserting shifts on the days with an insufficient number of nurses. However, when inserting shifts results in an infeasible solution, the assignment on the following day is deleted. The function $\mathscr{U}(LB,\ldots,UB)$ returns a value between LB and UB sampled from a uniform distribution. Line 1 iterates over the days with insufficient coverage. Lines 2 and 3 randomly select an idle nurse on day d and assign the missing shift and skill. If the resulting solution cs' is infeasible and d is not the last day of the scheduling horizon (line 4), the algorithm removes the assignment on the next day. If the resulting solution is feasible, it replaces the current solution (lines 7-9). The neighborhood's size is $O(|D_v||S_d||K_d|)$. The procedure terminates by returning the best solution found.

```
Input  : $D_v$ set of days with insufficient coverage ordered by most violated days,
         $D_a$ set of all days, $S_d$ set of shifts with insufficient coverage on day d, $K_d$ set of skills with insufficient
coverage on day d, $F_d$ set of idle nurses on day $d$
Output : Solution
1  foreach $d \in D_v, s \in S_d, k \in K_d$ do
2      n ← $\mathscr{U}(1,\ldots,|F_d|)$                // returns a random nurse not working on day d
3      $cs'$ ← assignShiftSkill(cs, n, d, s, k)                // returns null if infeasible
4      if $cs'$ is null and $d+1 \leq |D_a|$ then
5          $cs'$ ← assignDayOff($cs'$, n, (d+1))
6      end
7      if $cs' \neq null$ then                          // if an improved neighbor is found
8          cs ← $cs'$
9      end
10 end
11 return cs
```

Algorithm 9: Assign missing shift delete next.

4.5.8 Intensification and diversification neighborhood

Algorithm 10 details an intensification and diversification procedure for the NRRP. The objective here is to explore larger neighborhoods, thereby increasing the probability of finding a feasible solution when the other neighborhoods fail. Preliminary experiments on a subset of the instances revealed that the most suitable values for the input parameters *maxNoImprov*, *maxChanges*, *maxNoImprovDiv*, *maxChangesDiv* are 100, 3, 50 and 2, respectively. The algorithm is terminated after the maximum number of iterations without improvement is reached or if a feasible solution is found (line 3). The loop spanning lines 6-22 determines how many neighbors are generated in each iteration of the procedure's intensification phase. In this case, this value equals the number of nurses $|N|$. If the number of iterations without improvement is greater than a specific threshold, the diversification phase begins (line 30). In this phase, for each day, two random modifications are generated in the current solution (lines 32-38). After this diversification procedure, the local search operators *assignMissingShift, changeAssignMissingShift, assignDeleteShift, changeShift, swapShift* are called to further improve the solution (lines 39-40). The procedure terminates by returning the best solution found.

```
Input   : $D_v$ set of days with absent nurses, $N$ set of nurses, $S$ set of shifts, *bestSolution* current solution,
          *maxNoImprov* maximum number of iterations without improvement, *maxChanges* maximum number
          of changes, *maxNoImprovDiv* maximum number of iterations to start the diversification phase,
          *maxChangesDiv* maximum number of changes in the diversification phase
Output  : Solution
countNoImprov ← 0
cs ← bestSolution
while countNoImprov < maxNoImprov and hasHardViolation(cs) do
    bestNeighbor ← $\infty$
    $cs''$ ← null
    for $x \leftarrow 1$ to $|N|$ do
        randNumChange ← $\mathscr{U}(1,\dots,maxChanges)$     // returns a random number of changes
        for $z \leftarrow 1$ to randNumChanges do
            randNurse ← $\mathscr{U}(1,\dots,|N|)$     // returns a random nurse
            randDay ← $\mathscr{U}(1,\dots,|D|)$     // returns a random day
            randShift ← $\mathscr{U}(1,\dots,|S|)$     // returns a random shift
            $cs'$ ← assignShift(cs, randNurse, randDay, randShift)   // returns null if infeasible
            if $cs' \neq null$ and isNotTabu(randNurse, randDay, randShift) then
                addTabu(randNurse, randDay, randShift)     // add nurse,day,shift to tabu
                  list
                if OFV($cs'$) < bestNeighbor then
                    bestNeighbor ← OFV($cs'$)
                    $cs''$ ← $cs'$
                    break
                end
            end
        end
    end
    if $cs'' \neq null$ and OFV($cs''$) < OFV(cs) then
        countNoImprov ← 0
        cs ← $cs''$
        bestSolution ← cs
    end
    else
        countNoImprov ← countNoImprov + 1
        if countNoImprov > maxNoImprovDiv then
            cs ← bestSolution
            foreach $d \in D$ do
                foreach $y \leftarrow 1$ to maxChangesDiv do
                    randNurse ← $\mathscr{U}(1,\dots,|N|)$     // returns a random nurse
                    randShift ← $\mathscr{U}(1,\dots,|S|)$     // returns a random day
                    cs ← assignShift(cs, randNurse, d, randShift)
                end
            end
            cs ← assignMissingShift(cs);   cs ← changeAssignMissingShift(cs)
            cs ← assignDeleteShift(cs);   cs ← changeShift(cs);   cs ← swapShift(cs)
        end
    end
end
return bestSolution
```

Algorithm 10: Intensification and diversification procedure.

4.6 Computational results

This section analyzes a series of computational experiments to investigate whether the proposed IP formulation can be solved using a MIP solver for both small and large instances with multi-skilled nurses. The impact of relaxing soft constraints with respect to the original NRP in terms of solution quality and computation time is analyzed. Moreover, whether the degradation of solution quality is significant when rerostering a limited scheduling horizon and whether or not the proposed VND heuristic can generate competitive results compared to a MIP solver is also investigated.

4.6.1 Data sets and experimental setup

This section presents two sets of instances employed for the experiments. They cover both academic and realistic scenarios. This research contributes a rerostering version of the INRC-II instances. The Lisbon instances were, until this work, the only public set of instances available in the literature related to the NRRP. They were developed by Moz and Pato (2007) and based on real data provided by a Lisbon hospital. The INRC-II instances incorporate a large number of soft constraints related to the NRP and are therefore less restrictive in terms of hard constraints compared to the Lisbon instances.

4.6.1.1 INRC-II instances

Table 4.7 describes the constraints of the INRC-II for which there are two main sets. The first concerns those related to the NRP that have less weight in the objective function when rerostering and, consequently, less importance associated with avoiding their violations. The second set of constraints concerns the specific objectives related to the NRRP. The number of changes, which is considered the most important objective to minimize by the objective function is assigned a weight of 100. Meanwhile, the change in number of assigned shifts is regarded as less important and receives a weight of 50. An in-depth discussion of each constraint is provided by Ceschia et al. (2019).

Ingels and Maenhout (2015) simulate employee availability using a Bernoulli distribution. They performed simulations for short-term sick leave with a probability of 2.44% based on a study conducted by SD Worx (2013) in Belgium. Moreover, they reported that simulations using an absenteeism probability of 5% and 10% resulted in

Table 4.7: Hard and soft constraints in the INRC-II instances.

Index	Constraint description	Weight	Eq.
Nurse Rostering Constraints			
-	A nurse can be assigned to at most one shift per day	HC	C.2
-	Minimum number of nurses per day/shift/skill	HC	C.3
-	A shift type succession must belong to a valid succession (for example, a Night shift cannot be followed by an Early shift)	HC	C.4
-	A shift requiring nurses with a given skill must necessarily be fulfilled by a nurse having that skill	HC	C.5
1	Preferred coverage	30	C.7
2	Minimum consecutive assignments (working days)	30	C.8, C.9
3	Maximum consecutive assignments (working days)	30	C.15
4	Minimum number of consecutive days off	30	C.11, C.12
5	Maximum number of consecutive days off	30	C.13
6	Minimum consecutive assignments to the same shift	15	C.14, C.15
7	Maximum consecutive assignments to the same shift	15	C.16
8	Individual nurse's undesired working day/shift	10	C.17
9	Complete weekend	30	C.18, C.19
10	Minimum number of assignments over the scheduling period	20	C.20
11	Maximum number of assignments over the scheduling period	20	C.21
12	Total working weekends	30	C.18, C.22
Nurse Rerostering Constraints			
-	Absent nurses cannot be assigned to any shift	HC	4.2
13	Each change in the new roster is penalized	100	4.3, 4.4, 4.5
14, 15	The original number of assigned shifts should be maintained	50	4.6, 4.7

similar results, regarding sick leave probabilities.

In this study, absences were randomly generated based on statistics observed by Aguirre and Kerin (2014) in the U.S. They report absenteeism rates ranging from 5% to 10% among all employees, meaning that at any given time 5% to 10% of the workforce is missing from work. While this rate varies by sector, the emergency services and health-care, both known for their stressful working conditions, high rates of overtime, and are therefore unsurprisingly associated with the highest rates of absenteeism.

The first group of NRRP instances is named *Single-day Nurse Absence*, where absences are generated for randomly selected nurses and days based on an absenteeism rate of 5%. The instances have 35, 70 or 110 nurses and a scheduling horizon of either four or eight weeks.

The second group of instances is named *Consecutive-days Nurse Absence* and represents situations which simulate nurse illness. In these instances, a randomly selected nurse is absent for a sequence of days beginning from a first random day i until a later random day j which are chosen based on a uniform distribution. Instances with 35, 70 or 110 nurses and scheduling horizons of four or eight weeks have 5% of their associated nurses absent for a random number of consecutive days.

4.6.1.2 Lisbon instances

The Lisbon instances differ from the INRC-II in terms of constraints, shifts and nurses' skills. Rather than four shifts as per the INRC-II instances, the Lisbon instances have three shifts, namely: Early (08:00-16:00), Late (16:00-24:00) and Night (00:00-08:00). The nurses are single-skilled, while in the INRC-II instances they are multi-skilled. All constraints are hard and the objective simply concerns minimizing the number of changes compared to the original roster. The general model proposed in Section 4.4 and Appendix C details this subset of constraints. Table 4.8 presents the set of constraints and the related equations.

Table 4.8: Hard and soft constraints considered for the Lisbon instances.

Index	Constraint Description	Weight	Eq.
Nurse Rostering Constraints			
-	A nurse can be assigned to at most one shift per day	HC	C.2
-	Minimum number of nurses per day/shift/skill	HC	C.3
-	A shift type succession must belong to a valid succession	HC	C.4
-	Every seven days sequence, nurses must have 1 day off when the contract is 42 hours per week, and 2 days off when the contract is 35 hours per week	HC	C.6
Nurse Rerostering Constraints			
-	Absent nurses cannot be assigned any shift	HC	4.2
13	Each change in the new roster is penalized	1	4.3, 4.4, 4.5

Inconsistencies regarding these Lisbon instances required adaptations as the current roster, provided by the hospital's head nurse, violated several hard constraints. The following changes were made to render the instances feasible:

- Nurses with 30-hour contracts are now considered as having 35-hour contracts. In doing so, these nurses must have two days off per working week;
- The current solution for 32 nurses is infeasible due to some violations where the nurses' contracts permit them to work a maximum of 5 days every 7 days. However, some nurses work 6 days in the provided rosters. In this case, the nurses' contracts were adjusted to 42 hours, thereby permitting them to work 6 days every 7 days;
- The pattern file, provided in PDF format, was ignored during the conversion process since the patterns were not always respected in the roster provided by the hospital's head nurse.

4.6.1.3 Computational environment

All models and algorithms were implemented in Java and compiled with OpenJDK 1.8. The experiments were conducted on an AMD FX 8150 eight-core processor with 32 GB of RAM memory running Linux Ubuntu 16.04.3 64-bit. The commercial MIP solver employed was CPLEX version 12.7.1 with default parameters and configured to use eight threads. For the experiments with an open-source solver, Coin-OR CBC 2.9.9 was employed with eight threads. Relative gaps in solution quality were calculated as $gap = 100 \times \frac{UB-OPT}{OPT}$, where UB (upper bound) corresponds to the objective function value of the VND heuristic, and OPT corresponds to the optimum solution value obtained by CPLEX. For each experiment, the VND heuristic was executed ten times with different seed values for the random number generator.

4.6.1.4 VND neighborhoods and parameter tunning

The primary objective of the experiments in this section is to analyze to which degree each VND neighborhood impacts upon the heuristic's performance. Table 4.9 compares the results obtained by the MIP solver and different configurations of the VND heuristic. The complete scheduling horizon and all NRP and NRRP constraints were considered for these experiments. The following five VND heuristic configurations were investigated:

- *VND 1:* employs only the *assignMissingShift* neighborhood;
- *VND 2:* employs the *assignMissingShift* and *assignDeleteShift* neighborhoods;
- *VND 3:* employs the *assignMissingShift*, *assignDeleteShift* and *changeShift* neighborhoods;
- *VND 4:* employs all neighborhoods, except *intensDiverLS*;
- *VND:* employs all neighborhoods.

The second through fourth columns in Table 4.9 provide the objective values obtained by the MIP solver, the respective time to the optimum solution, and the time to prove optimality. The fifth through fourteenth columns show, for each configuration of the VND heuristic, the gap to the optimum objective value and the required computation time.

The average relative optimality gaps for *VND1*, *VND2*, *VND3*, *VND4* and *VND* are 1.10%, 1.07%, 0.62%, 0.61% and 0.61%, respectively. The VND heuristic generated

Table 4.9: VND neighborhoods evaluation - Complete scheduling horizon, NRP and NRRP constraints

	Single-day absences												
	MIP solver			VND 1		VND 2		VND 3		VND 4		VND	
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
n035w4	3206.0	3.7	5.7	0.8	0.7	0.8	0.9	0.7	1.0	0.6	1.0	0.6	1.0
n035w8	6490.5	8.5	14.0	1.1	0.9	1.1	1.3	1.0	1.8	1.0	1.8	1.0	1.8
n070w4	5850.0	5.8	16.6	0.7	1.0	0.7	1.4	0.0	2.4	0.0	2.8	0.0	2.8
n070w8	12379.0	51.5	186.3	1.5	1.2	1.4	2.2	0.4	4.4	0.4	6.6	0.4	6.6
n110w4	7495.0	12.1	42.1	0.9	1.1	0.8	2.1	0.7	3.0	0.7	3.9	0.7	3.9
n110w8	14366.0	78.7	237.5	1.5	1.5	1.5	2.9	1.0	5.8	1.0	10.2	1.0	10.2
average				1.10		1.07		0.62		0.61		0.61	
	Consecutive-days absences												
	MIP solver			VND 1		VND 2		VND 3		VND 4		VND	
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
n035w4	3167.0	5.3	9.6	2.4	0.8	1.7	1.0	1.4	1.1	1.3	1.2	1.3	1.2
n035w8	6383.5	7.5	28.6	2.7	0.9	2.2	1.3	2.1	1.8	1.9	1.9	1.9	1.9
n070w4	5564.5	13.8	41.9	0.8	1.0	0.7	1.5	0.4	2.4	0.3	2.5	0.3	2.5
n070w8	11595.0	109.2	208.0	1.3	1.2	1.3	2.0	1.0	4.3	0.9	6.2	0.9	6.2
n110w4	7039.0	28.3	84.5	1.2	1.2	1.2	2.1	0.5	4.1	0.5	5.0	0.5	5.0
n110w8	13670.5	229.6	598.9	2.5	1.6	2.5	2.9	1.8	6.7	1.9	9.7	1.9	9.7
average				1.82		1.60		1.20		1.11		1.11	

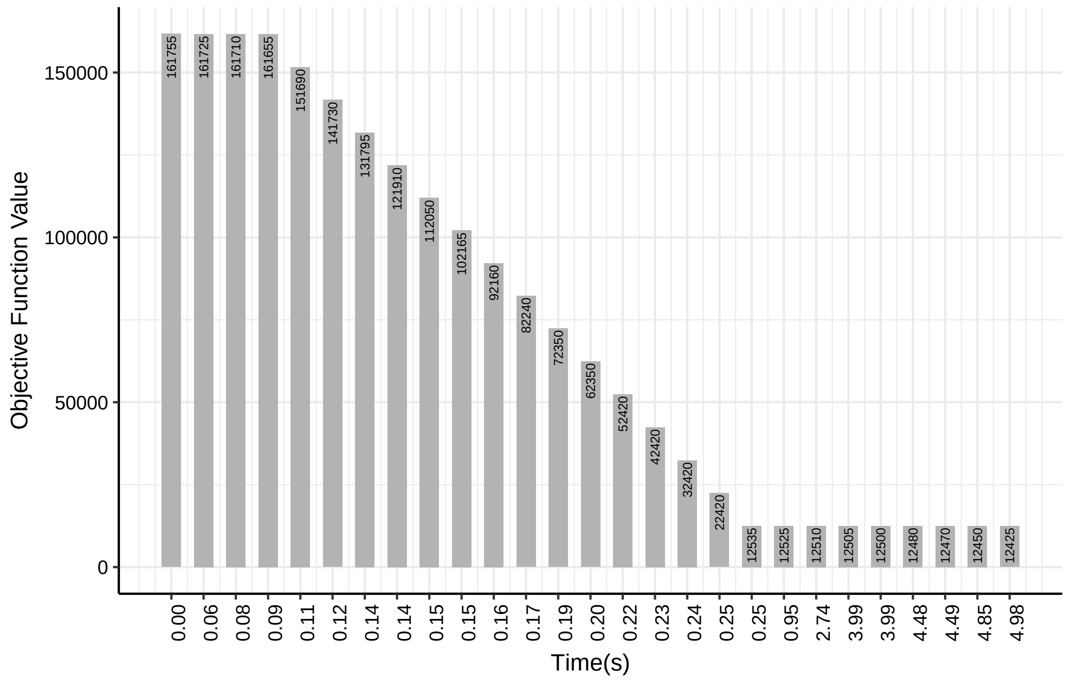
near-optimum solutions within only a few seconds. It is worth noting that due to the numerous dynamic situations which may occur in the real world, fast management decisions and therefore short computation times are critical to ensure the best decision is made as quickly as possible, considering both the institution and employee preferences.

The second part of Table 4.9 reports the results for the instances with consecutive-days absences. The average relative optimality gaps are 1.82%, 1.60%, 1.20%, 1.11% and 1.11%, respectively for the five VND heuristic variants. Again, the computation times were much lower compared to those required by the MIP solver for reaching its near-optimum solutions. For example, on the largest instances, the MIP solver spent 229.6 seconds to reach the optimum value, while the VND heuristic required only 9.7 seconds to reach a solution within 1.11% of the optimum solution. Experiments with different versions of the VND heuristic demonstrate that all the implemented components are important in contributing to obtaining near-optimum solutions or to solve infeasibilities. Moreover, when all neighborhoods are used, the algorithm generated the best results. In all remaining experiments, the VND heuristic is the one employed with all neighborhoods.

The VND heuristic has two main parameters. The first parameter determines after how many iterations the intensification and diversification procedure is called and was set to 30 in order to limit the algorithm's runtime while still providing sufficient possibilities to solve infeasibilities in the Lisbon instances. For the INRC-II instances, the intensification and diversification procedure was not required to solve infeasibilities. However, for the Lisbon instances this neighborhood was essential and solved infeasibilities in 10 out of 64 instances. The second main parameter is the maximum number of top-level loop iterations which was set to 100 to avoid infinite loops in the algorithm whenever an instance did not have a feasible solution. Table D.1 (in Appendix D) details the number of top-level loop iterations for different instances and strategies. Considering the complete scheduling horizon and all constraints, the average number of trials was less than two for the INRC-II instances, 12.9 for the Lisbon instances with 19 nurses and 4.8 for the Lisbon instances with 32 nurses.

Figure 4.1 presents the evolution of the objective function value throughout the VND heuristic's execution on an INRC-II instance with 110 nurses and a scheduling horizon of eight weeks. The algorithm begins with an initial objective value of 161755 and after 4.98 seconds ends with an objective function value of 12425. After some initial small improvements, the algorithm quickly finds several significant improvements. The algorithm then ends like it began: with a series of minimal improvements. This experiment demonstrates how the algorithm fulfills its two primary objectives: to find high-quality solutions within short computational runtimes.

Figure 4.1: OFV evaluation throughout VND execution.



4.6.2 Computational results for the INRC-II instances

This section presents the experiments employing the INRC-II instances. All tables present average results for each group of instances. The first column details the Instance Id, where *n035, n070, n110* represent the number of nurses and where *w4* and *w8* correspond to the number of weeks. Each group contains 10 instances for a total of 60. The column *std. dev.* provides the standard deviation on the average value which is reported in the previous column. Detailed computational results are publicly available online[1].

4.6.2.1 Complete scheduling horizon, complete set of NRP and NRRP constraints

In these experiments, the complete scheduling horizon and all the NRP and NRRP constraints are considered when rerostering. Table 4.10 presents the average results for both single-day absences and the consecutive-days absences instances. The first block (second and third columns) provides data concerning the initial infeasible solution, the second provides the NRP objective value, while the third provides the NRP+NRRP objective value. Note that to estimate the initial objective value, each unit below the minimum coverage violation was penalized with a weight of 10000.

The second block (fourth to ninth columns) shows the results obtained by the MIP solver. The fourth and fifth columns detail the NRP and the NRP+NRRP objective values, respectively. The sixth and eighth columns provide the time to reach the optimum solution and the time to prove it, while the seventh and ninth columns provide the respective standard deviations. The last block (tenth to twelfth columns) details the results regarding the VND heuristic. The tenth column provides the relative gap to the optimum value, while the eleventh and twelfth columns provide the time in seconds to reach the value and its respective standard deviation.

An interesting finding concerns instance *n035w4_2_9-9-2-1* that has one violation of nurses below the minimum coverage. This infeasibility was solved without any changes concerning working days, days off or shift changes. This was only possible since the number of scheduled nurses is higher than the required minimum and the nurses are multi-skilled. The formulation is designed in such a way that the solver was capable of finding a solution by only changing skills of nurses who were already assigned to shifts. Other noteworthy observations concern instances *n070w4_0_3-6-5-1* and *n110w4_2_5-1-3-0*, which were feasible even with the randomly generated disruptions. This occurs

[1]<http://www.inf.ufrgs.br/~tiwickert/download/2017/reroster>

Table 4.10: Complete scheduling horizon NRP+NRRP constraints.

	Single-day absences										
	Initial infeasible solution		MIP						VND		
Instance Id	NRP OFV	NRP+NRRP OFV	NRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.	Gap(%)	Time(s)	Std. Dev.
n035w4	2370.0	39770.0	2546.0	3206.0	3.7	1.1	5.7	1.1	0.6	1.0	0.2
n035w8	4912.5	84712.5	5165.5	6490.5	8.5	5.5	14.0	4.5	1.0	1.8	0.5
n070w4	4704.5	42504.5	4665.0	5850.0	5.8	3.7	16.6	7.5	0.0	2.8	0.6
n070w8	10308.0	72908.0	9834.0	12379.0	51.5	37.7	186.3	104.2	0.4	6.6	1.1
n110w4	6183.5	34383.5	6065.0	7495.0	12.1	7.2	42.1	42.9	0.7	3.9	1.0
n110w8	11467.5	113867.5	11211.0	14366.0	78.7	42.8	237.5	130.9	0.9	10.2	1.9
average									0.6		
	Consecutive-days absences										
	Initial infeasible solution		MIP						VND		
Instance Id	NRP OFV	NRP+NRRP OFV	NRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.	Gap(%)	Time(s)	Std. Dev.
n035w4	2175.5	43745.5	2292.0	3167.0	5.3	2.0	9.6	8.0	1.4	1.2	0.3
n035w8	4075.0	117090.0	4548.5	6383.5	7.5	7.8	28.6	48.3	1.9	1.9	0.5
n070w4	4042.5	50032.5	4029.5	5564.5	13.8	23.0	41.9	34.4	0.4	2.5	0.6
n070w8	8856.0	85831.0	8880.0	11595.0	109.2	81.0	208.0	86.1	0.9	6.3	0.9
n110w4	5397.5	52967.5	5059.0	7039.0	28.3	24.1	84.5	57.3	0.5	4.9	0.6
n110w8	9927.0	175977.0	9740.5	13670.5	229.6	267.5	598.9	495.7	1.8	9.7	2.0
average									1.1		

because the absent nurses were generated on days where the number of nurses exceeded the minimum coverage, referred to as the preferred number of nurses. The solving times for instances with single-day absences were considerably quicker than for instances with consecutive-days absences, with the time required to reach the optimum solution for instances n110w8 being 78.7 for single-day absences compared to 229.6 seconds for when consecutive-day absences were generated. Using the VND heuristic, the time required to reach relative gaps of 0.6% and 1.1% were considerably shorter compared to the MIP solver, as can be observed in the eleventh column. Despite the MIP solver generating optimum results in tractable time limits, the VND heuristic still provides a good alternative whenever a MIP solver is not affordable or if an urgent change regarding the current roster must be performed online.

4.6.2.2 Complete scheduling horizon and ignoring the NRP's soft constraints

These experiments consider the complete scheduling horizon, NRRP hard/soft constraints, and NRP hard constraints. In contrast to Section 4.6.2.1, these experiments ignore the NRP's soft constraints when rerostering. The first block of Table 4.11 presents the Instance Id, while the second block provides the results obtained considering all NRP and NRRP constraints. The last block presents the results ignoring the NRP's soft constraints.

Table 4.11's eighth column shows how on average, the NRRP objective value was lower (highlighted in bold) when the NRP soft constraints were dropped. This occurred for 51 of the 60 instances with single-day absences and in 54 instances with consecutive-days absences. The advantage associated with ignoring the NRP soft constraints is the required computation time. Considering the single-day absence instances, Table 4.11 details the average computation time required to reach the optimum value when considering all constraints for the larger instances (n110w8), which was 78.8 seconds, while without the NRP soft constraints the average computation time decreases significantly to just 6.3 seconds. A similar observation occurs for the consecutive-days absences instances where the average times were 229.6 and 7.5 seconds with and without the NRP soft constraints for the *n110w8* instances, respectively.

Table 4.12 presents the same experiment employing the VND heuristic. The results detail a decrease regarding NRRP constraint violations (eighth column) when the NRP constraints are ignored compared to when all constraints are considered (third column). Moreover, when the results of the VND heuristic's relaxed constraints are compared against the MIP results, a smaller increase of the NRRP+NRP objective value (ninth column) is observed when compared against the ninth column of Table 4.11. These results are due to the VND heuristic being unable to reach optimum results in all the instances when the NRP constraints are ignored, benefiting in these cases, the NRP constraints.

It may, therefore, be concluded that the original NRP soft constraints (presented in Table 4.7) are important to consider for generating good quality solutions. However, in some situations it may be useful to ignore them, such as when, for example, an urgent surgery is scheduled and there are also nurse shortages. The surgery should be prioritized over nurse preferences or consecutive working and resting day restrictions/entitlements.

Table 4.11: MIP - Complete scheduling horizon, NRP soft constraints relaxation.

	Single-day absences											
	All constraints					NRP constraints relaxation						
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Opt Prove Time(s)	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	2546.0	660.0	3206.0	3.7	5.7	4332.0	**620.0**	4952.0	0.8	0.2	0.9	0.2
n035w8	5165.5	1325.0	6490.5	8.5	14.0	7583.5	**1165.0**	8748.5	2.7	0.7	3.7	2.3
n070w4	4665.0	1185.0	5850.0	5.8	16.6	8766.0	**970.0**	9736.0	1.7	1.0	2.0	0.9
n070w8	9829.0	2550.0	12379.0	51.5	186.3	16902.0	**1865.0**	18767.0	5.4	1.8	6.5	2.3
n110w4	6065.0	1430.0	7495.0	12.1	42.1	9643.0	**1240.0**	10883.0	1.5	0.8	2.7	2.6
n110w8	11211.0	3155.0	14366.0	78.7	237.5	16538.5	**2665.0**	19203.5	6.3	6.2	39.6	23.2
	Consecutive-days absences											
	All constraints					NRP constraints relaxation						
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Opt Prove Time(s)	NRP OFV	NRRP OFV	NRP+NRRP OFV	Opt Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	2292.0	875.0	3167.0	5.3	9.6	4077.0	**795.0**	4872.0	0.8	0.2	0.8	0.2
n035w8	4553.5	1830.0	6383.5	7.5	28.6	7122.0	**1720.0**	8842.0	2.5	0.3	2.6	0.4
n070w4	4039.5	1525.0	5564.5	13.8	41.9	7926.5	**1340.0**	9266.5	2.0	0.8	2.2	0.8
n070w8	8880.0	2715.0	11595.0	109.2	208.0	14879.5	**2425.0**	17304.5	5.0	2.0	6.3	2.1
n110w4	5059.0	1980.0	7039.0	28.3	84.5	8848.5	**1690.0**	10538.5	2.4	1.7	2.8	1.7
n110w8	9735.5	3935.0	13670.5	229.6	598.9	15089.5	**3470.0**	18559.5	7.5	4.5	16.4	14.0

Table 4.12: VND - Complete scheduling horizon, NRP soft constraints relaxation.

	Single-day absences									
	All constraints					NRP constraints relaxation				
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.
n035w4	2565.0	660.0	3225.0	1.0	0.2	2633.0	**625.0**	3258.0	0.6	0.0
n035w8	5208.0	1350.0	6558.0	1.8	0.5	5683.0	**1185.0**	6868.0	0.9	0.2
n070w4	4670.5	1180.0	5850.5	2.8	0.6	5099.5	**970.0**	6069.5	1.3	0.1
n070w8	9962.5	2467.9	12430.4	6.6	1.1	10983.0	**1875.0**	12858.0	1.9	0.1
n110w4	6124.0	1420.0	7544.0	3.9	1.0	6369.0	**1265.0**	7634.0	1.8	0.1
n110w8	11303.0	3194.7	14497.7	10.2	1.9	12172.0	**2725.0**	14897.0	2.7	0.2
	Consecutive-days absences									
	All constraints					NRP constraints relaxation				
Instance Id	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.	NRP OFV	NRRP OFV	NRP+NRRP OFV	Time(s)	Std. Dev.
n035w4	2330.5	880.0	3210.5	1.2	0.3	2492.0	**805.0**	3297.0	0.6	0.0
n035w8	4624.6	1878.1	6502.7	1.9	0.5	5048.0	**1800.0**	6848.0	0.9	0.2
n070w4	4130.6	1454.9	5585.5	2.5	0.6	4402.0	**1340.0**	5742.0	1.3	0.1
n070w8	9009.3	2688.8	11698.1	6.3	0.9	9488.0	**2465.0**	11953.0	2.0	0.1
n110w4	5103.1	1969.9	7073.0	4.9	0.6	5559.0	**1695.0**	7254.0	1.9	0.2
n110w8	9912.8	4002.9	13915.7	9.7	2.0	10939.5	**3565.0**	14504.5	2.7	0.2

4.6.2.3 Scheduling horizon relaxation

These experiments evaluate the impact of rerostering when considering different scheduling horizons. The complete scheduling horizon, the most straightforward approach, is analyzed in addition to restricted horizons which only reroster those days where nurses are absent, from the first absent day until the last absent day, or from the first absent day until the end of the scheduling horizon. All NRP and NRRP constraints were considered throughout these experiments.

Tables 4.13 and 4.14's fifth columns document the results when only rerostering on absent days employing the MIP solver and the VND heuristic, respectively. Whereas this restricted rerostering considers a problem which is more constrained, the computational results indicate only a slight deterioration concerning solution quality compared against the complete scheduling horizon. Values in bold indicate improvements obtained by restricting the rerostering horizon. This rerostering strategy, therefore, provides a good alternative when obtaining a solution is urgent and must be acquired within a very short period of time.

Tables 4.13 and 4.14 also present the results when the scheduling horizon is limited from the first absence to the last absence (third block), and until the end of the scheduling horizon (fourth block). The results are very similar to when the complete scheduling horizon is considered. The gaps reported in Table 4.14 demonstrate consistent performance of the VND heuristic under different strategies regarding scheduling horizon relaxations. For the consecutive-day absences the relative gaps are 1.1%, 1.0%, 1.7% and 1.1% for the complete scheduling horizon, only absent days, first absence to last absence, and first absence to the end of the scheduling horizon, respectively.

It can therefore be concluded that the VND heuristic generates near-optimum results (with gaps less than 2%), providing a good alternative to the MIP solver. When the new schedule has already been communicated to all employees and the new month has not yet begun, then rerostering the complete scheduling horizon provides the best alternative. Nevertheless, it is worthwhile to consider other strategies which restrict the scheduling horizon, given that the NRRP depends on when an employee communicates their absence. For example, if the new month has already begun and some employees communicate unavailabilities, the revised roster should not reconsider assignments from the past and consequently the beginning of the new scheduling horizon should instead be the first absent day.

Table 4.13: MIP - Comparison of scheduling horizons.

	Single-day absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
n035w4	3206.0	3.7	5.7	**3217.0**	0.5	0.6	3206.0	2.3	3.7	3206.0	3.0	4.7
n035w8	6490.5	8.5	14.0	**6504.0**	1.1	1.1	6490.5	6.7	10.9	6490.5	7.3	15.4
n070w4	5850.0	5.8	16.6	**5867.0**	2.4	2.5	**5857.0**	7.4	10.2	**5852.5**	5.6	16.3
n070w8	12379.0	51.5	186.3	**12452.0**	5.0	6.8	**12382.5**	41.0	160.6	12379.0	36.9	148.5
n110w4	7495.0	12.1	42.1	**7501.5**	3.9	5.3	**7496.5**	6.2	20.2	7495.0	9.0	25.1
n110w8	14366.0	78.7	237.5	**14433.5**	5.7	10.5	**14375.0**	67.0	199.3	14366.0	61.8	164.3
	Consecutive-days absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
n035w4	3167.0	5.3	9.6	**3177.5**	1.1	1.2	3167.0	4.4	5.5	3167.0	4.1	5.4
n035w8	6383.5	7.5	28.6	**6392.0**	2.2	2.6	6383.5	7.1	29.8	6383.5	6.9	40.9
n070w4	5564.5	13.8	41.9	**5581.0**	3.8	7.8	5564.5	7.0	27.1	5564.5	7.4	31.0
n070w8	11595.0	109.2	208.0	**11623.5**	6.0	11.5	11595.0	85.3	180.0	11595.0	73.3	190.3
n110w4	7039.0	28.3	84.5	**7055.5**	7.1	21.8	7039.0	27.7	65.9	7039.0	29.7	60.8
n110w8	13670.5	229.6	598.9	**13692.5**	54.0	84.4	**13674.0**	188.5	597.2	**13674.0**	178.9	466.9

4.6.3 Computational results for the Lisbon instances

Since there are no soft constraints in the Lisbon instances, only those strategies concerning the scheduling horizon are analyzed for this dataset. Tables 4.15 and 4.16 present the results for the Lisbon instances using the MIP solver and VND heuristic, respectively. In both tables, rerostering the complete scheduling horizon and only a limited part is evaluated. In Table 4.15 the columns labeled *OFV* report the optimum objective values obtained by the MIP solver for each scheduling horizon, while columns *opt time(s)* and *opt prove time(s)* are the times (in seconds) to reach the optimum value and to prove

Table 4.14: VND - Comparison of scheduling horizons.

	Single-day absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
n035w4	3225.0	0.6	1.0	**3233.5**	0.5	0.4	**3332.9**	3.8	1.0	3225.0	0.6	0.9
n035w8	6558.0	1.0	1.8	**6570.0**	1.0	0.6	**6614.0**	1.9	1.7	6558.0	1.0	1.7
n070w4	5850.5	0.0	2.8	**5867.5**	0.0	1.5	**5893.1**	0.6	2.3	**5853.0**	0.0	2.6
n070w8	12430.4	0.4	6.6	**12494.9**	0.3	3.5	**12456.4**	0.6	6.0	12430.4	0.4	6.3
n110w4	7544.0	0.6	3.9	**7550.5**	0.6	2.7	**7565.0**	0.9	3.2	7544.0	0.6	3.6
n110w8	14497.7	0.9	10.2	**14566.0**	0.9	6.2	**14554.2**	1.2	9.1	14497.7	0.9	9.8
		0.6			0.6			1.5			0.6	

	Consecutive-days absences											
	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
n035w4	3210.5	1.4	1.2	**3219.0**	1.3	0.6	**3223.5**	1.8	1.0	3210.5	1.4	1.1
n035w8	6502.7	1.8	1.9	**6506.4**	1.8	0.9	**6560.6**	2.7	2.0	6502.7	1.8	1.7
n070w4	5585.5	0.4	2.5	**5596.4**	0.3	1.7	**5610.4**	0.8	2.2	5585.5	0.4	2.3
n070w8	11698.1	0.9	6.2	**11716.6**	0.8	3.5	**11711.8**	1.0	5.7	11698.1	0.9	5.9
n110w4	7073.0	0.5	5.0	**7083.5**	0.4	3.9	**7171.6**	1.8	4.7	7073.0	0.5	4.6
n110w8	13915.7	1.8	9.7	**13931.2**	1.7	7.6	**13965.3**	2.1	9.0	**13919.2**	1.8	9.3
		1.1			1.0			1.7			1.1	

optimality, respectively. In Table 4.16 the *gap* is relative to the optimum value obtained by the MIP solver for each scheduling horizon.

Table 4.15 details the results when employing the MIP solver. All instances were quickly solved to optimality. In the worst case, the MIP solver proved the optimum solution within 2.4 seconds. Both rerostering the complete scheduling horizon and rerostering from first absence until the end of the scheduling horizon generated the best results, while rerostering only the absent days resulted in infeasibility for 8 of the 68 instances. Finally, rerostering from the first absence until the last absent day resulted in 7 instances being infeasible. Note that instance *II7_19* is infeasible for all the scheduling horizons.

Table 4.16 details the results obtained by the VND heuristic. The best results for the instances with 19 nurses were obtained by restricting the scheduling horizon to only the absent days while the worst solutions were obtained when considering the full scheduling horizon. This indicates that the algorithm's overall performance improves when restricting the available possibilities for rerostering. Increasing the allowed computation time of the VND heuristic to ten minutes did not considerably improve the average relative gaps. For instances with 32 nurses, the chosen strategy does not affect the average gaps significantly. Compared to the MIP solver, solutions within 1% of the optimum solutions are obtained in comparable computation time. Independent of which strategy was applied, the VND heuristic performed significantly better when more nurses are available for rerostering as this allowed for more possibilities to repair the infeasibilities.

Table 4.15: MIP - Experiments employing the Lisbon instances

	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)	OFV	Opt Time(s)	Opt Prove Time(s)
I1_19	3	0.3	0.3	3	0.0	0.1	3	0.0	0.1	3	0.0	0.1
I2_19	2	0.2	0.3	2	0.0	0.1	2	0.1	0.1	2	0.0	0.1
I3_19	9	0.5	0.8	9	0.1	0.1	9	0.1	0.1	9	0.1	0.1
I4_19	2	0.2	0.3	2	0.0	0.1	2	0.0	0.1	2	0.1	0.2
I5_19	15	0.3	0.6	17	0.1	0.1	17	0.2	0.2	17	0.1	0.1
I6_19	8	0.3	0.5	8	0.2	0.2	8	0.2	0.2	8	0.2	0.2
I7_19	19	0.3	0.7	20	0.3	0.4	20	0.3	0.3	20	0.3	0.3
I8_19	2	0.2	0.2	2	0.0	0.0	2	0.0	0.0	2	0.1	0.2
II1_19	1	0.2	0.2	1	0.0	0.0	1	0.0	0.0	1	0.1	0.1
II2_19	0	0.0	0.1	0	0.0	0.0	0	0.0	0.0	0	0.0	0.1
II3_19	5	0.2	0.3	5	0.1	0.1	5	0.0	0.2	5	0.1	0.2
II4_19	10	0.3	0.7	∞	0.1	0.2	12	0.1	0.3	12	0.1	0.3
II5_19	6	0.2	0.4	6	0.1	0.1	6	0.1	0.1	6	0.1	0.2
II6_19	16	0.3	0.5	16	0.1	0.3	16	0.1	0.3	16	0.1	0.3
II7_19	∞	-	-	∞	-	-	∞	-	-	∞	-	-
II8_19	3	0.2	0.3	6	0.1	0.2	6	0.1	0.2	5	0.1	0.3
III1_19	7	0.3	0.3	7	0.0	0.1	7	0.0	0.1	7	0.2	0.2
III2_19	9	0.3	0.6	∞	0.0	0.1	∞	0.0	0.1	12	0.5	0.5
III3_19	10	0.2	0.5	∞	0.0	0.1	∞	0.0	0.1	13	0.5	0.5
III4_19	7	0.2	0.4	7	0.1	0.2	7	0.1	0.2	7	0.2	0.3
III5_19	27	1.1	1.1	27	0.6	0.6	27	0.5	0.5	27	0.7	0.8
III6_19	26	0.8	0.8	28	0.3	0.5	26	0.6	0.6	26	0.6	0.6
III7_19	18	0.7	0.9	23	0.3	0.4	19	0.3	0.8	19	0.3	0.7
III8_19	10	0.3	0.7	10	0.1	0.3	10	0.1	0.3	10	0.2	0.3
IV1_19	8	0.2	0.7	∞	0.0	0.1	∞	0.0	0.1	9	0.3	0.4
IV2_19	11	0.3	0.8	∞	0.0	0.1	∞	0.0	0.2	12	0.4	0.4
IV3_19	10	0.2	0.6	∞	0.0	0.1	∞	0.0	0.1	10	0.3	0.3
IV4_19	26	0.3	0.5	∞	0.1	0.1	∞	0.1	0.2	26	0.2	0.5
IV5_19	17	0.3	0.9	19	0.2	0.6	19	0.7	1.0	17	0.3	0.6
IV6_19	21	0.9	1.1	25	0.1	0.3	23	0.3	0.4	23	0.6	0.6
IV7_19	9	0.3	0.6	9	0.1	0.2	9	0.3	0.6	9	0.3	0.6
IV8_19	9	0.2	0.5	9	0.1	0.3	9	0.2	0.4	9	0.2	0.5
I1_32	3	0.3	0.5	3	0.1	0.3	3	0.1	0.4	3	0.1	0.3
I2_32	3	0.3	0.4	3	0.1	0.2	3	0.1	0.2	3	0.1	0.1
I3_32	6	0.3	0.5	6	0.1	0.4	6	0.1	0.4	6	0.1	0.4
I4_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
I5_32	8	0.4	0.9	8	0.1	0.3	8	0.1	0.3	8	0.1	0.3
I6_32	12	0.3	0.5	12	0.1	0.2	12	0.1	0.2	12	0.1	0.2
I7_32	7	0.3	0.6	7	0.1	0.2	7	0.1	0.2	7	0.1	0.1
I8_32	8	0.3	0.6	8	0.1	0.2	8	0.1	0.1	8	0.1	0.2
II1_32	1	0.3	0.4	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
II2_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.1	0.1
II3_32	3	0.4	0.4	3	0.1	0.2	3	0.1	0.2	3	0.1	0.2
II4_32	7	0.4	0.7	7	0.1	0.3	7	0.1	0.4	7	0.2	0.6
II5_32	16	0.3	0.6	16	0.2	0.4	16	0.2	0.3	16	0.2	0.3
II6_32	20	0.3	0.7	20	0.1	0.4	20	0.1	0.4	20	0.1	0.4
II7_32	6	0.3	0.5	6	0.1	0.2	6	0.2	0.2	6	0.2	0.2
II8_32	5	0.3	0.5	5	0.1	0.2	5	0.2	0.3	5	0.2	0.3
III1_32	7	0.4	0.6	7	0.1	0.2	7	0.1	0.2	7	0.2	0.4
III2_32	5	0.3	0.4	5	0.1	0.1	5	0.1	0.1	5	0.2	0.3
III3_32	7	0.4	0.5	7	0.1	0.2	7	0.1	0.2	7	0.2	0.4
III4_32	6	0.3	0.5	6	0.1	0.2	6	0.1	0.2	6	0.3	0.4
III5_32	19	0.4	1.2	19	0.2	0.6	19	0.3	0.8	19	0.3	0.8
III6_32	36	0.4	1.4	36	0.3	1.2	36	0.3	1.3	36	0.3	1.5
III7_32	22	0.4	1.0	22	0.3	1.2	22	0.4	1.1	22	0.3	1.1
III8_32	25	0.4	1.0	25	0.3	1.0	25	0.3	1.1	25	0.3	1.1
IV1_32	4	0.3	0.6	4	0.1	0.2	4	0.1	0.3	4	0.3	0.4
IV2_32	5	0.3	0.6	5	0.1	0.3	5	0.1	0.2	5	0.3	0.5
IV3_32	4	0.3	0.5	4	0.1	0.1	4	0.1	0.2	4	0.3	0.5
IV4_32	12	0.3	0.5	12	0.1	0.2	12	0.1	0.2	12	0.3	0.5
IV5_32	11	0.4	0.9	11	0.2	1.1	11	0.2	1.0	11	0.3	0.9
IV6_32	1	0.3	0.3	1	0.1	0.1	1	0.1	0.1	1	0.3	0.3
IV7_32	10	0.4	1.1	10	0.2	0.8	10	0.2	0.7	10	0.3	0.9
IV8_32	22	0.4	1.8	22	0.4	1.3	22	0.3	1.3	22	0.3	1.4
V1_32	7	0.3	0.5	7	0.3	0.5	7	0.4	0.5	7	0.3	0.5
V2_32	19	0.4	1.7	19	0.4	1.8	19	0.4	1.7	19	0.4	1.7
V3_32	19	0.3	1.0	19	0.3	1.0	19	0.4	1.1	19	0.3	0.9
V4_32	87	1.4	2.4	87	1.4	2.4	87	1.4	2.4	87	1.4	2.4

4.6.4 Limits of the solution approaches

The previous results demonstrated that CPLEX was able to find optimum solutions for all feasible instances while requiring very little computation time. This section further

Table 4.16: VND - Experiments employing the Lisbon instances

	Complete scheduling			Only absent days			First absence to last absence			First absence to end scheduling		
Instance Id	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)	OFV	Gap(%)	Time(s)
I1_19	3.0	0.0	0.2	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I2_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.1	2.0	0.0	0.1
I3_19	9.8	8.9	0.5	9.6	6.7	0.1	9.6	6.7	0.2	9.6	6.7	0.2
I4_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.0	2.0	0.0	0.2
I5_19	19.5	30.0	0.5	19.5	14.7	0.2	19.0	11.8	0.2	19.5	14.7	0.2
I6_19	9.5	18.8	0.6	9.5	18.8	0.2	9.5	18.8	0.3	9.5	18.8	0.1
I7_19	27.9	46.8	10.9	27.0	35.0	10.5	27.0	35.0	2.1	27.0	35.0	0.1
I8_19	2.0	0.0	0.2	2.0	0.0	0.1	2.0	0.0	0.1	2.0	0.0	0.1
II1_19	1.0	0.0	0.2	1.0	0.0	0.1	1.0	0.0	0.0	1.0	0.0	0.2
II2_19	0.0	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.2
II3_19	6.5	30.0	0.5	5.5	10.0	0.9	∞	-	-	6.5	30.0	0.2
II4_19	16.5	65.0	0.8	∞	-	-	17.0	41.7	0.6	16.5	37.5	0.3
II5_19	12.0	100.0	0.8	9.0	50.0	2.6	11.0	83.3	1.6	12.0	100.0	0.3
II6_19	25.6	60.0	0.9	25.6	60.0	0.4	25.7	60.6	0.6	25.6	60.0	0.2
II7_19	∞	-	-	∞	-	-	∞	-	-	∞	-	-
II8_19	7.1	136.7	0.7	7.1	18.3	0.2	∞	-	-	7.1	42.0	0.3
III1_19	7.0	0.0	0.2	7.0	0.0	0.1	∞	-	-	7.0	0.0	0.3
III2_19	12.2	35.6	23.7	∞	-	-	∞	-	-	14.0	16.7	0.4
III3_19	14.3	43.0	24.3	∞	-	-	∞	-	-	14.8	13.8	0.3
III4_19	9.8	40.0	0.6	9.8	40.0	0.1	7.9	12.9	0.2	9.8	40.0	0.4
III5_19	34.9	29.3	35.3	33.7	24.8	11.2	∞	-	-	33.0	22.2	0.4
III6_19	33.2	27.7	34.5	32.4	15.7	66.2	32.3	24.2	5.7	33.5	28.8	0.3
III7_19	25.0	38.9	1.6	24.7	7.2	329.6	25.7	35.3	1.4	25.2	32.6	10.1
III8_19	15.3	53.0	0.7	12.1	21.0	7.0	13.6	36.0	1.8	15.3	53.0	0.3
IV1_19	11.9	48.8	0.7	∞	-	-	∞	-	-	11.9	32.2	0.6
IV2_19	14.7	33.6	0.8	∞	-	-	∞	-	-	15.6	30.0	0.4
IV3_19	13.9	39.0	0.8	∞	-	-	∞	-	-	14.1	41.0	0.4
IV4_19	39.2	50.8	6.9	∞	-	-	∞	-	-	39.2	50.8	0.4
IV5_19	30.6	80.0	0.8	23.8	25.3	9.5	21.8	14.7	4.9	30.6	80.0	0.5
IV6_19	34.9	66.2	8.1	27.9	11.6	75.7	35.6	54.8	2.0	34.9	51.7	0.4
IV7_19	10.2	13.3	0.6	10.2	13.3	0.3	10.2	13.3	0.8	10.2	13.3	0.4
IV8_19	10.2	13.3	0.6	10.2	13.3	0.3	10.0	11.1	0.7	10.2	13.3	0.4
average		35.8			16.1			21.9			27.9	
I1_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I2_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.1
I3_32	6.0	0.0	0.6	6.0	0.0	0.2	6.0	0.0	0.2	6.0	0.0	0.2
I4_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
I5_32	8.0	0.0	0.5	8.0	0.0	0.2	8.0	0.0	0.2	8.0	0.0	0.2
I6_32	12.0	0.0	0.5	12.0	0.0	0.1	12.0	0.0	0.3	12.0	0.0	0.1
I7_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.1
I8_32	8.0	0.0	0.5	8.0	0.0	0.1	8.0	0.0	0.1	8.0	0.0	0.1
II1_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
II2_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.2
II3_32	3.0	0.0	0.5	3.0	0.0	0.1	3.0	0.0	0.1	3.0	0.0	0.2
II4_32	7.0	0.0	0.5	7.0	0.0	0.2	7.0	0.0	0.3	7.0	0.0	0.3
II5_32	16.0	0.0	0.5	16.0	0.0	0.3	16.0	0.0	0.5	16.0	0.0	0.3
II6_32	20.0	0.0	0.5	20.0	0.0	0.2	20.0	0.0	0.3	20.0	0.0	0.2
II7_32	6.0	0.0	0.5	6.0	0.0	0.2	6.0	0.0	0.2	6.0	0.0	0.3
II8_32	5.0	0.0	0.5	5.0	0.0	0.2	5.0	0.0	0.3	5.0	0.0	0.3
III1_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.4
III2_32	5.0	0.0	0.5	5.0	0.0	0.1	5.0	0.0	557.1	5.0	0.0	0.3
III3_32	7.0	0.0	0.5	7.0	0.0	0.1	7.0	0.0	0.2	7.0	0.0	0.4
III4_32	6.0	0.0	0.5	6.0	0.0	0.1	6.0	0.0	0.2	6.0	0.0	0.4
III5_32	19.0	0.0	0.5	19.0	0.0	0.3	19.0	0.0	0.4	19.0	0.0	0.3
III6_32	36.9	2.5	14.4	37.5	4.2	10.1	36.6	1.7	9.3	37.5	4.2	10.1
III7_32	22.0	0.0	0.5	22.0	0.0	0.4	22.0	0.0	0.3	22.0	0.0	0.3
III8_32	27.0	8.0	0.9	27.0	8.0	0.6	27.0	8.0	0.5	27.0	8.0	0.6
IV1_32	4.0	0.0	0.5	4.0	0.0	0.1	4.0	0.0	0.1	4.0	0.0	0.4
IV2_32	5.0	0.0	0.5	5.0	0.0	0.1	5.0	0.0	0.1	5.0	0.0	0.4
IV3_32	4.0	0.0	0.5	4.0	0.0	0.1	4.0	0.0	0.1	4.0	0.0	0.4
IV4_32	12.0	0.0	0.5	12.0	0.0	0.1	12.0	0.0	0.2	12.0	0.0	0.5
IV5_32	11.0	0.0	0.5	11.0	0.0	0.3	11.0	0.0	0.4	11.0	0.0	0.4
IV6_32	1.0	0.0	0.5	1.0	0.0	0.1	1.0	0.0	0.1	1.0	0.0	0.4
IV7_32	10.0	0.0	0.5	10.0	0.0	0.3	10.0	0.0	0.4	10.0	0.0	0.4
IV8_32	22.0	0.0	0.5	22.0	0.0	0.4	22.0	0.0	0.4	22.0	0.0	0.4
V1_32	7.0	0.0	0.4	7.0	0.0	0.4	7.0	0.0	0.4	7.0	0.0	0.4
V2_32	20.0	5.3	12.5	20.0	5.3	12.1	21.0	10.5	11.0	20.0	5.3	11.6
V3_32	19.7	3.7	9.3	19.7	3.7	9.6	19.9	4.7	8.7	19.7	3.7	9.1
V4_32	102.0	17.2	126.3	102.0	17.2	119.7	102.0	17.2	67.4	102.0	17.2	120.5
average		1.0			1.1			1.2			1.1	

challenges the proposed integer programming model by investigating the performance of an alternative MIP solver and analyzing the performance of the solution approaches on large-scale problem instances.

Table 4.17 compares the performance of CPLEX against that of Coin-OR CBC, one of the leading open-source MIP solver projects (LOUGEE-HEIMER, 2003). The third and seventh columns provide the number of feasible solutions found for each group of ten INRC-II instances. The fourth and fifth columns detail the time required by CPLEX to reach the reported objective value and the time required to prove optimality, while the eighth and tenth columns show these times for Coin-OR CBC. The reported standard deviations are always relative to the times in the preceding column. A dash (-) indicates that no feasible solution was found within the imposed time limit of one hour.

Table 4.17: Open-source solver - complete scheduling horizon NRP+NRRP constraints.

	CPLEX				Coin-OR CBC					
Instance Id	OFV	#Feasible	Opt Time(s)	Opt Prove Time(s)	OFV	#Feasible	Time(s)	Std. Dev.	Opt Prove Time(s)	Std. Dev.
n035w4	3167.0	10	5.3	9.6	3167.0	10	183.1	113.9	200.5	135.2
n035w8	6383.5	10	7.5	28.6	6420.5	10	1742.6	724.8	-	-
n070w4	5564.5	10	13.8	41.9	5565.0	10	1299.4	472.0	-	-
n070w8	11595.0	10	109.2	208.0	-	0	-	-	-	-
n110w4	7039.0	10	28.3	84.5	7062.0	10	2406.4	599.1	-	-
n110w8	13670.5	10	229.6	598.9	-	0	-	-	-	-

In general, the computation times of Coin-OR CBC were much longer than those of CPLEX. Consequently, Coin-OR CBC could prove optimality only for the smallest instances with 35 nurses and a scheduling horizon of four weeks and was unable to find feasible solutions for the larger instances containing 70 and 110 nurses and a scheduling horizon of eight weeks. However, on instances for which feasible solutions were obtained, Coin-OR CBC performed only slightly worse than CPLEX, indicating that the open-source solver is a suitable alternative when the number of nurses is limited and when short computation times are not crucial.

To investigate the performance of the proposed solution approaches on large problem instances, ten additional larger instances containing 150, 200, 300, 400 and 500 nurses were generated based on the INRC-II constraints and problem characteristics. Table 4.18 presents the results using CPLEX and the VND heuristic for these much larger instances. For each instance, the complete scheduling horizon and all NRP and NRRP constraints were considered. Note that Coin-OR CBC is not included in this comparison as Table 4.17 already demonstrated that instances with 110 nurses are beyond its capabilities.

The first column in Table 4.18 describes the instance size ranging from 150 to 500 nurses and scheduling horizon of four and eight weeks. The second and seventh columns show the percentage of instances for which a feasible solution was found. The third and eight columns provide the objective function values, while the fourth and ninth columns detail the gap relative to the lower bound obtained by CPLEX. The fifth and tenth columns provide the required computation time in seconds. The sixth and eleventh columns present the standard deviation of the computation time. Infeasible solutions were not taken into account for these calculations.

Table 4.18: Large instances - complete scheduling horizon NRP+NRRP constraints.

	CPLEX					VND				
Instance Id	Feasible(%)	OFV	Gap(%)	Time(s)	Std. Dev.	Feasible(%)	OFV	Gap(%)	Time(s)	Std. Dev.
n150w4	100	126550.5	0.0	318.1	146.2	100	126889.3	0.3	39.1	10.3
n150w8	100	317429.5	0.0	2317.2	897.5	100	318278.3	0.3	107.7	32.1
n200w4	100	177228.0	0.0	557.9	165.4	100	177844.5	0.4	79.6	22.6
n200w8	100	442882.5	0.6	3421.7	234.3	100	444443.5	1.0	275.8	94.3
n300w4	100	270278.0	0.0	1925.7	991.6	100	270989.6	0.3	176.0	51.9
n300w8	100	689322.0	1.3	3241.5	444.0	100	690517.9	1.5	722.6	237.3
n400w4	100	358380.0	0.0	2233.5	678.7	100	359583.0	0.3	489.3	63.7
n400w8	70	903917.1	1.8	2385.3	1069.8	100	904319.2	1.9	1877.2	253.3
n500w4	100	453763.6	0.1	3394.9	283.3	100	454904.0	0.3	527.4	53.6
n500w8	0	-	-	3600.0	-	100	1375360.4	-	3790.7	589.2

CPLEX manages to consistently find feasible solutions for problems with up to 400 nurses and a scheduling horizon of four weeks. Even for the instances with 400 nurses and scheduling horizon of eight weeks, a feasible solution was found for the majority of instances (7 out of 10). An interesting observation was that when CPLEX can solve the initial infeasibility, it quickly found (near-)optimum solutions in very limited computation time. For the instances with 500 nurses and scheduling horizon of eight weeks, CPLEX was unable to find any feasible solutions within the time limit. By contrast, the VND heuristic generated feasible solutions for all these instances in considerably shorter running time, with exception of the instances with 500 nurses and eighth weeks where the running time was on average 3791 seconds. The solutions obtained by the VND heuristic were near-optimum with an average gap of only 0.7%, demonstrating how it provides the best solution approach for large-scale problems with hundreds of nurses if low computation times are required.

4.7 Conclusions

The primary contribution of this work is the evaluation of novel rerostering strategies such as the relaxation of the NRP soft constraints and various rerostering scheduling horizons. Additionally, a general integer programming formulation considering multi-skilled nurses and a large set of constraints from both the NRP and the NRRP was introduced. A third contribution is a VND heuristic, which provides an alternative to commercial solvers and significantly reduces the required computation time at the expense of very small reductions in solution quality. Furthermore, a new set of instances derived from those proposed by Moz and Pato (2007) and Ceschia et al. (2019), which are used throughout the computational experiments, have been made publicly available online[2].

Besides the NRRP constraints, the computational study has revealed that maintaining the original NRP's constraints is important for obtaining high-quality NRRP solutions. However, ignoring the NRP's soft constraints provides a good alternative when urgent demands require online changes to the current roster, such as when, for example, it is necessary to cover a shortage of professionals for a surgery. If only the days on which absenteeism occurs are evaluated, less time is required to reach solutions and this, therefore, represents a good alternative strategy when very little time is available for rerostering. Only considering the period from the first absent day until the last absent day or until the end of the scheduling horizon generated similar results, but both scheduling horizons are important to consider when an absence is communicated during the current month.

Results also demonstrated that some solutions remain valid despite nurse absenteeism, with this type of roster robustness being a desirable solution property given how it minimizes the impact when personnel shortages occur. The next chapter addresses this issue by proposing a metric for quantifying and generating robust rosters.

[2]<http://www.inf.ufrgs.br/~tiwickert/download/2017/reroster>

5 ROBUST ROSTERING

5.1 Introduction

A recent study reports that Belgian organizations suffer from a short-term (less than one month) absenteeism rate of 2.44% (SD WORX, 2013). On average, an employee was absent for 9.4 days in 2008, while this increased to 12.1 days by 2017. Other studies report absence rates between 3% and 6% in Europe and 2.8% in the United States (EDWARDS; GREASLEY, 2010; BUREAU OF LABOR STATISTICS, 2017). The direct and indirect effects of employee absenteeism are significant (KOCAKULAH et al., 2016). The Society for Human Resource Management (2016) reports that 67% of the questioned organizations perceived absences to have a moderate to significant impact on productivity, while 62% said absences disrupted the work of others, and 51% reported them to increase stress. Moreover, employees with supervisory responsibility spent on average 3.3 hours per week responding to absences by searching for replacements, adjusting workflow and providing additional training.

Unforeseen absences typically have a significant impact on staff rosters, quickly rendering them sub-optimal or even infeasible (DRAKE, 2014). To address this issue, the academic literature has introduced the concept of robustness which enforces solutions less sensitive to disruptions (BERTSIMAS; SIM, 2004). Algorithms which generate such robust solutions are referred to as proactive methods. By contrast, algorithms used to repair infeasible solutions are referred to as reactive methods and are typically employed in more dynamic problem settings.

In the context of project scheduling, there is a large body of research concerning methods for creating robust schedules (HERROELEN; LEUS, 2004; HERROELEN; LEUS, 2005). Such strategies include adding idle time between activities (LAMBRECHTS; DEMEULEMEESTER; HERROELEN, 2008) or allocating additional resources (LAMBRECHTS; DEMEULEMEESTER; HERROELEN, 2011). Robustness is also a desirable property in timetabled services such as airlines or public transport (SAFAK; GÜREL; AKTÜRK, 2017; BURGGRAEVE et al., 2017). Crew scheduling for airlines has received particular attention given that it is considered a crucial component with respect to achieving high-quality service levels. Dück et al. (2012) and Ionescu and Kliewer (2011) propose methodologies based on column generation for improving the robustness of crew schedules.

For staff rostering, common proactive strategies include maximizing employee substitutability, designing effective hiring and overtime policies, using reserve shifts (INGELS; MAENHOUT, 2017b), and efficiently employing available staff (GUO et al., 2014). Reactive procedures, used to repair disruptions which have occurred after the baseline roster was generated, have been proposed by Moz and Pato (2003), Maenhout and Vanhoucke (2011), Bäumelt et al. (2016) and Wickert, Smet and Vanden Berghe (2019). Although these procedures have been demonstrated to be effective, they always negatively impact employees' personal lives due to inevitable last-minute changes to the roster (WILLIAMS; LAMBERT; KESAVAN, 2017). Flexible baseline rosters, that is, robust solutions, are essential in mitigating such negative effects.

Since currently does not exist a standardized means of quantifying staff roster robustness, this research proposes a metric which enables one to calculate the degree to which a given real-world roster is immune to unforeseen absences, taking into account organizational and personal constraints as well as complex skill structures. By including the proposed robustness metric in an integer programming model, a methodology is established for generating rosters with a predefined minimum level of robustness by way of assigning employees to reserve shifts in addition to their regular working shifts. Since the focus is on addressing employee absenteeism, demand is considered deterministic. The integer programming formulation is designed such that the robust rosters generated remain feasible for every possible realization of the assigned reserve shifts, meaning that reserve shifts may be either removed from the roster or converted into a working shift without violating any employee's contractual constraints.

The proposed approach is validated using a three-stage procedure which simulates employee absences based on a given probability distribution, and which repairs disruptions using a reactive re-rostering procedure. A computational study involving nurses in a hospital ward demonstrates the trade-off between roster robustness and the operational costs incurred by repairing disruptions. Results demonstrate how significant cost reductions may be achieved by increasing the robustness of rosters.

The remainder of the chapter is organized as follows. Section 5.2 reviews literature related to robust staff rostering. Section 5.3 introduces the staff rostering problem, while Section 5.4 introduces how robustness is quantified for both single- and multi-skilled employees. The methodology employed to generate robust rosters and validate the proposed measures is detailed throughout Section 5.5. Section 5.6 describes the computational experiments, while Section 5.7 presents conclusions and future research possibilities.

5.2 Related work

Van den Bergh et al. (2013) identified three main categories of uncertainty in staff rostering: i) uncertainty of demand in which the expected workload is not deterministic, ii) uncertainty of capacity in which actual available manpower may not be the same as the planned manpower, and iii) uncertainty of arrival which refers to a non-deterministic arrival pattern of the workload. Uncertainty of demand is common in hospitals since the workload depends on patient admissions, however, most hospitals operate under the assumption of fixed demand. Uncertainty of capacity occurs in every organization working with human resources given the unavoidable occurrence of employee absenteeism. Uncertainty of arrival is strongly related to uncertainty of demand and has an impact only if individual tasks require scheduling.

There are two general approaches which currently exist for introducing robustness in staff rosters: those focusing on horizontal robustness and those focusing on vertical robustness. Horizontal robustness may be obtained by extending shift durations or by permitting overtime to compensate for employee shortages (INGELS; MAENHOUT, 2018). Meanwhile, vertical robustness is introduced through different types of buffers which manage uncertainty of demand and capacity through the use of surplus resources. Capacity buffers involve assigning more resources than originally required by the nominal problem by, for example, rostering up to a preferred staffing level rather than a minimum level (INGELS; MAENHOUT, 2017b). A similar strategy is the use of reserve shifts, where employees are assigned to special reserve shifts which, if necessary, are converted into working shifts to cover disruptions (INGELS; MAENHOUT, 2015). Alternative strategies include introducing replaceability among employees such that absences may be covered without affecting roster feasibility (INGELS; MAENHOUT, 2017a) or by enabling overlap between working shifts (LUSBY et al., 2012).

The different approaches introduced in the academic literature typically have one or more parameters responsible for indirectly controlling the robustness which is enforced in a roster. For example, the number of reserve shifts required in a solution determines how robust a roster will be against uncertainty of capacity and demand (INGELS; MAENHOUT, 2015). The best value of this parameter is set based on an extensive empirical study of different strategies specific for the considered experimental setting. Similarly, Lusby et al. (2012) control the level of robustness by increasing the expected workload and by adding delays to the start time of tasks. However, there is no direct quantifiable relation between these parameters and the degree to which solutions are immune to unforeseen events.

Although there is a history of strategies for enforcing robustness in existing literature, these approaches lack a formal metric for quantifying the robustness of a roster. In contrast, the present research introduces a metric for measuring the robustness of staff rosters in advance using an analytical function rather than relying on the outcome of simulations to evaluate the robustness. Given a roster, the proposed metric quantifies the degree to which a solution is immune to uncertainty of capacity based on the robustness introduced by capacity and reserve shifts considering both single- and multi-skilled employees. An estimation of the robustness level has significant importance for institutions because they can estimate of how immune the current roster is when unexpected employee shortages do occur.

5.3 The staff rostering problem

The studied staff rostering problem is defined by a set of employees $N = \{1,\ldots,|N|\}$, a set of days $D = \{1,\ldots,|D|\}$, a set of shifts $S = \{1,\ldots,|S|\}$, and a set of skills $K = \{1,\ldots,|K|\}$. The goal is to find an assignment of shifts to employees subject to a variety of contractual constraints. An employee may be assigned to at most one shift per day. Moreover, there are forbidden shift successions which prevent a pair of shifts from being assigned on two consecutive days. For example, an early shift may never be assigned on day d if a Late shift was assigned on day $d-1$. Other constraints which are taken into account are related to employee skills, personal requests, the minimum number of days worked in the planning horizon and the maximum number of consecutive assignments. Employees are allowed to work overtime by working more days than the maximum contracted. To correctly evaluate these constraints, data from the preceding scheduling periods are taken into account to avoid constraint violations at the beginning of each planning period (SMET; SALASSA; VANDEN BERGHE, 2017). To avoid issues regarding infeasibility, the demand requirements are relaxed to soft constraints with a very high penalty incurred when not meeting the minimum required levels. The objective is to minimize the (weighted) sum of the soft constraint violations along with employee wage costs incurred by overtime. A detailed description of all problem parameters and constraints, as well as an integer programming formulation of this rostering problem, are presented in Appendix E.

5.4 Measuring robustness

In contrast to other problem domains, a metric for measuring robustness has hitherto received minimal attention in the academic literature. The primary objective of measuring robustness is to estimate how efficiently a given roster can accommodate disruptions before becoming infeasible. For a general overview of robustness in the context of machine scheduling and project scheduling, interested readers are referred to Goren and Sabuncuoglu (2008) and Hazır, Haouari and Erel (2010). This section begins by defining robustness as it is considered in the present research before subsequently introducing how robustness is quantified for staff rostering problems with single- and multi-skilled employees.

5.4.1 Definition of robustness

A roster is considered robust whenever a low cost is associated with repairing disruptions caused by uncertainty of capacity. Two strategies to obtain this type of robustness are considered: i) assigning more employees than the minimum required (capacity buffers), and ii) assigning employees to reserve shifts. The first strategy does not require absent employees to be replaced because more than the minimum are available on the same shift. The second strategy is more flexible as employees in reserve shifts may be deployed to replace any shift. Note that this assumes that a reserve shift may be converted into any of the working shifts without violating the employee's contractual constraints. This is not a trivial assumption and requires a dedicated solution approach to generate such rosters, as will be discussed throughout Section 5.5.

5.4.2 Single-skilled employees

Rosters are usually constructed respecting a constraint which enforces a minimum number of employees required for each day and shift. The proposed metric is expressed relative to this minimum requirement: 0 indicates that none of the employees assigned to working shifts can be replaced and 1 indicates that all employees assigned to working shifts can be substituted. Values higher than 1 may be obtained when a disruption is covered by multiple employees. Table 5.1 details the notation used in the definition of the robustness measures for problems with single-skilled employees.

Table 5.1: Symbols and definitions for problems with single-skilled employees

Symbol	Definition
Parameters	
s'	reserve shift index
$\hat{x}_{nds} \in \{0,1\}$	equals 1 if employee n is assigned to shift s on day d, 0 otherwise
$\hat{x}_{nds'} \in \{0,1\}$	equals 1 if employee n is assigned to reserve shift s' on day d, 0 otherwise
m_{ds}	minimum number of employees required on day d, shift s
Variables	
$\hat{r}_d \in \mathbb{R}_{\geq 0}$	robustness on day d using reserve shifts
$\hat{r} \in \mathbb{R}_{\geq 0}$	reserve shift-robustness for the complete roster
$\hat{o}_{ds} \in \mathbb{N}_{\geq 0}$	number of assigned shifts above the minimum on day d, shift s
$\hat{o}_d \in \mathbb{R}_{\geq 0}$	robustness on day d considering number of shifts above the minimum
$\hat{o} \in \mathbb{R}_{\geq 0}$	capacity buffer-robustness for the complete roster
$rob_d \in \mathbb{R}_{\geq 0}$	robustness on day d
$rob \in \mathbb{R}_{\geq 0}$	robustness of the complete roster

Robustness induced by reserve shifts is calculated as shown in Equation (5.1). The numerator equals the number of employees assigned to reserve shifts on day d, while the denominator equals the total minimum demand summed over all shifts. As the minimum demand is, de facto, a hard constraint, $\hat{r}_d$ expresses how many employees are assigned to reserve shifts relative to the number of employees assigned to regular working shifts.

$$\hat{r}_d = \frac{\sum_{n \in N} \hat{x}_{nds'}}{\sum_{s \in S} m_{ds}} \tag{5.1}$$

Daily robustness calculated by Equation (5.1) may be generalized by averaging $\hat{r}_d$ over all days, as shown in Equation (5.2). Note that by averaging, precision may be lost due to a compensation effect such that the robustness on each day d is not guaranteed to be $\hat{r}$.

$$\hat{r} = \frac{\sum_{d \in D} \hat{r}_d}{|D|} \tag{5.2}$$

Robustness obtained by capacity buffers may be quantified per day d and per shift s. Equation (5.3) calculates, for each day and shift, the number of employees assigned over the minimum required, again expressed relative to the minimum demand. From a robustness point-of-view, the higher this number is, the more unexpected absences on this day and shift may be covered using the surplus employees.

$$\hat{o}_{ds} = \frac{\sum_{n \in N} \hat{x}_{nds} - m_{ds}}{m_{ds}} \tag{5.3}$$

This capacity buffer-robustness may be averaged over all shifts for each day to obtain a more general calculation of this type of robustness. Equation (5.4) calculates this averaged robustness for day $d \in D$.

$$\hat{o}_d = \frac{\sum_{s \in S} \hat{o}_{ds}}{|S|} \tag{5.4}$$

Similar to the generalization step applied between Equations (5.1) and (5.2), $\hat{o}_d$ may be averaged over all days, resulting in Equation (5.5).

$$\hat{o} = \frac{\sum_{d \in D} \hat{o}_d}{|D|} \tag{5.5}$$

The complete roster robustness *rob* considering both capacity buffers and reserve shifts, averaged over all days in the planning horizon, is calculated by summing $\hat{o}$ and $\hat{r}$, as shown in Equation (5.6).

$$rob = \hat{r} + \hat{o} \tag{5.6}$$

The advantage of averaging over all days is that the robustness is represented by a single value. However, the disadvantage is that the robustness might be concentrated in some days, with others exhibiting far lower rates. In such cases, a more refined robustness quantification is required which is specified for day $d \in D$ as shown in Equation (5.7).

$$rob_d = \hat{r}_d + \hat{o}_d \tag{5.7}$$

5.4.3 Multi-skilled employees

Skills represent an additional challenge when rostering an organization's staff since in most cases not all employees are qualified for all tasks (DE BRUECKER et al., 2015). In hospitals, for example, a regular nurse may be qualified for medical tasks for which a caregiver may not have had the necessary training. In such situations, caregivers cannot assume the work of a regular nurse. However, more complex situations may occur such as when there exists a hierarchy of qualifications or an arbitrary skill structure in which each employee has an individual subset of skills. The robustness quantifications introduced in Section 5.4.2 will now be extended for multi-skilled employees where each individual employee may have one or more skills. Table 5.2 introduces the additional notation that is required.

We first consider robustness obtained by using reserve shifts. Note that employees with more than one skill are counted only once when they are assigned to reserve shifts.

Table 5.2: Additional symbols and definitions for problems with multi-skilled employees

Symbol	Definition
Parameters	
$\hat{x}_{ndsk} \in \{0,1\}$	1 if employee n is assigned to shift s on day d with skill k, 0 otherwise
$\hat{x}_{nds'k} \in \{0,1\}$	1 if employee n is assigned to reserve shift s' on day d with skill k, 0 otherwise
m_{dsk}	minimum number of employees required in shift s on day d with skill k
Variables	
$\hat{r}_{dk} \in \mathbb{R}_{\geq 0}$	robustness on day d, skill k, using reserve shifts
$\hat{r}_d \in \mathbb{R}_{\geq 0}$	robustness on day d, using reserve shifts
$\hat{o}_{dsk} \in \mathbb{N}_{\geq 0}$	number of assigned shifts above the minimum on day d, shift s with skill k
$\hat{o}_{ds} \in \mathbb{R}_{\geq 0}$	robustness on day d and shift s, considering shifts above the minimum
$\hat{o}_d \in \mathbb{R}_{\geq 0}$	robustness on day d, considering shifts above the minimum
$rob_{dk} \in \mathbb{R}_{\geq 0}$	robustness on day d for skill k
$rob_d \in \mathbb{R}_{\geq 0}$	robustness on day d
$rob \in \mathbb{R}_{\geq 0}$	robustness for the complete roster

The re-rostering procedure, however, is not limited to use these employees only for the skill they were counted for, allowing them to be assigned to any shift for which they are qualified. Equation (5.8) calculates the robustness for each day $d \in D$ and skill $k \in K$. Note that this is almost identical to the single-skilled case except for the skill index.

$$\hat{r}_{dk} = \frac{\sum_{n \in N} \hat{x}_{nds'k}}{\sum_{s \in S} m_{dsk}} \tag{5.8}$$

Equation (5.9) calculates the average robustness for each day $d \in D$ by averaging $\hat{r}_{dk}$ over all skills. As with previous generalizations, this step makes the quantification less accurate as compensation may occur among different skills.

$$\hat{r}_d = \frac{\sum_{k \in K} \hat{r}_{dk}}{|K|} \tag{5.9}$$

Results may be averaged over all days as shown in Equation (5.10). The resulting robustness quantification again becomes less accurate.

$$\hat{r} = \frac{\sum_{d \in D} \hat{r}_d}{|D|} \tag{5.10}$$

The robustness on day $d \in D$, shift $s \in S$ and skill $k \in K$ induced by capacity buffers may be calculated by Equation (5.11).

$$o_{dsk} = \frac{\sum_{n \in N} \hat{x}_{ndsk} - m_{dsk}}{m_{dsk}} \tag{5.11}$$

As with the previous equations, $\hat{o}_{dsk}$ may be averaged over all shifts, skills and finally days resulting in aggregated, less accurate, robustness quantification, as shown in

Equations (5.12) - (5.14).

$$\hat{o}_{dk} = \frac{\sum_{s \in S} o_{dsk}}{|S|} \tag{5.12}$$

$$\hat{o}_{d} = \frac{\sum_{k \in K} \hat{o}_{dk}}{|K|} \tag{5.13}$$

$$\hat{o} = \frac{\sum_{d \in D} \hat{o}_{d}}{|D|} \tag{5.14}$$

Finally, the complete roster robustness for problems with multi-skilled employees is calculated using Equation (5.15) by summing $\hat{r}$ and $\hat{o}$.

$$rob = \hat{r} + \hat{o} \tag{5.15}$$

Despite the advantage of a single value quantifying a roster robustness as calculated by Equation (5.15), it may not be sufficiently accurate as it presents an average over all days and skills. More detailed quantifications of robustness may be obtained using Equations (5.16) and (5.17) which calculate robustness per day, and per day/skill, respectively.

$$rob_d = \hat{r}_d + \hat{o}_d \tag{5.16}$$

$$rob_{dk} = \hat{r}_{dk} + \hat{o}_{dk} \tag{5.17}$$

5.4.4 Numerical example

Table 5.3 presents an example which considers a hospital setting with ten multi-skilled nurses and a planning horizon of seven days. The shifts are *early* (E), *late* (L) and *night* (N). Meanwhile, there are two skills: *head nurse* (H) and *nurse* (N). On each shift and day, at least one *head nurse* and one *nurse* are required. Nurses 1-5 are qualified for both *head nurse* and *nurse* skills, while Nurses 6-10 only have the *nurse* skill. This means that, for example, Nurses 6-10 may be replaced with Nurses 1-5, while the opposite is not true as Nurses 6-10 do not have the skill *head nurse*.

On the first day, $\hat{r}_d = 0.33$ since only Nurse5 and Nurse9 are assigned to a reserve shift. Although Nurse5 has both *head nurse* and *nurse* skills, she is only counted towards the highest qualification (in this case *head nurse*) and therefore $r_{dH} = 1$ and $r_{dN} = 1$. The last day has both types of robustness: capacity buffers and reserve shifts. Nurse8 and

Nurse10 are assigned to reserve shifts, resulting in $\hat{r}_d = 0.33$. Nurse9 is scheduled in excess of the minimum required for the Night shift and skill *nurse* resulting in a capacity buffer. Variable $\hat{o}_d$ subsequently assumes a value of 0.17. For the last day, the total robustness is $rob_d = 0.33 + 0.17 = 0.50$, contributing towards a general roster robustness of $rob = 0.3571$.

Table 5.3: Proposed robustness measure applied to multi-skilled employee roster.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
N01 [H, N]	E[H]	E[H]	E[H]	N[H]	N[H]	N[H]	-
N02 [H, N]	L[H]	L[H]	L[H]	-	**R**	-	E[H]
N03 [H, N]	N[H]	N[H]	N[H]	-	**R**	-	L[N]
N04 [H, N]	-	**R**	$\boxed{\textit{N[H]}}$	L[H]	L[H]	L[H]	L[H]
N05 [H, N]	**R**	-	$\boxed{\textit{E[H]}}$	E[H]	E[H]	E[H]	N[H]
N06 [N]	E[N]	E[N]	E[N]	-	**R**	-	N[N]
N07 [N]	L[N]	L[N]	L[N]	L[N]	L[N]	L[N]	E[N]
N08 [N]	N[N]	N[N]	N[N]	-	**R**	-	**R**
N09 [N]	**R**	-	$\boxed{\textit{E[N]}}$	E[N]	E[N]	E[N]	$\boxed{\textit{N[N]}}$
N10 [N]	-	**R**	$\boxed{\textit{N[N]}}$	N[N]	N[N]	N[N]	**R**
r_{dH}	1	1	0	0	2	0	0
r_{dN}	1	1	0	0	2	0	2
$\hat{r}_d$	0.33	0.33	0.00	0.00	0.67	0.00	0.33
o_{dEH}	0	0	1	0	0	0	0
o_{dLN}	0	0	0	0	0	0	0
o_{dNH}	0	0	1	0	0	0	0
o_{dEN}	0	0	1	0	0	0	0
o_{dLH}	0	0	0	0	0	0	0
o_{dNN}	0	0	1	0	0	0	1
$\hat{o}_d$	0.00	0.00	0.67	0.00	0.00	0.00	0.17
rob	0.3571						

5.5 Generating and validating robust rosters

Once having a standardized means by which to calculate the robustness of a given roster, it is possible to formulate additional constraints which enforce a minimum robustness level when generating initial rosters. Constraints based on the proposed robustness quantification are added to the integer programming formulation presented in Appendix E.

For example, if a minimum robustness level ρ_d is required on each day $d \in D$ in a multi-skilled setting, Constraints (5.18) are added to model (E.1)-(E.10). Note that this additional constraint is almost identical to Equation (5.13) but substitutes the given assignments $\hat{x}_{ndsk}$ with decision variables x_{ndsk}.

$$\frac{\sum_{s\in S}\sum_{k\in K}(\frac{\sum_{n\in N} x_{ndsk} - m_{dsk}}{m_{dsk}})}{|S||K|} \geq \rho_d \tag{5.18}$$

To validate a roster's ability to cope with uncertainty concerning capacity, a three-phase procedure similar to the approaches of Abdelghany, Abdelghany and Ekollu (2008) and Ingels and Maenhout (2015) is employed. First, a baseline roster is generated using the aforementioned proactive approach. Note that, in addition to minimizing soft constraint violations, the objective function also includes employee wages. Introducing robustness in this phase thus increases the objective value as we assume that employees assigned to a reserve shift receive 10% of their salary, regardless of whether they are called on duty or not. The generated rosters ensure that every reserve shift can be converted into any working shift by carefully modeling the employee's contractual and personal constraints.

The second step involves simulating employee absences which lead to roster disruptions. These disruptions are manifested as employees who become unavailable on one (or more) days in the scheduling period, thereby possibly introducing infeasibilities by violating their contractual constraints, and leading to violations of the demand requirements.

The third step applies a reactive re-rostering method which attempts to restore the solution's feasibility in terms of demand. The integer programming model introduced in Appendix F is employed for this re-rostering phase. Solving this optimization problem converts reserve shifts into working shifts in addition to other modifications such as changes in shift assignments or converting a day off into a working day. The main objective is now to minimize the overall cost regarding the number of changes, that is, the feasibility of the roster must be restored with as few changes as possible. Moreover, it is more preferable for a reserve shift to be converted into a working shift rather than converting a day off into a working shift or changing a shift of an already scheduled nurse. The reason behind this assumption is that unexpected last-minute calls to nurses with days off or changes to an existing work schedule negatively impact their personal lives to a significant degree.

5.6 Computational study

This section conducts a computational study on the methodology proposed throughout Section 5.5 and the robustness quantification introduced in Section 5.4. The primary purpose of this empirical study is to investigate both the effectiveness of the proposed robustness quantification and the impact of capacity and reserve shifts on operational costs. Moreover, insights are provided concerning various strategies for introducing robustness to a roster: i) specifying a general minimum robustness level leaving it to the solver to distribute it among the days and employees, or ii) specifying a minimum robustness level per day while the solver chooses over which skills it should be distributed, or iii) specifying a minimum robustness level per day and skill.

5.6.1 Experimental setup and data

All models were implemented in Java and compiled with OpenJDK 1.8. The experiments were conducted on an AMD FX 8150 eight-core processor at 1400 MHz with 32 GB of RAM memory running Linux Ubuntu 16.04.3 64-bit. CPLEX version 12.7.1 was employed to solve the integer programming formulation using default parameters.

Computational experiments were conducted using the instances from the Second International Nurse Rostering Competition (CESCHIA et al., 2019) as they realistically represent general practice in staff rostering. Since the primary objective of this research is to evaluate the cost associated with disruptions and the influence of enforcing robustness, all soft constraints were treated as hard constraints. The minimum number of nurses per day, shift and skill were adjusted until a feasible solution was generated. By doing so, the objective function models only operational costs and thus avoids compensation by soft constraint violation penalties. For the same reason, constraints related to the minimum number of consecutive working days, days off and days working the same shift were also removed. In addition to regular rostering, robust rostering minimizes the costs associated with nurse wages, overtime and reserve shifts. In the re-rostering step, the objective function is to minimize both the original rostering objective and the number of changes with respect to the baseline solution, that is, feasibility is restored by modifying as few assignments as possible in the roster.

There are numerous re-rostering costs which are additional to those in the original rostering phase: calling a nurse from a reserve shift to a working shift (10% of the nurse's wage, in addition to the regular wage), calling a nurse from a day off or cancelling a working shift (150% of the nurse's wage) and modifying the nurse's assigned shift (100% of the the nurse's wage). In the simulation step, disruptions are generated using a Bernoulli distribution with an absenteeism rate of 2.44%. This percentage is based on a study regarding absenteeism in Belgian organizations (SD WORX, 2013).

Instances employed for computational experiments have 35 nurses and a planning horizon of four weeks. There are four regular shifts *early (E)*, *late (L)*, *day (D)* and *night (N)* in addition to the *reserve shift (R)*. In the multi-skill setting, the considered skills are $K = \{H, N, C, T\}$ which correspond to *head nurse*, *nurse*, *caretaker* and *trainee*, respectively. Nurses have one of four possible subsets of these skills: $\{H, N, C\}$, $\{N, C, T\}$, $\{C, T\}$ or $\{T\}$.

5.6.2 Robustness measure evaluation

First, the effectiveness of the proposed robustness metric to approximate the actual robustness of solutions is evaluated. All reported results are averages obtained by repeating the simulation of disruptions and re-rostering 100 times, as detailed in Section 5.5.

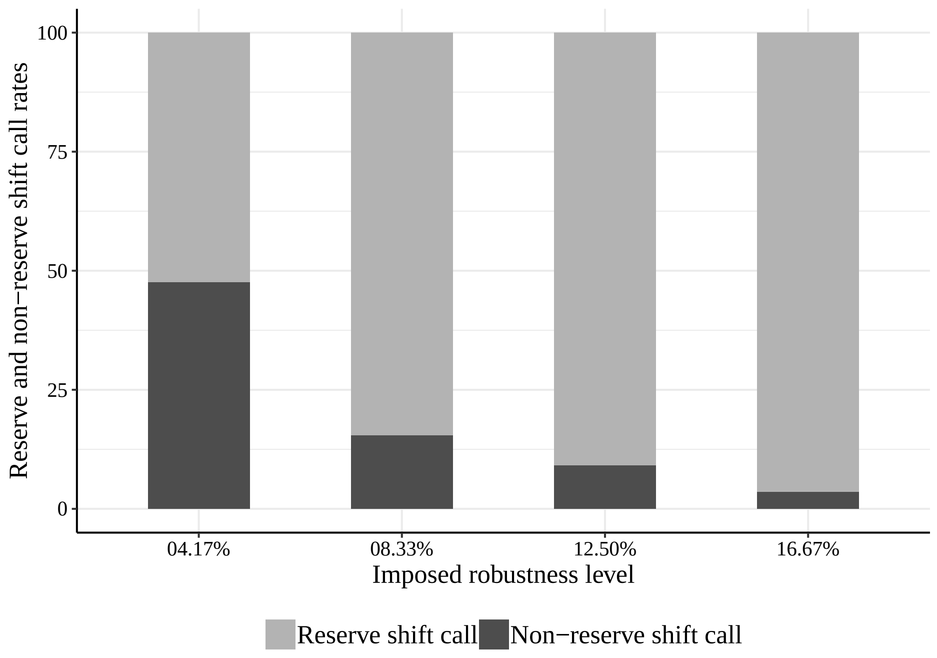
5.6.2.1 Single-skilled nurses

The first set of experiments involves single-skilled employees and considers four different minimum robustness levels $R = \{4.17\%, 8.33\%, 12.50\%, 16.67\%\}$. Based on the experimental setup, these values correspond to assigning one to four employees per day to reserve shift or capacity buffers. Note that higher robustness levels are not attainable due to the limited number of employees, compared to the coverage requirements.

Figure 5.1 details the results of the first robustness strategy which enforces a minimum robustness level by adding a general constraint based on Equation (5.6) to model (E.1)-(E.10). Dark-gray bars represent the proportion rate of disruptions solved by converting days off into working shifts, while light-gray bars indicate the proportion rate of disruptions which were solved by converting reserve shifts into working shifts.

Without proactively adding robustness to the roster, disruptions can only be solved by converting days off into working shifts. However, by increasing the required robustness, a reduction concerning the number of unexpected employee calls is observed and disruptions are primarily solved by converting the reserve shifts to working shifts. For the highest robustness level of 16.67%, 96.45% of the disruptions were repaired by using

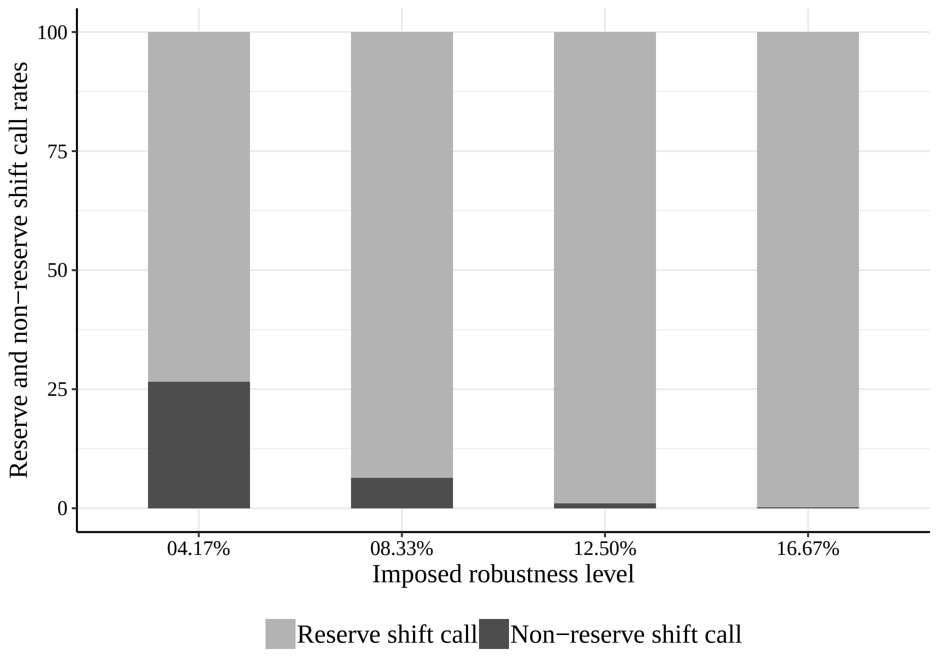
Figure 5.1: Effect of enforcing robustness with a general constraint upon reserve and non-reserve shift call rates.



reserve shifts.

Figure 5.2 plots the results of the second strategy by enforcing robustness on each day in the planning horizon using constraints based on Equation (5.7). In contrast to Equation (5.6), which enforces robustness as a general constraint, now the robustness is ensured per day.

Figure 5.2: Effect of enforcing robustness per day upon reserve and non-reserve shift call rates.



Results show that with a robustness level of 16.67%, 99.76% of the disruptions could be repaired using reserve shifts enforced per day, against 99.45% when reserve shifts were ensured with a general constraint. For the single-skilled setting, therefore, it can be concluded that adding robustness constraints on each day reduces the number of unexpected employees calls the most, as reserve shifts are more uniformly distributed over the planning horizon. Moreover, a correlation between the proposed metric and the roster's actual robustness can be clearly observed: higher robustness levels significantly reduce the number of unexpected calls necessary to repair roster infeasibility.

Computation times for generating both the initial and the re-rostering solutions considering these different robustness levels are reported in Table G.1. There is a tendency that higher robustness levels require more computation time. For example, when robustness is ignored (0.00%) the required time for the initial solution is 1.5 seconds, while when a robustness level of 16.67% is enforced the generation of the initial solution demanded 5.5 seconds. In the re-rostering phase, similar computation times between 1.3 and 1.6 seconds were observed, independent of the required level of robustness. Note that all initial and re-rostering solutions were executed until the optimality was proved by the MIP solver.

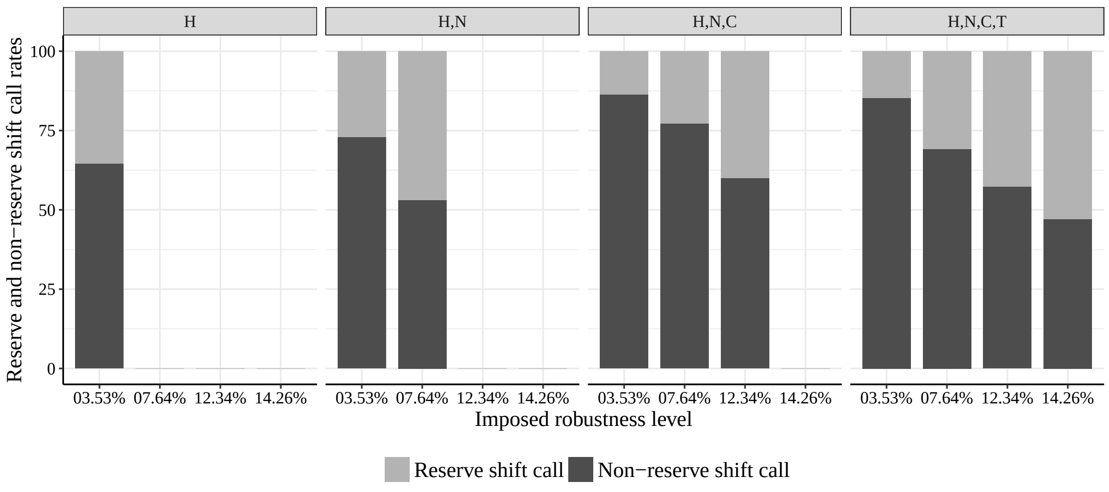
5.6.2.2 Multi-skilled nurses

The second series of experiments analyze the impact of robustness in a multi-skill setting. All graphs presented in this section are divided into four groups, corresponding to the four (sub)sets of skills: $K_1 = \{H\}$, $K_2 = \{H,N\}$, $K_3 = \{H,N,C\}$ and $K_4 = \{H,N,C,T\}$. In the first group, only employees with the skills in K_1 can be assigned to reserve shifts. For the second group, the employees are restricted to those with skills in K_2, and so on. Again, four minimum robustness levels are considered $R = \{3.53\%, 7.64\%, 12.34\%, 14.26\%\}$. Preliminary experiments showed that higher robustness levels are impossible due to the limited number of employees available. Moreover, when only a subset of skills is considered, such as in K_1, K_2 and K_3, high robustness levels could not be attained since the number of employees per skill is limited.

Figure 5.3 details the results obtained by requiring a minimum level of robustness using a constraint based on Equation (5.15). For each of the skill groups, the parameter K in Equations (5.8) and (5.9) is set appropriately.

Figure 5.3 demonstrates a reduction of unexpected employee calls (dark-gray bars) when the robustness is increased. Analyzing the effect of increasing robustness over skill groups confirms the intuition that it is more beneficial to assign head nurses to reserve

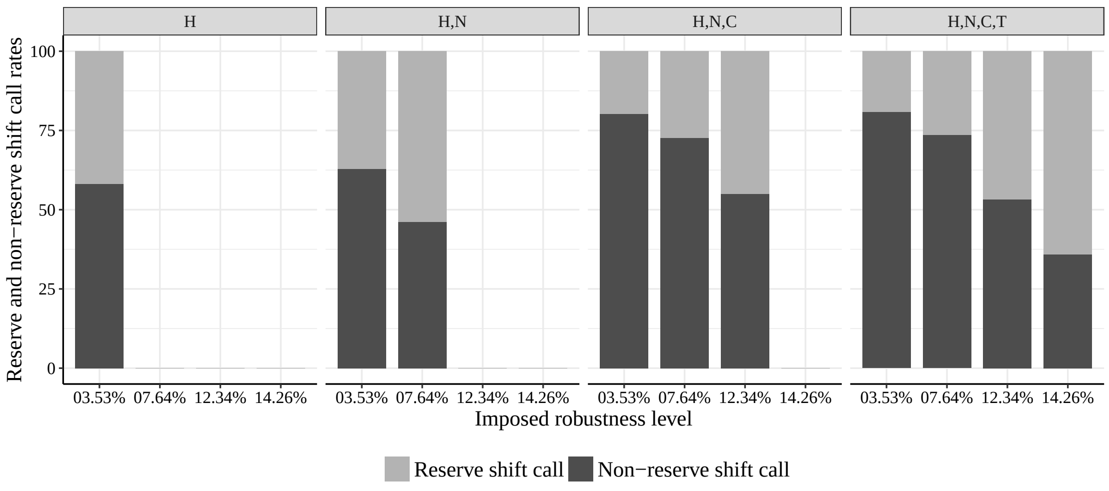
Figure 5.3: Effect of enforcing robustness with a general constraint upon reserve and non-reserve shift call rates.



shifts rather than caretakers or trainees. For example, for a robustness level of 3.53%, the best result was obtained when only employees with skill subset K_1 were assigned to reserve shifts, while a decrease of reserve shifts calls was observed when employees of all skills were included.

Figure 5.4 details the results when robustness is proactively enforced on each day separately using constraints based on Equation (5.16). The results exhibit the same tendency as when robustness is enforced as an average over all days of the planning horizon. However, one distinction lies in the proportion of disruptions solved using reserve shifts. For example, considering a robustness level of 14.26%, when the robustness was induced using a general constraint, 52.96% of the disruptions were solved by using reserve shifts (Figure 5.3). In contrast, when robustness is enforced each day, this percentage improved the use of reserve shifts to 64.16% (Figure 5.4).

Figure 5.4: Effect of enforcing robustness per day upon reserve and non-reserve shift call rates.



The impact of specifying the required robustness in the most detailed fashion, that is, on each day and each skill separately, is demonstrated in Figure 5.5. Here, constraints were added to the rostering formulation based on Equation (5.17). Note that the possible robustness values which could be realized per day and per skill are even more restricted due to the limited number of employees. Robustness is obtained by including one employee of each subset of skills $K_1 = \{H\}$, $K_2 = \{H,N\}$, $K_3 = \{H,N,C\}$ and $K_4 = \{H,N,C,T\}$, which translates into robustness levels of 3.53%, 7.64%, 12.34% and 14.26%.

Figure 5.5: Effect of enforcing robustness per day and skill upon reserve and non-reserve shift call rates.

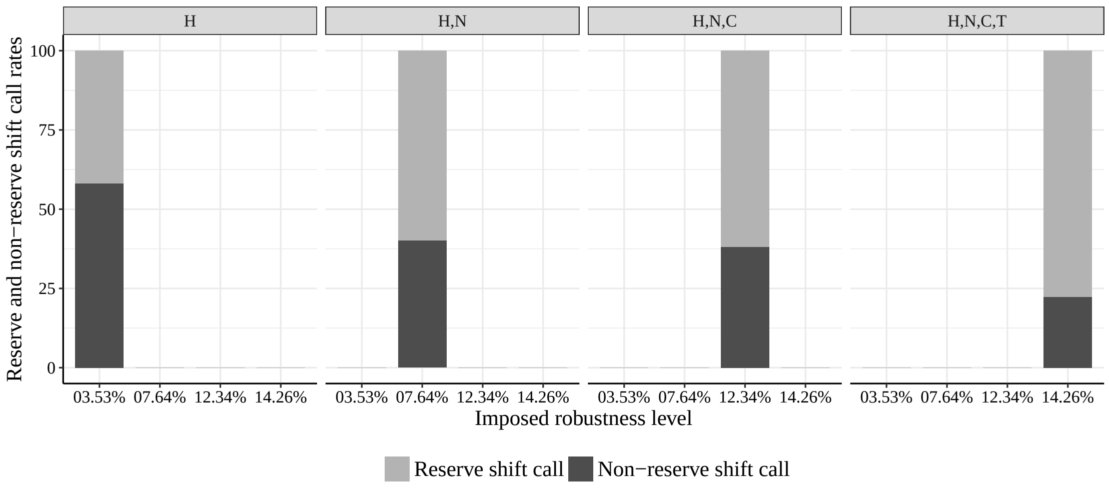


Figure 5.5 shows a drop of unexpected employee calls when the required robustness is increased. The percentage of disruptions solved using reserve shifts was 41.09%, 59.79%, 61.91% and 77.72% for robustness levels of 3.53%, 7.64%, 12.34% and 14.26%, respectively.

The results for problems with multi-skilled employees all follow a similar trend: increasing the robustness reduces the number of unexpected calls which necessitated changing a day off into a working shift. These results also confirm the validity of how robustness is quantified, as introduced in Section 5.4 since there is a clear correlation to the robustness defined as a constraint for each experiment. Overall, defining robustness constraints in a more detailed manner, per day and skill, generated better results compared against when they were only defined per day or as a general constraint. Moreover, assigning high-skilled employees to reserve shifts is better than low-skilled employees. This is simply due to the higher probability of high-skilled employees being able to assume the work of an absent employee.

Detailed computation times for the multi-skilled scenarios are reported in Table G.2. A trend similar to the single-skilled problem is observed: increasing the robustness level requires more computation time to generate the initial solutions. Compared to the single-skilled problems, more computation time was required for finding initial solutions for the multi-skilled problems. The results for the re-rostering phase demonstrate similar trends, however, the computation times are in general lower when higher robustness levels are considered.

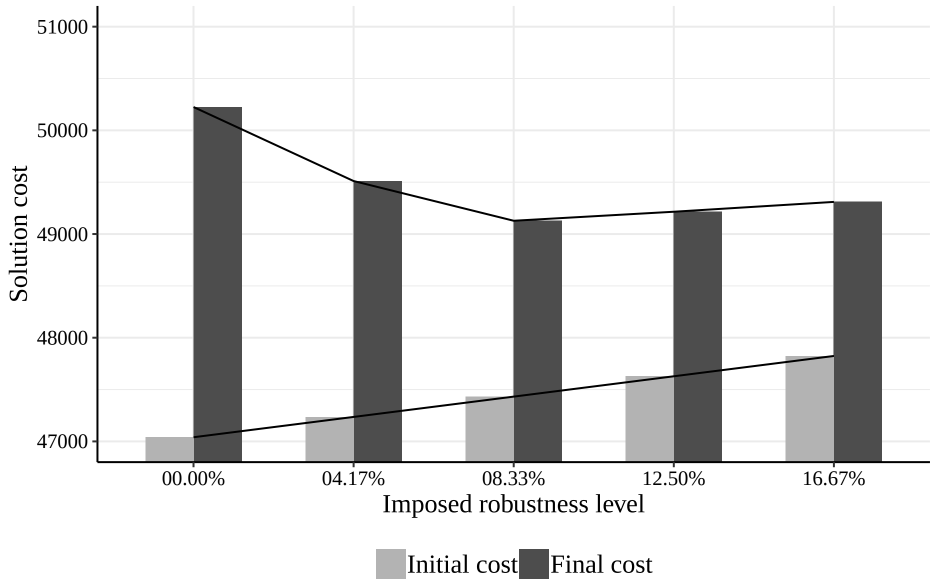
5.6.3 Impact on operational costs

The previous analysis ignored a crucial element of the objective function concerning the operational costs induced by the employees' wages associated with a roster. This section investigates the trade-off between managing the operational costs and robustness of a roster. As in Section 5.6.2, the impact on problems with both single- and multi-skilled employees is evaluated.

5.6.3.1 Single-skilled nurses

Figure 5.6 shows the initial cost of the baseline roster and the final cost after disruptions were solved, for different robustness levels specified as a general constraint. The difference between the initial and final costs is due to the additional wage costs incurred by calling employees to cover the absences.

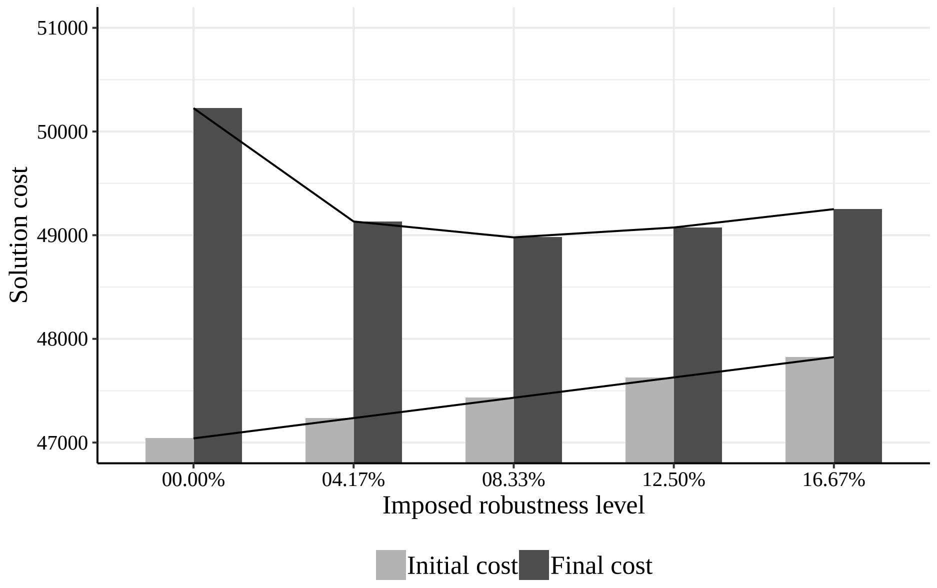
Figure 5.6: Comparison of general robustness levels and respective initial cost and final cost after repair.



Note that the higher the robustness level, the higher the initial cost. While the roster without enforced robustness has the lowest initial cost, it is ultimately associated with the highest final cost. The best result was obtained with a robustness level of 8.33%. Increasing the robustness level to 12.50% and 16.67% generated higher initial costs, while final costs were also higher than those obtained with a robustness level of 8.33%. The explanation is that more robustness was introduced than necessary.

Figure 5.7 presents the same robustness level evaluation as the experiment shown in Figure 5.6, however, instead of adding a general robustness constraint, the daily robustness constraint enforces the solver to evenly distribute the buffers over all days in the planning horizon.

Figure 5.7: Comparison of robustness levels per day and respective initial cost and final cost after repair.



As in the previous experiment, a robustness level of 8.33% generated the best results. However, the total cost reduction was higher compared to defining robustness as a general constraint (2.48% vs 2.19%). This observation confirms the results from Section 5.6.2.1 which showed fewer unexpected calls when a robustness constraint was added per day, rather than on a general level.

5.6.3.2 Multi-skilled nurses

Figure 5.8 compares the final costs for different robustness levels using the subsets of skills detailed in Section 5.6.2.2. The leftmost set of bars represents the final cost without any robustness in the roster and is used as a baseline for the comparison. Bars are not shown for a skill subset if the robustness level could not be attained. This is likely due to the limited number of available employees with those specific skills.

With a robustness level of 3.53%, the best results were obtained when only head nurses are assigned to a buffer. When the robustness level was increased to 7.64%, results were better when allowing head nurses and nurses to be assigned to reserve shifts. Such results are expected due to the greater number of disruptions that can be covered by high-skilled nurses.

Figure 5.8: Comparison of general robustness levels and respective costs.

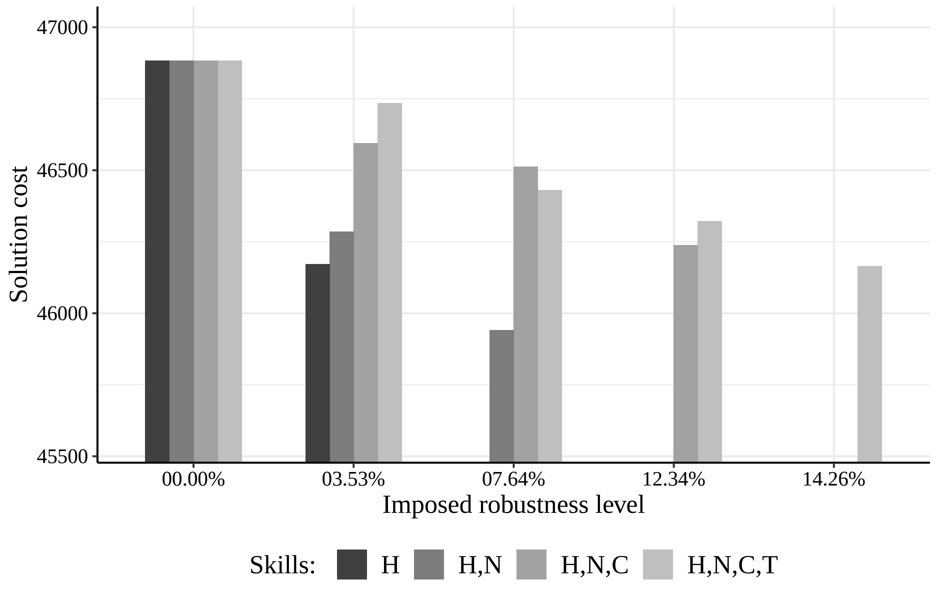


Figure 5.9 compares the final costs when robustness is enforced as a constraint per day. Similar to previous experiments when a general robustness level was defined, only allowing high-skilled employees to buffers resulted in lower final costs. For example, when enforcing a robustness level of 3.53% only with head nurses and nurses significantly better results can be observed compared to when caretakers and trainees were also assigned to reserve shifts.

Figure 5.10 compares final costs when the required robustness level constraints are specified per day and skill. Better results were obtained with robustness levels higher than or equal to 7.64%. The lowest final cost was achieved using employees with skills head nurse, nurse and caretaker and a robustness level of 12.34%. Despite an increase of the robustness level to 14.26% when employees with skill trainee are assigned to reserve shifts, this does not translate into a reduction of the final cost as trainees only can cover absences of other trainees. Since disruptions often require higher qualified employees, trainees have a low contribution as regards covering disruptions.

Figure 5.9: Comparison of robustness levels per day and respective costs.

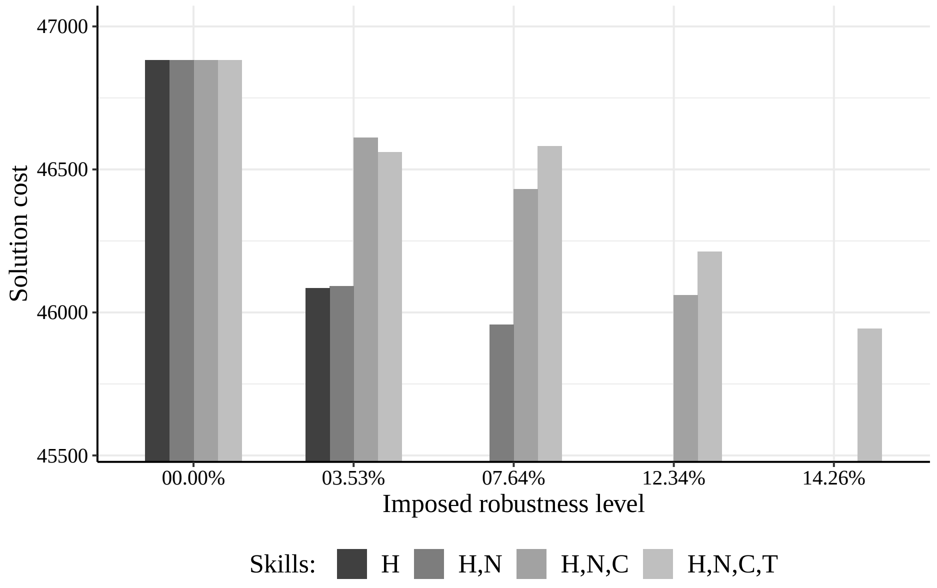
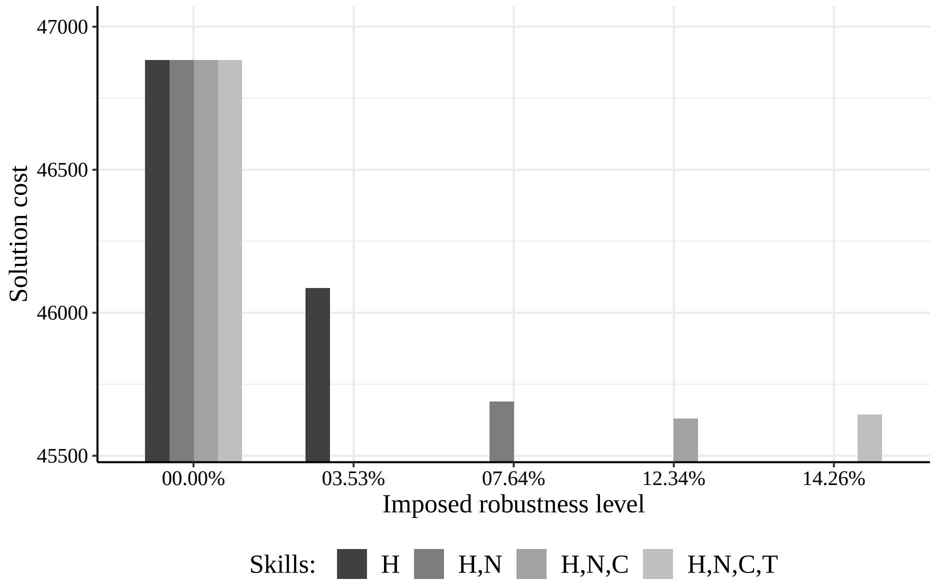


Figure 5.10: Comparison of robustness levels per day and skill, and respective costs.

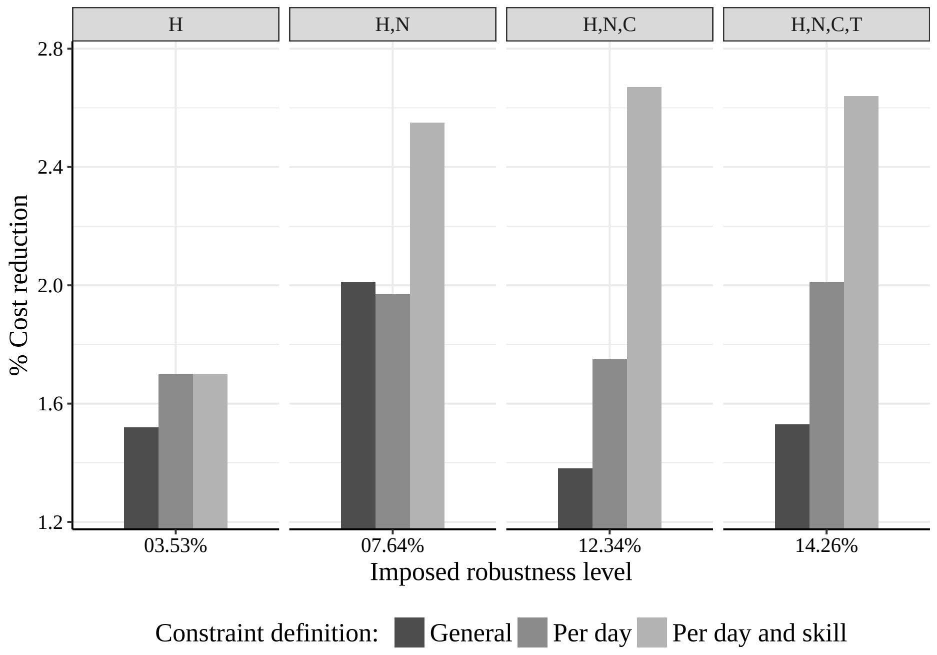


5.6.3.3 Comparison of robustness strategies

Figure 5.11 summarizes the performance of the different strategies used to increase robustness. The best results were obtained when constraints were specified per day and skill, while a general robustness constraint (averaged over all days and skills) resulted in the highest final costs. The primary reason for these results is that constraints enforcing robustness per day, or per day and skill generate a uniform distribution of the reserve shifts over the scheduling horizon, which best matches the distribution of disruptions. Employing a general constraint, on the other hand, may concentrate the reserve shifts on some days, leaving others without coverage. Note that for this study, statistics were only available regarding employee absenteeism in general, without details concerning the distribution of these disruptions. To improve the effectiveness of a proactive approach, organizations should observe historical data to determine which days or months have higher probabilities of absenteeism. Based on this data, a statistical distribution that better fits the real absenteeism rate could be utilized to predict the most critical days and thus increase the buffers on these days.

Moreover, specifying constraints considering employees' skills generated better results compared to incorporating daily robustness constraints since the solver tends to assign less qualified employees to reserve shifts who have comparably lower costs associated with them. When employees' skills are considered, higher qualified employees are assigned to reserve shifts. Since they also have a higher probability of being able to replace absent employees, this strategy performs better.

Figure 5.11: Comparison of robustness strategies for multi-skilled employees



5.7 Conclusions

The present study sought to remedy a shortcoming in the literature by introducing a general robustness metric for both single- and multi-skilled staff rostering problems. This metric provides a means of quantifying the robustness of a roster and thus an a priori estimate of how well it will accommodate unexpected employee shortages. In addition, an integer programming formulation was presented with the option of incorporating these alternative robustness strategies. The target robustness can be enforced as a general constraint, per day, or per day and skill when the problem concerns multi-skilled employees.

Enforcing relatively low robustness levels appeared sufficient to result in the lowest final cost for both single- and multi-skilled employees. Higher robustness levels, meanwhile, resulted in higher initial and final costs since too many employees were unnecessarily assigned to reserve shifts. Similar results were observed for multi-skilled employees, however, since employees have skills and can only substitute for others when they share the same skills, a higher robustness level is required compared to single-skilled problems.

Another important factor to consider when increasing the robustness of a roster is the distribution of the capacity buffers and reserve shifts. Ideally, these buffers should be concentrated on days with a higher probability of absenteeism. In this study, a Bernoulli distribution was employed to generate disruptions. The best results were obtained when robustness was enforced as constraints per day for single-skilled employees. Meanwhile, the best results for multi-skilled employees were obtained when the minimum robustness level was enforced per day and skill. Moreover, assigning higher qualified employees to reserve shifts generated better results than assigning, for example, trainee nurses. These results were to be expected. Despite the fact that higher qualified employees are associated with higher costs, they are also capable of replacing a far greater percentage of personnel shortages which occur. Our experiments demonstrated a strong correlation between the robustness metric and the computational results. Future research regarding staff rostering will benefit from utilizing the proposed metric in order to standardize the robustness quantification of a roster.

6 CONCLUSIONS

Bridging the gap between theory and practice is crucial for academic research to translate into meaningful results. This thesis investigated challenging real-world rostering scenarios and proposed solving methods such that they become acceptable in practice and can replace manual planners. A wide range of research questions regarding constraint analysis and various types of rosters such as non-cyclic, cyclic, rerostering and robust rostering were addressed. In addition to the academic domain, the present thesis also contributes a licensed rostering solver, used at HCPA, which positively impacts rostering organization, employee satisfaction and patient care. To render the developed solver acceptable in practice, there are four basic elements which must be considered: *(i)* the return on investment, *(ii)* generality, *(iii)* future maintenance of the solver and *(iv)* fairness between the scheduled employees.

The return on investment is achieved through the reduction of overtime and time effort required by planners to organize the rosters. Other elements that are more difficult to be quantified were not measured, such as better roster organization, reduction of mistakes and improvements of patient care. Another important feature is the generality of the solving method. Ideally, the solver should be general enough such that it can be deployed within various hospital units, contributing to the cost reduction. Furthermore, another important element of the solver is the possibility of accepting manual changes after the roster is generated. Such a feature enables a physician, for example, to swap the schedule of specific shifts with another due to last-minute unavailability. Despite the fact that these changes usually render worse solutions in terms of costs to the hospital compared against the original roster, they are commonly accepted by the managers because they improve employee satisfaction rates.

Information technology professionals working at HCPA typically hold a basic understanding of computer science and technology engineering. The solving method must therefore be as simple as possible without compromising the generation of good results in short computation time, enabling these professionals to provide future maintenance. The proposed method achieved this objective by using general-purpose solvers to solve IP models, thus facilitating the application of them in practice.

The majority of employees work on business days which implies that they do not work nights or weekends. However, healthcare professionals often have to work during these unpopular shifts because hospitals operate on a 24/7 basis. Such a situation directly impacts upon the social interaction between healthcare professionals and those that work during business days. Common problems concern childcare, interaction with the family and social life. Reducing such negative effects is therefore of fundamental importance. Note that to address this issue, the proposed solving method balances the working hours during non-business days by assigning employees an ideal number of hours specified according to their working contract and seniority.

Although multi-skilled employees frequently occur in the industry, existing academic literature has thus far only approached single-skilled rerostering problems. Moreover, in contrast to recent literature which is mainly based on heuristic methods, this research showed that large instances with up to 500 multi-skilled nurses can be solved to optimality using an exact method such as the one presented in Chapter 4. These results indicate that effective IP formulations using standalone solvers provide good alternatives for addressing rerostering problems and restoring feasibility after disruptions occur.

A metric for quantifying robustness concerning machine scheduling was proposed more than ten years ago. Despite the importance of knowing in advance the robustness of a roster, this subject has since been ignored in academic literature. This research represents a breakthrough on this front by proposing a metric for quantifying the robustness of a roster a priori, enabling an estimation of how well it will accommodate unexpected employee shortages. Appendices H, I and J provide three physician rosters, generated using the proposed metric with robustness levels of 0.00%, 2.64% and 7.78%. These robustness levels are obtained by assigning physicians to capacity buffers during business days. The first roster (Appendix H), without robustness, results in a total of 298 hours of overtime. The second roster (Appendix I), when physicians were assigned to capacity buffers during Early shifts, has 394 hours of overtime. The third roster (Appendix J) where physicians were assigned to capacity buffer for both Early and Late shifts, resulted in 496 hours of overtime. All three rosters were presented to the hospital managers. They decided to use the second option which provides a lower robustness level, but which also has a lower cost associated concerning paid overtime for the physicians.

These findings reinforce the applicability of the proposed metric in real-world scenarios. Moreover, it was interesting to see that the option selected by the hospital managers has a strong correlation with the conclusions of the robust rostering research (Chapter 5). That is, high robustness levels increase operational costs too much and, therefore, lower robustness levels are more likely to be implemented in practice.

Despite the simplicity of adapting existing non-cyclic IP models such that they can be used in cyclic rostering problems, this enabled constraints from practice to be included in challenging multi-skilled scenarios. It was shown that instances based on data provided by a research institute considering multi-skilled employees can still be solved to optimality employing the proposed model in reduced computation time. Moreover, state-of-the-art results were generated when considering instances from the literature. A series of previously proposed methods including heuristics, constraint programming and methods based on SAT solvers were all outperformed. Compared to the previous IP formulation which was incapable of finding feasible solutions in hours of computation time, the proposed method solved these instances in only a few seconds.

Results presented by this thesis also raised new research questions that should be approached in future work. A general model including constraints from both the nurse and physician rostering problems might be possible and enable the design of a more generic solving method which is capable of addressing multiple problems. Note that this proposal differs from several existing methods which are not only focused on single problem domains but are also restricted to one type of problem instance. Such case-specific methods are very restrictive and, therefore, difficult to implement in real-world scenarios. When we look to other problem domains such as vehicle routing, in the twelfth edition of the vehicle routing competition the solving methods will be evaluated employing different variants of the problem. Similarly, a solving method capable to address more variants of personnel rostering problems would be interesting for both the research community and the industry. Referring back to the general research question, therefore, can be concluded that future literature focusing on more general solving methods concerning personnel rostering would heighten the chances that academic progress actually translates into practical implementation.

REFERENCES

ABDELGHANY, K. F.; ABDELGHANY, A. F.; EKOLLU, G. An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, v. 185, n. 2, p. 825 – 848, 2008. ISSN 0377-2217.

AGUIRRE, A.; KERIN, A. *Shiftwork Practices 2005*. 2014. [Online; accessed 20-July-2018]. Available from Internet: <https://create.piktochart.com/output/2598161-absenteeism>.

BARTHOLDI, J. J. A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering. *Operations Research*, v. 29, p. 501–510, 06 1981.

BÄUMELT, Z.; DVOŘÁK, J.; ŠŮCHA, P.; HANZÁLEK, Z. A novel approach for nurse rerostering based on a parallel algorithm. *European Journal of Operational Research*, v. 251, n. 2, p. 624 – 639, 2016. ISSN 0377-2217.

BBC. *NHS 'not fit for 21st Century', says chief hospital inspector*. 2017. [Online; accessed 2-October-2017]. Available from Internet: <http://www.bbc.com/news/uk-41451162>.

BEAULIEU, H.; FERLAND, J. A.; GENDRON, B.; MICHELON, P. A mathematical programming approach for scheduling physicians in the emergency room. *Health Care Management Science*, v. 3, n. 3, p. 193–200, Jun 2000. ISSN 1572-9389.

BECKER, T.; STEENWEG, P. M.; WERNERS, B. Cyclic shift scheduling with on-call duties for emergency medical services. *Health Care Management Science*, Jul 2018. ISSN 1572-9389.

BERTSIMAS, D.; SIM, M. The price of robustness. *Operations Research*, v. 52, p. 35–53, 2004.

BILGIN, B.; DEMEESTER, P.; MISIR, M.; VANCROONENBURG, W.; VANDEN BERGHE, G.; WAUTERS, T. A hyper-heuristic combined with a greedy shuffle approach to the nurse rostering competition. *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT'10)*, 2010.

BRUNI, R.; DETTI, P. A flexible discrete optimization approach to the physician scheduling problem. *Operations Research for Health Care*, v. 3, n. 4, p. 191 – 199, 2014. ISSN 2211-6923.

BRUNNER, J. O.; BARD, J. F.; KOLISCH, R. Flexible shift scheduling of physicians. *Health Care Management Science*, v. 12, n. 3, p. 285–305, 2009. ISSN 1572-9389.

BRUNNER, J. O.; EDENHARTER, G. M. Long term staff scheduling of physicians with different experience levels in hospitals using column generation. *Health Care Management Science*, v. 14, n. 2, p. 189–202, 2011. ISSN 1572-9389.

BUREAU OF LABOR STATISTICS. *Absences from work of employed full-time wage and salary workers by occupation and industry*. 2017. [Online; accessed 06-June-2018]. Available from Internet: <https://www.bls.gov/cps/cpsaat47.htm>.

BURGGRAEVE, S.; BULL, S. H.; VANSTEENWEGEN, P.; LUSBY, R. M. Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research Part C: Emerging Technologies*, v. 77, n. Supplement C, p. 134 – 160, 2017. ISSN 0968-090X.

BURKE, E. K.; CURTOIS, T. *New computational results for nurse rostering benchmark instances*. [S.l.], 2011. [Online; accessed 7-July-2016]. Available from Internet: <http://www.cs.nott.ac.uk/~psztc/new_computational_results_for_nurse_rostering_benchmark_instances.pdf>.

BURKE, E. K.; CURTOIS, T.; POST, G.; QU, R.; VELTMAN, B. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, v. 188, n. 2, p. 330 – 341, 2008. ISSN 0377-2217.

BURKE, E. K.; DE CAUSMAECKER, P.; PETROVIC, S.; VANDEN BERGHE, G. Metaheuristics for handling time interval coverage constraints in nurse scheduling. *Applied Artificial Intelligence*, Taylor & Francis Group, v. 20, n. 9, p. 743–766, 2006.

BURKE, E. K.; DE CAUSMAECKER, P.; VANDEN BERGHE, G.; VAN LANDEGHEM, H. The state of the art of nurse rostering. *Journal of scheduling*, Kluwer Academic Publishers, v. 7, n. 6, p. 441–499, 2004.

BURKE, E. K.; LI, J.; QU, R. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, v. 203, n. 2, p. 484 – 493, 2010. ISSN 0377-2217.

CESCHIA, S.; DANG, N.; DE CAUSMAECKER, P.; HASPESLAGH, S.; SCHAERF, A. The second international nurse rostering competition. *Annals of Operations Research*, v. 274, n. 1, p. 171–186, Mar 2019. ISSN 1572-9338.

CLARK, A.; MOULE, P.; TOPPING, A.; SERPELL, M. Rescheduling nursing shifts: scoping the challenge and examining the potential of mathematical model based tools. *Journal of Nursing Management*, v. 23, n. 4, p. 411–420, 2015. ISSN 1365-2834.

CROCE, F. D.; SALASSA, F. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, v. 218, n. 1, p. 185–199, Jul 2014. ISSN 1572-9338.

DE BRUECKER, P.; VAN DEN BERGH, J.; BELIËN, J.; DEMEULEMEESTER, E. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, v. 243, n. 1, p. 1 – 16, 2015. ISSN 0377-2217.

DE CAUSMAECKER, P.; VANDEN BERGHE, G. A categorisation of nurse rostering problems. *Journal of Scheduling*, v. 14, n. 1, p. 3–16, Feb 2011. ISSN 1099-1425.

DONNELLY, L. *Nurses so overworked they are forced to leave patients to die alone, survey finds*. 2017. [Online; accessed 2-October-2017]. Available from Internet: <http://www.telegraph.co.uk/news/2017/09/29/nurses-overworked-forced-leave-patients-die-alone-survey-finds>.

DRAKE, R. G. The 'robust' roster: exploring the nurse rostering process. *Journal of advanced nursing*, Wiley Online Library, v. 70, n. 9, p. 2095–2106, 2014.

DÜCK, V.; IONESCU, L.; KLIEWER, N.; SUHL, L. Increasing stability of crew and aircraft schedules. *Transportation Research Part C: Emerging Technologies*, v. 20, n. 1, p. 47 – 61, 2012.

EDWARDS, P.; GREASLEY, K. *Absence from work*. 2010. [Online; accessed 06-June-2018]. Available from Internet: <https://www.eurofound.europa.eu/observatories/eurwork/comparative-information/absence-from-work>.

ERHARD, M.; SCHOENFELDER, J.; FÜGENER, A.; BRUNNER, J. O. State of the art in physician scheduling. *European Journal of Operational Research*, v. 265, n. 1, p. 1 – 18, 2018. ISSN 0377-2217.

ERKINGER, C.; MUSLIU, N. Personnel scheduling as satisfiability modulo theories. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. [S.l.: s.n.], 2017. p. 614–621.

ERNST, A.; JIANG, H.; KRISHNAMOORTHY, M.; SIER, D. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, v. 153, n. 1, p. 3–27, 2004.

FELICI, G.; GENTILE, C. A polyhedral approach for the staff rostering problem. *Manage. Sci.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 50, n. 3, p. 381–393, mar. 2004. ISSN 0025-1909.

FORBES. *The Causes And Costs Of Absenteeism In The Workplace*. 2013. [Online; accessed 16-July-2018]. Available from Internet: <https://www.forbes.com/sites/investopedia/2013/07/10/the-causes-and-costs-of-absenteeism-in-the-workplace>.

GOMES, R. A. M.; TOFFOLO, T. A. M.; SANTOS, H. G. Variable neighborhood search accelerated column generation for the nurse rostering problem. *Electronic Notes in Discrete Mathematics*, v. 58, p. 31 – 38, 2017. ISSN 1571-0653. 4th International Conference on Variable Neighborhood Search.

GOREN, S.; SABUNCUOGLU, I. Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions*, Taylor & Francis, v. 40, n. 1, p. 66–83, 2008.

GUNAWAN, A.; LAU, H. C. Master physician scheduling problem. *Journal of the Operational Research Society*, v. 64, n. 3, p. 410–425, 2013. ISSN 1476-9360.

GUO, M.; WU, S.; LI, B.; RONG, Y. Maximizing the efficiency of use of nurses under uncertain surgery durations: A case study. *Computers & Industrial Engineering*, v. 78, p. 313 – 319, 2014. ISSN 0360-8352.

HASPESLAGH, S.; DE CAUSMAECKER, P.; SCHAERF, A.; STØLEVIK, M. The first international nurse rostering competition 2010. *Annals of Operations Research*, v. 218, n. 1, p. 221–236, Jul 2014. ISSN 1572-9338.

HAZIR, Ö.; HAOUARI, M.; EREL, E. Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research*, v. 207, n. 2, p. 633 – 643, 2010. ISSN 0377-2217.

HERROELEN, W.; LEUS, R. The construction of stable project baseline schedules. *European Journal of Operational Research*, Elsevier, v. 156, n. 3, p. 550–565, 2004.

HERROELEN, W.; LEUS, R. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, Elsevier, v. 165, n. 2, p. 289–306, 2005.

HUELE, C. V.; VANHOUCKE, M. Analysis of the integration of the physician rostering problem and the surgery scheduling problem. *Journal of Medical Systems*, v. 38, n. 6, p. 43, 2014. ISSN 1573-689X.

INGELS, J.; MAENHOUT, B. The impact of reserve duties on the robustness of a personnel shift roster: An empirical investigation. *Computers & Operations Research*, v. 61, n. Supplement C, p. 153 – 169, 2015. ISSN 0305-0548.

INGELS, J.; MAENHOUT, B. Employee substitutability as a tool to improve the robustness in personnel scheduling. *OR Spectrum*, v. 39, n. 3, p. 623–658, Jul 2017. ISSN 1436-6304.

INGELS, J.; MAENHOUT, B. Optimised buffer allocation to construct stable personnel shift rosters. *Omega*, 2017. ISSN 0305-0483.

INGELS, J.; MAENHOUT, B. The impact of overtime as a time-based proactive scheduling and reactive allocation strategy on the robustness of a personnel shift roster. *Journal of Scheduling*, v. 21, n. 2, p. 143–165, Apr 2018.

IONESCU, L.; KLIEWER, N. Increasing flexibility of airline crew schedules. *Procedia-Social and Behavioral Sciences*, Elsevier, v. 20, p. 1019–1028, 2011.

KOCAKULAH, M. C.; KELLEY, A. G.; MITCHELL, K. M.; RUGGIERI, M. P. Absenteeism problems and costs: causes, effects and cures. *The International Business & Economics Research Journal (Online)*, The Clute Institute, v. 15, n. 3, p. 89, 2016.

LAMBRECHTS, O.; DEMEULEMEESTER, E.; HERROELEN, W. A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, v. 111, n. 2, p. 493 – 508, 2008. ISSN 0925-5273. Special Section on Sustainable Supply Chain.

LAMBRECHTS, O.; DEMEULEMEESTER, E.; HERROELEN, W. Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research*, v. 186, n. 1, p. 443–464, Jun 2011. ISSN 1572-9338.

LEGRAIN, A.; OMER, J.; ROSAT, S. A rotation-based branch-and-price approach for the nurse scheduling problem. *Mathematical Programming Computation*, Oct 2019. ISSN 1867-2957.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; STÜTZLE, T.; BIRATTARI, M. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43 – 58, 2016. ISSN 2214-7160.

LOUGEE-HEIMER, R. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, IBM, v. 47, n. 1, p. 57–66, 2003.

LUSBY, R.; DOHN, A.; RANGE, T. M.; LARSEN, J. A column generation-based heuristic for rostering with work patterns. *Journal of the Operational Research Society*, Springer, v. 63, n. 2, p. 261–277, 2012.

MAENHOUT, B.; VANHOUCKE, M. An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*, v. 38, n. 10, p. 1400 – 1411, 2011. ISSN 0305-0548.

MAENHOUT, B.; VANHOUCKE, M. Reconstructing nurse schedules: Computational insights in the problem size parameters. *Omega*, v. 41, n. 5, p. 903 – 918, 2013. ISSN 0305-0483.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097 – 1100, 1997. ISSN 0305-0548.

MOZ, M.; PATO, M. V. An integer multicommodity flow model applied to the rerostering of nurse schedules. *Annals of Operations Research*, v. 119, n. 1, p. 285–301, 2003. ISSN 1572-9338.

MOZ, M.; PATO, M. V. Solving the problem of rerostering nurse schedules with hard constraints: New multicommodity flow models. *Annals of Operations Research*, v. 128, n. 1, p. 179–197, Apr 2004. ISSN 1572-9338.

MOZ, M.; PATO, M. V. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, v. 34, n. 3, p. 667 – 691, 2007. ISSN 0305-0548. Logistics of Health Care ManagementPart Special Issue: Logistics of Health Care Management.

MUSLIU, N. Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research*, v. 2, n. 4, p. 309–326, 2006. ISSN 0973–1873.

MUSLIU, N.; GÄRTNER, J.; SLANY, W. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, v. 118, n. 1, p. 85 – 98, 2002. ISSN 0166-218X. Special Issue devoted to the ALIO-EURO Workshop on Applied Combinatorial Optimization.

MUSLIU, N.; SCHUTT, A.; STUCKEY, P. J. Solver independent rotating workforce scheduling. In: HOEVE, W.-J. van (Ed.). *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Cham: Springer International Publishing, 2018. p. 429–445. ISBN 978-3-319-93031-2.

PATO, M. V.; MOZ, M. Solving a bi-objective nurse rerostering problem by using a utopic pareto genetic heuristic. *Journal of Heuristics*, v. 14, n. 4, p. 359–374, 2008. ISSN 1572-9397.

PETROVIC, S.; VANDEN BERGHE, G. A comparison of two approaches to nurse rostering problems. *Annals of Operations Research*, v. 194, n. 1, p. 365–384, 2012. ISSN 1572-9338.

PUENTE, J.; GÓMEZ, A.; FERNÁNDEZ, I.; PRIORE, P. Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. *Computers & Industrial Engineering*, v. 56, n. 4, p. 1232 – 1242, 2009. ISSN 0360-8352.

ROCHA, M.; OLIVEIRA, J. F.; CARRAVILLA, M. A. Cyclic staff scheduling: optimization models for some real-life problems. *Journal of Scheduling*, v. 16, n. 2, p. 231–242, Apr 2013. ISSN 1099-1425.

RÖMER, M.; MELLOULI, T. A direct MILP approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. *Proceedings of the 11th International Confenference on Practice and Theory of Automated Timetabling (PATAT-2016)*, p. 549 – 551, 2016.

ROUSSEAU, L.-M.; PESANT, G.; GENDREAU, M. A general approach to the physician rostering problem. *Annals of Operations Research*, v. 115, n. 1, p. 193–205, 2002. ISSN 1572-9338.

SAFAK, Ö.; GÜREL, S.; AKTÜRK, M. S. Integrated aircraft-path assignment and robust schedule design with cruise speed control. *Computers & Operations Research*, v. 84, n. Supplement C, p. 127 – 145, 2017. ISSN 0305-0548.

SALASSA, F.; VANDEN BERGHE, G. A stepping horizon view on nurse rostering. *Proceedings of the 9th International Confenference on Practice and Theory of Automated Timetabling (PATAT-2012)*, p. 161 – 174, 2012.

SANCHOTENE, T. C.; BURIOL, L. S. *Abordagem Heurística para Solução do Problema de Alocação de Médicos do HCPA*. 2018. [Online; accessed 10-October-2018]. Available from Internet: <https://www.lume.ufrgs.br/bitstream/handle/10183/175530/Poster_52197.pdf>.

SANTOS, H. G.; TOFFOLO, T. A. M.; GOMES, R. A. M.; RIBAS, S. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, Springer US, p. 1–27, 2014. ISSN 0254-5330.

SD WORX. *2012: A record year for absenteeism*. 2013. Available from Internet: <https://www.sdworx.be/nl-be/sd-worx-r-d/publicaties/persberichten/2013-03-27-recordjaar-ziekteverzuim>.

SMET, P. Constraint reformulation for nurse rostering problems. *Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling*, p. 69 – 80, 2018.

SMET, P.; SALASSA, F.; VANDEN BERGHE, G. Local and global constraint consistency in personnel rostering. *International Transactions in Operational Research*, v. 24, n. 5, p. 1099–1117, 2017.

STOLLETZ, R.; BRUNNER, J. O. Fair optimization of fortnightly physician schedules with flexible shifts. *European Journal of Operational Research*, v. 219, n. 3, p. 622 – 629, 2012. ISSN 0377-2217. Feature Clusters.

THE SOCIETY FOR HUMAN RESOURCE MANAGEMENT. *Total Financial Impact of Employee Absences*. 2016. [Online; accessed 03-July-2018]. Available from Internet: <https://www.shrm.org/hr-today/trends-and-forecasting/research-and-surveys/Documents/Total%20Financial%20Impact%20of%20Employee%20Absences%20Report.pdf>.

TRISKA, M.; MUSLIU, N. A constraint programming application for rotating workforce scheduling. In: ____. *Developing Concepts in Applied Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 83–88. ISBN 978-3-642-21332-8.

UNITED NATIONS. *World Population Ageing 2015*. 2015. [Online; 1-October-2017]. Available from Internet: <http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2015_Report.pdf>.

VALOUXIS, C.; GOGOS, C.; GOULAS, G.; ALEFRAGIS, P.; HOUSOS, E. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, v. 219, n. 2, p. 425 – 433, 2012. ISSN 0377-2217.

VAN DEN BERGH, J.; BELIËN, J.; DE BRUECKER, P.; DEMEULEMEESTER, E.; DE BOECK, L. Personnel scheduling: A literature review. *European Journal of Operational Research*, v. 226, n. 3, p. 367–385, 2013. ISSN 0377-2217.

WICKERT, T. I.; SARTORI, C.; BURIOL, L. S. *A Fix-and-Optimize VNS Algorithm Applied to the Nurse Rostering Problem*. IRIDIA, Institut de Recherches Interdisciplinaires et de D'eveloppements en Intelligence Artificielle, 2016. [Online; accessed 15-October-2018]. Available from Internet: <http://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2016-007.pdf>.

WICKERT, T. I.; SMET, P.; Vanden Berghe, G. The nurse rerostering problem: Strategies for reconstructing disrupted schedules. *Computers & Operations Research*, v. 104, p. 319 – 337, 2019. ISSN 0305-0548.

WILLIAMS, J. C.; LAMBERT, S.; KESAVAN, S. *How The Gap Used an App to Give Workers More Control Over Their Schedules*. 2017. URL: <https://hbr.org/2017/12/how-the-gap-used-an-app-to-give-workers-more-control-over-their-schedules> [Online; accessed 25-April-2018].

ZHENG, Z.; LIU, X.; GONG, X. A simple randomized variable neighbourhood search for nurse rostering. *Computers & Industrial Engineering*, v. 110, n. Supplement C, p. 165 – 174, 2017. ISSN 0360-8352.

APPENDIX A — PHYSICIAN ROSTER - APRIL 2019

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Name	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue
P1		N[2]			N[2]				N[2]				E[2]L[2]	N[2]		N[2]							N[2]			N[2]				N[2]
P2			L[1]	N[1]		E[1]L[1]	E[1]L[1]			L[1]	N[1]		N[1]					N[1]						L[1]	N[1]					
P3			E[1]	E[1]	E[1]		N[1]			E[1]	E[1]	E[1]					E[1]	E[1]	N[1]		E[1]L[1]			E[1]	E[1]	E[1]				
P4				L[1]				N[1]						N[1]	N[1]			L[1]				N[1]			L[1]		E[1]L[1]	E[1]L[1]	N[1]	
P5	N[2]			L[2]		E[2]L[2]		N[2]			L[2]				N[2]				E[2]L[2]	N[2]		N[2]			L[2]					
P6			N[2]		E[2]			E[2]		N[2]		E[2]		E[2]L[2]	E[2]		N[2]					E[2]		N[2]		E[2]	N[2]	N[2]		
P7	N[2]			E[2]		E[2]L[2]	N[2]	N[2]			E[2]																		N[2]	
P8		N[5]							N[5]											E[5]L[5]	N[5]		N[5]							
P9		L[2]		N[2]					L[2]							L[2]		N[2]		E[2]L[2]	N[2]		L[2]		N[2]					
P10		N[2]					E[2]L[2]		N[2]		N[2]					N[2]			E[2]L[2]	N[2]			N[2]							N[2]
P11			N[1]						L[1]	N[1]			E[1]L[1]				N[1]						L[1]	N[1]			N[1]	N[1]		L[1]
P12	L[1]					E[1]L[1]	E[1]L[1]	L[1]	N[1]							N[1]						L[1]	N[1]					N[1]		N[1]
P13			N[2]						E[2]	N[2]			E[2]L[2]	E[2]L[2]		E[2]	N[2]						E[2]	N[2]				N[2]		
P14	N[5]					E[5]L[5]	N[5]															N[5]								
P15		N[1]							N[1]		L[1]		E[1]L[1]		L[1]	N[1]			E[1]L[1]	N[1]									L[1]	N[1]
P16		N[3]		L[3]		E[3]L[3]	N[3]		N[3]							N[3]		L[3]		E[3]L[3]	N[3]		N[3]		L[3]					N[3]
P17			L[2]		N[1]	N[1]	N[1]					N[1]		E[1]L[1]			L[1]													
P18		E[1]		N[1]							N[1]							N[1]	N[1]		E[1]L[1]		E[1]		N[1]		N[1]			E[1]
P19		L[3]		N[3]		N[3]					N[3]					L[3]		N[3]							N[3]		E[3]L[3]	E[3]L[3]		L[3]
P20	L[3]		N[3]				E[3]L[3]	L[3]		N[3]					L[3]		N[3]		E[3]L[3]	N[3]				N[3]					L[3]	
P21	N[4]				L[4]	E[4]L[4]				L[4]		E[4]	E[4]L[4]	N[4]	N[4]		L[4]									E[4]			N[4]	
P22	E[1]L[4]		N[4]					E[1]L[4]		N[4]		L[4]		E[1]L[1]	E[1]L[4]		N[4]		E[4]L[4]	N[4]		E[1]L[4]		N[4]					E[1]L[4]	
P23		E[4]		N[4]		N[4]	N[4]		E[4]		N[4]					E[4]							E[4]		N[4]			E[4]L[4]		E[4]
P24					E[3]			N[3]				E[3]	E[3]L[3]	E[3]L[3]	N[3]							N[3]				E[3]	N[3]		N[3]	
P25			N[1]		L[1]					N[1]		L[1]	N[1]	N[1]			N[1]							N[1]		L[1]	E[1]L[1]	E[1]L[1]		
P26	L[5]			N[5]							N[5]		E[5]L[5]	E[5]L[5]				N[5]				L[5]			N[5]		N[5]		L[5]	
P27				L[4]					N[4]		L[4]					N[4]		L[4]	N[4]		E[4]L[4]				L[4]	L[4]	N[4]			N[4]
P28	E[4]	N[4]		E[4]			E[4]L[4]	E[4]			E[4]		N[4]		E[4]			E[4]				E[4]	N[4]		E[4]				E[4]	
P29	N[1]					N[1]		N[1]							N[1]					E[1]L[1]	N[1]	N[1]				N[1]			N[1]	
P30			E[2]		N[2]					E[2]		N[2]					E[2]		N[2]		E[2]L[2]			E[2]		N[2]	N[2]			
P31			E[3]	E[3]	L[3]						N[2]	N[2]		N[3]				N[2]		E[2]L[2]	N[2]				E[2]	L[3]				
P32			L[3]		N[3]					L[3]		N[3]	N[3]				L[3]		N[3]		E[3]L[3]			L[3]		N[3]				
P33															L[5]	N[5]			N[5]		E[5]L[5]		L[5]	E[3]						N[5]
P34			L[5]					N[5]		L[5]					N[5]		L[5]		E[5]L[5]	N[5]				L[5]				E[5]L[5]	N[5]	
P35			E[6]L[6]				N[6]	N[6]		E[6]L[6]		N[6]					E[6]L[6]		E[6]L[6]		E[6]L[6]			E[6]L[6]		N[6]				
P36			E[4]		N[4]			N[4]		E[4]				E[4]L[4]						E[4]L[4]	N[4]	N[4]		E[4]		N[4]				
P37		L[6]	N[5]		L[5]	N[5]			L[6]	N[5]		L[5]				L[6]	N[5]						L[6]	N[5]		L[5]	E[5]L[5]	N[5]		L[6]
P38				L[5]	N[5]		E[5]L[5]				L[5]	N[5]	N[5]	N[5]				L[5]							L[5]	N[5]				
P39	N[6]		N[6]		L[6]							L[6]			N[6]				N[6]	N[6]		N[6]				L[6]		E[6]L[6]	N[6]	
P40		L[4]	L[4]		E[4]				L[4]			N[4]				L[4]	E[4]	N[4]					L[4]	L[4]			E[4]L[4]	N[4]		L[4]
P41		L[5]		N[2]			E[2]L[2]	L[5]	L[5]				N[2]	N[2]		L[5]						N[2]			N[2]					L[5]
P42	L[2]				N[1]			L[2]				N[1]	N[2]		L[2]							L[2]	N[1]				E[2]L[2]	E[2]L[2]	L[2]	
P43	N[3]					N[2]			L[3]		L[3]	L[3]			N[2]							L[3]	L[3]				E[2]L[2]	N[3]	N[2]	
P44	L[6]			N[6]		E[6]L[6]	E[6]L[6]	L[6]			N[6]		N[6]		L[6]			N[6]				L[6]							L[6]	
P45	N[1]	N[1]			L[2]	N[2]	N[2]			L[2]		L[2]					L[2]							L[2]		N[1]		E[2]L[2]		
P46				E[6]L[6]		N[6]			N[6]		E[6]L[6]							E[6]L[6]		E[6]L[6]	N[6]		N[6]		E[6]L[6]					N[6]
P47		E[6]			E[6]				E[6]	N[6]		E[6]		N[6]		E[6]	N[6]						E[6]	N[6]		E[6]	E[6]L[6]	N[6]		E[6]
P48	E[6]	N[6]			N[6]			E[6]					E[6]L[6]	E[6]L[6]	E[6]	N[6]						E[6]			N[6]		N[6]		E[6]	

E=Early, L=Late, N=Night; 1=Area 1, 2=Area 2, 3=Area 3, 4=Area 4, 5=Area 5, 6=Area 6

APPENDIX B — PHYSICIAN ROSTER - MAY 2019

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Name	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri
P1			N[2]				N[2]					E[2]L[2]		N[2]			N[2]				N[2]				N[2]	N[2]		N[2]			
P2		N[1]			E[1]L[1]				N[1]		N[1]																			N[1]	
P3	N[1]		E[1]					E[1]	E[1]						E[5]	E[5]	E[5]		E[1]L[1]			E[5]	E[5]	E[5]	N[1]				E[5]	E[5]	E[5]
P4											N[1]		N[1]							N[1]					E[1]L[1]	N[1]	N[1]				
P5	E[2]L[2]																		E[2]L[2]	N[2]							N[2]			L[2]	
P6	E[2]L[2]		E[4]	E[2]L[2]	E[2]L[2]	E[4]		N[2]		E[4]			E[4]		N[2]		E[2]			E[2]		N[2]		E[2]			E[4]		N[2]		E[4]
P7	N[2]			N[2]		N[2]			E[4]			E[2]L[2]	N[2]			E[2]				N[2]			E[2]				N[2]			E[4]	
P8							N[5]											E[5]L[5]	N[5]		N[5]							N[5]			
P9		N[2]					L[2]		N[2]		E[2]L[2]	N[2]		L[2]		N[2]							N[2]							N[2]	
P10		N[2]					N[2]							N[2]		N[2]		E[2]L[2]	N[2]		N[2]				E[2]L[2]			N[2]			
P11							L[1]	N[1]			E[1]L[1]	N[1]																	N[1]		
P12	E[1]L[1]					L[3]	N[1]					E[1]L[1]	L[3]	N[1]				N[1]		L[3]	N[1]						L[3]	N[1]			
P13				E[2]L[2]	N[2]		E[4]	N[2]						E[4]	N[2]						E[2]	N[2]			N[2]			E[4]	N[2]		
P14																				N[5]						E[5]L[5]	N[5]				
P15	E[1]L[1]	L[1]				L[2]	N[1]		L[2]				L[2]	N[1]				E[1]L[1]	N[1]		N[1]							N[1]			
P16		L[3]		E[3]L[3]	E[3]L[3]		N[3]		L[3]		N[3]	N[3]		N[3]		L[3]					N[3]		L[3]	L[3]				N[3]		L[3]	
P17			N[1]							N[1]					N[1]			N[1]	N[1]			N[1]		N[1]		E[1]L[1]					N[1]
P18		N[1]		N[1]	N[1]				N[1]					E[5]		N[1]					E[5]		N[1]			E[1]L[1]		E[5]		N[1]	
P19	N[3]	N[3]					L[3]		N[3]					L[3]		N[3]			E[3]L[3]		L[3]	L[3]	N[3]		N[3]			L[3]		N[3]	
P20				N[3]	N[3]			N[3]			E[3]L[3]	E[3]L[3]			N[3]		L[3]					N[3]							N[3]		L[3]
P21	N[4]		N[4]			N[4]		L[4]			E[4]L[4]	N[4]	N[4]				E[4]			E[4]				E[4]			N[4]				
P22	N[1]		L[4]	N[1]		E[1]L[4]		N[4]		L[4]	E[1]L[1]		E[5]L[4]		N[4]					E[5]L[4]	L[4]	N[4]		L[4]			E[5]L[4]		N[4]		L[4]
P23		N[4]		E[4]L[4]					N[4]							E[4]	N[4]		N[4]		E[4]		E[4]	N[4]							
P24			L[3]			N[3]				L[3]			N[3]					E[3]L[3]	N[3]	N[3]					E[3]L[3]	N[3]	N[3]				
P25										L[5]					N[1]		L[5]	E[1]L[1]	E[1]L[1]			N[1]		L[5]	N[1]				N[1]		L[5]
P26	N[5]	N[5]		N[5]		L[5]						E[5]L[5]	L[5]			N[5]				L[5]			N[5]				L[5]			N[5]	
P27	E[4]L[4]	L[4]					N[4]		L[4]			E[4]L[4]				L[4]	L[4]	N[4]			N[4]		L[4]					N[4]		L[4]	
P28		E[4]			E[4]L[4]					N[4]	N[4]			N[4]																	
P29				E[1]L[1]	N[1]	N[1]				N[1]			N[1]							N[1]					E[1]L[1]	N[1]	N[1]				
P30	N[2]		N[2]							N[2]					E[2]		N[2]	N[2]				E[2]		N[2]		E[2]L[2]					N[2]
P31			N[1]					N[1]			E[2]L[2]	E[1]L[1]						N[2]	N[2]				N[1]	N[1]							N[1]
P32	E[3]L[3]		N[3]					L[3]		N[3]					L[3]		N[3]	N[3]						N[3]		E[3]L[3]			L[3]		N[3]
P33			N[5]				L[5]	N[5]						L[5]	L[5]	L[5]			E[5]L[5]		L[5]	N[5]			N[5]	N[5]		L[5]	L[5]		
P34					E[5]L[5]	N[5]		L[5]			N[5]		N[5]																		
P35								E[6]L[6]		L[6]	N[6]	N[6]			E[6]L[6]		N[6]			E[6]		E[6]L[6]		N[6]		E[6]L[6]	E[6]		E[6]L[6]		N[6]
P36				N[4]	N[4]			E[4]							E[4]					N[4]		E[4]			E[4]L[4]	E[4]L[4]			E[4]	N[4]	N[4]
P37			L[5]	E[5]L[5]	N[5]		L[6]		N[5]		E[5]L[5]	N[5]		L[6]		L[6]	N[5]				L[6]		L[6]	N[5]				L[6]	N[5]		
P38	E[5]L[5]	L[5]							L[5]	N[5]				N[5]	N[5]			N[5]					L[5]		E[5]L[5]					L[5]	N[5]
P39			L[6]		E[2]L[2]	N[6]				N[6]			N[6]				L[6]	N[6]		N[6]				L[6]	E[6]L[6]	N[2]	N[6]				L[6]
P40							L[4]							L[4]	L[4]	N[4]		E[4]L[4]	E[4]L[4]			L[4]	N[4]		N[4]	N[4]		L[4]	L[4]		
P41						N[2]		L[2]	N[2]		N[2]	N[2]						E[2]L[2]	E[2]L[2]			L[5]	N[2]							N[2]	
P42				N[2]	N[2]					N[2]			N[2]		L[2]	N[1]	N[1]								E[2]L[2]	E[2]L[2]					N[2]
P43	N[6]			N[6]		L[6]							L[6]			N[6]		E[6]L[6]	E[6]L[6]	L[6]			N[6]				L[6]			N[6]	
P44		E[6]		E[1]L[1]	E[1]L[1]	N[1]			E[6]	L[2]	N[2]	N[1]				E[6]	N[1]							N[2]						E[6]	
P45																			N[6]		N[6]		E[6]			N[6]		N[6]		L[6]	
P46		L[6]	E[6]	E[6]L[6]	N[6]		E[6]		L[6]	E[6]				E[6]	N[6]		E[6]				E[6]	N[6]		E[6]	N[6]			E[6]	N[6]		E[6]
P47	E[6]L[6]	N[6]	N[6]		E[6]L[6]	E[6]	N[6]	N[6]	N[6]		E[6]L[6]	E[6]L[6]	E[6]	N[6]																	

E=Early, L=Late, N=Night; 1=Area 1, 2=Area 2, 3=Area 3, 4=Area 4, 5=Area 5, 6=Area 6

APPENDIX C — NURSE REROSTERING - GENERAL INTEGER PROGRAMMING FORMULATION FOR THE NURSE ROSTERING PROBLEM

Table C.1 provides the sets, decision and auxiliary variables employed in the formulation. The objective function minimizes the cost associated with the violation of the soft constraints.

Table C.1: Indices, sets, decision and auxiliary variables employed in the problem formulation.

Symbol	Definition
Input Data	
$n \in N$	n is the index of the nurse, and N is the set of nurses;
$d \in D$	d is the index of the day, and D is the set of days;
$s \in S$	s is the index of the shift, and $S = \{1,2,3,4\}$ is the set of shifts, where 1 corresponds to Early, 2 to Day, 3 to Late and 4 to Night;
$k \in K$	k is the index of the skill, and $K = \{1,2,3,4\}$ is the set of skills, where 1 corresponds to HeadNurse, 2 to Nurse, 3 to Caretaker and 4 to Trainee;
$(n,k) \in \tilde{K}$	set containing the pairs of forbbiden skill k of nurse n;
$r_{dsk} \in \mathbb{N}_0$	number of required nurses on day d, shift s, having skill k;
$(s1,s2) \in \hat{S}$	contains the pairs of invalid shift sequences, for example, $(4,1) \in \hat{S}$ means that a Night shift cannot be followed by an Early shift;
T^w	set of patterns $T^w = \{T^w_t : t \in \{1,2,\dots,p^w\}\}$, where p^w is the minimum number of consecutive working days minus one. T^w_t is a binary vector of dimension $t+2$, with one zero in the first position and one zero in the last position, being t the number of ones that appear in vector T^w_t. For example, considering 4 as the minimum number of working days, the patterns to search are $T^w = \{T^w_1 = (0,1,0), T^w_2 = (0,1,1,0), T^w_3 = (0,1,1,1,0)\}$. If the first pattern is found in the schedule, it represents three violations, the second pattern two violations, and the third pattern a single violation.
T^r	follows the same idea of T^w, and represents a set of patterns $T^r = \{T^r_t : t \in \{1,2,\dots,p^r\}\}$, where p^r is the minimum number of consecutive days off minus one.

T^s	follows the same idea of T^w, and represents a set of patterns $T^s = \{T^s_{t_s} : t_s \in \{1,2,\ldots,p^s\}\}$, where p^s is the minimum number of consecutive working days minus one at shift s.
$w \in W$	w is a Saturday index and W the set of all Saturdays indexes;
$M_h \in \{5,6\}$	set of maximum working days every 7 days. $M_1 = 5$ and $M_2 = 6$ depending of the nurses' contract;
α^1_{dsk}	preferred number of nurses for day d, shift s, skill k;
β^i_n	limit of soft constraint $2,...,5$ and $10,...,12$ for nurse n, that is, minimum/maximum consecutive working days, minimum/maximum consecutive days off, minimum/maximum number of assignments over the scheduling period and total working weekends;
γ^i_s	limit of soft constraint 6 and 7 for shift s, that is, minimum/maximum consecutive assignments to the same shift;
ω^i	weight for violating the lower and/or upper limits of soft constraint i.

Decision Variables

$x_{ndsk} \in \{0,1\}$	1 if nurse n is allocated on day d, shift s with skill k, 0 otherwise;
$y_{nw} \in \{0,1\}$	1 if nurse n works at weekend w, 0 otherwise.

Auxiliary Variables

$a^1_{dsk} \in \mathbb{N}_0$	preferred number of nurses violations for day d, shift s, skill k;
$b^i_{ndt} \in \mathbb{N}_0$	minimum number of consecutive working days and days off violations, $i \in 2,4$ for nurse n on day d, pattern t;
$c^i_{nd} \in \mathbb{N}_0$	maximum number of consecutive working days and days off violations, $i \in 3,5$ for nurse n on day d;
$e^6_{ndst} \in \mathbb{N}_0$	minimum number of consecutive assignments to the same shift violations, for nurse n on day d, shift s, pattern t;
$f^7_{nds} \in \mathbb{N}_0$	maximum number of consecutive assignments to the same shift violations, for nurse n on day d, shift s;
$g^8_{nds} \in \mathbb{N}_0$	number of nurse's undesired working day/shift violations, for nurse n on day d, shift s;
$h^9_{nw} \in \mathbb{N}_0$	number of complete weekends violations, for nurse n on weekend w;

$j_n^i \in \mathbb{N}_0$	minimum/maximum number of assignments over the scheduling period violations, maximum number of worked weekends violations, $i \in \{10, 11, 12\}$ for nurse n.

Constant

C	constant with value 10.

Minimize:

$$\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} a^1_{dsk}\omega^1 + \sum_{n\in N}\sum_{d\in D}\sum_{t\in T_t}\sum_{i\in\{2,4\}} b^i_{ndt}\omega^i + \sum_{n\in N}\sum_{d\in D}\sum_{i\in\{3,5\}} c^i_{nd}\omega^i + \sum_{n\in N}\sum_{d\in D}\sum_{s\in S}\sum_{t\in T_t} e^6_{ndst}\omega^6 + \sum_{n\in N}\sum_{d\in D}\sum_{s\in S} f^7_{nds}\omega^7 + \sum_{n\in N}\sum_{d\in D}\sum_{s\in S} g^8_{nds}\omega^8 + \sum_{n\in N}\sum_{w\in W} h^9_{nw}\omega^9 + \sum_{n\in N}\sum_{i\in\{10,11,12\}} j^i_n\omega^i \tag{C.1}$$

Subject to:

$$\sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 1 \qquad \forall n\in N, d\in D \tag{C.2}$$

$$\sum_{n\in N} x_{ndsk} \geq r_{dsk} \qquad \forall d\in D, s\in S, k\in K \tag{C.3}$$

$$\sum_{k\in K}(x_{nds1k} + x_{n(d+1)s2k}) \leq 1 \qquad \forall n\in N, d\in D\setminus\{|D|\}, (s1,s2)\in\hat{S} \tag{C.4}$$

$$\sum_{d\in D}\sum_{s\in S} x_{ndsk} = 0 \qquad \forall (n,k)\in\tilde{K} \tag{C.5}$$

$$\sum_{d'=d}^{6+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} \leq M_h \qquad \forall n\in N, d\in D \tag{C.6}$$

$$\sum_{n\in N} x_{ndsk} + a^1_{dsk} \geq \alpha^1_{dsk} \qquad \forall d\in D, s\in S, k\in K \tag{C.7}$$

$$S1_{ndt} + b^2_{ndt} \geq \beta^2_n \qquad \forall n\in N, t\in\{1,2,\dots p^w\}, d\in\{1,2,\dots,|D|-(t+2)\} \tag{C.8}$$

$$S1_{ndt} = \sum_{d'=d}^{t+d+1}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} + \sum_{d'\in\{d,\ t+d+1\}}\sum_{s\in S}\sum_{k\in K} x_{nd'sk}C + \sum_{d'=d+1}^{t+d}(1-\sum_{s\in S}\sum_{k\in K} x_{nd'sk})C \tag{C.9}$$

$$\sum_{d'=d}^{\beta^3_n+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} - c^3_{nd} \leq \beta^3_n \qquad \forall n\in N, d\in\{1,\dots,|D|-\beta^3_n\} \tag{C.10}$$

$$S2_{ndt} + b^4_{ndt} \geq \beta^4_n \qquad \forall n\in N, t\in\{1,2,\dots p^r\}, d\in\{1,2,\dots,|D|-(t+2)\} \tag{C.11}$$

$$S2_{ndt} = \sum_{d'=d}^{t+d+1} (1 - \sum_{s \in S} \sum_{k \in K} x_{nd'sk}) + \sum_{d' \in \{d, t+d+1\}} (1 - \sum_{s \in S} \sum_{k \in K} x_{nd'sk})C + \sum_{d'=d+1}^{t+d} \sum_{s \in S} \sum_{k \in K} x_{nd'sk}C \quad (C.12)$$

$$\sum_{d'=d}^{\beta_n^5+d} (1 - \sum_{s \in S} \sum_{k \in K} x_{nd'sk}) - c_{nd}^5 \leq \beta_n^5 \qquad \forall n \in N, d \in \{1, \dots, |D| - \beta_n^5\} \quad (C.13)$$

$$S3_{ndst} + e_{ndst}^6 \geq \gamma_s^6 \qquad \forall n \in N, s \in S, t_s \in \{1, 2, \dots p^s\}, d \in \{1, 2, \dots, |D| - (t_s + 2)\} \quad (C.14)$$

$$S3_{ndst} = \sum_{d'=d}^{t_s+d+1} \sum_{k \in K} x_{nd'sk} + \sum_{d' \in \{d,\ t_s+d+1\}} \sum_{k \in K} x_{nd'sk}C + \sum_{d'=d+1}^{t_s+d} (1 - \sum_{k \in K} x_{nd'sk})C \quad (C.15)$$

$$\sum_{d'=d}^{|\gamma_s^7|+d} \sum_{k \in K} x_{nd'sk} - f_{nds}^7 \leq \gamma_s^7 \qquad \forall n \in N, s \in S, d \in \{1, \dots, |D| - \gamma_s^7\} \quad (C.16)$$

$$g_{nds}^8 - \sum_{k \in K} x_{ndsk} = 0 \qquad \forall (n, d, s) \in U \quad (C.17)$$

$$\sum_{s \in S} \sum_{k \in K} (x_{nwsk} + x_{n(w+1)sk}) \leq 2y_{nw} \qquad \forall n \in N, w \in W \quad (C.18)$$

$$\sum_{s \in S} \sum_{k \in K} (x_{nwsk} + x_{n(w+1)sk}) + h_{nw}^9 \geq 2y_{nw} \qquad \forall n \in N, w \in W \quad (C.19)$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} + j_n^{10} \geq \beta_n^{10} \qquad \forall n \in N \quad (C.20)$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{k \in K} x_{ndsk} - j_n^{11} \leq \beta_n^{11} \qquad \forall n \in N \quad (C.21)$$

$$\sum_{w \in W} y_{nw} - j_n^{12} \leq \beta_n^{12} \qquad \forall n \in N \quad (C.22)$$

Constraints (C.2) ensure a single shift per day. Constraints (C.3) ensure the minimum number of nurses per days, shift, and skill. Constraints (C.4) ensure that a shift succession must be valid. Constraints (C.5) ensure a nurse can only be scheduled on a shift if they have the required skill. Constraints (C.6) ensure maximum M_h worked days, every 7 days. Constraints (C.7) calculate the preferred coverage violations. Constraints (C.8) and (C.9) calculate the minimum consecutive assignments (working days) violations. In the equations, $S1$ is calculated as the *(sum of the working days) + (two border bits* $\times$ *C) + (complement of middle bits* $\times$ *C)*. Constraints (C.10) calculate the maximum number of consecutive assignments (working days) violations. Constraints (C.11) and (C.12) calculate the minimum number of consecutive days off violations. $S2$ is evaluated similarly to Equations (C.8) and (C.9), however, the bits are inverted and the sum is related to free days instead of working days. Constraints (C.13) calculate the maximum number of consecutive days off violations. Constraints (C.14) and (C.15) calculate the minimum number of consecutive assignments to the same shift violations. $S3$ is evaluated similarly to Equations (C.8) and (C.9), however, the violations are stored by nurse/day/shift/pattern. Constraints (C.16) calculate the maximum of consecutive assignments to the same shift violations. Constraints (C.17) calculate the undesired day/shift assignments violations. Constraints (C.18) calculate if nurse n works on weekend w. Constraints (C.19) calculate the complete weekend violation. Constraints (C.20) calculate the minimum number of total working days violations over the whole scheduling period. Constraints (C.21) calculate the maximum number of total working days violations over the whole scheduling period. Constraints (C.22) calculate the total number of working weekends violations.

APPENDIX D — NURSE REROSTERING - VND ITERATIONS

Table D.1: VND - Average number of iterations

	VND iterations - single-day absences			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
n035w4	1.1	1.0	1.1	1.1
n035w8	1.4	1.0	1.4	1.4
n070w4	1.7	1.5	1.7	1.7
n070w8	1.9	1.8	1.9	1.9
n110w4	1.4	1.4	1.4	1.4
n110w8	1.9	1.8	1.9	1.9
	VND iterations - consecutive-days absences			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
n035w4	1.4	1.2	1.4	1.4
n035w8	1.5	1.5	1.5	1.5
n070w4	1.5	1.4	1.5	1.5
n070w8	1.9	1.4	1.9	1.9
n110w4	2.0	1.9	2.0	2.0
n110w8	1.8	1.8	1.8	1.8
	VND iterations - Lisbon instances			
	Complete scheduling	Only absent days	First absence to last absence	First absence to end scheduling
19 nurses	12.9	38.1	32.8	15.3
32 nurses	4.8	4.8	4.8	4.8

APPENDIX E — ROBUST ROSTERING - INTEGER PROGRAMMING FORMULATION FOR THE STAFF ROSTERING PHASE

Table E.1 details the constraints and costs employed in the mathematical formulation. The objective function cost is calculated per shift and is not only related to the nurses' wages but also guarantees service quality. The latter is modeled as a penalty for not meeting the minimum number of nurses per day/shift/skill. Table E.2 provides the input sets and variables employed in the rostering formulation.

Table E.1: Hard and soft constraints - rostering phase

Constraint description	Weight
Hard constraints	
A nurse can be assigned to at most one shift per day	HC
A shift type succession must belong to a valid succession	HC
A shift requiring nurses with a given skill must always be fulfilled by a nurse having that skill	HC
Maximum consecutive assignments (working days)	HC
Maximum consecutive assignments to the same shift	HC
Shift or day off request	HC
Minimum number of assignments over the scheduling period	HC
Soft constraints - rostering costs	
Minimum number of nurses per day/shift/skill	5000
Overtime cost per worked shift if a nurse works more shifts than contracted (150% of the wage)	[150,105,75,45]
Head nurse wage per worked shift	100
Nurse wage per worked shift	70
Caretaker wage per worked shift	50
Trainee wage per worked shift	30
Assign a nurse to a reserve shift (10% of the wage)	[10,7,5,3]

Table E.2: Indices, sets and variables used in the rostering formulation.

Symbol	Definition
Input Data	
$n \in N$	n is the index of the nurse, and N is the set of nurses;
$d \in D$	d is the index of the day, and D is the set of days;
$s \in S$	s is the index of the shift, and S is the set of all shifts;
s'	reserve shift index;

$k \in K$	k is the index of the skill, and K is the set of skills;
$(n,k) \in \tilde{K}$	set containing the pairs of forbbiden skill k of nurse n;
m_{dsk}	minimum number of required nurses on day d for shift s with skill k;
$(s1,s2) \in \tilde{S}$	set containing the pairs of invalid shift successions;
$(n,d,s) \in U$	triple with the undesired working day d, shift s for nurse n;
β_n^1	maximum number of consecutive working days;
β_n^2	maximum number of consecutive working days on Night shifts;
β_n^3	minimum number of working days over the planning horizon;
β_n^5	maximum number of working days over the planning horizon;
$c_{ndsk} \in \{0,1\}$	Constant values representing the current feasible solution, 1 if nurse n is allocated to shift s, day d with skill k, and 0 otherwise;

Constraint Weights

$\omega_{ns}^1 \in \mathbb{N}_{\geq 0}$	cost of nurse n to work on shift s;
$\omega_n^5 \in \mathbb{N}_{\geq 0}$	cost of nurse n per worked shift exceeding the contracted number of shifts;
$\omega^6 \in \mathbb{N}_{\geq 0}$	cost of understaffing;

Decision Variables

$x_{ndsk} \in \{0,1\}$	1 if nurse n is allocated to shift s on day d with skill k, 0 otherwise;

Auxiliary Variables

$v_n^5 \in \mathbb{N}_{\geq 0}$	number of worked shifts above the maximum contracted for nurse n;
$v_{dsk}^6 \in \mathbb{N}_{\geq 0}$	number of nurses below the minimum for day d, shift s, and skill k;

$$\textbf{Minimize:} \quad \sum_{n\in N}\sum_{d\in D}\sum_{s\in S}\sum_{k\in K} x_{ndsk}\omega_{ns}^1 + \sum_{n\in N} v_n^5\omega_n^5 + \sum_{d\in D}\sum_{s\in S}\sum_{k\in K} v_{dsk}^6\omega_n^6 \tag{E.1}$$

Subject to:

$$\sum_{s\in S}\sum_{k\in K} x_{ndsk} \leq 1 \qquad \forall n \in N, d \in D \tag{E.2}$$

$$\sum_{n\in N} x_{ndsk} + v_{dsk}^6 \geq m_{dsk} \qquad \forall d \in D, s \in S, k \in K \tag{E.3}$$

$$\sum_{k\in K} (x_{nds1k} + x_{n(d+1)s2k}) \leq 1 \qquad \forall n \in N, d \in D \setminus \{|D|\}, (s1,s2) \in \tilde{S} \tag{E.4}$$

$$\sum_{d\in D}\sum_{s\in S} x_{ndsk} = 0 \qquad \forall (n,k) \in \tilde{K} \tag{E.5}$$

$$\sum_{d'=d}^{\beta_n^1+d}\sum_{s\in S}\sum_{k\in K} x_{nd'sk} \leq \beta_n^1 \qquad \forall n \in N, d \in \{1,\ldots,|D|-\beta_n^1\} \tag{E.6}$$

$$\sum_{d'=d}^{\beta_n^2+d} \sum_{k \in K} x_{nd's'k} \leq \beta_n^2 \qquad \forall n \in N, d \in \{1, \ldots, |D| - \beta_n^2\} \tag{E.7}$$

$$\sum_{k \in K} x_{ndsk} = 0 \qquad \forall (n,d,s) \in U \tag{E.8}$$

$$\sum_{d \in D} \sum_{s \in S \setminus \{s'\}} \sum_{k \in K} x_{ndsk} \geq \beta_n^3 \qquad \forall n \in N \tag{E.9}$$

$$\sum_{d \in D} \sum_{s \in S \setminus \{s'\}} \sum_{k \in K} x_{ndsk} - v_n^5 \leq \beta_n^5 \qquad \forall n \in N \tag{E.10}$$

Objective function (E.1) minimizes the overall cost, including wages, overtime and the costs related to understaffing. Constraints (E.2) ensure a nurse works a single shift per day. Constraints (E.3) ensure the minimum number of nurses per day, shift and skill. Constraints (E.4) ensure that a shift succession must be valid. Constraints (E.5) ensure a nurse can only be assigned to a shift if they have the required skill. Constraints (E.6) ensure the maximum number of consecutive assignments (working days) is respected. Constraints (E.7) ensure the maximum number of consecutive assignments to Night shifts is not exceeded. Constraints (E.8) ensure undesired shifts or days are not assigned. Constraints (E.9) and (E.10) enforce the minimum and maximum number of assignments over the planning horizon, respectively.

APPENDIX F — ROBUST ROSTERING - INTEGER PROGRAMMING FORMULATION FOR THE STAFF RE-ROSTERING PHASE

Table F.1 details the soft constraint weights for the re-rostering phase. Table F.2 provides the input sets and variables employed in the re-rostering formulation. An additional penalty is induced for undesired roster modifications that may affect individual nurses' rosters.

Table F.1: Soft constraints - re-rostering costs

Constraint description	Weight
Call a nurse from a reserve shift (10% of the wage)	[10,7,5,3]
Call a nurse from a day off or cancel a working shift (150% of the wage)	[150,105,75,45]
Change the nurse's assigned shift (100% of the wage)	[100,70,50,30]
Change the nurse's assigned skill	0

Table F.2: Additional indices, sets and variables used in the re-rostering formulation.

Symbol	Definition
Input Data	
$\hat{N}$	set of absent nurses;
$\hat{c}_{nd} \in \{0,1\}$	1 if nurse n is absent on day d, and 0 otherwise;
Constraint Weights	
$\omega_n^2 \in \mathbb{N}_{\geq 0}$	cost of nurse n to change a shift (excluding the reserve shifts);
$\omega_n^3 \in \mathbb{N}_{\geq 0}$	cost of nurse n to change a reserve shift to a working shift;
$\omega_n^4 \in \mathbb{N}_{\geq 0}$	cost of nurse n to change from a day off to a working shift or vice versa;
Auxiliary Variables	
$y'_{nds} \in \{0,1\}$	1 if nurse n works on the original schedule or on the new roster, 0 otherwise;
$y''_{nds} \in \{0,1\}$	auxiliary variable to calculate the number of changes compared to the original roster;
$y'''_{nds} \in \{0,1\}$	auxiliary variable representing the number of changes compared to the original roster;
$v_{nd}^2 \in \mathbb{N}_{\geq 0}$	number of shift changes compared to the original roster excluding the reserve shifts for nurse n on day d;
$v_{nd}^3 \in \mathbb{N}_{\geq 0}$	number of changes from the reserve shift to any working shift for nurse n on day d;
$v_{nd}^4 \in \mathbb{N}_{\geq 0}$	number of times a day off is replaced with a working shift and vice versa;

$$\textbf{Minimize:} \quad (\text{E.1}) + \sum_{n \in N} \sum_{d \in D} \sum_{i \in \{2,..,4\}} v^i_{nd} \omega^i_n \tag{F.1}$$

Subject to:

$$\hat{c}_{nd} + \sum_{s \in S} \sum_{k \in K} x_{ndsk} \leq 1 \qquad \forall n \in N, d \in D \tag{F.2}$$

$$\sum_{k \in K} (c_{ndsk} + x_{ndsk}) \leq 2y'_{nds} \qquad \forall n \in N \setminus \hat{N}, d \in D, s \in S \tag{F.3}$$

$$\sum_{k \in K} (c_{ndsk} + x_{ndsk}) + y''_{nds} \geq 2y'_{nds} \qquad \forall n \in N \setminus \hat{N}, d \in D, s \in S \tag{F.4}$$

$$\sum_{s \in S} y''_{nds} - 2y'''_{nd} \leq 0 \qquad \forall n \in N \setminus \hat{N}, d \in D \tag{F.5}$$

$$\sum_{s \in S \setminus \{s'\}} \sum_{k \in K} (c_{ndsk} + x_{ndsk}) - 1 - v^2_{nd} \leq 1 - y'''_{nd} \qquad \forall n \in N \setminus \hat{N}, d \in D \tag{F.6}$$

$$\sum_{s \in S} \sum_{k \in K} x_{ndsk} + \sum_{k \in K} c_{nds'k} - 1 - v^3_{nd} \leq 1 - y'''_{nd} \qquad \forall n \in N \setminus \hat{N}, d \in D \tag{F.7}$$

$$\textstyle\sum_{s \in S} \sum_{k \in K} c_{ndsk} + \sum_{s'' \in S \setminus \{s'\}} \sum_{k \in K} x_{nds''k} + v^4_{nd} \geq 2(y'''_{nd} - \sum_{k \in K} c_{nds'k}) \qquad \forall n \in N \setminus \hat{N}, d \in D \tag{F.8}$$

$$\sum_{s \in S} \sum_{k \in K} x_{ndsk} \geq \sum_{k \in K} c_{nds'k} \qquad \forall n \in N \setminus \hat{N}, d \in D \tag{F.9}$$

The objective function (F.1) minimizes the overall cost regarding the original objective function (E.1) and costs associated with the re-rostering. Constraints (F.2) ensure that an absent nurse is not scheduled to work. Constraints (F.3), (F.4) and (F.5) calculate the number of changes compared to the original roster and store them in auxiliary variables. Constraints (F.6) calculate the shift changes excluding the reserve shifts. Constraints (F.7) calculate the shift changes concerning only reserve shifts. Constraints (F.8) calculate the working day to day off changes and vice versa. Constraints (F.9) ensure a reserve shift is not replaced with a day off.

APPENDIX G — ROBUST ROSTERING - EXPERIMENTS COMPUTATION TIMES

Tables G.1 and G.2 provide the computation times for the single- and multi-skilled instances, respectively.

Table G.1: Computation time (seconds) using single-skilled instances.

	Initial solution		Re-rostering solution	
Robustness level	General constraint	Robustness constraint per day	General constraint	Robustness constraint per day
0.00%	1.5	1.4	1.4	1.3
4.17%	1.8	2.0	1.4	1.3
8.33%	2.1	2.2	1.5	1.4
12.50%	5.4	5.3	1.5	1.5
16.67%	5.5	5.5	1.6	1.5

Table G.2: Computation time (seconds) using multi-skilled instances.

	Initial solution											
Robustness	General constraint				Robustness constraint per day				Robustness constraint per day and skill			
level	H	H,N	H,N,C	H,N,C,T	H	H,N	H,N,C	H,N,C,T	H	H,N	H,N,C	H,N,C,T
3.53%	3.6	6.1	2.7	3.6	407.5	11.4	3.1	3.0	290.9	-	-	-
7.64%	-	5.3	27.8	3.4	-	10.4	613.8	5.2	-	284.5	-	-
12.34%	-	-	61.2	84.8	-	-	1073.0	141.5	-	-	416.4	-
14.26%	-	-	-	102.4	-	-	-	60002.3	-	-	-	620.3
	Re-rostering solution											
Robustness	General constraint				Robustness constraint per day				Robustness constraint per day and skill			
level	H	H,N	H,N,C	H,N,C,T	H	H,N	H,N,C	H,N,C,T	H	H,N	H,N,C	H,N,C,T
3.53%	4.3	4.6	5.4	5.5	4.3	5.1	6.3	4.5	3.7	-	-	-
7.64%	-	6.4	4.7	3.9	-	2.8	4.4	3.2	-	4.2	-	-
12.34%	-	-	4.5	6.8	-	-	2.6	3.2	-	-	2.6	-
14.26%	-	-	-	5.3	-	-	-	2.6	-	-	-	13.6

APPENDIX H — 0.00% ROBUST ROSTER - JUNE 2019

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Name	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Mon	Tue	Wed	Thu	Fri
P1				N[2]				N[2]	N[2]		N[2]			N[2]				N[2]							N[2]				E[2]L[2]	E[2]L[2]
P2	N[1]	N[1]			L[3]	N[1]						L[1]	N[1]		E[1]L[1]	E[1]L[1]			L[3]								N[1]			
P3							E[4]					E[4]	E[1]	E[1]	N[1]	N[1]			E[4]		E[4]	E[1]L[1]	E[1]L[1]				E[4]	E[4]		
P4	N[1]		N[1]					E[1]L[1]	N[1]	N[1]							N[1]			E[1]L[1]				N[1]						
P5			N[2]					E[2]L[2]	E[2]L[2]	N[2]			L[2]		E[2]L[2]	E[2]L[2]	N[2]													
P6			E[4]		N[2]			E[2]L[2]	E[2]L[2]	E[4]		N[2]		E[4]	E[2]L[2]	E[2]L[2]	E[4]		N[2]					E[4]		N[2]				
P7			N[2]							N[2]			E[1]				N[2]					E[2]L[2]	E[2]L[2]	N[2]					N[2]	N[2]
P8				N[5]					N[5]		N[5]											E[5]L[5]			N[5]					
P9				L[5]		N[2]					L[5]		N[2]			N[2]		L[5]		E[2]L[2]					L[5]		N[2]		N[2]	
P10				N[2]		N[2]		N[2]	N[2]		N[2]							N[2]		E[2]L[2]		E[2]L[2]			N[2]					
P11					N[1]							N[1]			E[1]L[1]	E[1]L[1]			N[1]							N[1]			N[1]	N[1]
P12	E[1]L[1]	E[1]L[1]		N[1]				N[1]	N[1]		N[1]							N[1]							N[1]					
P13	E[2]L[2]	E[2]L[2]			N[2]							N[2]			N[2]	N[2]			N[2]							N[2]				
P14	N[5]									N[5]							N[5]						E[5]L[5]							
P15				N[1]							N[1]							N[1]				N[1]	N[1]		N[1]				E[1]L[1]	E[1]L[1]
P16			L[3]	N[3]				E[3]L[3]	E[3]L[3]		N[3]		L[1]	L[1]	N[3]	N[3]		N[3]						L[3]	N[3]		L[3]			
P17					E[4]	E[4]	N[1]	N[1]												E[1]L[1]	N[1]	N[1]				E[4]	E[1]	N[1]		
P18				E[4]		N[1]		E[1]L[1]	E[1]L[1]		E[4]							E[4]		N[1]			N[1]		E[4]		N[1]			
P19				L[3]		N[3]		N[3]	N[3]		L[3]							L[3]							L[3]		N[3]		E[3]L[3]	E[3]L[3]
P20		E[3]L[3]																	N[3]			N[3]	N[3]			N[3]				
P21	N[4]	N[4]								N[4]				L[4]	E[4]L[4]	E[4]L[4]	N[4]				L[4]			N[4]				L[4]		
P22			L[4]		N[4]		L[4]			L[4]		N[4]			N[2]		L[4]		N[4]	N[2]		E[4]L[4]	E[2]L[2]	L[4]		N[4]				
P23													N[4]	N[4]						N[4]			E[4]L[4]				N[4]			N[4]
P24			N[3]							N[3]				L[3]	E[3]L[3]	E[3]L[3]	N[3]							N[3]					N[3]	N[3]
P25					N[1]		L[3]					N[1]		L[1]	N[1]	N[1]			N[1]		L[3]	E[1]L[1]	E[1]L[1]			N[1]		L[3]		
P26			L[5]			N[5]		N[5]		L[5]			N[5]				L[5]			N[5]				L[5]			N[5]		E[5]L[5]	E[5]L[5]
P27				N[4]		L[4]		E[4]L[4]	E[4]L[4]				L[4]					N[4]				N[4]	N[4]		N[4]		L[4]			
P28	E[4]L[4]	E[4]L[4]		L[4]		N[4]		N[4]	N[4]		N[4]		E[4]					L[4]							L[4]					
P29	E[1]L[1]	E[1]L[1]	N[1]							N[1]				N[1]			N[1]							N[1]					N[1]	N[1]
P30	E[2]L[2]	E[2]L[2]			E[6]		N[2]							N[2]					E[6]			N[2]	N[2]			E[6]		N[2]		
P31	N[2]	N[2]					N[1]														N[2]						N[2]		E[1]L[1]	E[1]L[1]
P32	E[3]L[3]	N[3]				L[3]	N[3]					L[3]								N[3]	N[3]		E[3]L[3]			L[3]		N[3]		
P33					N[5]		L[5]					L[5]	N[2]		E[5]L[5]	E[5]L[5]		N[5]		N[2]							L[5]		N[5]	
P34		E[5]L[5]	N[5]		L[5]							N[5]			N[5]	N[5]			L[5]	E[5]L[5]				N[5]		L[5]				
P35	N[6]	N[6]			L[6]		N[6]			N[6]		E[6]L[6]							L[6]		N[6]			E[6]		L[6]		L[6]	E[6]L[6]	E[6]L[6]
P36																				E[4]L[4]	N[4]							N[4]	N[4]	
P37	E[5]L[5]	N[5]		L[6]		L[5]	N[5]				L[6]		L[3]	L[5]				L[6]		E[3]L[3]	N[5]				L[6]			N[5]		N[5]
P38								E[5]L[5]	E[5]L[5]				L[5]	N[5]					N[5]		L[5]	N[5]	N[5]			N[5]		L[5]		
P39			N[6]				L[6]							L[6]	E[6]L[6]	E[6]L[6]	N[6]		N[6]		L[6]		N[6]	N[6]				N[6]	N[6]	
P40			N[4]		L[4]		N[4]				L[4]	L[4]			N[4]	N[4]			L[4]							L[4]			E[4]L[4]	E[4]L[4]
P41																	L[3]					E[3]L[3]	N[2]	N[2]				N[2]		N[2]
P42	N[3]	N[1]							E[1]L[1]			N[3]	N[3]	N[3]																
P43			L[6]			N[6]			E[6]L[6]	L[6]			N[6]	N[6]			L[6]			N[6]		N[6]								
P44	N[2]	N[2]			N[3]		N[2]			L[3]											N[2]								E[2]L[2]	E[2]L[2]
P45				N[6]		E[6]L[6]					N[6]		E[6]L[6]		N[6]	N[6]		N[6]				E[6]L[6]	E[6]L[6]		N[6]		E[6]L[6]			
P46				E[6]	N[6]		E[6]	E[6]L[6]	N[6]		E[6]	N[6]		E[6]				E[6]		E[6]L[6]	E[6]			L[6]	E[6]	N[6]		E[6]		N[6]
P47	E[6]L[6]	E[6]L[6]	E[6]					N[6]		E[6]			N[1]	N[1]			E[6]			N[1]	N[1]						N[6]	N[1]		
P48																						N[2]								
R[1]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=1	L=1	E=1			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[2]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=1	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[3]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
R[5]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
$\hat{\delta}_d$	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

E=Early, L=Late, N=Night; 1=Area 1, 2=Area 2, 3=Area 3, 4=Area 4, 5=Area 5, 6=Area 6
Overtime: 298

APPENDIX I — 2.64% ROBUST ROSTER - JUNE 2019

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Name	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Mon	Tue	Wed	Thu	Fri
P1				N[2]				N[2]	N[2]		N[2]			N[2]				N[2]							N[2]				E[2]L[2]	E[2]L[2]
P2					L[4]	N[1]						L[1]	N[1]		N[1]	N[1]										L[4]	N[1]		E[1]L[1]	E[1]L[1]
P3					E[4]		E[4]					E[4]	E[1]	E[1]	E[1]L[1]	E[1]L[1]			E[4]		E[4]	N[1]	N[1]			E[4]		E[4]		
P4	E[1]L[1]	E[1]L[1]	N[1]						N[1]	N[1]							N[1]			N[1]				N[1]						
P5			N[2]					E[2]L[2]	E[2]L[2]	N[2]			L[2]		E[2]L[2]	E[2]L[2]								N[2]						
P6			E[2]		N[2]		E[2]			E[2]		N[2]			E[2]L[2]	E[2]L[2]	E[2]		N[2]							N[2]		E[2]	E[2]L[2]	E[2]L[2]
P7			N[2]					E[2]L[2]	N[2]	N[2]			E[1]				N[2]					E[2]L[2]	N[2]	N[2]						
P8	N[5]			N[5]							N[5]							N[5]		E[5]L[5]					N[5]					
P9				L[5]		N[2]					L[5]		N[2]		N[2]	N[2]		L[5]		E[2]L[2]					L[5]		N[2]			
P10	N[2]	N[2]		N[2]					E[2]L[2]		N[2]							N[2]		E[2]L[2]					N[2]		N[2]			
P11					N[1]			E[1]L[1]	E[1]L[1]			N[1]			N[1]	N[1]			N[1]							N[1]				
P12	E[1]L[1]	E[1]L[1]	L[3]	N[1]							N[1]						L[3]	N[1]						L[3]	N[1]				N[1]	N[1]
P13				E[2]	N[2]						E[2]	N[2]			N[2]	N[2]		E[2]	N[2]			E[2]L[2]	E[2]L[2]			N[2]				
P14																	N[5]			N[5]				N[5]						E[5]L[5]
P15				N[1]							N[1]							N[1]				N[1]	N[1]		N[1]				E[1]L[1]	E[1]L[1]
P16				N[3]		L[3]				L[3]	N[3]		L[1]	L[1]	N[3]	N[3]		N[3]				E[3]L[3]	E[3]L[3]		N[3]		L[3]			
P17						E[1]	N[1]	N[1]				E[1]							E[1]	E[1]L[1]	N[1]		E[1]L[1]			E[1]	E[1]	N[1]		
P18	N[1]	N[1]		E[4]		N[1]					E[4]							E[4]				E[1]L[1]	E[1]L[1]		E[1]		N[1]			
P19				L[3]		N[3]		N[3]	N[3]		L[3]							L[3]							L[3]		N[3]		E[3]L[3]	E[3]L[3]
P20		E[3]L[3]																	N[3]			N[3]	N[3]			N[3]				
P21	N[4]	N[4]					L[4]	E[4]L[4]	E[4]L[4]	N[4]				L[4]			N[4]				L[4]			N[4]				L[4]		
P22			E[4]L[4]		N[4]					E[4]L[4]		N[4]					L[4]					E[4]L[4]	E[2]L[2]	E[2]L[4]		N[4]			N[2]	N[2]
P23													N[4]	N[4]						N[4]			E[4]L[4]		E[4]		N[4]			N[4]
P24			N[3]							N[3]				L[3]	E[3]L[3]	E[3]L[3]	N[3]							N[3]					N[3]	N[3]
P25					N[1]		L[3]	N[1]	N[1]			N[1]		L[1]	E[1]L[1]	E[1]L[1]			N[1]		L[3]					N[1]		L[3]		
P26			L[5]			N[5]		E[5]L[5]	E[5]L[5]	L[5]			N[5]				L[5]							L[5]			N[5]		N[5]	N[5]
P27	E[4]L[4]	E[4]L[4]		N[4]		L[4]					N[4]		L[4]		N[4]	N[4]									N[4]		L[4]			
P28						E[4]		N[4]	N[4]				E[4]				E[4]	N[4]	N[4]					E[4]			E[4]		E[4]L[4]	E[4]L[4]
P29			N[1]					E[1]L[1]	E[1]L[1]					N[1]			N[1]							N[1]					N[1]	N[1]
P30	E[2]L[2]	E[2]L[2]			E[2]		N[2]	N[2]						N[2]						N[2]	N[2]							N[2]		
P31	E[2]L[2]	E[2]L[2]			N[3]		N[2]														E[2]	N[2]	N[2]					N[2]		
P32	E[3]L[3]	N[3]			L[3]		N[3]		E[3]L[3]										L[3]	N[3]	N[3]					L[3]		N[3]		
P33	E[5]L[5]	E[5]L[5]			N[5]	N[2]		N[5]	N[5]			L[5]	N[2]														L[5]	L[5]		
P34			N[5]		L[5]					N[5]		N[5]			N[5]	N[5]			L[5]			E[5]L[5]	E[5]L[5]			L[5]				
P35					E[6]L[6]		N[6]			N[6]		E[6]L[6]		L[6]	E[6]L[6]	E[6]L[6]			E[6]L[6]			N[6]	N[6]			E[6]L[6]		N[6]		
P36																				E[4]L[4]	N[4]							N[4]	N[4]	
P37	N[3]	N[5]		L[6]		L[5]	L[5]				L[6]		L[3]	N[5]				L[6]		E[3]L[3]	N[5]				L[6]	N[5]			E[5]L[5]	
P38							N[5]						L[5]	L[5]	E[5]L[5]	E[5]L[5]			N[5]		L[5]	N[5]	N[5]					N[5]		
P39	E[6]L[6]	N[6]	N[6]				L[6]	E[6]L[6]	N[6]					N[6]			N[6]				L[6]			N[6]		N[6]		L[6]		
P40			N[4]			N[4]					L[4]	L[4]		E[4]	E[4]L[4]	E[4]L[4]		L[4]	L[4]			N[4]	N[4]		L[4]					
P41												L[3]					N[2]				N[2]	N[2]							N[2]	N[2]
P42	N[2]	N[2]						E[3]L[3]				N[3]	N[3]	N[3]																
P43	N[6]		L[6]			N[6]			E[6]L[6]	L[6]			N[6]				L[6]			N[6]	N[6]									
P44	N[1]	N[1]		L[4]			N[4]			N[1]										E[1]L[1]		E[1]L[1]						N[1]		
P45		E[6]L[6]		N[6]		E[6]L[6]					N[6]		E[6]L[6]		N[6]	N[6]		N[6]		E[6]L[6]					N[6]		E[6]L[6]			
P46				E[6]	N[6]		E[6]				E[6]	N[6]		E[6]				E[6]	N[6]		E[6]	E[6]L[6]	E[6]L[6]	L[6]	E[6]			E[6]	N[6]	N[6]
P47			E[6]				N[1]	N[6]		E[6]			N[1]	N[1]			E[6]			N[1]	N[1]			E[6]			N[6]		E[6]L[6]	E[6]L[6]
P48																				N[2]										
R[1]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=1	L=1	E=1			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[2]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=1	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[3]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
R[5]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
$\hat{o}_d$	0.00%	0.00%	4.17%	4.17%	4.17%	4.17%	4.17%	0.00%	0.00%	4.17%	4.17%	4.17%	8.33%	4.17%	0.00%	0.00%	4.17%	4.17%	4.17%	0.00%	4.17%	0.00%	0.00%	0.00%	4.17%	4.17%	4.17%	4.17%	0.00%	0.00%

E=Early, L=Late, N=Night; 1=Area 1, 2=Area 2, 3=Area 3, 4=Area 4, 5=Area 5, 6=Area 6

Overtime: 394

APPENDIX J — 7.78% ROBUST ROSTER - JUNE 2019

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Name	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Mon	Tue	Wed	Thu	Fri
P1				N[2]				N[2]	N[2]		N[2]			N[2]				N[2]							N[2]				E[2]L[2]	E[2]L[2]
P2	N[1]	N[1]			L[1]	N[1]						L[1]	N[1]		E[1]L[1]	E[1]L[1]			L[1]							L[1]	N[1]			
P3	E[1]L[1]	E[1]L[1]			E[4]		E[4]					E[4]	E[1]	E[1]	N[1]	N[1]			E[4]		E[4]					E[4]		E[4]		
P4			N[1]			L[1]		E[1]L[1]	N[1]	N[1]							N[1]							N[1]					E[1]L[1]	N[1]
P5			N[2]					E[2]L[2]	E[2]L[2]	N[2]			L[2]		E[2]L[2]	E[2]L[2]								N[2]			L[2]			
P6			E[2]		N[2]		E[2]			E[2]		N[2]			E[2]L[2]	E[2]L[2]			N[2]		E[2]	E[2]L[2]	E[2]L[2]			N[2]		E[2]		
P7			N[2]							N[2]			E[1]				N[2]			E[2]L[2]			E[2]L[2]	N[2]					N[2]	N[2]
P8				N[5]							N[5]							N[5]		E[5]L[5]		N[5]			N[5]					
P9				L[2]		N[2]							N[2]		N[2]	N[2]		L[2]		E[2]L[2]							N[2]			
P10	E[2]L[2]	E[2]L[2]		N[2]				N[2]	N[2]		N[2]							N[2]							N[2]					
P11	N[1]				N[1]						L[1]	N[1]			E[1]L[1]	E[1]L[1]			N[1]	N[1]					L[1]	N[1]				
P12			L[1]	N[1]				N[1]	N[1]		N[1]						L[1]	N[1]				E[1]L[1]	E[1]L[1]	L[1]	N[1]					
P13				E[4]	N[2]			E[2]L[2]	E[2]L[2]		E[4]	N[2]			N[2]	N[2]		E[4]	N[2]							N[2]				
P14										N[5]							N[5]			N[5]			E[5]L[5]							
P15				N[1]						L[1]	N[1]							N[1]				N[1]	N[1]		N[1]				E[1]L[1]	E[1]L[1]
P16			L[3]	N[3]		L[3]		E[3]L[3]		L[3]	N[3]		L[3]	L[1]	N[3]	N[3]		N[3]		E[3]L[3]				L[3]	N[3]		L[3]			
P17					E[1]	E[1]	N[1]	N[1]				E[1]							E[1]	E[1]L[1]	N[1]						E[1]	N[1]	N[1]	
P18				E[1]		N[1]		E[1]L[1]	E[1]L[1]		E[1]							E[1]				N[1]	N[1]		E[1]		N[1]			
P19				L[3]		N[3]		N[3]	N[3]		L[3]						L[3]	L[3]							L[3]		N[3]		E[3]L[3]	E[3]L[3]
P20		E[3]L[3]																	N[3]			N[3]	N[3]			N[3]				
P21	N[4]	N[4]					L[4]	E[4]L[4]	E[4]L[4]	N[4]				L[4]			N[4]				L[4]			N[4]				L[4]		
P22			E[4]L[4]	L[4]	N[4]					E[4]L[4]		N[4]					E[2]L[4]					E[2]L[2]	E[4]L[4]	E[2]L[4]		N[4]			N[2]	N[2]
P23													N[4]	N[4]						N[4]		E[4]L[4]	N[4]		E[4]		N[4]			
P24			N[3]				L[3]			N[3]				L[3]	E[3]L[3]	E[3]L[3]	N[3]							N[3]				L[3]	N[3]	N[3]
P25	E[1]L[1]	E[1]L[1]			N[1]		L[1]					N[1]		L[1]	N[1]	N[1]			N[1]		L[1]					N[1]		L[1]		
P26	E[5]L[5]	E[5]L[5]	L[5]			N[5]				L[5]			N[5]				L[5]							L[5]			N[5]		N[5]	N[5]
P27				N[4]		L[4]							L[4]		E[4]L[4]	E[4]L[4]		N[4]							N[4]		L[4]		N[4]	N[4]
P28	E[4]L[4]	E[4]L[4]	N[4]			E[4]		N[4]	N[4]				E[4]				E[4]		N[4]					E[4]			E[4]			
P29			N[1]							N[1]				N[1]			N[1]					E[1]L[1]	E[1]L[1]	N[1]					N[1]	N[1]
P30	E[2]L[2]	E[2]L[2]					N[2]							N[2]						N[2]	N[2]		N[2]			E[2]		N[2]		
P31	N[2]	N[2]			N[3]																L[3]	E[3]L[3]	E[3]L[3]				N[2]	N[2]		
P32	E[3]L[3]	N[3]			L[3]		N[3]		E[3]L[3]			L[3]							L[3]	N[3]	N[3]					L[3]		N[3]		
P33				L[5]	L[5]	N[2]					L[5]	N[5]	N[2]					L[5]				N[2]	N[5]		L[5]				E[5]L[5]	E[5]L[5]
P34			N[5]		N[5]			E[5]L[5]	E[5]L[5]			L[5]			N[5]	N[5]			L[5]					N[5]		L[5]				
P35	N[6]	N[6]			E[6]L[6]		N[6]					E[6]L[6]			E[6]L[6]	E[6]L[6]			E[6]L[6]		N[6]			N[6]		E[6]L[6]		L[6]		
P36																				E[4]L[4]	N[4]	N[4]						N[4]		
P37	N[5]	N[5]		L[6]		L[5]	N[5]				L[6]		L[1]	N[5]				L[6]		E[1]L[1]	L[5]	E[5]L[5]			L[6]		L[5]	N[5]		
P38							L[5]	N[5]	N[5]				L[5]	L[5]	E[5]L[5]	E[5]L[5]			N[5]		N[5]					N[5]		L[5]		
P39			N[6]				L[6]	E[6]L[6]	N[6]	N[6]				L[6]			N[6]			E[6]L[6]	L[6]		N[6]			N[6]		N[6]		
P40					L[4]	N[4]	N[4]				L[4]	L[4]		E[4]	N[4]	N[4]		L[4]	L[4]							L[4]			E[4]L[4]	E[4]L[4]
P41												L[2]					N[2]					N[2]	N[2]					N[1]		E[1]L[1]
P42	N[3]	N[1]							E[1]L[1]			N[3]	N[3]	N[3]																
P43			L[6]			N[6]			E[6]L[6]	L[6]			N[6]	N[6]			L[6]			N[6]		N[6]								
P44	N[2]	N[2]					N[2]				N[4]										N[2]				L[4]				E[2]L[2]	E[2]L[2]
P45				N[6]		E[6]L[6]					N[6]		E[6]L[6]					N[6]				E[6]L[6]	E[6]L[6]		N[6]		E[6]L[6]		N[6]	N[6]
P46				E[6]	N[6]		E[6]				E[6]	N[6]		E[6]	N[6]	N[6]		E[6]	N[6]		E[6]			L[6]	E[6]			E[6]	E[6]L[6]	E[6]L[6]
P47	E[6]L[6]	E[6]L[6]	E[6]				N[1]	N[6]		E[6]			N[1]	N[1]			E[6]			N[1]	N[1]			E[6]			N[6]			
P48																				N[2]										
R[1]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=1	L=1	E=1			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[2]			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=1	E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2		E=2 L=2			E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2	E=2 L=2		
R[3]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
R[5]			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1	E=1	E=1			E=1	E=1	E=1		E=1			E=1	E=1	E=1	E=1	E=1		
$\hat{\delta}_d$	0.00%	0.00%	12.50%	12.50%	12.50%	12.50%	12.50%	0.00%	0.00%	12.50%	12.50%	12.50%	12.50%	16.67%	0.00%	0.00%	8.33%	12.50%	12.50%	0.00%	12.50%	0.00%	0.00%	8.33%	12.50%	12.50%	12.50%	12.50%	0.00%	0.00%

E=Early, L=Late, N=Night; 1=Area 1, 2=Area 2, 3=Area 3, 4=Area 4, 5=Area 5, 6=Area 6

Overtime: 496

APPENDIX K — PUBLICATIONS, CONFERENCES AND SEMINARS

International peer-reviewed journals or conferences:

- Wickert, Toni I.; Smet, Pieter; Vanden Berghe, Greet: The nurse rerostering problem: strategies for reconstructing disrupted schedules, *Computers and Operations Research*, volume 104, pages 319-337, 2019. DOI: <http://dx.doi.org/10.1016/j.cor.2018.12.014>;
- Wickert, Toni I.; Kummer Neto, Alberto F.; Buriol, Luciana S.: An integer programming approach for the physician rostering problem, *In Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2018)*, pages 53-67, Vienna – Austria, 28/08/2018 – 31/08/2018;
- Portella, Victória; Buriol, Luciana S.; Wickert, Toni I.: Metaheurística Late Acceptance Hill Climbing Aplicada ao Problema de Escalonamento de Enfermagem, *In Proceedings of the Brazilian Symposium on Operations Research (SBPO 2018)*, Rio de Janeiro, RJ, Brazil, 06/08/2018 – 09/08/2018;
- Wickert, Toni I.; Sartori, Carlo; Buriol, Luciana S.: A Fix-and-Optimize VNS Algorithm Applied to the Nurse Rostering Problem, *In proceedings of Sixth International Workshop on Model-based Metaheuristics (Matheuristic 2016)*, pages 1-12, Brussels – Belgium, 04/09/2016 – 07/09/2016.

Abstract or extended abstract presented in international conferences:

- Wickert, Toni I.; Mosquera Nuñez, Federico; Smet, Pieter; Thanos, Emmanouil: Optimizing city-wide vehicle allocation for car sharing, *32nd Annual Conference of the Belgian Operation Research Society (ORBEL 32)*, Liege, Belgium, 1/2/2018 - 2/2/2018;
- Wickert, Toni I.; Smet, Pieter; Vanden Berghe, Greet: A mathematical modeling approach for robust nurse rostering, *29th European Conference On Operational Research (EURO 2018)*, Valencia, Spain, 8/7/2018 – 11/7/2018;
- Wickert, Toni I.; Smet, Pieter; Vanden Berghe, Greet: Reconstructing Disrupted Nurse Rosters, *Operational Research Applied to Health Services (ORAHS2017)*, Bath, United Kingdom, 30/7/2017 - 4/8/2017;
- Wickert, Toni I.; Sartori, Carlo; Buriol, Luciana S.: A fix-and-optimize heuristic applied to the Nurse Rostering Problem,*Workshop on Applied Combinatorial Optimization Methods (WACOM 2016)*, Ouro Preto – Brazil, 21/03/2016 – 23/03/2016.

Seminars:

- Wickert, Toni I.: Strategies for reconstructing disrupted schedules. 2nd workshop on personnel planning and management. Place: Vrije Universiteit Brussel (VUB) - Room C2.07a (building C, 2nd floor) Date: 15/09/2017;
- Wickert, Toni I.: Reconstructing Disrupted Nurse Rosters. Place: KU Leuven - Technologiecampus, Gebr. De Smetstraat 1, 9000 Gent. Date: 13/07/2017 11:00.