DANIEL GERARD KREEFT

DEVELOPMENT AND IMPLEMENTATION OF A COMPUTER-AIDED METHOD FOR PLANNING RESIDENT SHIFTS IN A UNIVERSITY HOSPITAL

Mémoire présenté
à la Faculté des études supérieures et postdoctorales de l'Université Laval
dans le cadre du programme de maîtrise en sciences de l'administration
pour l'obtention du grade de *Maître ès sciences* (M.Sc.)

OPÉRATIONS ET SYSTÈMES DE DÉCISION
FACULTÉ DES SCIENCES DE L ' A D M I N I S T R A T I ON
UNIVERSITÉ LAVAL
QUÉBEC

2012

Résumé

Ce mémoire propose une formulation pour le problème de confection d'horaire pour résidents, un problème peu étudiée dans la litérature. Les services hospitaliers mentionnés dans ce mémoire sont le service de pédiatrie du CHUL (Centre Hospitalier de l'Université Laval) et le service des urgences de l'Hôpital Enfant-Jésus à Québec.

La contribution principale de ce mémoîre est la proposition d'un cadre d'analyse pour l'analyse de techniques manuelles utilisées dans des problèmes de confection d'horaires, souvent décrits comme des problèmes d'optimisation très complexes. Nous montrons qu'il est possible d'utiliser des techniques manuelles pour établir un ensemble réduit de contraintes sur lequel la recherche d'optimisation va se focaliser. Les techniques utilisées peuvent varier d'un horaire à l'autre et vont déterminer la qualité finale de l'horaire. La qualité d'un horaire est influencée par les choix qu'un planificateur fait dans l'utilisation de techniques spécifiques; cette technique reflète alors la perception du planificateur de la notion *qualité de l'horaire*. Le cadre d'analyse montre qu'un planificateur est capable de sélectionner un ensemble réduit de contraintes, lui permettant d'obtenir des horaires de très bonne qualité. Le fait que l'approche du planificateur est efficace devient clair lorsque ses horaires sont comparés aux solutions heuristiques. Pour ce faire, nous avons transposées les techniques manuelles en un algorithme afin de comparer les résultats avec les solutions manuelles.

Mots clés: Confection d'horaires, Confection d'horaires pour résidents, Creation manuelle d'horaires, Heuristiques de confection d'horaires, Méthodes de recherche locale

Abstract

This thesis provides a problem formulation for the resident scheduling problem, a problem on which very little research has been done. The hospital departments mentioned in this thesis are the paediatrics department of the CHUL (Centre Hospitalier de l'Université Laval) and the emergency department of the Hôpital Enfant-Jésus in Québec City.

The main contribution of this thesis is the proposal of a framework for the analysis of manual techniques used in scheduling problems, often described as highly constrained optimisation problems. We show that it is possible to use manual scheduling techniques to establish a reduced set of constraints to focus the search on. The techniques used can differ from one schedule type to another and will determine the quality of the final solution. Since a scheduler manually makes the schedule, the techniques used reflect the scheduler's notion of *schedule quality.* The framework shows that a scheduler is capable of selecting a reduced set of constraints, producing manual schedules that often are of very high quality. The fact that a scheduler's approach is efficient becomes clear when his schedules are compared to heuristics solutions. We therefore translated the manual techniques into an algorithm so that the scheduler's notion of schedule quality was used for the local search and show the results that were obtained.

Key words: Timetable scheduling, Resident scheduling, Manual scheduling, Heuristic schedule generation, Local search methods

Samenvatting

Deze doctoranduscriptie beschrijft een probleem-formulatie voor het werkrooster van medische studenten, een probleem waar tot op heden weinig onderzoek naar is gedaan. De ziekenhuisafdelingen waar mee werd samengewerkt in het kader van deze doctoranduscriptie zijn de afdeling pediatrie van het CHUL (Centre Hospitalier de l'Universite Laval) en de eerste-hulp afdeling van het Hôpital Enfant-Jesus in Québec.

Deze doctoranduscriptie stelt een onderzoekskader voor voor de analyse van handmatige technieken gebruikt in werkrooster-problemen, vaak omschreven als complexe optimisatie-problemen. We laten zien dat het met behulp van deze technieken mogelijk is een kleiner aantal restricties, of nevenvoorwaarden, te gebruiken voor de exploratie van alle mogelijke oplossingen. De gebruikte technieken kunnen verschillen van werkrooster tot werkrooster maar bepalen de uiteindelijke kwaliteit van de oplossing. Dit onderzoekskader richt zich op het voorstellingsvermogen dat de planner heeft van *werkrooster-kwaliteit* en modeliseert de technieken die een planner gebruikt voor het handmatig creëren van een werkrooster. Het onderzoekskader laat zien dat een planner in staat is een beperkt aantal constraints te selecteren en aan de hand hiervan een werkrooster op te stellen van zeer hoge kwaliteit. Het feit dat de aanpak van een planner efficiënt is wordt duidelijk als zijn werkroosters worden vergeleken met heuristieke oplossingen. Wij hebben hiervoor de technieken van het onderzoekskader omgezet in een een algoritme en rapporteren de met deze methode behaalde resultaten.

Trefwoorden: Werkroosters, Werkroosters medische studenten, Handmatig werkrooster, Heuristieke zoektechnieken, lokale zoektechnieken.

Acknowledgements

The completion of this Masters thesis has been possible by the advice, ideas and support of a great number of people. I would therefore like to thank everyone that in some way has contributed to this thesis. I would also like to thank my trusty computer that has stayed with me throughout these long hours while I was pounding the keyboard.

I would particularly like to thank Bernard Lamond and Angel Ruiz, in alphabetical order although both are dear to me, my thesis supervisors at the Université Laval. From the outset they have put a great trust in me and have helped me see this thesis thru to completion. I would also like to thank all the residents and hospital management that I worked with and provided me with the data necessary for this project. I would also like to thank my lovely wife that has supported me, even in the times where the research progress was slow. I would also like to thank my parents and my brother, who encouraged me to continue my studies.

I would finally like to thank my fellow students at the university with whom I have exchanged a large number of ideas on different aspects of my research. I would also like to thank everyone at the CIRRELT (Centre interuniversitaire sur les réseaux d'entreprise, la logistique, et le transport).

Foreword

This thesis is the final result of two years of work as a M.Sc. Student at the CIRRELT (Centre interuniversitaire sur les réseaux d'entreprise, la logistique, et le transport) under the supervision of Professors Bernard Lamond and Angel Ruiz. The framework presented in this thesis has been the subject of a conference paper (Kreeft et al. 2010) entitled: *"Allowing user interaction in timetable scheduling software",* presented at GISEH 2010.

This thesis provides a problem formulation of the resident scheduling problem, a problem on which very little research has been done. Additionnally, we present a formal framework for the analysis of manual techniques for creating a schedule, a description that has never been made within the field of scheduling. It is therefore difficult to directly associate this thesis to past works of researchers.

Perhaps the most difficult part of this thesis was to observe schedulers at work and translate our observations; schedulers use techniques that, to the eye of researchers in operations research in any case, have an impact on schedule quality. However, they are not capable of quantifying the impact a technique has on schedule quality. The hard part was therefore to translate our observations into scheduling techniques in such a way that it formed a clear and concise modelling framework.

Contents

List of Tables

List of Figures

List of Abbeviations

CIP	*Coverage Input Problem*
CI	*Coverage Input*
OR	*Operational Research*
MSP	*Manual Scheduling Process*
MRS	*Manually Restricted Space*
RAH	*Resident Assignement Heuristic*
RSP	*Resident Scheduling Problem*
TS	*Tabu Search*

Glossary

Sets

I	*Set of all residents available during the planning period*
I_l	*Set of residents belonging to seniority class l*
W	*Set of all wards k*
	W = {W(1),W(2)…W(K)}
WKD	*Set of all weekend shifts*
	(t = {5,6,7},{12,13,14},{19,20,21},{26,27,28})
WKDSET1	*Set of weekend shifts of first week of planning period*
WKDSET2	*Set of weekend shifts of second week of planning period*
WKDSET3	*Set of weekend shifts of third week of planning period*
WKDSET4	*Set of weekend shifts of fourth week of planning period*
WEEK	*Set of all week shifts*
	(t = {1,2,3,4},{8,9,10,11},{15,16,17,18},{22,23,24,25,26})
WEEK1	*Set of weekdays of first week of planning period*
WEEK2	*Set of weekdays of second week of planning period*
WEEK3	*Set of weekdays of third week of planning period*
WEEK4	*Set of weekdays of fourth week of planning period*
JUNIOR	*Set of all junior residents*
SENIOR	*Set of all senior residents*
PW	*Set of weekend shifts of last 2 weeks of previous planning period*
CW	*Set of weekend shifts of current planning period*
MONDAY	*Set of all Mondays of planning period*
TUESDAY	*Set of all Tuesdays of planning period*
WEDNESDAY	*Set of all Wednesdays of planning period*
THURSDAY	*Set of all Thursdays of planning period*
FRIDAY	*Set of all Fridays of planning period*
SATURDAY	*Set of all Saturdays of planning period*

SUNDAY	*Set of all Sundays of planning period*

Parameters

i	*Index for resident*	*(i= 1… m)*
t	*Index for day of planning period*	*(t = 1… n)*
k	*Index for wards*	*(k = 1 … K)*
RDO_{it}	*Constant parameter for when resident i requested a day off at day t*	
C_{it}	*Constant parameter for when resident i is on conference at day t*	
H_{it}	*Constant parameter for when resident i is on holiday at day t*	
MC_{tk}	*Constant parameter for the minimum number of residents needed at ward k on day t*	
MAX_i	*Maximum number of shifts resident i can work*	
ρ	*Maximum number of days off a resident can request*	
$\mathbb{Q}$	*Maximum number of allowed 48-hours rest cycles*	

Penalty values

PEN_MC	*Penalty attributed for non-respect of minimum coverage*
PEN_MAX	*Penalty attributed for non-respect of maximum number of shifts*
PEN_RDO	*Penalty attributed for non-respect of requested days-off*
PEN_C	*Penalty attributed for non-respect of conference days*
PEN_H	*Penalty attributed for non-respect of holidays*
PEN_EXC_72AFT	*Penalty attributed for non-respect of resting period after shift*
PEN_WKD	*Penalty attributed for non-respect of maximum number of weekends*
PEN_CWKD	*Penalty attributed for non-respect of maximum number of consecutive weekends*
PEN_DISP^-	*Penalty attributed for unfair dispersion of days (too much easy days)*
PEN_DISP^+	*Penalty attributed for unfair dispersion of days (too much difficult days)*
PEN_TSD^-	*Penalty attributed for unfair dispersion of shifts (too little shifts)*
PEN_TSD^+	*Penalty attributed for unfair dispersion of shifts (too much shifts)*
PEN_MO	*Cost attributed when resident i is working a shift on Monday*
PEN_TU	*Cost attributed when resident i is working a shift on Tuesday*
PEN_WE	*Cost attributed when resident i is working a shift on Wednesday*
PEN_TH	*Cost attributed when resident i is working a shift on Thursday*

PEN_FR	*Cost attributed when resident i is working a shift on Friday*
PEN_SA	*Cost attributed when resident i is working a shift on Saturday*
PEN_SU	*Cost attributed when resident i is working a shift on Sunday*
PEN_SINGSAT	*Penalty attributed when a resident works more than one Saturday*
PEN_BCON	*Penalty attributed when a resident works a shift before a conference*
PEN_ACON	*Penalty attributed when a resident works a shift after a conference*
PEN_DOC	*Penalty attributed when a doctor has to replace a resident*

Decision variables

x_{it}	*Variable for when resident i is working at day t*
x_{ik}	*Variable for when resident i is allowed to work on department k*
x_doc_t	*Variable for when doctors are working uncovered shifts*
rdo_{it}	*Variable for when resident i requested a day off at day t*
c_{it}	*Variable for when resident i is in conference at day t*
h_{it}	*Variable for when resident i is on holiday at day t*
s_{jt}	*Variable for when the junior resident j is working at day t*
rdo_exc_{it}	*Variable for when the number of requested days off is not respected*
c_exc_{it}	*Variable for when the number of conference days is not respected*
h_exc_{it}	*Variable for when the number of holidays is not respected*
exc_72bef_{it}	*Variable for when the legal resting period of 72 hours before a shift is not respected*
exc_72aft_{it}	*Variable for when the legal resting period of 72 hours after a shift is not respected*
$disp\_score_i$	*Variable for when the availability score of resident i differs from the average score*
$h\_bef\_exc_{i,WEEK}$	*Variable for excluding a resident from working the weekend before his holidays*
$h\_aft\_exc_{i,WEEK}$	*Variable for excluding a resident from working the weekend after his holidays*

Slack variables

$d\_MC_{kt}^{-}, d\_MC_{kt}^{+}$	*positive and negative slack variables for the total coverage on the department k*
$d\_max_i^{-}, d\_max_i^{+}$	*positive and negative slack variables for the maximum number of allowed shifts for resident i*

$d\_rdo_{it}^{-}, d\_rdo_{it}^{+}$	*positive slack variables for the number of requested days-off*
$d\_c_{it}^{+}, d\_c_{it}^{-}$	*positive and negative slack variables for the number of conference days*
$d\_h_{it}^{+}, d\_h_{it}^{-}$	*positive and negative slack variables for the number of holidays*
$d\_exc\_72bef_{it}^{-}, d\_exc\_72bef_{it}^{+}$	*positive and negative slack variables for the number of times the resting period before a shift is not respected*
$d\_exc\_72aft_{it}^{-}, d\_exc\_72aft_{it}^{+}$	*positive and negative slack variables for the number of times the resting period after a shift is not respected*
$d\_WKD_{i}^{-}, d\_WKD_{i}^{+}$	*positive and negative slack variables for the number of times the maximum number of weekend shifts is not respected*
$d\_CWKD_{i}^{-}, d\_CWKD_{i}^{+}$	*positive and negative slack variables for the number of times the maximum number of weekend shifts is not respected*
$d\_disp_{i}^{-}, d\_disp_{i}^{+}$	*positive and negative slack variables for the average score for days per resident*
$d\_tsd_{i}^{-}, d\_tsd_{i}^{+}$	*positive and negative slack variables for the average number of shifts worked per resident*
$d\_singsat_{i}^{-}, d\_singsat_{i}^{+}$	*positive and negative slack variables for the number of times a resident works more than a single Saturday*
$d\_bcon_{i}^{-} - d\_bcon_{i}^{+}$	*positive and negative slack variables for the number of times a resident works before attending a conference*
$d\_acon_{i}^{-} - d\_acon_{i}^{+}$	*positive and negative slack variables for the number of times a resident works after attending a conference*

Tabu search parameters

s^*	*Best known solution found by the tabu search*
s_0	*Initial solution found by the tabu search*
s^p	*Potential solution in the candidate list p=1…P*
s^k	*Best solution on the candidate list choosen from all potential solutions*
s^c	*Current solution used in the tabu search*
TL	*Tabu list size*
MI	*Maximum number of iterations of the search*
MNI	*Maximum number of iterations without improvement in the best known solution*
TAST	*Total available search time*
CL	*Candidate list of all potential solutions*
CDT_LIST	*Name of the candidate list in the prototype's algorithm*
SFT_LIST	*List of all shifts necessary to ensure full coverage throughout the planning period*
RSD_LIST	*List that contains the pre-scheduling score for each staff member calculated using the pre-filled calendar*
MAXSFT_LIST	*List that keeps track of the number of remaining shifts*
LIST_SCORELINE	*List that contains the score of each line of the current solution and represents* $f_i'(s^c)$.
$SCORE_i$	*Pre-scheduling score of resident i*
Assignment $score_t$	*Cost attributed when the shift on day t is assigned to a resident*

Intensification and diversification procedures

BAEP procedure	(*Best Available Exchange Possible*)

	Procedure used by the tabu search to select the best available move. This procedure is used in the intensification process for establishing the candidate list.
PDS procedure	(*Probabilistic Diversification Strategy*) Procedure used to avoid the iterative procedure of the tabu search from remaining trapped in a local optimum. This procedure is used for establishing the candidate list in the diversification process.
SELECT_LINE procedure	Selection procedure that keeps track of the line to choose.This procedure composes the list of the score of each resident in a list called LIST_SCORELINE.

Chapter I – Introduction

1.1 Background

A resident is a medical specialist-in-training and during his residency he practices medicine under direct or indirect supervision of a doctor. In the province of Québec there are approximately 3000 residents and their schedules are mostly made by hand. Particularly little help has been provided, in the form of software tools, to support them in their planning efforts. All activities done by residents are overseen by the FRMQ (Fédération des médecins résidents du Québec), a provincial instance that serves and protects the interests of hospital residents. This protection also means that there is a collective agreement that dictates the working conditions applicable to residents uniformly throughout the province. Besides the resident's regular workload of day and night shifts, a resident still receives education and attends conferences. All of these requirements can lead to exhausting schedules.

We have found that exhausting schedules are very often due to the fact that resident schedulers do not have the time to compare alternative schedules which unnecessarily increases the residents' work stress. Scheduling software is available in most hospitals nowadays and has frequently helped to reduce the stress caused by exhausting schedules[1]. Computer software leads to an improvement in productivity and allows the scheduler to consider better alternatives. It should be mentioned that not every department in a hospital benefits from scheduling software. There only is a potential gain for departments of reasonable size, where the time spent creating a schedule is a factor that can be connected to the quality of a schedule. An example of such a time factor was encountered during our study at the paediatrics department of a hospital where approximately 30 residents are scheduled on a monthly basis. It should be mentioned that

[1] A literature review by Burke et al (2004b) mentions a number of different software programs that have been used in real-life in different hospitals

departments with 10 to 18 residents are rather the standard in the Québec context. In such smaller departments the scheduling effort is less time-consuming.

Until now, computer programs leading to a strong improvement in productivity have only been considered by larger departments because the scheduling task becomes more complex. At the paediatrics department, scheduling activities take approximately 8-hours per month, whereas this would only be 3-hours in a smaller department. Software therefore can be useful in larger departments saving more than half the time.

The total time of creating a new schedule is not completely spent on assigning the shifts to residents. A large part of the job consists in adding the initial data, i.e. the names and time-off requests. In our study we measured the time it took to create a new schedule in two hospital wards and we found that the total times needed were very different. In the larger ward this took approximately 8 hours, whereas this only took 2 hours at the smaller ward. At both wards collecting and adding the initial data took approximately 1 hour. With a software program the time spent assigning shifts can be brought back to 1 hour in both the large and small hospital ward. Here, if we suppose that this larger ward would use scheduling software to create a schedule, the total time would therefore be situated at 2 hours, a 6-hours time saving. At the smaller ward assigning shifts using software could take as little as 30 minutes. A schedule could therefore be created in 1 ½ hours, a ½-hour gain.

A missing link often exists in scheduling research between theory and practice when a research project discusses a *real-life problem*. When discussing these types of problems in an article most research projects present three stages. First, the research problem is defined. Next, a model is formulated to provide a solution to the problem. Finally, an algorithm is developed and applied to the model. Two questions are not always addressed:

1. Is the algorithm used by people, other than researchers, to solve other instances of the real-world problem on a regular basis?
2. What does the program provided to the users look like?

The practical sides of the research project are seldom presented, or often merely presented as a footnote in the conclusion. It is often hard to verify if the presented results of the real-life problem were useful in a real-life situation.

In health care there are different employee groups such as nurses, doctors and residents. Each group is subject to a specific collective agreement that dictates their working conditions. On top of these legal conditions each hospital department has its own rules for scheduling employees. When researching the topic of scheduling we therefore have to partition the problems in different schedule categories each with their own particularities and each their own employee group.

The resident scheduling problem is a multi-period staff assignment problem with a predefined number of work-night shifts (of 12- or 24-hours shifts in length) that are assigned to staff members, while considering departmental staffing needs as well as residents' preferences; The different considerations result in a schedule where a resident works around a single shift per four days, in such a way that a single shift is isolated from others by blocks of resting days. This type of scheduling problem has hardly been researched in particular even if their working conditions are different from normal medical staff. The resident scheduling problem reduces to a simplified nurse scheduling problem where a set of shifts (typically morning, day, night timeslots) per day are replaced by a single daily shift and where a normal five day working week is replaced by a single shift per four days.

This thesis presents a prototype with an integrated optimization method for solving the resident scheduling problem. In this schedule type day shifts occur according to the same fixed schedule for all residents. However, when working a night shift, residents are exempt from active duty in the hospital by the rules of the collective agreement. Nevertheless, they can be required to be present at their mandatory educational and conference days directly after a night shift. It occassionnally happens that residents have to accept schedules that are very exhausting.

1.2 Objectives and research contribution

The objective of this thesis is twofold. First of all we provide a problem formulation for the resident scheduling problem, a problem on which very little research has been done. Additionnally, we present a formal framework for the analysis of manual techniques for creating a schedule, a description that (to our knowledge) has never been made within the field of scheduling. It is therefore difficult to directly associate this thesis to past works of researchers. This framework has furthermore translated the manual techniques into an optimization method that was integrated in the prototype that has resulted from this thesis.

Perhaps the most difficult part of this thesis was to observe schedulers at work and translate our observations; The hard part was therefore to translate our observations into scheduling techniques in such a way that it formed a clear and concise modelling framework. We show that it is possible to use manual scheduling techniques to establish a reduced set of constraints to focus the search on.

On a secondary level, the intention of this study is to contribute to the field of scheduling research by clarifying what the scheduler's notion of *schedule quality* exactly represents and integrate this notion in a search algorithm. The techniques used can differ from one schedule type to another and will determine the quality of the final solution. Since a scheduler manually makes the schedule, the techniques used reflect the scheduler's notion of *schedule quality.* To analyse the scheduler' definition of schedule quality and see how manual techniques influence the schedule quality we present a formal framework for the Manual Scheduling Process (MSP). We also wished to discover the way in which the scheduler wishes the search algorithm to perform the search. This model has been integrated in a prototype in order to investigate the scheduler-software interaction.

The first part of the project involved a literature review and the formulation of a mathematical model. The mathematical model applies to other hospitals across the province of Québec, with exception of a few constraints. The model can thus be used in other hospitals without any additional implementation efforts.

The goal of this thesis is to explain a solution approach that uses a methodology derived from intuitive techniques to build an initial solution and to perform a local search. We have formalised then programmed into our solving algorithm some of the intuitive ideas that managers already use to build schedules by hand. From this a heuristic method to build an initial solution has been conceived. The algorithm tries to improve the initial solution by a tabu search. It is worth mentioning that, unlike most of the works reported in the literature, the construction algorithm allows the user to influence the evolution of the schedule. We will furthermore describe the details of the program and discuss the results that came forward out of the tests performed with the residents.

The added value of this research project is to show that a project can present theoretical results as well as discussing the practical advantages software offers to its final users. By first formulating the problem and presenting the algorithm implemented to find solutions we wish to follow the normal steps of a research project. Next, we desire to show that our prototype has allowed the hospitals involved to develop new schedules on a regular basis and that this is done by a resident scheduler. Furthermore, we also provide a description of the used prototype.

This project initially started as a pilot project to model the constraints that apply to residents' wards in Québec. This led to a second phase in which a prototype of a schedule generator was developed. It was assumed first that an interface would be of little interest. However, the schedulers found it difficult to use the prototype without an interface and therefore interpret the quality of the resulting schedules. A part of the focus of this project has therefore moved to the development of a user-friendly prototype that could be used by residents of the Enfant-Jesus and CHUL in Québec City. It aims at providing the residents schedulers of these hospitals with software capable of producing mathematically optimal (or at least high quality) schedules. This prototype has been offered to resident schedulers free of cost in exchange for information on how they use the computer program.

1.3 Structure of the thesis

We will first present the mathematical model and the prototype that was developed as a result. Chapter II provides a brief tutorial on nurse scheduling as well as a literature review on papers related to resident and nurse scheduling and optimization methods implemented in/or tested on real-life instances. The review discusses articles in which computer systems were implemented in hospitals to illustrate the research that discussed the pratical side of projects. There is little research available on resident scheduling and it was therefore deemed more interesting to discuss the literature connected to nurse scheduling problems to provde an overview of the research in this area. The problem description, a formal framework of the manual scheduling process and the construction method and tabu search that we developed from this framework are provided in Chapter III. Next, Chapter IV explains how the data instances were obtained and how the prototype and constraints were validated with the resident schedulers. Chapter V describes the implementation and validation of the prototype with resident schedulers. Finally, Chapter VI presents the results for the test instances obtained. Future research directions and conclusions will be presented in Chapter VI.

Chapter II – Problem definition

2.1 Introduction

Given a hospital context, the resident scheduling problem is a multi-period staff assignment problem with a predefined number of work-night shifts (of 12- or 24-hours shifts in length) that are assigned to staff members, while considering departmental staffing needs as well as residents' preferences; The different considerations result in a schedule where a resident works around a single shift per four days, in such a way that a single shift is isolated from others by blocks of resting days. The residents' problem is never considered in particular although their working conditions are different from normal medical staff (nurses, doctors, etc.), for whom a number of literature reviews show that different solution approaches have been proposed (Bradley and Martin (1991), Jelinek and Kavois (1992), Ernst et al. (2004)) to deal with their working conditions. For the purpose of a literature review it is therefore more pertinent to discuss the literature connected to Nurse Rostering Problems (NRP's). However, we can consider that the discussed solution methods are equally valid for resident scheduling problems.

Nurse rostering is defined as the creation of a periodic (weekly, fortnightly, or monthly) schedule for the nursing staff of one or several wards, subject to constraints that Miller et al. (1976) called feasibility set- and nonbinding-constraints (also called as hard and soft constraints), such as legal regulations, personnel policies, nurses preferences and other hospital specific requirements. The formulation of cost functions and objectives can vary from one hospital to another. The variation in circumstances has resulted in different NRP-models and the development of different solution approaches. The resident scheduling problem can be considered as a reduced nurse scheduling problem where a set of shifts (typically morning, evening, night timeslots) per day are replaced by a single daily shift and where a normal five day working week is replaced by a single shift per four days. The literature review will discuss solution methods for NRP's because these methods are a good representation of resident scheduling problems.

There are different reasons that make hospital personnel scheduling problems important. Making schedules in an effective and efficient way is important because nursing salaries can account for up to 40 % (Boldy and O'Kane, 1982) of hospital budget costs. There also are different interest groups that benefit from a good schedule. Warner (1976) was one of the first to describe hospital staff as an interest group that has something to gain from good schedules. Sitompul and Randhawa (1990) as well as Oldenkamp and Simons (1995) mentioned that it direcly affects the care quality a patient receives. For hospital staff, a better schedule means less stress and more rest. A good schedule improves care quality for patients, speeding up the patient's recovery. We can also add the hospital interests; the hospital is concerned with using as little nursing staff as possible - in order to keep costs low - while providing a satisfactory level of care. Job satisfaction is often low when staff works shifts in an irregular way. When a staff member changes from night shift to day shifts in an irregular manner, he will experience the undesirable effect of 'jet fatigue'. Irregular working patterns have a negative impact on circadian rhythms and as such on job satisfaction of nursing staff

2.2 Nurse timetable scheduling: a brief tutorial

2.2.1 Types of nurse rostering problems

The category to which a scheduling problem-type belongs is determined according to the addressed problem-type and the application areas covered. The general categories of personnel scheduling problems that exist have been mapped by Ernst et al. (2004). They classified the papers according to the type of problem addressed, the application areas covered and the methods used. Examples of different types of problems are crew scheduling and days-off scheduling. Examples for the application areas are airlines, call centres, manufacturing and nurse scheduling. NRP's formed a category containing over 107 papers. Randhawa and Sitompul (1993) have developed a classification scheme for nurse scheduling models according to the type of scheduling and the solution approach.

Some of the methods used include mathematical programming, set covering, genetic algorithms, and simulation.

The description in this review will focus on the criteria that define NRP's. These criteria can be directly applied to classify resident problems. Within the category of nurse rostering problems a schedule can be described by the following characteristics (based on the terminology defined by Burke et al. (2004b)):

– *Planning Period*: the time interval over which staff is scheduled. The typical length of a planning period is 4 weeks;

– *Skill Category*: staff members have a particular level of qualification, skill or responsibility which allows them to perform specific tasks, or explicitly prohibits them from doing so;

– *Shift Length*: Shifts always have a well-defined start and end time. Many rostering problems are concerned with the three traditional shifts: early (e.g. 6:00 a.m.–2:00pm), late e.g. 2:00pm –10:00pm), and night (e.g. 10:00pm –6:00am), but two 12-hour shifts also occur (early 8:00am – 8:00pm, night 8:00pm-8:00am). This is also illustrated in Figure 2.1

– *Schedule type*: A schedule can be either cyclic or non-cyclic. If a schedule is non-cyclic staff members can indicate their preferences for working or being off on specific days.

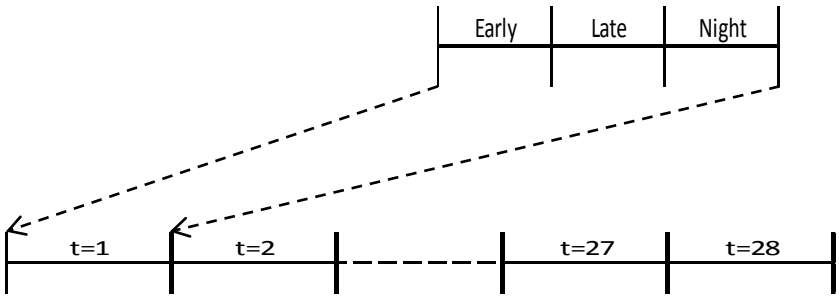


Figure 2.1 The division of shifts over a planning period

Figure 2.1 shows that throughout the day employees provide a 24-hours coverage by working early, late or night shifts. In nurse rostering problems this is the most used type of coverage.

To this day no formal classification has been created for NRP's. De Causmaecker and Vanden Berghe (2010) however made an attempt to start the development of a more general framework for categorizing nurse rostering problems. Osogami and Imai (2000)

developed an instance generator for the NRP-problem that can handle different parameters for the mentioned categories. Different instances can be created for sub-categories from the same data. For example, from the same input data a generator can create different instances for cyclic and non-cyclic schedules.

The evaluation of personnel scheduling problems could be considered as an element related to the classification of methods. Timetabling research is sometimes motivated by the search for higher efficiency but on other occasions by the development of new

Schedule assessment factors	
Optimality factor	Represents the degree in which nursing expertise is distributed over the different shifs
Completeness factor	Represents the degree in which the quantitative demands for occupation per shift are met
Proportionality factor	Degree with which each nurse have been given about the same amount of day and night shifts
Healthiness factor	Degree in which it has been taken of the welfare and health of the nurses
Continuity factor	Represents the degree in which there is continuity in the nursing crew during the different shifts

Table 2.1 Schedule assessment factors

methods. When researchers develop new methods a more objective evaluation can be necessary than the simple measurement of the penalty cost. Oldenkamp and Simons (1995) proposed five factors to evaluate nurse rostering problems as illustrated in Table 2.1. These factors can be used for the assessment of the solution quality and human-computer interaction quality and provide a more objective evaluation of the efficiency of the solution method.

2.2.2 Depicting nursing schedules

A *nurse roster* is defined as a calendar for a period of t days containing n persons. The nurse roster can be depicted as a two-dimensional data structure of decision variables. Figure 2.2 shows an example of a nurse roster, depicted as a R_{ij} matrix schedule, called a day-view. One dimension contains the set of staff members and the second dimension contains the set of days in the planning period. As illustrated, each line of the calendar indicates the schedule for a staff member, referred to as an individual schedule. A *nurse schedule* is defined as the list of tasks that is assigned to a specific individual for a time period. A nurse schedule is the formal term and is also used when referred to resident schedules'. In Figure 2.2 the days marked by a 1 stand for shifts worked, whereas 3 and 4 respectively indicate conference attendance and holidays on the days. We can read the

Nurse/ Resident	Mon	Tue	Wed	Thu	Fri	Sat	Sun
R-01	0	0	0	1	0	0	1
R-02	0	0	1	3	0	3	3
R-03	4	4	4	4	4	4	4

Figure 2.2 Example of a R_{ij} matrix nurse roster

nurse roster as follows: employee R-01 will be on duty Thursday and Sunday, R-02 is on duty Wednesday and absent for conference attendance on Thursday, Saturday and Sunday, R-03 is on holiday throughout the entire week.

In research literature the general term schedule is often used to indicate either a nurse roster or a nurse schedule. During the solution search the term *shift pattern* is used to designate an infeasible or feasible nurse schedule. The term “shift patterns” is also used whenever a timetable is cyclic due to the fact that different patterns are repeated throughout the schedule.

2.2.3 Complexity of nursing schedules

The criteria presented in §2.2.1 also determine the number of possible alternative solutions. To explain the number of alternative solutions we introduce the following variables and parameters and consider them not only to be useful to define nurse scheduling but also for resident scheduling problems:

Parameters

T: Total number of days of planning horizon

ɵ: Total number of shifts during a day

W: Total number of different wards

C: Total number of nurses/residents needed

Variables

M: Total number of shifts to be planned

N: Total number of nurses/residents available

During each day of a planning horizon several shifts can be planned. Shift lengths and the length of the planning horizon in the health care area are often determined by trade unions. The typical length of the planning horizon is T=28. For nurses, the length of each shift is 8 hours in most cases. The total number of shifts ɵ is therefore (ɵ = 3). For residents, the length is dependent on the type of duty performed.

The parameters T, ɵ, C and W determine M the total number of shifts of the problem. M grows for positive values of T, ɵ, C, and W and increases further if there are several co-existent shift lengths. E.g. during the day shift more nurses are active but not always on duty during an entire morning shift; some nurses can work on a part-time basis and only work half of the afternoon shift. In our project the shift length for residents is 12-hours on weekdays and 24-hours (ɵ = 1) on weekends. In our project the following values therefore apply:

$T = 28$

$ɵ = 1$

$W = 1 \text{ or } 2$

$C = 2 \text{ or more}$

$M = CTɵW$ *(2.1)*

The decision variable x_{it} is a binary variable introduced to identify what each staff member i does on each day t of the planning horizon. This variable defines on-duty shifts (*1*) and free shifts (*0*). This variable will be discussed in further detail in Chapter 3 to allow other events to be handled simultaneously. On-duty shifts can be defined to include a morning shift, an afternoon shift and a night shift. Free shifts are more complex since they include days off, public holidays, vacation leave, study day, unpaid leave, etc.

Miller et al. (1976) defined π_i as “the set of feasible patterns for nurse” /resident i, and the solution space as the Cartesian product of all feasibility regions $\pi_1, \pi_2, \pi_3, \ldots \pi_n$,. For a single employee with 4 shifts worked over a period of 28 days, a single feasibility region contains:

$$\binom{n}{k} = \frac{n!}{x!(n-k)!} = \binom{28}{4} = \frac{28!}{4!(28-4)!} = 491\,400\ schedules$$

However, the number of available solutions that really exists in practice is smaller. Because there are hard constraints on conferences and holidays, certain days cannot be considered when assigning a shift. Since not every day can be considered for scheduling a shift the number of combinations for each of them is lower than theoretically suggested. The total number of permutations is therefore also lower. If feasibility regions are defined, each region can be defined as an upper bound for the complexity of the problem.

Another factor in scheduling problems are upper and lower bounds on coverage constraints. The lower and upper bounds for the number of shifts are generally relaxed for different reasons, e.g. nurses can work overtime. Practical problems therefore have different types of coverage for the lower and upper bound and to take into account under- and overcoverage. Minimum coverage is defined as the minimum staff level to maintain a ward operational. Desired coverage is defined as the number of employees that hospital management wishes to have on duty. Overcoverage is defined as the maximum number of employees that hospital management will have operational at the same time. When these bounds are taken into account in Equation (2.1) this will become:

Bounds
MC: Minimum coverage
DC: Desired coverage
OC: Overcoverage

$$MC \leq DC \leq CT \theta W \leq M \leq OC \qquad (2.2)$$

Where MC, DC and OC ≥ 0. To solve this problem the number of planned shifts will ideally be situated between DC and OC. If M is larger than the upper bound OC the hospital will incur unnecessary extra costs for nursing. If M is below MC a hospital ward will not operate efficiently or not at all. In reality bounds are formulated so that M will be in between MC and DC.

2.2.4 Basic timetabling notions and framework approaches

Nurse Rostering problems generally have a constant workforce demand and constant availability and are therefore considered to be deterministic. Several deterministic models have been proposed during the last decade in order to resolve a range of diverse problems of nurse rostering problems.

To become more familiar with scheduling theory it can be useful to consult different tutorials. De Werra (1985) presents a timetabling tutorial based on graph-coloring formulations to explain search techniques and complexity issues for the class-teacher problem and course timetabling. Another introductory tutorial to staff scheduling is given in Blöchliger (2004) that presents the basic concepts of the scheduling problem and discusses some facets of staff scheduling. An alternative tutorial written by Ferland et al. (2001) associates a generalized assignment framework to several problem categories and illustrates a few scheduling examples, amongst which nurse scheduling.

Basic formulations of personnel scheduling problems are not as widespread as the classic travelling salesman problem and only a few papers try to address it thru a more general framework. A good standardized description of personnel scheduling is provided by the *minimum shift design problem* (MSD) by Gaspero et al. (2003). Their research provides a good insight into the theoretical aspects of scheduling and they show that the MSD reduces to a special case of the minimum edge-cost flow problem. They furthermore prove that the logarithmic approximation of the problem is NP-hard. Bilgin et al. (2008) also have attempted to look into complexity and scheduling problems.

2.2.5 Cyclical and non-cycliclal scheduling

As briefly mentioned in §2.2.1 and §2.2.2, schedules can be cyclic scheduling or non-cyclic. In a cyclic schedule a number of shifts are always grouped together and nurses/residents rotate from one pattern to another. In cyclic schedules a number of feasible shift patterns such as those shown in fig 2.1 are used to build a schedule. In a

non-cyclic schedule shifts are considered to be independent and every shift is assigned individually.

A pattern is a fixed order of shifts and days off. For example, a nurse/resident works Monday-Friday and is off on the weekend. Figure 2.3 shows three different patterns that commonly occur in a working week. Suppose that Example 1 is an infeasible pattern because a nurse cannot be required to work more than five days in a row. Then Example 2 would be a feasible pattern since it contains no more than five working days in a week and two resting days. Example 3 shows another feasible, (although unwanted) pattern where the working days are separated on/off days. In cyclic schedules a pattern often contains a sequence of days on duty followed by days off or the inverse because such patterns are more easily accepted than separated on/off days.

1	2	3	4	5	6	7
1	1	1	1	1	1	1

1	2	3	4	5	6	7
1	1	1	1	0	0	1

1	2	3	4	5	6	7
1	0	1	0	1	0	1

Figure 2.3 Examples of shift patterns

The notion of cyclic or non-cyclic became an important notion in personnel scheduling problems because it has an influence on the complexity of a problem. By regrouping a number of shifts together we can decrease the number of combinations of shifts assigned to nurses/residents.

Before illustrating the interest of cyclic schedules we first define a number of variables. We define *ST* to be the number of shift patterns or shift subsets. We furthermore have:

ST: Number of shift subsets
T: Length of planning horizon (days)
N: Total number of shifts
p_i*: shift pattern i (i =1…4)*

Suppose that we mix a number of patterns together similar to the examples and that we have to assign all of these patterns to a set of nurses/residents in order to create a schedule. Now suppose that we put all nurses/residents together in a room and ask them to distribute the patterns amongst each other. Imagine that the same would have to be done but each individual shift has to be assigned. Not only would such a distribution likely take more time but nurses/residents would probably divide the shifts in patterns to assign the shifts more quickly and have a more favourable working schedule for everyone. The use of shift patterns has a similar impact. It can reduce the complexity of a problem and it can result in better work patterns staff.

We can demonstrate this using an example. Say for example that for a hospital ward a two-week schedule (T= 14) has to be created with N shifts divided in a morning (1) and evening (2) shift each day. The number of shifts to be assigned if the schedule is non-cyclic would result in 28 variables. By defining 4 shift patterns it would be possible to reduce the complexity to 8 variables. For example, we could have the following patterns:

Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	1	1	1	0	0	0	1	1	1	1	0	0	0
2	2	2	2	0	0	0	0	0	0	0	2	2	2
0	0	0	0	2	2	2	0	0	0	0	1	1	1
0	0	0	0	1	1	1	2	2	2	2	0	0	0

Figure 2.4 Example of schedule assignment using shift patterns

p_1 = {Morning: Mon, Tue, Wed, Thu}

p_2 = {Night: Mon, Tue, Wed, Thu}

p_3 = {Morning: Fri, Sat, Sun}

p_4 = {Night: Fri, Sat, Sun}

Using p_1, p_2, p_3, p_4 we can obtain full coverage for an entire week. An example of a solution is illustrated in Figure 2.4. This means that for two weeks we could use the same patterns twice resulting in 8 variables. The number of variables in a problem is therefore strongly reduced in a cyclic schedule by grouping together N shifts into ST subsets. In this example we would end up with N/4 subsets = 28/4 = 8. Instead of scheduling n shifts to m staff members resulting in $\langle {n \atop m} \rangle$ solutions we end up with $\langle {n/ST \atop m} \rangle$ solutions for an employee.

The benefit of this planning method is that nurses know what shift pattern they are expected to make in advance. The drawback however is that it takes less into account individual preferences. There is an exchange that takes place between more fairness vs. less flexibility. Non-cyclic schedules take into consideration individual preferences when making a timetable. Depending on the problem type and the solution method there will be a preference for cyclic or non-cyclic schedules. This also depends on the scheduling practices maintained by hospital management. It can be preferable to maintain a cyclic schedule for nurses sometimes due to the size of a ward, or the high number of constraints.

2.2.6 Definition of hard and soft constraints

Constraints are divided into two classes: feasibility set constraints, and nonbinding constraints. Both are respectively referred to as hard and soft constraints. Miller et al. (1976) used the following definition for hard and soft constraints: “feasibility set constraints”, which define the sets of feasible nurse schedules, and nonbinding constraints, whose violation incurs a penalty cost that appears in the objective function. The hard constraints must be satisfied at all costs for a schedule to be feasible. Soft constraints are those that are desirable but which may need to be violated up to a certain degree in order to obtain a feasible solution. A soft constraint is complemented by additional side constraints that determine the size of the feasible region by stipulating upper and lower bounds for the constraint.

To measure the quality of a schedule the relative violation of soft constraints is generally used. The goal is always to schedule resources to meet the hard constraints while aiming at a high quality result with respect to soft constraints. Even soft constraints can only be violated up to a certain level; if an employee has a 35-hour work contract it could still be acceptable to work 40 hours, or even 50 hours. When this is 80 hours however this would result in an infeasibility problem. This is why it is necessary to define upper and lower bounds for soft constraints.

A soft constraint can be represented by a penalty function, a lower and upper bound/threshold value, as well as a range of acceptable values. In the mentioned example

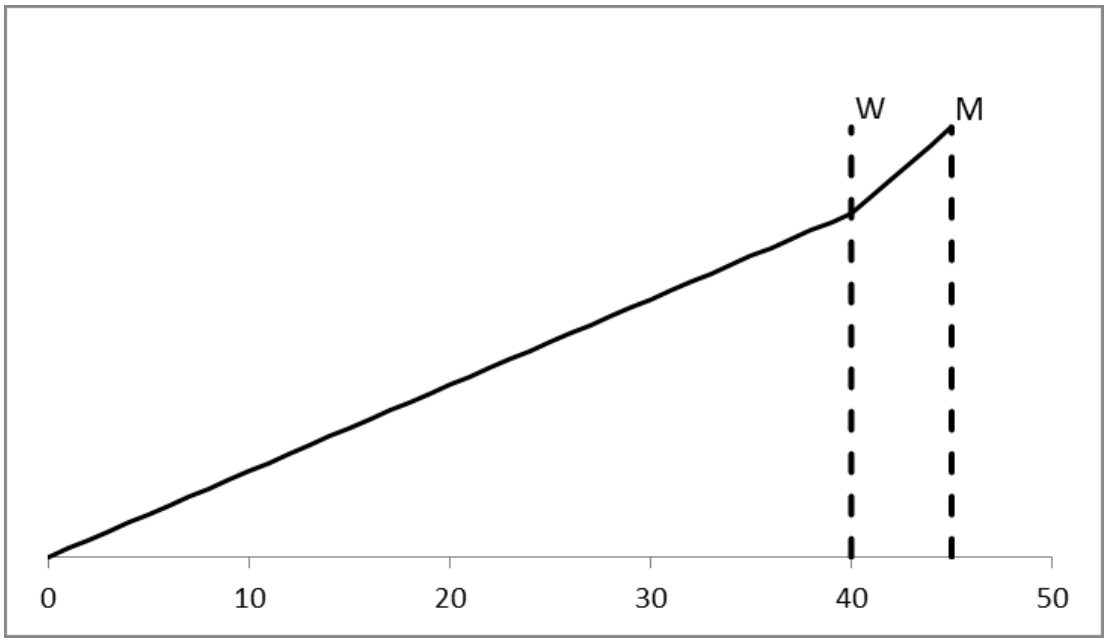


Figure 2.5 A piece-wise linear cost function and resulting feasible region

an acceptable range of values could be 20 – 60 hours. The level of satisfaction of the soft constraint is visible in the objective function.

Suppose a nurse has a 25-hour contract stipulating that she will be paid even if she is not on duty throughout the entire week. Legislation could stipulate that the hospital has to pay a fine if the nurse works more than 40 hours in a single week. Also suppose that legislation forbids the nurse to work more than 45 hours. Another constraint could be that a ward needs 40 hours of coverage by a single nurse.

Not scheduling would mean that the hospital pays for unused capacity, while overscheduling would result in a fine for the hospital. It is therefore necessary to define a soft constraint and two hard constraints. The first hard constraint could be defined as follows: *A minimum coverage of 40 hours is necessary*. The second hard constraint would be: *A nurse cannot work more than 45 hours*. The soft constraint then becomes: *Limit overtime for the nurse*.

Supposing that all other factors can be neglected and we have to schedule the nurse for 40 hours throughout the entire week. This can be formulated as a cost minimization function for the hospital in the following linear program (LP):

C: Number of hours of coverage needed

H: Number of hours worked

h_t: Number of hours worked within time period t $(1 \leq t \leq 3)$

Min: $h_1 + 10h_2 + 20h_3$

S.T.

$C = 40$	*(2.1)*
$H\text{-}C \geq 0$	*(2.2)*
$h_1 \leq 25$	*(2.3)*
$h_2 \leq 15$	*(2.4)*
$h_3 \leq 5$	*(2.5)*
$H\text{-}h_1\text{-}h_2\text{-}h_3 = 0$	*(2.6)*
$h_1, h_2, h_3 \geq 0$	*(2.7)*
h_1, h_2, h_3 : integer	*(2.8)*

The penalty function $h_1 + 10h_2 + 20h_3$ is intended to minimize the cost of scheduling the nurse for too many hours. Constraints (2.7) and (2.8) are Integrity constraints. Constraint (2.1) is considered a hard constraint since the LP would not be feasible if this constraint is not respected. The constraints (2.3) to (2.6) are considered as a single soft constraint. More exactly, (2.6) is the actual soft constraint but additional side constraint are necessary to define the allowed feasible region of the solution. The nurse can work anywhere between 25 to 45 hours. A soft constraint simply has a larger feasibility region, whereas a hard constraint only has a single acceptable value.

The resulting cost function is depicted in Figure 2.5. Notice that the objective function is defined as a piece-wise linear function. In reality Warner and Prawda (1972) reported that overtime costs show an exponential behaviour due to fines or other reasons.

Different penalty weights are included in the objective function to define the degree of violation associated to the soft constraint. If the nurse works anywhere beneath 25 hours the cost would remain constant. Working more than 25 hours means that the hospital would have to pay supplementary wages for every hour of overtime. If the nurse works more than 40 hours the hospital also has to pay a fine so the hourly cost would increase

even further. Line W shows the hard constraint for minimum coverage. Line M shows the legal hard constraint.

Since C = 40 and h_1 is bounded to 25 ($h_1 \leq 25$ constraint *(2.3))* we know that initially (2.6) will take the following values:

$H\text{-}C = 0 \Rightarrow H = 40 \Rightarrow H\text{-}h_1\text{-}h_2\text{-}h_3 = 0 \Rightarrow 25\text{-}h_2\text{-}h_3 = 40 \Rightarrow h_2\text{-}h_3 = 40\text{-}25$

$h_2\text{-}h_3 = 15$

Since h_2 is bounded to 15 we must have h_2 = 15 because the penalty cost is lower. If we solve the LP one variable at a time we will end up the the following optimal solution:

$C = 40;\ h_1 = 25;\ h_2 = 15;\ h_3 = 0;\ \text{Optimal value} = 175$

In nurse scheduling problems a number of constraints occur on a regular basis. Constraints are stipulated by hospital management, legislation and staff members. These constraints can be divided in different categories:

– *Hospital* (*Coverage) Constraints*: the number of staff members needed for every skill category and for every shift during the entire planning period more commonly referred to as personnel requirements.

Coverage Constraints		*Time Related Constraints*
Nurse skill levels and categories		Nurses workload (minimum/maximum)
Shift type(s) assignments (max. shift type, requirements for each shift types)		Consecutive same working shift (minimum/ maximum/exact number)
Constraints among groups/types of nurses, e.g., nurses not allowed to work together or nurses who must work together		Consecutive working shift/days (minimum/ maximum/exact number)
Other requirements in a shorter or longer time period other than the planning time period, e.g., every day in a shift must be assigned		
Constraints among shifts, e.g., shifts cannot be assigned to a person at the same time		
Requirements of (different types of) nurses or staff demand for any shift (minimum/maximum/exact number)		
Work Regulations Constraints		*Internal ward constraints*
Nurses preferences or requirements		Shift patterns
Nurses free days (minimum/maximum/consecutive free days)		Historical record, e.g., previous assignments
Free time between working shifts (minimum)		
Holidays and vacations (predictable), e.g., bank holiday, annual leave		
Working weekend, e.g., complete weekend		

Table 2.2 Cheang et al.'s list of common occurring constraints

– *Time Related Constraints*: all the restrictions on personal schedules, such as personal requests, personal preferences, and constraints on balancing the workload among personnel.

– *Work Regulations Constraints*. It sets a number of time related constraints for the nurses.

– *Internal ward constraints:* The practices applied by wards, such as attributing Saturday shifts to more junior staff members. These constraints are most often soft constraints.

Hospital management constraints, e.g. at least 1 nurse and no more than 2 nurses on the maternity ward during night shifts, are generally hard constraints. This is due to the nature of constraints. Regulatory constraints are generally also hard constraints because violation is not allowed. The algorithm also stipulates a number of Integrity constraints that are necessary for problem formulation. Constraints apply to each staff member.

Cheang et al. (2003) mention a list of constraints that occur commonly, which is illustrated in Table 2.2. All of these constraints mentioned amount to different levels of cost. In their ANROM model (Advanced Nurse Rostering Model) Burke et al. (2004b) mention an extended list of hard and soft constraints that resulted from the development of a general model for the nurse rostering model.

2.2.7 Interactive computing

A research trend in the field of nurse rostering problems is to give more importance to the interface used by schedulers and its impact on the quality of a timetable and the therefore related impacts on theoretical schedule quality and algorithmic performance. The field of human-computer interaction is a discipline that deals with the design, evaluation and implementation of such interactive systems.

Interactive computing refers to software that is told by humans what to do and is studied in different fields (Computer science, Cognitive psychology, Ergonomics, Artificial intelligence, Linguistics). Human-computer interaction has an impact on the ease-of-use and the theoretical aspects of a scheduling algorithm. The schedule quality in terms of the objective function will not end up being optimal because a number of theoretical aspects are sacrificed to gain an increased ease-of-use.

Throughout the 20th century this human-computer interaction gave rise to artificial intelligence. This can be done through the interface of a computer program or within the hard coding.

Expert systems that imitate the behaviour of users are classified as artificial intelligence systems. In the interface this means that the user can define search options, e.g. the constraints to be included in the cost function. Interactive interfaces are mentioned by several researchers as an important consideration for the development of their software (Burke et al. (2004b), Oldenkamp and Simons (1995)). The impact of these options is visible in the ease-of-use of the program and productivity of the user.

In the second sense, AI algorithms imitate human intelligence processes. For example, learning (store information and define rules for the use of this information), reasoning (use the rules to obtain approximate or definitive conclusions), and self-correction.

2.2.8 Existing decision support systems

Several different programs are already in use in different hospitals. Burke et al. (2004b) mention a number of programs developed by researchers and in use in hospitals. The table has been reproduced in Table 2.3[2]. Programs were either implemented in a single or several hospitals. They also mention algorithms that have been tested on real-life data. Programs mentioned being used in several hospitals have resulted in the development of software with interface modules that allow the use of several planning techniques. For a full reference of all the authors mentioned in Table 2.3, the reader can refer himself to Burke et al. (2004b). A few further authors that were not covered in their literature review

Not applied in practice but tested on real data	Applied in practice
	Approaches applied in 1 hospital
Abernathy et al. (1973)	Easton, Rossin, and Borders (1992): staffing
Berrada, Ferland, and Michelon (1996)	Jaszkiewicz (1997)
Petrovic, Beddoe and Vanden Berghe (2003)	Smith and Wiggins (1977)
Cheng, Lee, and Wu (1996, 1997)	Bellanti et al. (2004)
Okada and Okada (1988) and Okada (1992)	
Abdennadher and Schlenker (1999a, 1999b): INTERDIP	
de Vries (1987)	**Approaches applied in multiple hospitals**
Warner and Prawda (1972) and Trivedi and Warner (1976)	Warner, Keller, and Martel (1990): ANSOS,Warner (1976)
Miller, Pierskalla, and Rath (1976)	Jelinek and Kavois (1992): Medicus
Muslija, Gaertner, and Slany (2000)	Darmoni et al. (1995) Horoplan
Isken and Hancock (1991, 1998) and Isken (2004)	Meisels, Gudes, and Solotorevski (1997): EasyStaff
Jaumard, Semet, and Vovor (1998)	Meisels, Gudes, and Solotorevski (1996): TORANIT
Aickelin and Dowsland (2000) and Dowsland (1998)	Dowsland and Thompson (2000): CARE
Moz and Pato (2004)	Meyer auf'm Hofe (1997): ORBIS Dienstplan
Burke et al. (2001a, 2002, 2003)	Burke et al. (2001b, 2001, 1999), De Causmaecker and Vanden Berghe (2003): PLANE

Table 2.3 Burke et al.'s (2004b) overview of hospital software applied in practice and on real-life data

but who did present scheduling software are: Darmoni et al. (1994), Ozkarahan (1989), Weil et al. (1994).

These programs contain modules specifically aimed at helping schedulers in performing their task. The effects computer programs have on schedule-making has become a field of

[2] For a full reference of all the authors mentioned in Table 2.3, the reader can refer himself to Burke et al. (2004b). A few extra authors that were not mentioned but who did present scheduling software are: Darmoni et al. (1994), Ozkarahan (1989), Weil et al. (1994).

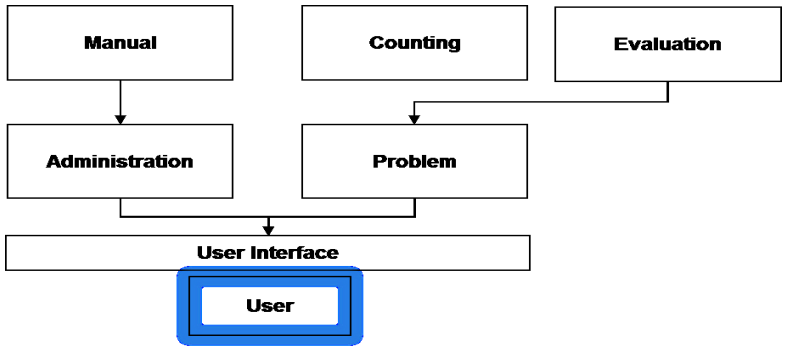


Figure 2.6 The general framework for scheduling decision support systems

study for human-computer interactivity. Warner (1976) called hand-made schedules the traditional approach. Nurse scheduling assisted by decision support systems has led to new methods of interaction. The baseline for decision support systems is that the program offers a number of different tools that provide information about what problems are encountered in a schedule to make work easier for schedulers. Roth and Woods (1989) have investigated the effects of decision support based tools on a cognitive task analysis. They found that decision support improved the task performance and that the variety of possible solutions for a specific problem increased. The cognitive tasks performed by schedulers have been described by Roth and Woods (1989) as well as by Mietus (1994).

As an example of a nurse scheduling system we can mention ANSOS, which is commercialized in the U.S (Warner, Keller and Martel (1991)). The system is built around a mathematical programming model. Costs for constraints can be adjusted by individual users. The program is also mentioned by Burke et al. (2004b) who provide a description of the four main modules of this system:

- The Position Control Module: keeps track of information for each employee (e.g. skills, types of shifts, maximum work stretch, etc.)
- The Scheduling Module: Based on all entered information it generates the schedule.
- The Staffing module: Determines the required staff levels for wards, based on demand data.
- The Management Reporting Module: Generates different types of reports

A general framework that can be extended to other decision support systems is illustrated in Figure 2.6. The Problem module can be used to create the schedule. The Administration Module is used in the early stages of the scheduling process to model job requirements. The tasks in the Adminstration Module include: Define shift-beginning and –ending times, determine number of staff that can be used in the planning period, enter pre-booked holidays and requested days-off, etc. The main algorithm for the creation of a schedule is contained within the Problem Module. This module program deals with counting, searching for personnel available for a shift, determine available staff, rank the availability of staff members, etc. Such specific tasks are generally handled by the mentioned submodules.

Other examples of nurse scheduling decision support systems are the ZKR system described by Mietus (1994) and the DSS system described by Randhawa and Sitompul (1993).

2.3 Literature review

In this partial review we present a number of articles that deal with nurse scheduling problems. Since there is a fairly large amount of literature available on nurse scheduling we will limit our review to articles in which computer systems were implemented in hospitals and articles that made a significant contribution to the subject.

Until the 60s, scheduling was mostly done by hand with the use of graphical tools. For an example off graphical tools and the considerations that have to be made you can consult Price (1970). Nowadays, methods from mathematical programming, articifial intelligence, heuristics and metaheuristics have all been employed to solve the nurse scheduling problems.

A number of trends are visible in the development of this field of research. From 1972 to 1988 researchers explored the subject by formulating appropriate models and implementing computer programs in hospital environments. The first important development was in the field of modelling. The research papers published since 1988 all

show similar modelling constraints. These constraints generally consider under- and overcoverage and nurse's preferences. The second major development was in the field of human-computer interactivity. Several authors mentioned that schedulers always made changes to the final solutions provided by software. Throughout this period programs always appeared with more and more features. Since 1988 the development of this field has been marked by new optimization methods and a few efforts to build a more general framework for nurse scheduling problems.

2.3.1 Resident scheduling literature

The first article that adressed resident scheduling was published by Ozkarahan (1994), to discuss their working conditions in the United States and propose a model that could be used as a decision support tool for making changes to working rules. Ozkarahan's model (1994) and the one introduced in the present study share similar constraints. Ozkarahan modelled the requirements of the residency program and the preferences of residents as to days off, weekends off and on-duty nights for a planning horizon of one week using a goal programming model.

Sherali et al. (2002) addressed the resident scheduling problem with constraints for ward staffing levels, skill requirements and residents' preferences. Their problem was modeled as a mixed integer program and was supported by different heuristic solution procedures to handle different scheduling scenarios.

Beliën and Demeulemeester (2006) describe a method using column generation for scheduling trainees at a Belgian hospital to solve the LP-relaxation of the long-term scheduling version using a decomposition scheme on the tasks. Cohn et al. (2009) studied a resident scheduling problem that spanned a 1-year planning period and included 3 different shift types. In addition, residents had to spend a number of time periods in different hospitals. Constraints concerned ward staffing levels, residency program requirements and resident preferences. The number of decision variables and constraints of the model was reduced because certain time periods had to be omitted from the model. The assignment of shifts in these time periods was done outside of the model. Cohn et al. (2009) modeled their problem as a goal programming model.

Ozkarahan and Topaloglu (2010) considered a resident scheduling problem over a planning period of four weeks considering constraints on residency program requirements, ward staffing levels and residents' preferences.

2.3.2 Published literature overviews

Over the years several different reviews have been published. Fries (1976) presented an overview of operations research techniques applied to health care, including scheduling techniques. Hung (1995) published a review of 128 articles on nurse scheduling. More recent reviews of the literature dedicated to this type of models are listed in Cheang et al**.** (2003) as well as in Burke et al. (2004b). Other literature reviews on hospital staff scheduling were furthermore performed by Bradley and Martin (1991) as well as Jelinek and Kavois (1992)*,* and Ernst et al. (2004)*.*

2.3.3 Mathematical programming

Research in mathematical programming has been done both in the general field of scheduling as well as on subproblems such as the nurse rostering problem. General scheduling problems have been treated in all fields of mathematical programming: *Linear programming, Integer programming, Mixed-integer programming, Non-linear programming, Goal programming approaches,* and *Network programming*. Nurse scheduling has been explored in a more limited number of fields: *Integer programming, Mixed-integer programming, Non-linear programming,* and *Goal programming approaches.*

Articles up to 1988 dealt mostly with the problem formulation. The newer methods that appeared in this field, have mainly focused on accelerating the search time without having to compromise the quality, and still obtaining an exact solution.

2.3.3.1 Non-linear programming

Warner and Prawda (1972) formulated a mixed-integer quadratic programming problem that was field-tested in six wards of a hospital with a cost function formulated to minimize hospital and legal penalties. They mentioned using a modified version of Balintfy and Blackburn's algorithm (Balintfy and Blackburn (1969)), which can be applied to goal-programming problems, based on a single-goal non-cyclic formulation. To accelerate the search process they simply used a primal resource-directive approach, decomposing the problem in quadratic subproblems.

Their problem formulation took into account skill categories and replacement between categories. Although their algorithm was field-tested they mentioned that they were insatisfied by the results obtained since a large amount of work still had to be done in order to provide satisfying schedules.

Before Warner and Prawda (1972) a number of researchers (Wolfe (1964), Wolfe and Young (1965a), (1965b)) studied nurse scheduling with dynamic staffing levels, assuming that the question of how high staffing levels should be was an integral part of the scheduling problem formulation. Warner and Prawda changed this formulation method and assumed that staffing level determination was independent from the scheduling problem. They considered that too much staff level constraints had a negative impact on the optimal solution. Instead they formulated constraints on preferred and minimum staffing levels. A more pratical reason was that staffing needs were determined in advance by hospital schedulers. It has led to nurse scheduling being formulated as deterministic problems with a constant workforce demand and constant availability.Their quadratic formulation was motivated by the fact that staff shortage costs often occur as an exponential function. As can be seen in Figure 2.7, they stated that the cost increased exponentially with the level of shortage, with staff shortage costs becoming zero if staffing levels were fully satisfied. This exponential behaviour was also described by Aickelin and Dowsland (2000) when testing a genetic algorithm as well as Berrada et al. (1996). A number of researchers refer to the mathematical program formulation used by Warner and Prawda (Miller et al. (1976), Ozkaharan and Bailey (1988), Okada and Okada (1988)) since skill categories and replacement between departments often occur in practice.

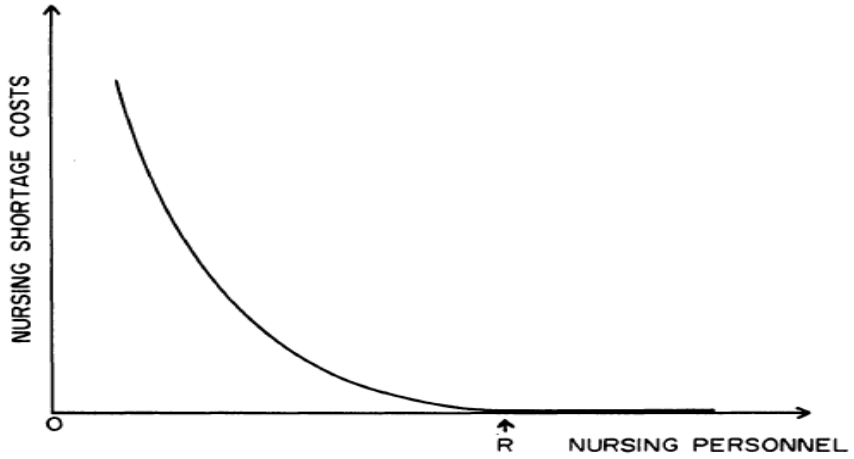


Figure 2.7 Warner and Prawda's exponential cost function for nursing care shortage

Warner (1976) reformulated the mathematical program of his previous research (Warner and Prawda (1972)) to include constraints corresponding to shift preferences as requested by nurses. The original mixed-integer quadratic programming problem solved by Balintfy and Blackburn's algorithm went through some modifications to make better use of the 0-1 structure of the model and to incorporate a certain number of improvements for the developed application. He made use of a heuristic exchange procedure to assign a set of a priori generated schedules to each nurses' schedule. The system ended up being implemented in 16 hospital wards, where the staff sizes ranged from 19 to 47 nurses.

A number of human-computer interactive principles were considered to ease the pratical use of the program were also mentioned by the author. Even when not all coverage constraints were necessarily met, the model was designed to solve the problem despite infeasibility. The program allowed also for minor manual changes by the scheduler. Warner also mentioned an improvement in productivity with time savings for a 6-week schedule with 46 nurses. The planning time decreased from 18-24 hours to about 1 hour, with a CPU time of 40-80 seconds. Warner also noted that a slight increase in the quality of schedules resulted from the use of the system.

This work has been acknowledged to be one of the first modern day works. His article is often mentioned for distinguishing three different fields within staff scheduling: staffing, scheduling and allocation of nurses. We can also note that he described five criteria for the scheduling problem. These criteria and scheduling approaches are mentioned in Table

Warner's scheduling approaches	
Traditional Approach	The schedules are generated by hand
Cyclical Scheduling	Generally provides good schedules but it is difficult to address personal requests
Computer Aided Traditional Scheduling	Allows a faster search for a higher number of good schedules.
Warner's scheduling criteria	
Coverage	Criteria pertaining to scheduling the preferred or required number of people for a task
Quality	How fair schedules are, judged by the violation of soft constraints
Stability	How the nurses perceive the schedules (in terms of consistency, predictable on/off days and weekend work)
Flexibility	How well the system can adapt to changes in the problem parameters
Cost	How many resources are consumed in making the decision: e.g. personnel manager's time or computer time.

Table 2.4 Warner's scheduling approaches and criteria

2.4. Several articles are indeed direct offspring of Warner's formulation (Miller et al. (1976), Ozkaharan and Bailey (1988), Okada and Okada (1988)).

2.3.3.2 Goal programming approaches

Arthur and Ravindran (1981) formulated a nurse rostering problem as a goal-programming problem which ended up being used by hospital schedulers for building real-life schedules. The goals used are indicated in Table 2.5. The hospital implementation was possible due to the fact that the complexity of the problem was very small: The problem describes the ward over a two-week scheduling period. The total number of variables per nurse was 5 for a two week period.

They constructed their solution with the help of a two-phase algorithm. In the first phase, the zero-one goal programming algorithm was used to assign shifts to a day-on/day-off pattern on to obtain an initial solution. The final solution then was created by using a heuristic procedure affecting the day-on/day-off patterns to nurses. A number of

Objectives (minimization)		Wgt
Priority 1	Minimum staffing requirements	Not mentioned
Priority 2	Desired staffing requirements	
Priority 3	Nurses' preferences	
Priority 4	Nurses' special requests	

Table 2.5 Arthur and Ravindran's minimization objectives

assumptions were used to reduce the initial problem's size, i.e. it was assumed that substitution between skill classes was not allowed and that each skill class could therefore be planned independently of the others.

The work of Arthur and Ravindran (1981) is considered an important contribution to nurse scheduling because they are considered as the first researchers to apply goal programming to this field. Their work was motivated by the fact that they considered that single-objective mathematical programming models were not flexible enough in terms of relative rankings assigned to various types of goals.

In their conclusion they make an interesting reference to human-computer interactivity. They admitted that the obtained schedule was not the final step in the process but that it needed to be manually refined by schedulers. Although an algorithm could produce an optimal schedule that took into account the various objectives, they admitted that preferences needed to be flexible: Meaning that schedulers should be able to manually adjust the model's priorities. This testifies that ease-of-use was considered by researchers as had been the case with Warner (1976), who allowed manual changes to schedules.

Musa and Saxena (1984) defined a 0-1 goal programming formulation which was solved by an algorithm based on the ideas presented in Balas's additive algorithm and coupled to a problem-specific version of the Garrod and Moore procedure. A test instance resulted in a formulation of 154 decision variables and 120 constraints for 11 nurses over a two-week period. For reference, the optimization priorities and their respective weights are given in Table 2.6. Their formulation was fairly similar to that used by Arthur and Ravindran concerning coverage requirements, and preference satisfaction. They also mentioned that the calculation time was approximately 28.3 seconds.

Objectives to minimize		Wgt
Priority 1	Achieve contracted days	9
Priority 2	Achieve minimum number of nurses's goals	7
Priority 3	Satisfy weekend preferences for full-time nurses, assign at least one weekend or two days off for full-time nurses, and do not violate the three-consecutive days off constraint. Achieve desired number of nurses for patient care	5
Priority 4	Satisfy weekend preferences for part-time nurses	3

Table 2.6 Musa and Saxena's minimization objectives

Their research is mentioned as a reference because it made an advance in the area of human-computer interactivity / artificial intelligence: It was the first computer program allowing users to make adjustments to the goals' relative weights from one period to another. Their application developed built on the conclusions of Arthur and Ravindran (1981) who suggested that it would be preferable for users to allow manual changed.

Ozkarahan and Bailey (1988) used a 0-1 GP model with a formulation based on the set-covering model. Their method used a two-phase system that integrated the time-of-day (TOD) problem with the Day-Off (DO) problem, as discussed in the initial works of Bailey (1985). Once both problems were solved to optimality, a heuristic would affect the optimal work patterns from the DOW problem to the TOD problem. Their decision support system incorporated several artificial intelligence techniques in the nurse scheduling process. The goals defined for their model are mentioned in Table 2.7.

In their findings, similarly to Arthur and Ravindran (1981) as well as Musa and Saxena (1984), it was reported a scheduler makes different changes in the algorithm's final solution. Their research was innovative because it integrated artificial intelligence techniques in the interface of a decision support system to facilitate manual changes (e.g. to be able to cope with changing assumptions, finding alternative solutions, manually changing solutions and evaluate different substitutes in case of two conflictive objectives). In their opinion most systems had shortcomings in the flexibility (i.e. artificial intelligence) level. As they mention: *"In reality, a nurse scheduler would form, perhaps with an AI front end program, the two or three goals that are psychologically most appealing."*

Objectives (Minimization)		Wgt
Priority 1	Minimize deviations from the required number of nurses needed for each day of a weekend	Not mentioned
Priority 2	Minimize deviations in both directions from the limited staff size	
Priority 3	Minimize deviations in both directions from the required number of nurses for Thursday through Sunday	

Table 2.7 Ozkarahan and Bailey's minimization objectives

It was reported that hospital management was satisfied during implementation because producing the system-generated schedules was more efficient than the paper-and-pencil schedules and because over- and understaffing were minized. Nurses also appeared appreciative of the system because single days on duty, and separated days off were eliminated.

For their research Berrada et al. (1996) used data taken from a Canadian hospital. In their works, Berrada et al. (1996) compared a sequential algorithm, an equivalent weights algorithm and a tabu search algorithm. The sequential and equivalent weight techniques made use of a branch-and-bound algorithm. They were able to split the problem into three single-shift problems since each nurse always worked the same shift, thus reducing the complexity of their model.

Their work provides an interesting comparative on the performance of different algorithms. They reported a higher CPU time for the tabu search than for the two exact procedures which they related to the experimental status of their tabu search. It should be noted that the tabu search also needed to be formulated as a non-linear program, thus resulting in a higher solution time. The way in which the problems were formulated was different as well; the deviation measuring constraints had to be formulated as non-linear constraints more demanding in their solution time. Schedulers were satisfied with most of the results and even decided to implement some of them.

Azaiez and Al Sharif (2005) used a 0-1 goal program that accounts for hospital objectives and nurses' preferences. Their research mentions wards of varying size. They divided a ward in subgroups according to nurse's qualifications and workloads. For the mentioned example of 13 nurses they obtained a problem with 1135 hard constraints and 1068 soft constraints consisting of 1092 binary decision variables and 2054 non-negative deviation variables.

They provided interesting information concerning the testing of their model. In most of the tested cases optimal solutions were obtained. Where this was not the case violations only occurred for soft constraints with the lowest importance weights. Their largest running time was approximately 20 minutes for a ward with 22 nurses. They reported that savings of 14% in over-time costs had been obtained over a 6-month testing period. According to

their estimations it would be possible to realize savings of $ 100 000 once implemented. They also pointed out that one of the problems perceived by the nurses was that less overtime deprived them of a good source of income.

2.3.3.3 Branch-and-price approaches

Jaumard et al. (1998) presented an exact algorithm that made use of techniques specifically aimed at accelerating the branching process. Based on a goal-programming formulation, their aim is to minimize salary costs, nurse preferences, and experienced/less-experienced staff while satisfying ward coverage. They defined a master program to find a configuration of schedules and generated new feasible schedules by solving a resource constrained shortest path problem based on a depth-first process.

Their algorithm was tested on a real-life instance from a Canadian hospital stretching a 6-week planning horizon with 41 permanent nurses. The system was capable of providing a linear relaxation solution for the instance concerned in less than 40 minutes. They mentioned that when the same problem is solved by a partial branch-and-bound it took approximately 16,5 hrs to get a solution. Creating a paper-and-pencil schedule took a full working day for the scheduler. In terms of search speed this algorithm could be considered slow but they made mention of this fact. They explained that the search could be stopped by the scheduler during any point of the process and still obtain satisfactory feasible solutions.

In their published research Bard and Purnomo (2005a, 2005b) used a multi-objective column generation approach that tries to ensure sufficient coverage while taking into account staff preferences. Their approach allowed for infeasible schedules by the use of outside nurses to fill up gaps in the schedule. Their algorithm was tested on instances of up to 100 nurses for a four week planning horizon producing schedules in 10 minutes. To reduce the size of rows of the model staff demand was expressed in periods rather than shifts. They generate only columns that are feasible for nurses to reduce the number of candidate schedules. To build new columns a double swapping heuristic was used where periods with staff shortages or surpluses were preffered for exchanges. Reports tell us that the algorithm has been implemented at different hospitals.

In a latter paper Bard and Purnomo (2007) used a Lagrangian decomposition scheme to lower the number of variables for the same instances. Maenhout and Vanhoucke (2007) investigated several different branching strategies to speed up the search time for an exact solution. These papers were not implemented in hospitals but show the trend that takes place in finding new solving methods for the existing problems.

2.3.4 Artificial intelligence methods

In the 1980's and later, artificial intelligence techniques for nurse scheduling (declarative approaches, constraint programming, expert systems) were investigated with some success. Although it has not resulted in any export systems up-to-date, a number of different methods have been proposed. Chiaramonte and Chiaramonte (2008) proposed a heuristic using a competitive agent-based negotiation that focused on nurses' preferences. Some of these approaches are still relevant to today's research issues (Chan and Weil, (2001); Chiarandini et al. (2000); Meyer auf'm Hofe (1997)).

Weil et al. (1995) presented a constraint programming (CP) model for nurse scheduling with the purpose of demonstrating the applicability of this technique as both a modelling and resolution tool. To model the problem they combined object programming and CP techniques as a tool for which they used ILOG-Solver. They defined an object class called "nurse" that contains information about the nurses' identity (name, title, etc…) and the constrained variables used to generate the schedule.

The scheduling horizon of their work was a 14-day calendar. They tested their system on two examples. The first one included less than 12 nurses, with a theoretical complexity of 10^{100}. The second problem contained 30 nurses from the same skill class. The problem contained 420 variables and 1470 constraints and had a theoretical complexity of 10^{250}. They reported a solution time of 12 seconds.

Okada and Okada (1988) developed a scheduling algorithm based on declarative programming with help of the logic programming language Prolog that followed a manual-

like method. Okada (1992) built on this method to develop a system that was able to handle constraints that were specific to different hospitals.

2.3.5 Heuristic methods

Miller et al. (1976) used a cyclic descent algorithm to solve schedules. They considered an objective function aimed at mimimizing staffing over- and undercoverage and nurse dissatisfaction in a single-objective formulation. The algorithm starts with an initial configuration of the schedule. From the nurses on the ward it selects one nurse to improve the schedule. Only if the algorithm finds a schedule that results in a lower cost, the lowest present cost and the best schedule are updated. If no cheaper configuration can be found during *I* consecutive tests, the algorithm stops.

Their contribution was compared to a branch-and-bound algorithm and contained a good description of the number of alternative schedules that can be obtained. The example they mention is when 4 days are given off over a 14-day period. The number of alternative schedules then amounts to $\binom{14}{4}$ > 1001 schedules. They mentioned that this number is lower when feasibility set constraints (hard constraints), previous schedules and special requests are taken into account.

They performed their algorithmic comparison on a 4-nurse problem, over a 20-day period. The optimal solution was a cost of 7,55 and with the cyclic descent they obtained a 12,3, with a CPU-time of respectively 10,509 sec vs. 0,367 sec. It should be noted that the upper-bound in the branch-and-bound algorithm was the final solution generated by the algorithm and was used to reduce the search time.

The also discuss their results of a general testing phase over a six-month period on groups of 5-7 nurses. They mentioned that the schedule was equitable over a planning horizon and that the algorithm generated a lower number of split weekends, something which was unwanted by the nurses. For the problems tested the generation time was generally between 2.5 sec. – 8 sec. Their estimation was that each example contained an average of 200 feasible schedules.

2.3.6 Metaheuristics

The group metaheuristics contains a number of different fields. Up-to-date approaches have been developed using local search and population-based metaheuristics. Methods presented for local search metaheuristics are limited to tabu search algorithms since only these have been implemented in decision support systems. Some population-based algorithms have also been presented although they have currently not been integrated directly in decision support systems.

2.3.6.1 Tabu search

Burke et al. (1998) developed a decision support system named Plane for Belgian hospitals. They reported that the number of constraints was higher than encountered in most literature problems researched. Their objective function formulation focused on the planning requirements including under- and overstaffing, and taken into account the preferred requirements. They mention that the cost function could be modelled by the scheduler himself.

Their system was based on the use of a tabu search with diversification strategies. They used random initialization to create new schedules and after the tabu search process two diversification strategies could be used: If necessary they relaxed a number of constraints to improve the weekend coverage. They also used a greedy shuffle to list all pairs of moves possible until the list was depleted. They mentioned that this latter technique was taken from seeing the scheduler at work. Schedulers were often capable to improve the results of the search by making a small exchange between two staff members.

They compared the performance of their tabu search to a steepest descent (SD) algorithm. Their results based on an example for a ward with 20 nurses and a planning horizon of 4 weeks showed that the steepest descent method was outperformed by the tabu search. Their conclusion was that schedulers often put more emphasis on a higher quality solution than the search time needed.

Dowsland (1998) used the strategic oscillation techniques proposed by Glover and Laguna (1993) to explore new techniques for solving nurse scheduling problems. Ferland et al. (2001) also make mention of a tabu search for providing schedules. Burke et al. (2001) used a hybrid tabu search coupled to a genetic algorithm to explore a solution approach that produces more robust solutions.

2.3.6.2 Genetic algorithms

A number of papers have been published on the use of genetic algorithms for generating schedules but none of them have actually been implemented in hospitals. Aickelin and Dowsland (2000) tested a genetic algorithm on an instance and encountered an exponential behaviour which diminishes the efficiency of a standard genetic algorithm. Maenhout and Vanhoucke (2006b) provided a comparison of different crossover operators for genetic algorithms.

2.3.6.3 Other metaheuristics

Maenhout and Vanhoucke (2005, 2006a) developed experimental algorithms for nurse rostering problems. They presented an electromagnetism meta-heuristic and a scatter search algorithm. Goodman et al. (2007) reported a hybrid grasp-knapsack algorithm. These papers show that newer models have mainly been focused on accelerating the search process.

Chapter III – Formulation and solution approach

3.1 Introduction

The contribution of this Chapter is twofold. In the first section, it models and presents a new problem, the Resident Scheduling Problem (RSP), whose main objective is to assign a set of shifts (typically 12- or 24-hours shifts) to staff members and where an employee should work around a single shift per four days in order to minimize costs. The Resident Scheduling Problem is a real-life scheduling problem which arises often in hospitals or in other contexts where manpower needs to be assigned to shifts to cover non-stop activities.

The second section proposes an efficient method to obtain a good quality solution, called the Manually Restricted Space (MRS) for the Resident Scheduling Problem. This construction heuristic, inspired from the manual techniques followed by schedulers, proposes an initial solution by selecting a region around a reference solution and exploring neighbouring solutions. The Manually Restricted Space is a space restriction approach thus providing an initial solution for a neighbourhood-based heuristic; instead of transforming the solution space, a region is selected around a reference solution.

The concept of delimiting the solution space to be explored is a logical way to deal with large scale problems. In his doctoral thesis Pecora (2008) defines this concept as the Restricted Space (RS): *"a subspace of the universal set of solutions which has two highly desirable characteristics (1) it should be small enough to be thoroughly explored and (2) it should have a high possibility of containing near-optimal solutions."* Several authors can be referred to as examples of studying the problem structure to reduce the solution space. Aickelin and Dowsland (2000) used a genetic algorithm to exploit the structure of their problem. Maenhout and Vaenhoucke (2009) used a branching strategy that branched on the original variables and progressively fixed the assigments in the individual schedules. To advocate their approach they mentioned that personnel scheduling problems are compatible with pricing problems.

The aim of the present study was to deduce the common characteristics and integrate the schedulers' knowledge in search methods. To reflect these considerations the manual techniques used by schedulers were integrated in a formal framework and ultimately led to the formalisation of the Manual Scheduling Process (MSP). The MSP reflects the schedulers' techniques, and their common scheduling behaviour. To integrate the MSP in a search method the construction heuristic was developed. The construction heuristic was obtained by combining different OR (Operational Research) heuristics. Finally, this resulted in the Manually Restricted Space.

3.2 Problem description

The constraints used in this Chapter have been formulated to provide a general model suitable for residency programs with ward-shifts. However, these constraints can be modified to take into account different types of residency programs such as resident home-carers or residents on standby at home. During our testing phase we have encountered instances where some postulates of this model had to be modified to be able to comply with hospitals' regulations. In this chapter not all constraints used within the schedule planning software will be mentioned to make the model easier to understand. The complete mathematical model can be found in Appendix A.

3.2.1 Problem dimensions

The characteristics previously described in § 2.2.1 and § 2.2.3 for the nurse scheduling problem, are the following for the resident scheduling problem:

1. *The planning horizon is 28 days;*
2. *Shifts only have to be planned for night shifts (day shifts are not planned by resident scheduler's). During the week shifts start at 8 P.M. and end at 8 A.M. During the weekend shifts start at 8 A.M. and end at 8A.M. Shift length is therefore 12 hours on weekdays and 24 hours on weekends;*
3. *There are 1 or more wards for each scheduling problem;*

4. *Minimum coverage level(MC) per ward: 1 resident;*
5. *Infeasible solutions are allowed at a cost, replacement shifts are done by doctors.*

Weekend shifts are the most difficult to assign because of their 24-hour length. Demand fluctuation is strong during weekends; on Saturdays demand is at its' lowest, making it the most difficult day because, according to residents, it is very monotonous and boring. Also, note that coverage infeasibility is allowed whenever no resident can be scheduled. If this occurs, the feasibility of the solution is restored by adding replacement shifts which will be done by a doctor. However, a doctor that does a night shift leads to a high penalty cost in the objective function. Therefore, a doctor should only be called upon if necessary. In the data samples that we gathered the frequency of shifts done by doctors was rather low.

Table 3.1 summarizes the constraints encountered in this problem. The next sections introduce formulations for each of these constraints groups. The integrity constraints model the allowable characteristics of the Resident Scheduling Problem. It includes the number of staff members needed for every skill category and for every shift during the entire planning period. The legal constraints model all the restrictions on personal schedules, such as personal requests and personal preferences. The hospital constraints contain the practices applied by wards, such as attributing Saturday shifts to more junior staff members and constraints on balancing the workload among personnel. We will use numerical examples to clarify the main constraints.

Integrity Constraints
Required number of residents for each day
Maximum of one shift assigned to a person at the same time
One day-type assigned to a person at the same time

Internal ward constraints
Proportionate dispersion of days among residents
Proportionate dispersion of total shifts among residents

Legal Constraints	
No more than the maximum number of shifts	Respect resting cycles after night shift
Respect requested days off	No more than two weekends per scheduling period
Respect congress days	No more than two consecutive weekends
Respect holidays	

Table 3.1 Resident Scheduling Problem constraints

The characteristics of the numerical example are the following:

— *The planning horizon is 28 days*

— *The hospital has 18 residents working on 2 wards: Emergency and Paediatrics. Resident 1 to 8 can only work at the Emergency ward and resident 11 to 18 at the paediatrics ward. Residents 9 and 10 can work on both wards. At each ward MC = 1*

3.2.2 Objective function

The objective function designed for this model uses slack variables to calculate the penalty cost function. Penalty coefficients are selected to adequately weigh each of the slack terms. This function is defined as follows:

$$Minimize: \tag{3.1}$$

$$PEN\_MC \sum_{t=1}^{n}\sum_{k=1}^{W} d\_MC_{kt}^{-} + PEN\_MAX \sum_{i=1}^{m} d\_max_i^{+} + PEN\_RDO \times \varrho \times \frac{1}{\sum_{i=1}^{m} d_{rdo_i}^{+}} + PEN\_C \sum_{i=1}^{m} d\_c_i^{+}$$
$$+ PEN\_H \sum_{i=1}^{m} d\_h_i^{+} + PEN\_EXC\_72AFT \sum_{t=1}^{n} d\_exc\_72aft_i^{+} + PEN\_WKD \sum_{i=1}^{m} d\_WKD_i^{+}$$
$$+ PEN\_CWKD \sum_{i=1}^{m} d\_CWKD_i^{+} + PEN\_TSD^{-} \sum_{i=1}^{m} d\_TSD_i^{-} + PEN\_TSD^{+} \sum_{i=1}^{m} d\_TSD_i^{+}$$
$$+ PEN\_DISP^{-} \sum_{i=1}^{m} d\_DISP_i^{-} + PEN\_DISP^{+} \sum_{i=1}^{m} d\_DISP_i^{+}$$

Apart from the penalty values the objective function also contains the parameter ϱ which sets the number of days off to be respected.

3.2.3 Integrity constraints

Integrity constraints define the constraints among shifts and are used for the algorithm used for the search process.

Required number of residents for each day

Constraint *(3.2)* has been formulated to handle undercoverage, i.e. allow a shortage in staff coverage. In some scheduling problems staff shortages can lead to infeasible solutions. To avoid this from occurring, undercoverage is allowed by defining two different levels of coverage. The first level is minimum coverage, the smallest number of staff members that are needed at a ward, and is reflected by the variables containing MC. The second level is the real coverage level, reflecting the actual number of residents that will be scheduled on a ward. The deviation between the minimum coverage and real coverage is measured by $d\_MC_{it}^{-}$. We consider every resident *i* that belongs to the same department, denoted as $\sum_{i \in W(k)} x_{it}$.

$$\sum_{i \in W(k)} x_{it} + d\_MC_{kt}^{-} - d\_MC_{kt}^{+} = MC_{kt} \qquad \forall k; \forall t \qquad (3.2)$$

This constraint furthermore considers that the same shift cannot be assigned twice.

For example, for the 1st day of the planning period constraint *(3.2)* for the emergency unit (denoted as *k=1*) can be written as:

$$x_{1;1} + x_{2;1} + \ldots + x_{8;1} + x_{9;1} + x_{10;1} + d\_MC_{1;1}^{-} - d\_MC_{1;1}^{+} = MC_{1;1}$$

For the paediatrics unit (*k=2*) this constraint is:

$$x_{11;1} + x_{12;1} + \ldots + x_{18;1} + x_{9;1} + x_{10;1} + d\_MC_{2;1}^{-} - d\_MC_{2;1}^{+} = MC_{2;1}$$

Maximum of one shift assigned to a person at the same time

Since residents can be active on different wards we have to assure that any person cannot be assigned to more than one shift at a time. Constraint *(3.3)* considers all different wards at which a resident can work by defining the summation of all a resident's shift variables on different wards.

$$\sum_{k}^{K} x_{i\in \mathrm{W(k)t}} \leq 1 \qquad \forall\, \mathrm{i}; \forall\, \mathrm{t} \qquad (3.3)$$

For example, the residents 9 and 10 can work on both departments. Therefore for *t=1*, we would have to include the following constraints:

$\mathrm{x}_{9;1} + \mathrm{x}_{9;2} \leq 1$

$\mathrm{x}_{10;1} + \mathrm{x}_{10;2} \leq 1$

One day-type assigned to a person at the same time

Constraints *(3.4)-(3.8)* set the day-type of a resident on day *t*. Where the binary variables in *(3.4)* and *(3.5)* will depend on the parameters of the constraints (3.6)-(3.8). The parameters and variables are mentioned below:

Parameters

RDO_{it}	*1*	*If resident i requested a day off at day t*
	0	*Otherwise*
C_{it}	*1*	*If resident i is at conference at day t*
	0	*Otherwise*
H_{it}	*1*	*If resident i is on holiday at day t*
	0	*Otherwise.*

Decision variables

x_{it}	*1*	*If resident i is working at day t*
	0	*Otherwise*
rdo_{it}	*1*	*If resident i requested a day off at day t*
	0	*Otherwise*
c_{it}	*1*	*If resident i is in conference at day t*
	0	*Otherwise*
h_{it}	*1*	*If resident i is on holiday at day t*
	0	*Otherwise*

$$\mathrm{x_{it}} + \mathrm{c_{it}} + \mathrm{h_{it}} \leq 1 \qquad \forall\, \mathrm{i}; \forall\, \mathrm{t} \qquad (3.4)$$

$$\mathrm{rdo_{it}} + \mathrm{c_{it}} + \mathrm{h_{it}} \leq 1 \qquad \forall\, \mathrm{i}; \forall\, \mathrm{t} \qquad (3.5)$$

$$rdo_{\mathrm{it}} = RDO_{\mathrm{it}}\,,\, c_{\mathrm{it}} = C_{\mathrm{it}}\,,\, h_{\mathrm{it}} = H_{\mathrm{it}} \qquad \forall\, \mathrm{i}; \forall\, \mathrm{t} \qquad (3.6) - (3.8)$$

The problem has been formulated in such a manner that a number of specific day-types can prohibit a resident from working night shifts on particular days. Constraints (3.4)-(3.8) are therefore intricately linked to constraints (3.9) and (3.11)-(3.16). These are legal constraints that set the limitations on day-types. The variables $\mathrm{rdo_{it}}$, $\mathrm{c_{it}}$, $\mathrm{h_{it}}$ will therefore also be used in the constraints (3.11) – (3.16) which are intended to assure the respect of the day-types.

The variables $\mathrm{x_{it}}$ and $\mathrm{rdo_{it}}$ have not been mentioned in the same equation but rather in separate equations - *(3.4)* and *(3.5)* - for the following reason; it might occur that $\mathrm{rdo_{it}} = 1$ but that a resident would have to work despite this value. Therefore, $\mathrm{rdo_{it}}$ and $\mathrm{x_{it}}$ do not exclude each other, since it is possible to not assign a day off to a resident who requested one. On the other hand, $\mathrm{rdo_{it}}$, $\mathrm{c_{it}}$ and $\mathrm{h_{it}}$ cannot occur simultaneously. We therefore have to define the constraint *(3.5)*, $\mathrm{rdo_{it}} + \mathrm{c_{it}} + \mathrm{h_{it}} \leq 1$, to avoid $\mathrm{rdo_{it}}$, $\mathrm{c_{it}}$ and $\mathrm{h_{it}}$ from happening simultaneously. For the same reason, we also have to assure that $\mathrm{x_{it}}, \mathrm{c_{it}}$ and $\mathrm{h_{it}}$ do not occur simultaneously, by defining constraint *(3.4)*, $\mathrm{x_{it}} + \mathrm{c_{it}} + \mathrm{h_{it}} \leq 1$.

The parameters $RDO_{\mathrm{it}}, C_{\mathrm{it}}, H_{\mathrm{it}}$ are set to indicate which day-type is valid, if, for example, on *t=5* resident 3 is attending a conference:

$\mathrm{x_{3;5}} + \mathrm{c_{3;5}} + \mathrm{h_{3;5}} \leq 1$

$\mathrm{rdo_{3;5}} + \mathrm{c_{3;5}} + \mathrm{h_{3;5}} \leq 1$

$rdo_{3;5} = 0\,,\, c_{3;5} = 1\,,\, h_{3;5} = 0$

3.2.4 Legal constraints

No more than the maximum number of shifts allowed per resident

Constraint *(3.9)* defines the number of shifts allowed, which is proportional to the number of days of availability of a resident.

$$\sum_{t=1}^{n} x_{i\in \mathrm{W(k)}t} + d\_max_i^- - d\_max_i^+ = \ MAX_i \qquad \forall \text{ i} \qquad (3.9)$$

Conferences and holidays are part of the entitled legal days off and diminish the number of availability days. The parameter MAX_i is the maximum number of shifts and is determined by equation *(3.10)*:

$$MAX_i = \left\lceil \frac{(28 - \ legal\ off\ days)}{28} * 6 \right\rceil \qquad (3.10)$$

The maximum number of legally allowed shifts is 6. This number is multiplied by the ratio (*remaining days/ total duration of planning horizon*) to obtain the number of shifts a resident can work in reality.

Suppose that resident 9 attends a 3-day conference and takes a week of holiday. Constraint *(3.10)* then becomes:

$$MAX_9 = \left\lceil \frac{(28 - \ 10)}{28} * 6 \right\rceil = \ \lceil 3{,}8571 \rceil = 4$$

$$x_{9;1;} + x_{9;1} + \ \ldots + x_{9;28} + \ x_{9;28} + d\_max_7^- - d\_max_7^+ = \ 4$$

Respect requested days off

According to the legal convention a resident is entitled to 2 requested days off per scheduling period. However, the number of requested free days is generally much higher than the 2 days allowed. The constraints *(3.11)* and *(3.12)* are therefore ment to avoid any violation of a requested free day and to allow more flexibility to the model because the resident scheduler sets the number of days off requests that have to be respected. In the objective function ϱ sets the number of days off to be respected. Any violations to constraint (3.11) will be inversely penalized with respect to ϱ in the objective function. Constraint *(3.15)* has therefore been formulated to handle penalties all requests for days off.

$$x_{it} + \ rdo_{it} \ - rdo\_exc_{it} \leq 1 \qquad \forall \text{ i; } \forall \text{ t} \qquad (3.11)$$

$$\sum_{t=1}^{n} rdo\_exc_{it} \ + \ d\_rdo_i^- - d\_rdo_i^+ \ = 0 \qquad \forall \text{ i} \qquad (3.12)$$

Constraint *(3.11)* is defined for each resident and each day in order to count the number of times a requested day off is not respected. Constraint *(3.12)* is the summation that has been formulated to determine if less or more than the ϱ requested preferred days have been respected. If in the objective function $\sum_{t=1}^{n} rdo\_exc_{it} \leq \rho$ the penalty will be higher than for $\sum_{t=1}^{n} rdo\_exc_{it} \geq \rho$. In the former case, less than the requested number of days have been respected and should therefore be penalized more strictly.

For example, if resident 16 handed in a request form indicating that he wished to be off during the first weekend of the planning period and if $\rho = 0$:

$x_{16;5} + rdo_{16;5} - rdo\_exc_{16;5} \leq 1$

$x_{16;6} + rdo_{16;6} - rdo\_exc_{16;6} \leq 1$

$x_{16;7} + rdo_{16;7} - rdo\_exc_{16;7} \leq 1$

$rdo_{16;5} = rdo_{16;6} = rdo_{16;7} = 1$

$rdo\_exc_{16;1} + rdo\_exc_{16;2} + \ldots + rdo\_exc_{16;28} + d\_rdo_{16}^{-} - d\_rdo_{16}^{+} = 0$

Respect conference days

When a resident attends a conference he cannot be expected to work. Constraint *(3.13)* is therefore defined for each resident and each day in order to count the number of times a conference day is not respected. Constraint *(3.14)* is the summation formulated to count the number of times a conference day is not respected.

$$x_{it} + c_{it} - c\_exc_{it} \leq 1 \qquad \forall \text{ i}; \forall \text{ t} \qquad (3.13)$$

$$\sum_{t=1}^{n} c\_exc_{it} + d\_c_i^{-} - d_{c_i}^{+} = 0 \qquad \forall \text{ i} \qquad (3.14)$$

For example, the conference resident 9 attended took place during the first week of the planning period:

$x_{9;2} + c_{9;2} - c\_exc_{9;2} \leq 1$

$x_{9;3} + c_{9;3} - c\_exc_{9;3} \leq 1$

$x_{9;4} + c_{9;4} - c\_exc_{9;4} \leq 1$

$c_{9;2} = c_{9;3} = c_{9;4} = 1$

$c\_exc_{9;1} + c\_exc_{9;2} + \ldots + c\_exc_{9;28} + d\_c_9^{-} - d\_c_9^{+} = 0$

Respect holidays

Constraint *(3.15)* is defined for each resident and each day in order to count the number of times a holiday is not respected. Constraint *(3.16)* is the summation that counts the number of times constraint *(3.15)* is not respected.

$$x_{it} + h_{it} - h\_exc_{it} \leq 1 \qquad \forall i, \forall t \qquad (3.15)$$

$$\sum_{t=1}^{n} h\_exc_{it} + d\_h_{it}^{-} - d_{h_{it}}^{+} = 0 \qquad \forall i \qquad (3.16)$$

When a resident is on holiday the collective agreement does not allow a violation of his holidays. When leaving on holiday, the resident also has to be off duty the weekend before and after his week of holidays. This therefore imposes a limitation on the attribution of weekend shifts to residents. This constraint is mentioned in Appendix A.

For example, the week of holiday resident 9 took was during the last week of the planning period:

$x_{9;22} + h_{9;22} - h\_exc_{9;22} \leq 1$

$x_{9;23} + h_{9;23} - h\_exc_{9;23} \leq 1$

…

$x_{9;28} + h_{9;28} - h\_exc_{9;28} \leq 1$

$h_{9;22} = h_{9;23} = \ldots = h_{9;28} = 1$

$h\_exc_{9;1} + h\_exc_{9;2} + \ldots + h\_exc_{9;28} + d\_h_{9}^{-} - d\_h_{9}^{+} = 0$

Respect resting cycles after night shift

The constraints *(3.17)-(3.21)* are necessary to respect the different cycles of being on- and off-duty.

$$x_{i(t-2)} + x_{i(t-1)} + x_{it} + x_{i(t+1)} + x_{i(t+2)} \leq 1 \qquad \forall i ; \forall t \qquad (3.17)$$

$$x_{i(t-3)} + x_{it} - exc\_72bef_{it} \leq 1 \qquad \forall i ; \forall t \qquad (3.18)$$

$$x_{i(t+3)} + x_{it} - exc\_72aft_{it} \leq 1 \qquad \forall i; \forall t \qquad (3.19)$$

Cycle type	1	2	3	4	5	6	7	8	9	10
48-hours resting cycle	0	0	1	0	0	1	0	0	0	0
72-hours resting cycle	0	0	1	0	0	0	1	0	0	0

Table 3.2 Rest cycles as described by the legal convention

$$\sum_{t=1}^{n} exc\_72aft_{it} + d\_exc\_72aft_i^- - d\_exc\_72aft_i^+ = \mathbb{Q} \qquad \forall \text{ i} \tag{3.20}$$

$$\sum_{t=1}^{n} exc\_72bef_{it} + d\_exc\_72bef_{it}^- - d\_exc\_72bef_{it}^+ = \mathbb{Q} \qquad \forall \text{ i} \tag{3.21}$$

The collective agreement prescribes two types of resting cycles, which are illustrated in Table 3.2. The first cycle type is a 72-hours rest period after each night shift, to allow the resident's circadian rhythm to come back to a normal rhythm. However, a second cycle type allows the limitation of the rest period to 48 hours. The legal convention allows a resident to work this second cycle type once every period. These cycle types will be referred to as 72-hours and 48-hours rest cycles respectively. In other words, a resident will only work a shift once every five days if working only 72-hours cycles. In constraints *(3.20)* and *(3.21)* the number of allowed 48-hours shifts -without incurring any penalties- in the resident's working pattern is set by $\mathbb{Q}$,. Although the legal convention limits the number of 48-hours shifts, some wards give residents a second 48-hours shift. This is allowed in the eyes of the legal convention if the resident accepts this workload. Normally $\mathbb{Q} = 1$, although it could set to be a higher number.

The slack variables $d\_exc\_72aft_{it}^+$ and $d\_exc\_72bef_{it}^+$ in (3.18) and (3.19) count the number of 48-hours resting cycle result and constraints (3.20) and (3.21) force the summation of $\sum_{i=1}^{m} exc\_72aft_{it}$ and $\sum_{i=1}^{m} exc\_72bef_{it}$ to penalize all 48-hours resting cycles over $\mathbb{Q}$.

A value of $\mathbb{Q}$ that is larger than 1 is unwanted, but cannot always be avoided. Table 3.3 shows the patterns that will be allowed as a function of $\mathbb{Q}$. For example, if no 48-hours resting cycles are allowed only pattern 1 would respect constraints (3.17)-(3.21). If $\mathbb{Q}$=1 pattern 2 would be allowed to occur, whereas this pattern would lead to a penalty if $\mathbb{Q} = 0$.

		1	2	3	4	5	6	7	8	9	10	11	12	13
Pattern 1	Q=0	0	0	1	0	0	0	1	0	0	0	1	0	0
Pattern 2	Q=1	0	0	1	0	0	1	0	0	0	1	0	0	0
Pattern 3	Q=2	0	0	1	0	0	1	0	0	1	0	0	0	0

Table 3.3 Allowed rest cycle patterns in a resident's schedule depending on $\mathbb{Q}$

Pattern 2 shows that a resident is working on $t=3$, is off on $t=\{3,4\}$ and back on duty on $t=6$. Such a set of days is a 48-hours rest cycle. If pattern 3 would occur while $\mathbb{Q} = 1$ a penalty would be attributed. During implementation of the prototype pattern 3 occurred regularly while $\mathbb{Q} = 1$. If this occured, a resident could only work such a schedule if he formally accepted it.

No more than two weekends per scheduling period

Constraint *(3.22)* expresses the limitation on the number of weekend shifts, due to the fact that weekend shifts are longer.

$$\sum_{t \in WKD} x_{it} + d\_WKD_{it}^{-} - d\_WKD_{it}^{+} = 2 \qquad \forall \text{ i} \qquad (3.22)$$

Every resident can only be on duty two weekends per scheduling period. Hospital management also indicated that it was preferable that no resident makes more than a single Saturday per calendar-period. This constraint is mentioned in Appendix A.

For resident 5 this results in the following constraint:

$$x_{5;5} + x_{5;6} + x_{5;7} + \ \ldots \ + \ d\_WKD_{5}^{-} - \ d\_WKD_{5}^{+} = 2$$

No more than two consecutive weekends

Constraints *(3.23)-(3.27)* ensure that a resident cannot work more than two consecutive weekends during a planning period. This legal constraint also has to be respected during two consecutive planning horizons.

$$\sum_{t \in PW} x_{it} + \sum_{t \in WKDSET1} x_{it} - \ CWKD_{ia} = 2 \qquad \forall \text{ i} \qquad (3.23)$$

$$\sum_{t=26 \in PW}^{28} x_{it} + \sum_{t \in WKDSET1} x_{it} + \sum_{t \in WKDSET2} x_{it} - CWKD_{ib} = 2 \qquad \forall \text{ i} \qquad (3.24)$$

$$\sum_{t \in WKDSET1} x_{it} + \sum_{t \in WKDSET2} x_{it} + \sum_{t \in WKDSET3} x_{it} - CWKD_{ic} = 2 \qquad \forall \text{ i} \qquad (3.25)$$

$$\sum_{t \in WKDSET2} x_{it} + \sum_{t \in WKDSET3} x_{it} + \sum_{t \in WKDSET4} x_{it} - CWKD_{id} = 2 \qquad \forall \text{ i} \qquad (3.26)$$

$$CWKD_{ia} + CWKD_{ib} + CWKD_{ic} + CWKD_{id} + d\_CWKD_{i}^{-} - d\_CWKD_{i}^{+} = 0 \qquad \forall \text{ i} \qquad (3.27)$$

	PW						WKD											
							WKDSET1			WKDSET2			WKDSET3			WKDSET4		
	19	20	21	26	27	28	5	6	7	12	13	14	19	20	21	26	27	28
Pattern 1	$x_{it} = 1$			$x_{it} = 1$			$x_{it} = 1$											
Pattern 2				$x_{it} = 1$			$x_{it} = 1$			$x_{it} = 1$								
Pattern 3							$x_{it} = 1$			$x_{it} = 1$			$x_{it} = 1$					
Pattern 4										$x_{it} = 1$			$x_{it} = 1$			$x_{it} = 1$		

Table 3.4 Forbidden consecutive weekend patterns over two consecutive planning horizons

The forbidden consecutive weekend patterns are shown in Table 3.2, where PW represents the set of all weekends of a previous planning horizon, and WKD the set of all weekends of the current planning horizon. The sets WKDSET1 to WKDSET4 represent the weekends of weekend 1 to weekend 4. For example, if a resident works two weekend shifts in set PW, pattern 1 forbids him to work a weekend shift in WKDSET1. The constraints reflect the patterns 1 to 4. For example, equation *(3.26)* corresponds to pattern 4. If one of the patterns is present in the schedule constraint *(3.27)* will penalize the number of violations.

For example, for resident 8 this constraint should become:

$x_{8;19\in PW} + \ldots + x_{8;28\in PW} + x_{8;5} + \ldots + x_{8;7} = 2$

$x_{8;26\in PW} + \ldots + x_{8;28\in PW} + x_{8;5} + \ldots + x_{8;14} = 2$

$x_{8;5} + \ldots + x_{8;21} = 2$

$x_{8;12} + \ldots + x_{8;28} = 2$

$CWKD_{8a} + CWKD_{8b} + CWKD_{8c} + CWKD_{8d} + d\_CWKD_i^- - d\_CWKD_i^+ = 0$

3.2.5 Hospital constraints

The hospital defined a few constraints that were not legally defined but were still considered to be an important factor in the fairness of each schedule. This third set of constraints reflects the hospital's concern that all residents are treated fairly.

Proportionate dispersion of days among residents

Every time constraint *(3.28)* is violated the dispersion score of resident *i* will deviate from the average penalty score.

$$disp\_score_i \; + \; d\_disp_i^- \; - \; d_{disp}{}_i^+ \; = \; \frac{\left(\sum_{i \in PW} disp_{score_i}\right)}{m} \qquad \forall \text{ i} \qquad (3.28)$$

Each resident has his own dispersion score, defined as the total penalty cost associated to a shift pattern, formulated in equation (3.29) as:

$$disp\_score_i = \qquad (3.29)$$
$$MO\_PEN \sum_{t \in MONDAY} x_{it} + TU\_PEN \sum_{t \in TUESDAY} x_{it} + WE\_PEN \sum_{t \in WEDNESDAY} x_{it} + TH\_PEN \sum_{t \in THURSDAY} x_{it}$$
$$+ FR\_PEN \sum_{t \in FRIDAY} x_{it} + SA\_PEN \sum_{t \in SATURDAY} x_{it} + SU\_PEN \sum_{t \in SUNDAY} x_{it}$$

Suppose that resident 4 works three Tuesday shifts and one Sunday shift. If $MO\_PEN = 3$ and $SU\_PEN = 6$ then $disp\_score_4 \; = 15$. If $\frac{\left(\sum_{i \in PW} disp_{score_i}\right)}{m} = 23$, resident 4 will work a better schedule than the average resident which results in a penalty cost. The penalty cost can be decreased by assigning resident 4 to more shifts or by assigning him to other shifts.

Proportionate dispersion of total shifts among residents

Constraint (3.30) is ment to ensure a fair distribution of the number of shifts between residents.

$$\sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{i \in W_k} x_{it} \; + \; d\_tsd_i^- - \; d_{tsd}{}_i^+ \; = \; \frac{\sum_{i=1}^{m} \sum_{t=1}^{n} x_{it}}{m} \qquad \forall \text{ i} \qquad (3.30)$$

Suppose that the average number of total shifts distributed between residents, $\frac{\sum_{i=1}^{m} \sum_{t=1}^{n} x_{it}}{m}$ = 5 and that we look at the shift pattern of resident 4. Since this resident would work four shifts in the current schedule his shift pattern would be of from the average shift total resulting in a penalty.

3.3 Proposed heuristics

This section provides a macro-view of the manual process used by resident scheduler's and of the proposed algorithms included in the construction heuristic, which is inspired by the manual process. From a theoretical perspective, a scheduler has rich knowledge on the characteristics often found in good solutions and uses his experience to create schedules. By combining different OR methods – a min-knapsack problem and a best-fit decreasing heuristic - into a construction heuristic it is possible to imitate the manual process and focus on the common characteristics of good solutions. The construction heuristic generates a promising subspace within the feasible solution space, called the Manually Restricted Space. The initial solution provided by this mechanism is thoroughly searched to find (hopefully) better solutions by a Tabu Search (TS) procedure. To illustrate the steps of the Construction Heuristic we will use as example the emergency department already described (in §3.2.1).

3.3.1 The manual scheduling process and its algorithmic equivalent

The resident scheduler creates a restricted space by using a Manual Scheduling Process (MSP) that leads to a solution contained in a feasible subspace. By combining different OR heuristics it is possible to imitate the manual process and focus on the common characteristics of good solutions. The combination of heuristics is also an efficient method to provide a solution in a reasonable amount of time. In this context, using heuristics is also a good way to ellicitate the schedulers' considerations, deduce their common characteristics and integrate this knowledge in search methods, thus recreating a RS (Restricted Space). Initial interviews led to the formalisation of the MSP and allowed the distinction of the different activities that were performed. A review of the literature allowed the identification of heuristic equivalents to these activities. The activities performed by schedulers are mentioned in Table 3.5 as well as the heuristics used for imitation.

Activity	Method	Heuristic equivalent
Determine minimum coverage level needs	Calculate number of shifts to work	Knap-sack relaxation
Establish initial schedule	Choose available resident	Best-fit decreasing heuristic (Bin-Packing pb)
Conflict resolution (before search)	Visual checks	Feasibility check in best-fit decreasing heuristic
Conflict resolution (during search)	Exchange with other residents	Tabu search

Table 3.5 Manual scheduling activity, corresponding method and heuristic equivalent

Solving the staffing problem by determining the coverage level needs has been used by Aickelin and Dowsland (2000) to ensure the satisfaction of coverage constraints. They used additional dummy (By the authors referred to as 'bank') nurses to compensate for undercoverage and provide initial solutions that were feasible. Ferland et al. (2001) used a best-fit decreasing heuristic on a nurse-scheduling problem to provide initial solutions of relatively good quality. We did not find an equivalent heuristic for the activity of *conflict resolution* during the construction of the initial solution. This was therefore interpreted more freely by taking into account one of the main problem characteristics. The challenge was that 72-hours resting cycles had to be taken into account before and after shifts. A constraint-verification was therefore introduced to verify if allocating the shift still allowed a 72-hours rest cycle before and after work. The final activity of *conflict resolution* was a search for a better solution. In the manual process, the scheduler would go over the calendar a few more times to see if all shift patterns had been assigned fairly. In optimization terms there are numerous methods. Nevertheless, a tabu search provided the best equivalent to the manual process where a move operator performs exchanges in a neighbourhood, opposed to evolutionary methods that use population-based search methods.

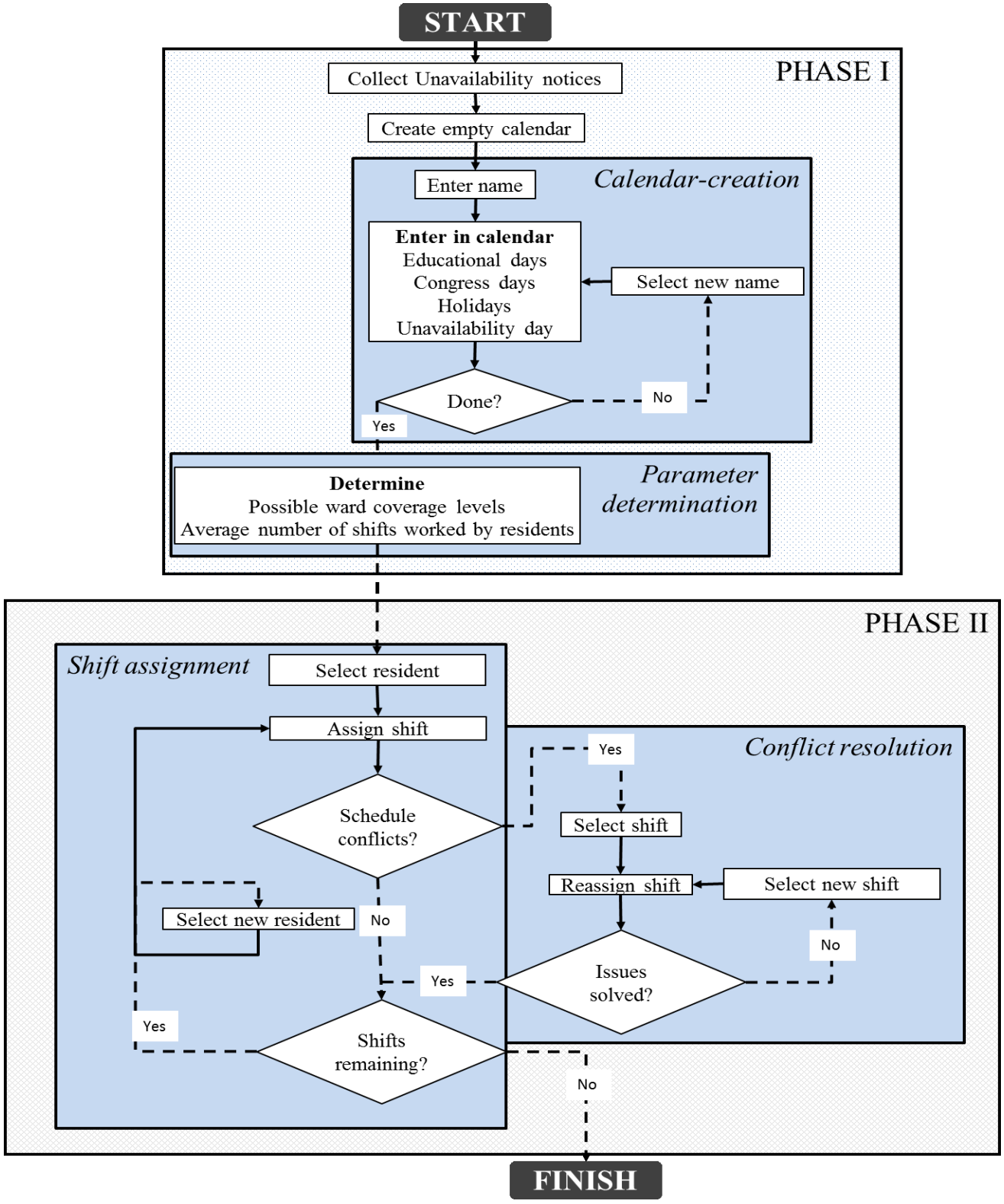


Figure 3.1 Manual Scheduling Process framework

Once the activites had been identified a more formal framework, called the Manual Scheduling Process (MSP), was developed. This framework is illustrated in Figure 3.1 and describes the process used by schedulers to manually create a schedule. This process distinguishes two phases:

- *Phase I: coverage input* - Residents provide the scheduler with their time-off requests which he uses to fill an empty calendar;
- *Phase II: shift assignment* - The scheduler creates the schedule and continues to solve occuring conflicts until he judges the schedule fair enough.

In simple terms schedulers design an initial schedule in the first phase and in the following phase they try to improve this schedule to obtain a feasible solution of good quality.

Phase *I* consists of two main tasks: *calendar-creation* and *parameter determination.* Before executing the task *calendar-creation* the scheduler gathers all the Unavailability Notices to create an empty calendar. The *calendar-creation* is a repeated task of filling in the names and the availability of each resident. Once this task is completed the scheduler will determine the ward coverage level together with the average number of shifts residents will work.

Phase II consists of two parallel tasks: *shift assignment* and *conflict resolution*. They are parallel because a scheduler can simultaneously manage the shift assignment in the neighbourhood, the evaluation of the schedule, as well as conflict resolving. The first task is the determination of schedule needs. The first task is *shift assignment*, a repeated process of assigning shifts to residents. The *conflict resolution* is triggered as a parallel task whenever the *shift assignment* results in the violation of rules. In phase *II* the position of the assigned shifts is subject to different constraints. The scheduler can assign shifts until a conflict with these constraints occurs. More precisely, the scheduler will start the *shift assignment* by selecting a resident. He will verify the number of shifts already assigned to this resident and decide whether or not affecting the shift to him. Whenever a conflict arises the scheduler will have to perform a *conflict resolution* by exchanging shift assignments with another resident.

	Method	Input	Output	Objectives
Phase I Initialisation	Integer knap-sack	Total availability of residents	Number of shifts covered by residents and number of shifts covered by doctors	Determine whether there will be under- or over-coverage
	Best-fit decreasing	Sorted SFT- and RDS-lists	Initial solution based on users' parameters	Provide an initial solution of good quality
Phase II Search	Tabu search	Initial solution	The best solution found with the tabu search	Explore the solution thoroughly

Table 3.6 Summary of the solution methods

The development of the MSP allowed the definition of the different tasks for the prototype. Table 3.6 provides a summary of the inputs and outputs of the different solution methods within the prototype. The integer knapsack problem takes the empty calendar information and determines whether any departments will be short-staffed throughout the planning horizon. The best-fit decreasing heuristic will receive the information from the integer knapsack and uses the raw data from the empty calendar and the staffing requirements of each department to transform it in an initial solution. The raw data consists of two sorted lists, the SFT- and the RDS-list, The SFT-list contains all the shifts to be assigned throughout the planning period. The RDS-list contains the residents in increasing order of pre-scheduling score based on different criteria. The initializaiton of these lists will be discussed in §3.3.3. The initial solution will be improved by a tabu search that will explore the different solutions and return the best solution found. The different activites were integrated in the prototype and the results were presented to schedulers to obtain their feedback on the quality of solutions.

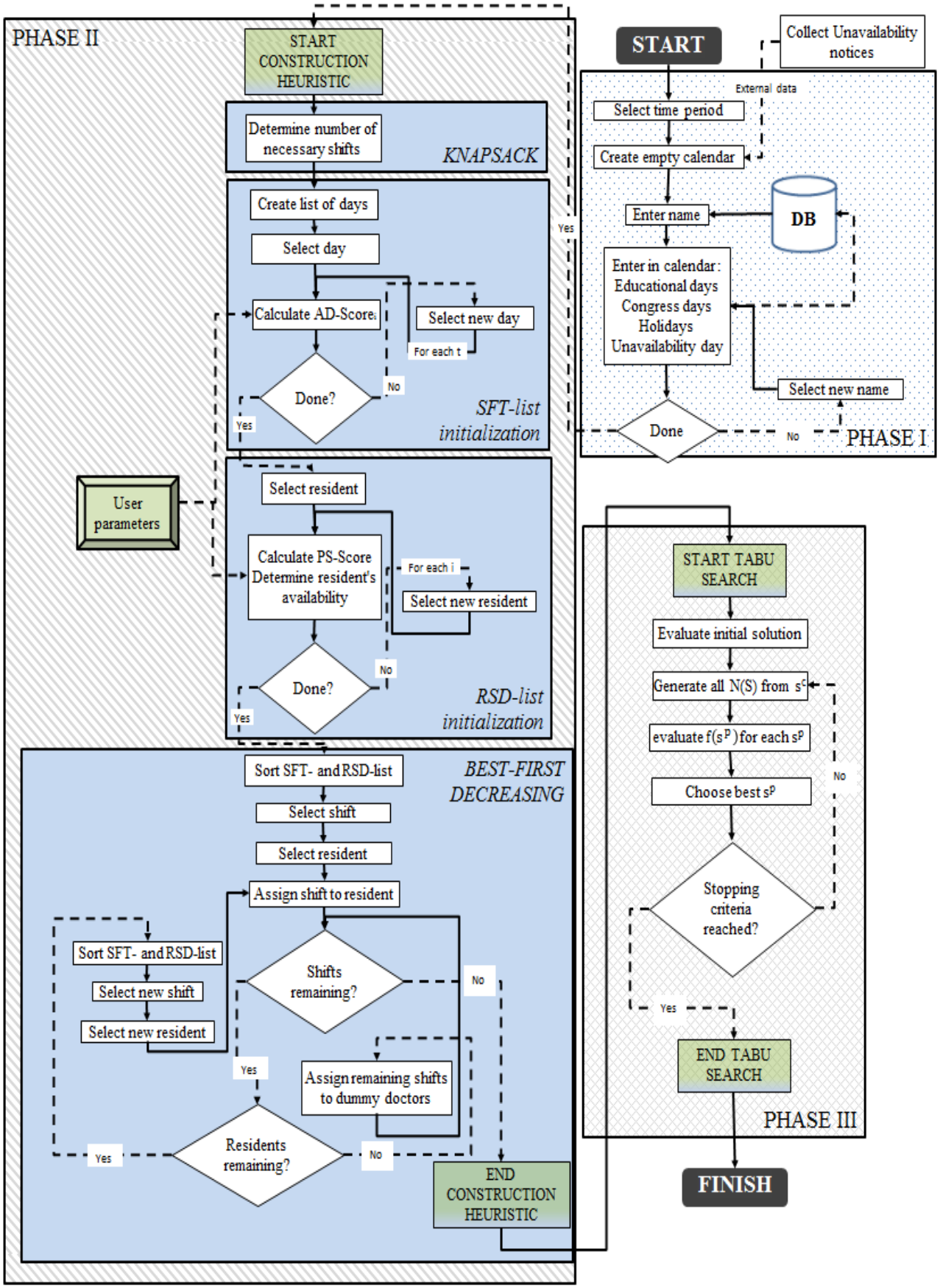


Figure 3.2 General algorithm

Figure 3.2 presents the general algorithm. This process is organized in three different phases:

- *Phase I: initial data input* - The scheduler enters the unavailability information in the prototype;
- *Phase II: restricted space construction* – The heuristics described build an initial solution to obtain the Manually Restricted Space;
- *Phase III: Optimization* – The tabu search thoroughly explores the Manually Restricted Space to find, hopefully, better solutions.

In Phase I the information provided by the scheduler is stored in the database. Phase I is done in the user interface of the prototype that allows user-computer interactivity. This Phase is similar to Phase I of the MSP.

Phase II of the algorithm is similar to Phase II of the MSP. Four different procedures are executed in this phase. First of all, the *Knapsack* procedure determines the number of shifts necessary to satisfy the covering needs throughout the planning period. Then the procedure *SFT-list initialization* is executed resulting in a list that contains all the shifts to be assigned throughout the planning period. In turn, the procedure *RSD-list initialization* creates the availability list of all residents. This list mentions the total number of shifts each resident can work throughout the planning period. Then, the next task is to assign all shifts by help of the *Best-fit decreasing* procedure. In case not all shifts can be assigned to residents, dummy doctors are assigned to the remaining shifts.

Finally within the algorithm, a tabu search is executed to explore the Manually Restricted Space and, with any luck, find (near-to)-optimal solutions. The procedures used in the algorithm will be described in more detail in the following sections.

3.3.2 A knapsack IP-relaxation of the staffing problem

The staffing problem (Staffing Problem) is presented below as a knapsack formulation of a min-knapsack problem. The resolution of the staffing problem can be used to determine coverage feasibility. This information is used to determine whether all night shifts throughout a scheduling period can be covered by residents and if doctors replacing

residents are required. This information is referred to as the Coverage Input (CI). This approach has been used by Aickelin and Dowsland (2000) to satisfy coverage constraints and provide better initial solutions. For the staffing problem the resolution corresponds to the satisfaction of constraints (3.2) and (3.9) (*Required number of residents for each period, No more than the maximum number of shifts allowed per resident*). The resolution results in the creation of an ordered list that contains all residents and the number of shifts they can be assigned to.

When formulating the Coverage Input as a knapsack problem we have:

Min $\Sigma a_i r_i + \Sigma b_j s_j$ *(3.31)*

$\Sigma n_i r_i + \Sigma s_j d_j \geq N$ *(3.32)*

Where

Parameters

N *is the total number of shifts that have to be worked;*

n_i *Is the number of shifts resident i can make, i=1…m;*

s_j *Is the number of shifts doctor j can make, j=1..D;*

a_i *Is the cost associated to scheduling resident i, i=1…m;*

b_j *Is the cost associated to scheduling doctor j, j=1..D.*

Decision variables

r_i *is 1 if resident i is available to work, 0 otherwise;*

d_j *is 1 if doctor j is available to work, 0 otherwise.*

To solve the integer problem we relax the Integrity constraint (3.32) on the assignment variables. N can be considered as the lower bound of the problem. By solving the IP-relaxation of the problem it is possible to determine whether there will be undercoverage. Thus, if $N - \Sigma n_i^* r_i \geq 0$, there will be a shortage in residents. By reformulating *(3.32)* the number of shifts that doctors have to make can be determined by solving (3.33):

$\Sigma s_j * d_i \leq N - \Sigma n_i * r_i$ *(3.33)*

If at any stage $\Sigma n_i * r_i$ – which will provide us with the total number of shifts that can be covered by residents - is greater than or equal to N, the corresponding scheduling problem

is feasible. Otherwise, if $\Sigma n_i * r_i$ is smaller than N, the number of residents available will not allow the satisfaction of minimum coverage constraints and doctors have to be scheduled to compensate for undercoverage. A doctor never covers more than a single shift when standing in. The doctors substitute residents in case of undercoverage and help us to estimate the total level of undercoverage throughout the scheduling period.

To demonstrate this we will use the characteristics of §3.21 for a numerical example. On the Emergency ward we had 10 residents available ($i=1...10$). We will suppose that each resident is allowed to work 4 shifts ($n_1=n_2$=... $n_{18}=4$). However, we will suppose that resident 9 can work only up to 2 shifts ($n_9=2$) at the emergency ward and that resident 10 can work only 1 shift on the emergency ward ($n_{10}=1$). Minimum coverage is 1 resident and the planning horizon is 28 days. A total number of 28 shifts ($N=28$) would therefore have to be worked to ensure sufficient coverage. We will look at the coverage in a simplified example.

$$\Sigma n_i = n_1+n_2+...n_{10} = 8*4 +2 +1= 36 \text{ shifts}$$

Because $\Sigma n_i = 36$, we could assume that all shifts to be worked could be assigned and that the set of solutions $\neq \emptyset$. Therefore, a feasible solution exists that satisfies *(3.31)* for the knapsack problem and that constraint *(3.32)* can be satisfied, therefore:

Min $\Sigma a_i r_i + \Sigma b_j s_j \Leftrightarrow \Sigma n_i * r_i \geq N$ *(3.34)*

If $\Sigma n_i * r_i \geq N$ is satisfied the following statement is true as well:

$\Sigma n_i * r_i \geq N \Leftrightarrow \Sigma s_j * d_j = 0$ *(3.35)*

When $\Sigma n_i * r_i \geq N$ for the Emergency department there will not be any undercoverage and the problem is feasible without needing the help of doctors to work shifts. Otherwise, if $\Sigma s_j * d_j \geq 0$ and we would have to solve *(3.33)*. This would mean that there would be a need to have doctors work shifts. In both cases the solution would give a feasible solution. If statement (3.34) is satisfied a feasible solution exists where only residents will work, otherwise a feasible solution will include shifts worked by doctors and residents.

3.3.3 Lists initialization

The construction heuristic starts by initializing the lists of residents and shifts. The first list to be initialized is the *SFT_LIST,* the list of all shifts necessary to ensure full coverage throughout the planning period. Before the start of the best-fit decreasing heuristic, the difficulty score for each day is already indicated by the scheduler in the prototype, which will be explained further on. These values will be used to determine the order in which the

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Score	4	4	4	2	7	10	7	4	4	4	2	7	10	7

t	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Day	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Score	4	4	4	2	7	10	7	4	4	4	2	7	10	7

Figure 3.3 Assginment of score to all days of planning period

shifts are selected to be assigned. This is illustrated in Figure 3.3. This Figure also shows the complete list of shifts for the emergency ward where the coverage level is 1 employee and the score that the scheduler gave to them.

For example, the scheduler gave Mondays a weight of 4 points, whereas Saturdays received 10 points. This means that Saturday shifts are more difficult to assign than Monday shifts and therefore that violation of Saturday's constraints will be penalized harder. Using this information the *SFT_LIST* is created. For example, the coverage levels for the emergency and the paediatrics ward are 1 resident each. The *SFT_LIST* will therefore contain a single shift for which *t =1, ward = emergency* and *score = 4*.

SFT_LIST	6	13	20	27	5	7	12	14	19	21	26	28	1	2
	Sat	Sat	Sat	Sat	Fri	Sun	Fri	Sun	Fri	Sun	Fri	Sun	Mon	Tue
	10	10	10	10	7	7	7	7	7	7	7	7	4	4

3	8	9	10	15	16	17	22	23	24	4	11	18	25
Wed	Mon	Tue	Wed	Mo	Tue	Wed	Mon	Tue	Wed	Thu	Thu	Thu	Thu
4	4	4	4	4	4	4	4	4	4	2	2	2	2

Figure 3.4 Example of SFT_LIST

An example of the SFT_LIST for the emergency ward is illustrated in Figure 3.4. The *SFT_LIST* is sorted in decreasing order of the difficulty score. For example, the days t = 6, 13, 20, 27 are the most difficult shifts to assign, which corresponds to the set of Saturday shifts. The Friday and Sunday shifts are equally difficult and will therefore appear at the same level of assignement.

Once the list of shifts has been initialized the construction heuristic will create the resident list (*RSD_LIST)*. The *RSD_LIST* contains the pre-scheduling score for each staff member which is calculated using the pre-filled calendar from phase I. The pre-scheduling score is defined as follows:

$$SCORE_i = pen\_c \sum_{t=1}^{m} c_{it} + \ pen\_h \sum_{t=1}^{m} h_{it} \ + \ pen\_prper \sum prper \qquad \forall\, i \qquad (3.36)$$

Where,

Parameters

pen_c		*Is the penalty value given to a resident for a day of conference attendance*
pen_h		*Is the penalty value given to a resident for a day of holidays*
pen_prper		*Is the penalty value given to a resident that worked a weekend shift in the course of the last two weekends of the previous period*
Decision variables		
c_{it}	*1*	*If resident i was attending a conference during day t*
	0	*Otherwise*
h_{it}	*1*	*If resident i if resident i was on holidays during day t*
	0	*Otherwise*
$\sum prper$		*Is the number of weekend shifts resident i worked during the last two weeks of the prevous period.*

For example, suppose that resident 9 has a week-long holiday and that he worked on a Sunday during the previous planning period. Let us assume that the scheduler has decided to set penalty weights as follows:

pen_c = 20; pen_h = 30; pen_pper = 8

Therefore, the resident score will be

$Score_9 = 30 * 7 + 8 * 1 = 218$

	Factor										
Resident		1	2	3	4	5	6	7	8	9	10
Congress	20	0	0	0	0	0	0	3	3	0	0
Holiday	30	0	0	0	0	7	0	0	0	7	0
PWKD	8	0	1	1	0	1	0	1	1	1	0
Score		**0**	**8**	**8**	**0**	**218**	**0**	**68**	**68**	**218**	**0**

RSD_LIST

1	4	6	10	2	3	7	8	5	9
0	**0**	**0**	**0**	**8**	**8**	**68**	**68**	**218**	**218**

Figure 3.5 Example of initializing a RSD_LIST

The resulting scores for each resident will be sorted in increasing order and saved in a list called *RSD_LIST*. An example of how the RSD_LIST is created is illustrated in Figure 3.5.

The first table in Figure 3.5 shows the calculation of the pre-scheduling score for residents 1 to 10. For example, for resident 9 the total pre-scheduling score was 218, the same score was obtained by resident 5. Their high scores are due to the fact that they are attending a conference and are on holidays for a week. The table illustrated in Figure 3.5 shows in what way conference attendance, holidays and, previous weekends (PWKD) affect the pre-scheduling score. These results are used to create the *RSD_LIST* illustrated below the table. The *RSD_LIST* contains the residents in increasing order of pre-scheduling score. For example, resident 5 and 9 had the highest pre-scheduling score and therefore will be positioned last in the *RSD_LIST*. Associated to the RSD_LIST is a secondary list, called MAXSFT_LIST that contains the number of shifts each resident can work. For example, resident 1 will not be absent due to conference attendance or holidays, nor did he work the weekend of the previous period. In a planning period of 28 days he would therefore be available to work 6 shifts. Hence, a workload of 6 shifts can therefore be associated to his status.

3.3.4 Best-fit decreasing heuristic

The staffing problem is best described as a shift-assignment formulation where the most difficult shift is directly assigned to the resident with the most possibilities, i.e. the one having a lower fairness score. The previous section showed how lists can be created for

this purpose. This resulted in the list of all residents containing the number of shifts they can work, as well as a list of all shifts that have to be assigned.

Ferland et al. (2001) used a bin-packing heuristic to assign shifts to hospital staff. One of such heuristics, a best-fit decreasing heuristic, commonly used for bin-packing problems in operational research, imitates the shift-assignment phase closely. Using this approach the list of residents will be considered as the list of bins and their corresponding capacity. In a bin-packing problem the goal is to pack objects into a finite number of bins of capacity C in a way that minimizes the number of bins used. The problem can be formulated as follows:

$$Min: \sum_1^w y_b \tag{3.37}$$

$$Subject\ to:$$

$$C * y_b - \sum_{a=1}^{v} c_{ab} x_{ab} \geq 0 \qquad \forall j \tag{3.38}$$

$$\sum_1^n x_{ab} = 1 \qquad \forall j \tag{3.39}$$

$$x_{ab}, y_b \in \{0, 1\} \qquad \forall i, \forall j \tag{3.40}$$

Where we have,

Indexes

a *index for item a ($1 \leq a \leq vn$)*
b *index for bin b ($1 \leq b \leq wm$)*

Parameters
C *Maximum storage capacity of each bin*
c_{ab} *Weight of item a in bin b ($c_{a1} = .. = c_{aw}$)*

Decision variables

x_{ab}	*1*	*If item a is packed in bin b*
	0	*Otherwise*
y_b	*1*	*If bin b is used*
	0	*Otherwise*

When associated to the assignment of shifts, this problem can be solved by a resident-assignment heuristic (RAH) where the goal is to assign shifts to a finite number of residents with a maximum workload C in a way that minimizes the number of residents used. To avoid infeasibility, the second goal is the minimization of the number of doctors added to the schedule for replacement. By supposing that enough doctors can always be found to avoid any shortages, all shifts can always be assigned. This problem can be formulated as follows:

$$Min: \sum_1^m y_i + \sum_1^D v_{dt} \quad (3.41)$$

$$Subject\ to:$$

$$C_i * y_i - \sum_{s=1}^{S} c_{si} x_{si} \geq 0 \qquad \forall\, i \quad (3.42)$$

$$\sum_1^S x_{si} + v_{dt} = 1 \qquad \forall\, i \quad (3.43)$$

$$x_{si}, y_i, v_{dt} \in \{0, 1\} \qquad \forall\, i;\ \forall\, s; \forall\, d \quad (3.44)$$

Where we have,

Indexs

j	*Index for resident i*	*(1 ≤ i ≤ m)*
i	*Index for shift s*	*(1 ≤ s ≤ S)*
d	*Index for doctor d*	*(1 ≤ d ≤ D)*
t	*Index for day t*	*(1≤ t ≤ T)*

Parameters

C_i	*Maximum workload for resident i*
c_{si}	*Workload of shift s for resident i* ($c_{s1} = .. = c_{sm}$)

Decision variables

x_{si}	*1*	*If shift i is assigned to resident j*
	0	*Otherwise*
y_{ji}	*1*	*If resident j is assigned*
	0	*Otherwise*
v_{dt}	*1*	*If doctor d is assigned on day t*
	0	*Otherwise*

The assignment strategy for shifts will be the best fit rule: Choose the assignment that results in the strongest decrease in workload over all shifts. The algorithm for the *RAH* can be described as follows:

- *Sort SFT_List in decreasing order*
- *While there are shifts remaining*
  - *Sort RSD_LIST in increasing order*
  - *Select resident i for whom* $\text{SCORE}_\text{i} \leq (\text{SCORE}_\text{i}\ 1 \leq \text{i} \leq \text{n})$ *for all other residents*
  - *Perform feasibility checks*
  - *Assign shift to resident*
  - *Update* SCORE_i *from RSD_LIST:*

$$\text{SCORE}_\text{i} = \text{SCORE}_\text{i} - \text{difficulty score}_\text{t}$$

  - *Eliminate assigned shift from SFT_LIST*
- *End While*

During an iteration of the RAH the following tasks will be executed. The resident with the highest pre-scheduling score, on top of the SFT_LIST, is affected to the most difficult shift. Once the shift has been assigned RSD_LIST and MAXSFT_LIST will be updated. The resident's score $\mathrm{SCORE_i}$ will be updated in RSD_LIST and a shift will be substracted from his availability in MAXSFT_LIST. RSD_LIST will be sorted again before the re-execution of the assignment-loop. The RSD_LIST is updated by subtracting the day score of current shift, in the algorithm denoted as $\mathrm{difficulty\ score_t}$. The difficulty score is the value of the assigned in the SFT_LIST.

The MAXSFT_LIST keeps track of the number of remaining shifts in order to perform feasibility check 1.
The feasibility checks were added to the best-fit decreasing heuristic to be capable to provide a better imitation of the scheduler manual process by trying to avoid conflicts. Hence, the RAH-algorithm performs the following feasibility checks:

1. *Adding the shift does not bring the number of worked shifts above MAX_i*
2. *No shifts have been assigned within a range of 3 days before/after the current day*
3. *No holidays are situated within a range of 3 days before/after the current day*

The feasibility checks are performed directly before the assignment of the shift to a resident (*Assign shift to resident).* If the RAH is executed without feasibility checks, conflicts arising most repeatedly are those concerning shifts that are situated within the 3-day range before or after the current day and shifts that were scheduled too close to holidays. If during the feasibility check the number of shifts will fall below 0 for any resident the assignment-loop will move on to the next resident. In case two residents have the same number of days available the shift will be allocated according to the position of the resident in the list.

Planning staff on certain days will result in a higher likeliness that more constraints can be violated. Hence, RAH can be seen as a technique to reduce the probability that constraints will be violated. Algorithmically speaking it also has a number of advantages. The solution space is diminished because only feasible solutions will be considered. Furthermore, two hard constraints are satisfied from the outset. On each day there will be a sufficient

SFT_LIST

6	13	20	27	5	7	12	14	19	21	26	28	1	2
Sat	Sat	Sat	Sat	Fri	Sun	Fri	Sun	Fri	Sun	Fri	Sur	Mor	Tue
~~20~~	**10**	**10**	**10**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**4**	**4**

3	8	9	10	15	16	17	22	23	24	4	11	18	25
Wed	Mon	Tue	Wed	Mo	Tue	Wed	Mon	Tue	Wed	Thu	Thu	Thu	Thu
4	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**2**	**2**	**2**	**2**

Figure 3.6 Update of SFT_List for shifts still to be assigned

RSD_LIST

1	4	6	10	2	3	7	8	5	9
0	**0**	**0**	**0**	**8**	**8**	**68**	**68**	**218**	**218**

RSD_LIST

4	6	10	2	3	7	8	1	5	9
0	**0**	**0**	**8**	**8**	**68**	**68**	**100**	**218**	**218**

Figure 3.7 Update of RSD_List for resident

Individual schedule

t	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
L_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Individual schedule

t	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
L_1	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figure 3.8 Updating the individual schedule by adding a shift

number of residents, thus satisfying the coverage constraint. The number of allowed shifts per resident will also remain intact. In such a manner it therefore simultaneously allows the satisfaction of daily staff level demands as well as the number of shifts allowed for each staff member.

An example of the steps in the algorithm is illustrated in the Figures 3.6 to 3.8. In the current example, the RAH will assign the first shift to a resident. As shown in Figure 3.6, the first shift to be assigned is the Saturday shift of day t=6. The first resident to be considered will be resident 1 because he has the lowest score and thus the lowest position in RSD_LIST, as illustrated in the first table of Figure 3.7. Before the shift is assigned the feasibility checks are performed. Figure 3.8 shows resident 1's schedule currently as empty, enabling us to assign him/her a shift. The Saturday shift will therefore be assigned to resident 1's schedule, which is updated, as shown in Figure 3.8. The second table in Figure 3.7 shows how RSD_LIST is updated. For resident 1 the score is updated by the formula $\mathrm{SCORE_i} = \mathrm{SCORE_i} - \mathrm{Assignment\ score_t}$. Once this task is complete RSD_LIST is sorted so that the next available resident can be selected and the first Saturday shift is deleted in SFT_LIST.

At the previous stage the question was to know whether there were sufficient residents available to execute all shifts, as well as obtaining an initial schedule. At the end of this stage all shifts have to been assigned to residents and we end up with a schedule that is feasible from the outset.

The methodology of giving a score to both working days and individual employees can be applied to different fields of timetabling and seems to be a promising way to reduce the search time. The advantage of the method is that it can be adapted to different types of situations. It is therefore possible to determine one's own way of giving a score to a staff member, or the scheduling difficulty of a day/shift.

3.3.5 Tabu search algorithm

The tabu search (TS) is a descent heuristic using an iterative procedure that prevents the search of visiting recently explored (tabu) solutions for a number of iterations. The TS examines solutions moving from one neighbourhood of the current solution to another neighbourhood. A neighbourhood is defined as the set of solutions differing from the current one of a given maximal distance. The TS procedure is characterized by the possibility of moving from the current solution to a neighbour one, even if doing so the current objective value is deteriorated. This mechanism avoids the search from being captured into local optima and therefore to leave already explored areas and be guided towards new areas in the solution space. To forbid revisiting solutions a short-term memory, called the tabu list, keeps track of the most recent movements. The efficiency of a tabu search scheme is strongly influenced by the intensification and diversification strategies (Glover and Laguna (1993)) which are used to find the right balance between concentrating the search in a given region of the solution space and enlarging the search throughout the whole space.

The following paragraphs introduce the main elements of the tabu search algorithm that we designed for the Resident Scheduling Problem, followed by a thorough description of important implementation issues. We also discuss the diversification strategies used during the search.

3.3.5.1 Mechanisms of the tabu algorithm

Tabu list: A list of the most recently made moves (or solutions). The number of moves in the list is determined by the tabu list length, denoted by TL. The list operates on a first-in-first-out base. The size of the list (TL) can be static or dynamic during the search.

Candidate List: The list of all feasible solutions in the neighbourhood of the current solution. The TS selects the best move from the candidate list to become the current solution and proceed with the search.

Intensification and diversification strategies: Local search heuristics spend most of their search in a limited portion of the search space. To enlarge the search field of view, specific strategies - included in the algorithmic structure and executed during run-time - are often used to control and guide the search path. Intensification strategies are used in an attractive neighbourhood to visit as much close-by neighbourhoods as possible to get the best solution available in that region. Diversification strategies are aimed at examining unvisited regions to find solutions that differ significantly from previous explored solutions.

Penalized objective function: The objective function of a solution s is denoted by $f(s)$ and is calculated by means of the cost function. Since a move from the current solution to a neighbour one only affects specific, rather small parts of the schedule's structure, it is more efficient to evaluate the potential of such a move by computing its incremental cost, named $f'(s)$, instead of recalculating the total objective cost. Computing incremental costs requires only a partial evaluation of a schedule so it is less demanding in computational time and is therefore useful to speed up the search process.

Stopping criteria: Are used to terminate the search process. Usually, the TS algorithm is stopped after reaching a preset number of iterations set by the user. An alternative stopping criterion is a predetermined number of iterations during which the current solution has not been improved. This indicates that the search is not able to find new local optima, motivating it to terminate the search.

Neighbourhood: Formally, a neighbourhood includes all the solutions that are situated within a given distance (or which differ in less than a certain number of characteristics) from the incumbent solution. The original schedule is denoted S and the set of all neighbour solutions as $N(S)$, $N(S)$ being defined as a subset of the search space. In practical applications of nurse rostering problems $N(S)$ is defined as the set of all feasible moves that are possible within the same staff member's line. Therefore, a single move is the swap of two cells in the same column between residents i and j. Because exchanging two days between residents that are not working does not affect the total cost, it is more efficient to only consider moves between a resident that is already working and a resident that is not yet working. This is also a useful consideration to diminish the computational time.

Acceptation criteria: The acceptation of the solution $N(S)^t$ resulting from the search in the neighbourhood $N(S)$ depends on the cost $f'(s^t)$ of this solution. As mentioned earlier, this is affected by the method used for calculating the cost function.

Although several exploration criteria can be used to select the new solution to move on from the available solutions in $N(S)$, the most popular strategy in tabu search schemes is the so called *best improvement* which consists in selecting the solution having the best cost within $N(S)$ and that is not tagged as tabu. In a tabu search the acceptation of a new solution is defined as the acceptation of the best new solution available within $N(S)$. The best move can either diminish the current cost, or increase it. When we define $f(s^*)$ as the cost function of the best known solution and all K solutions in the candidate list have a cost function $f_k(s')$ and for all solutions we have $f_k(s') \geq f(s^*)$ we will choose $\mathrm{Min}\{f_1(s,), f_2(s,), .., f_k(s,)\}$, the solution resulting in the lowest differtial cost increase.

Allowing infeasible solutions: The choice of including feasible or infeasible solutions is affected by the method used for the cost evaluation. When an infeasible solution is accepted the cost function is affected by a penalty cost reflecting the degree of infeasibility of the solution.

3.3.5.2 Tabu algorithm

The elements used in the tabu search are as follows:

N(S): Neighbourhood consisting of all exchanges between resident i and j, where resident i is working and resident j is not scheduled or requested a day off.

P: Number of exchanges

s^**: Best known solution*

s^c*: Current solution*

s^p*: Neighbour solution*

s^k*: Best solution on candidate list*

The neighbourhood $N(S)$ contains all potential feasible solutions s^p that can be obtained from the current solution. All of these solutions are enumerated in the list of candidate solutions that is of length P. Whenever an exchange concerning resident i and j is performed on s^c resulting in s^p, the schedule cost will only be affected in the contribution of these particular residents. The incremental cost functions are therefore computed only on the contributions of residents i and j, which reduces the computational time of the evaluation throughout the search.

The general algorithmic structure used to perform the search, given an initial solution, for the resident scheduling problem is as follows:

1. *Choose an initial solution s_0 obtained by ManualInit. Set $s^c = s_0$*
2. *From s^c select* $\mathrm{Max}\{f_1^c(s^c); f_2^c(s^c); \ldots; f_m^c(s^c)\}$ *as resident, the resident r with the highest contribution to the cost function, for the coming iteration.*
3. *Generate the neighbourhood N(S) by forming all the possible solutions, assigning one of the wards of resident i's at a time to every other resident, and evaluate* $f(s^p)$ *for each s^p*
4. *Choose the best solution s^p in N(S) from the candidate list of potential solutions*
5. *Set $s^c = s^k$. If $s^p \leq s^*$ then $s^* = s^p$.*
6. *Return to step 2 until reaching the stop criterion.*

The tabu search starts its search from the initial solution s_0, obtained by the construction heuristic. To proceed with the search, the resident r with the highest incremental cost is selected to generate $N(S)$ from, thus obtaining a candidate list. Next, the TS selects the best solution available on the candidate list, s^k, which becomes the current solution, s^c, at the end of each iteration. In parallel, if the solution s^k is better than the best known solution s*, this solution s^k becomes the best known solution.

Search intensification is done by occassionnally restarting the search with elite solutions; if the search does not result in a better solution than s* after MNI iterations, then the search jumps back to s* and restarts its search. This intensification process is handled by the BAEP-procedure (*Best Available Exchange Possible*) that selects the best available move. To avoid the iterative procedure from remaining trapped in a local optimum the PDS-procedure (*Probabilistic Diversification Strategy*) is used for diversification. In this procedure, the search restarts at s* and is forced in lesser explored neighbourhoods. The

probabilistic tabu search contributes to diminishing the risk of cycling. The parameters and mechanisms supporting BAEP- and PDS-procedure will be described in more detail in the following paragraphs.

3.3.5.3 Neighbourhood search technique

The TS establishes the candidate list of potential solutions through a search in a neighbourhood. The TS exchange heuristic focuses exclusively on exchanges where a shift can be substituted for another day because these exchanges have the highest potential of improvement. We define

R *as the resident with the worst current score*

SFT^R *as the set of all shifts worked by R*

where $SFT^r \subset N(s^*)$. The exchange heuristic collects all elements belonging to SFT^r = *{working, working,…,working}.* All residents that are still available would be considered for possible candidate exchanges, except during holidays or conference days.

Suppose we have a schedule with 20 residents. A single, already assigned, shift can be reassigned to 19 other residents. We can have up to 6 shifts exchanged for a single resident. The order is of importance in this case and by combination we can have up to 19 x 6! = 13 680 possible combinations with 6 shifts. On a single pass through the neighbourhood the search would only return 19 x 6 = 114 with 6 shifts.

In the diversification procedure (*PDS*) the solution is furthermore decomposed into a week subproblem and a weekend subproblem. Therefore, we define the following sets:

*WEEK: The set of weekdays of s**

*WKD: The set of weekends of s**

The search is performed in one of both sets while the elements of the other set remain unchanged. So, the search looks for exchanges in the *WEEK*-set while keeping the *WKD*-set unchanged and inversely in the *WKD*-set while keeping the *WEEK*-set intact.

3.3.5.4 Evaluation of the cost function

An exchange is defined as the swap of two cells in the same column between residents i and j within the same planning horizon. Since the exchanges made by the tabu search only concern two residents at a time, – only the lines concerned by the exchanges are evaluated – incremental costs are easily computed, making it unnecessary to reevaluate the entire solution. In this perspective, the elements used for the cost function are defined as follows:

$f(s^*)$: *Total cost function of best known solution s**
$f(s_0)$: *Total cost function of initial solution* s_0
$f(s^p)$: *Total cost function of neigbour solution* s^p
$f(s^c)$: *Total cost function of current solution* s^c
$f'_r(s^c)$: *Contribution to the cost function of resident r in current solution* s^c
$f'_r(s'_p)$: *Contribution to the cost function of resident r in neighbour solution* s^p $p=1...P$

The value of the cost function of the neighbour solution, $f(s^p)$, is obtained by subtracting the score differential (resulting from the exchanges) from the cost value of the current solution, $f(s^c)$. The score of the current solution is also used to update the score of the candidate solution. The cost function associated to a neighbourhood move is determined as:

$$f(s^p) = f(s^c) - \left(f'_i(s^c) - f'_i(s^p)\right) + \left(f'_j(s^c) - f'_j(s^p)\right) \quad (3.45)$$

Three outcomes are possible:

$$1 \qquad \left(f'_i(s^c) - f'_i(s^p)\right) + \left(f'_j(s^c) - f'_j(s^p)\right) > 0 \quad (3.46)$$

The cost resulting from this exchange will decrease because of an improvement in the schedule. Therefore $f(s^k) < f(s^*)$. In this case, the exchange will decrease the incremental cost of both residents or the improvement of the incremental cost of one resident will be higher than the deterioration of the other resident.

$$2 \qquad < \left(f_i'(s^c) - f_i'(s^p)\right) + \left(f_j'(s^c) - f_j'(s^p)\right) < 0 \qquad (3.47)$$

The cost resulting from this exchange will increase because of deterioration in the schedule. Therefore $f(s^p) > f(s^*)$. In this case, the exchange will increase the incremental cost of both residents or the deterioration of the incremental cost of one resident will be higher than the improvement of the other resident.

$$3 \qquad \left(f_i'(s^c) - f_i'(s^p)\right) + \left(f_j'(s^c) - f_j'(s^p)\right) = 0 \qquad (3.48)$$

This results in $f(s^p) = f(s^c)$. In this case, the cost value resulting from this exchange will not change because neither resident will have anything to gain from the change in the schedule. The improvement of one resident's incremental cost function could also be cancelled by the deterioration of one resident's incremental cost function.

The contribution to the cost function of a resident consists of only those constraints whose penalty value could be affected by the exchange. These constraints are indicated in Table 3.7. For example, constraints (3.13)-(3.16) could be eliminated from this evaluation because no exchange will be performed if either resident is attending a conference or is on holiday the day the intended exchange should be performed. Evaluating the incremental cost function takes only approximatively *1/m* of the evaluation time of that of a complete evaluation. The total evaluation time is therefore approximately a *2/m* factor of the total evaluation time.

For the evaluation we defined a list called LIST_SCORELINE that contains the score of each line of the current solution and represents $f_i'(s^c)$.

Internal ward constraints
Proportionate dispersion of days among residents
Proportionate dispersion of total shifts among residents

Legal Constraints
No more than the maximum number of shifts
Respect requested days off
Respect resting cycles after night shift
No more than two weekends per scheduling period
No more than two consecutive weekends

Table 3.7 Incremental cost function constraints

3.3.5.5 Stopping criteria

The stopping criteria are the maximum number of iterations (*MI*) and the total available search time *(TAST)*. The search is stopped as soon as one of them is reached.

3.3.5.6 Soft diversification techniques

The diversification is handled by the PDS-procedure. This diversification procedure is based on the notion of probabilistic tabu search. Glover and Laguna (1993) note that the use of probabilities based on past performance, as an underlying measure of randomization yields efficient and effective means of diversification. During the probabilistic diversification, the tabu status of neighbourhoods is circumvented and only the probability of selection of a line is taken into account. The diversification procedure is called if the search has remained in the same region for MN iterations, which is verified by evaluating the percentage of change in the total cost of the solution (α%) over the last MN iterations.

To describe the diversification procedure we define

F(i) *as the cumulative distribution function*

p_i *as the probability of visiting line (resident) i*

v_i *as the number of times neighbourhood i has been visited*

rnd *as the random number that is generated at each iteration* (*uniformly distributed, 0≤ rnd <1)*

Where p_i is a discrete probability function because v_i is a discrete variable and is determined by (3.49):

$$p_i = 1 - \frac{v_i}{\sum_{i=1}^{m} v_i} \tag{3.49}$$

With the cumulative distribution function:

$$F_i = \begin{cases} F_0 = 0, & i = 0 \\ F_i = p_1 + p_2 + \cdots + p_i & i \geq 0 \end{cases} \tag{3.50}$$

The probability p_i describes the probability of selection of a line; this probability is inversely related to the number of times a line has been explored. The chances of a regularly explored line being revisited are lower if it has been explored regularly. The probability p_i has been designed to give a higher likeliness to the lines that have not been visited often to be explored next.

The algorithm for the diversification procedure is as follows:

1. *Set* $s^c = s^*$
2. *Generate rnd and select i for which* $F_{i-1} \leq rnd \leq F_{i+1}$ *as neighbourhood N(s) for the coming iteration*
3. *Generate neighbourhood N(S) from resident i's schedule and evaluate* $f(s^p)$ *for each* s^p
4. *Choose the best solution* s^p *in N(S) from the candidate list of potential solutions and assign this solution to* s^k
5. *Set* $s^c = s^k$. *If* $s^k \leq s^*$ *then* $s^* = s^k$.

If there is still search time available return to step 2.

At the start of the PDS-procedure the search starts from the best current solution s*, decomposing the problem in a WEEK solution set containing all week shifts and a WKD solution set with all weekend shifts. At each iteration, a random number (*rnd*) is generated. The line (resident r) to be explored is selected based on *rnd* and the probability distribution function of visited neighbourhoods. We therefore choose the x^{th} line from the solution if it has been rarely visited. From the x^{th} line a candidate list of solutions is generated and the best solution s^k is selected as the current solution for the next iteration.

3.3.5.7 Short term Tabu Search List

For the BAEP-procedure the candidate solutions are obtained at each execution in the neighbourhood $N(s^c)$ of the current solution. We define the candidate list as CDT_LIST. All available residents would be considered for possible candidate exchanges, except when

on holiday or away at a conference. The candidate list is formed of all exchanges leading to new feasible solutions.

For example let us consider the schedule of resident 3, indicated as whose schedule is illustrated in Figure 3.9. Let us furthermore assume that he has the worst possible schedule and that he worked 4 different shifts: on day 6, 13, 18, and 23. In total there are 34 possible exchanges. By iterating through all possible exchanges we obtain the CDT_LIST. In Figure 3.9 this is illustrated as follows. Each arrow points to the different exchanges possible with other residents for the considered shift. For example, the Saturday shift *(t=6)* is currently worked by resident 3. The arrow pointing to the table directly below shows all available exchanges with the other residents. The first column identifies which resident is concerned, the column 2 this resident's current assignment, column 3 whether it can be added to the CDT_LIST and column 4 the information that will be stored in the CDT_LIST.

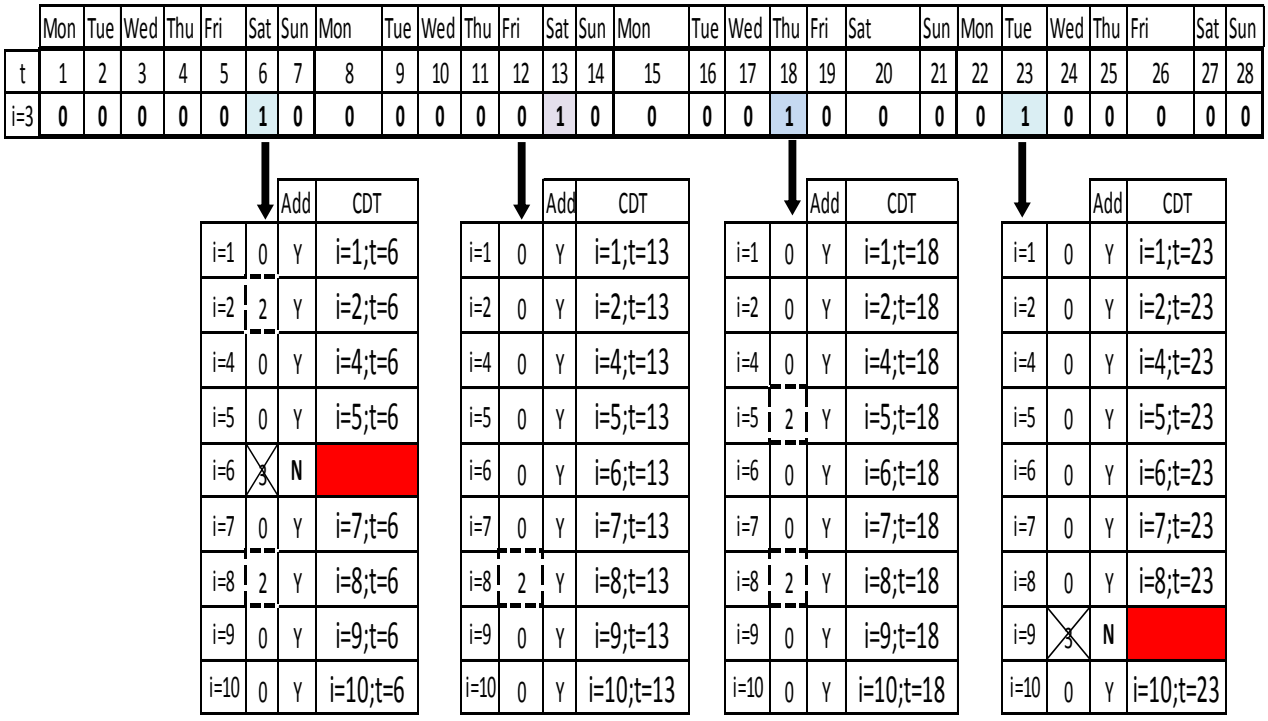


Figure 3.9 Establishing the CDT_LIST by considering all possible exchanges of 1 shift with all other residents

For example, let us consider the possible exchanges for t=6. For $i=1$, the 0 in column 2 in means that the resident is available and that he can therefore be added to CDT_LIST. For $i=2$ the 2 in column 2 means that the resident requested a day off, nevertheless he can be added to CDT_LIST. Figure 3.9 also shows a few exceptions that cannot be added to CDT_LIST. For example, for $i=6$ column 2 is marked 3, meaning that the resident is at a conference. This excludes resident 6 from being added to the CDT_LIST.

CDT_LIST	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
	i=1;t=6	i=2;t=6	i=4;t=6	i=5;t=6	i=7;t=6	i=8;t=6	i=9;t=6	i=10;t=6	i=1;t=13	i=2;t=13	i=4;t=13	i=5;t=13	i=6;t=13	i=7;t=13	i=8;t=13	i=9;t=13	i=10;t=13
	18	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
	i=1;t=18	i=2;t=18	i=4;t=18	i=5;t=18	i=6;t=18	i=7;t=18	i=8;t=18	i=9;t=18	i=10;t=18	i=1;t=23	i=2;t=23	i=4;t=23	i=5;t=23	i=6;t=23	i=7;t=23	i=8;t=23	i=10;t=23

Figure 3.10 The resulting unsorted CDT_LIST, before evaluation, once all possible exchanges have been added

Figure 3.10 shows the resulting CDT_LIST once the algorithm has iterated through all potential exchanges. This list is still unsorted though. Each candidate solution from CDT_LIST would be evaluated on the differential between the new line and the old line resulting in the evaluation of four lines. Let us assume that after evaluation of CDT_LIST the most interesting potential solution s'_k would be s'_{13}. Since i=6 and t=13, resident 6 would be now scheduled for the shift on day 13 instead of resident 3.

At the end of each iteration the new solution becomes the current solution denoted as s* = min[s*,s'_k]. The tabu-list will be updated with the positions in the R_{ij}-matrix of the exchanged shifts.

The tabu search consists of three procedures. The first procedure is SELECT_LINE, which is a selection procedure that keeps track of the line to choose. This procedure composes the list of the score of each resident in a list called LIST_SCORELINE. The two other procedures are the BAEP- and PDS-procedure used for establishing the candidate list. During the search process a static tabu list of length l keeps track of the last treated line and is updated at the end of each iteration. The length of the tabu list should always be inferior to the number of residents on duty so that SELECT_LINE will be able to select a new line.

Suppose that the tabu search is at the beginning of an iteration and that better local optima are still available but unknown. The first step will be to select the line that has the worst score from LIST_SCORELINE in the SELECT_LINE-procedure. The next step is to verify if the line is tabu or not. If so, the SELECT_LINE-procedure will select the next worst line. Every time a move is tabu another line will be selected and the list will be updated. The next step is to establish the candidate list that contains all feasible candidate solutions. This is mainly done throughout the BAEP-procedure. However, after MN iterations without improvement by the BAEP-procedure the PDS-procedure will establish this list. From the candidate list the best candidate will be chosen and the current solution s^* will become s^k. It is can also happen that no better solution will be found after a while into the search. The different stopping criteria define when the search will end.

Chapter IV – Prototype implementation and validation

4.1 Introduction

The suitability of our developments with the consulted hospitals was strongly dependent on the architectural design of the software structure. This Chapter therefore explains the different stages of software development. We begin by describing the first stage with a focus on the design and the validation process of the mathematical model accomplished in strong collaboration with the users (resident schedulers). Then, we provide information about the system development and implementation of the prototype.

As mentioned before, the prototype's development was executed in two separate stages that will be referred to as stage I (exploratory research project) and stage II (system development process) throughout this chapter. The first stage was aimed at formulating a mathematical model for the resident scheduling problem. During the course of this stage it was noted that schedulers define/apply a number of constraints that determine the fairness of individual resident's schedules that were difficult to define by merely interviews. On validation of the Resident Scheduling Model (RSP) it was therefore decided to develop a prototype, to allow the elicitation of further constraints by seeing schedulers in action. The second stage concerns the system development process. This process was completed by using an iterative development approach because the users were unable to identify the requirements of an information system. The learning experience allowed an increase in the efficiency and user-computer interactivity of earlier versions of the prototype to a level that allowed real-time usage.

4.2 Validation of the constraints and prototype

To develop the Resident Scheduling Model initial data had to be collected. This data collection process connected to stage I is illustrated in a decision tree in Figure 4.1. At the start of this project, data-collection (A) and interviews (B) were conducted in parallel at the Hôpital Enfant-Jésus and CHUL to obtain the necessary data. The schedulers provided data samples of implemented solutions and explained the constraints that had to be taken into account. In both hospitals this led to an initial mathematical model (C) that was submitted to the schedulers for additional comments. The feedback they provided allowed the verification of the formulation proposed (E) and led to the revision of several constraints (F) of the model before the final validation (G).

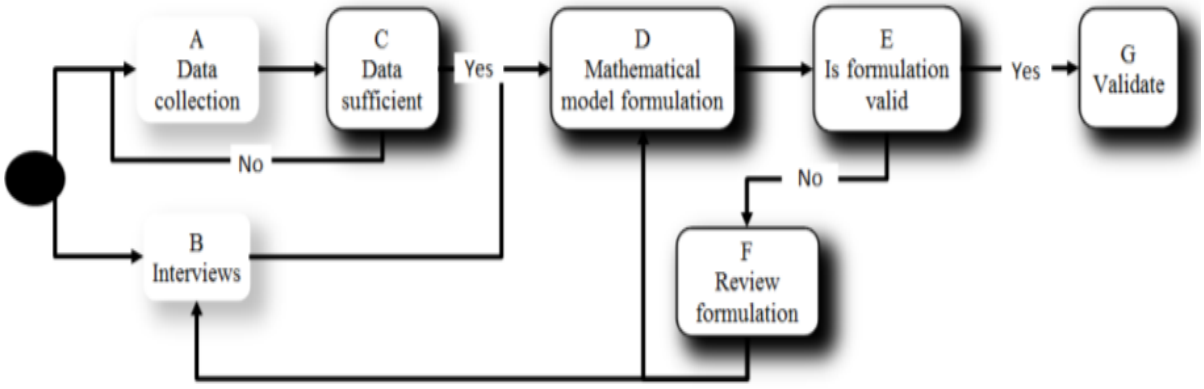


Figure 4.1 Mathematical model validation decision tree

During the data collection process, we discussed data examples with the resident schedulers in order to provide estimates for the constraint weights for the legal constraints and the internal ward constraints. The scheduler's initial scores for the RSP are indicated in Table 4.1. Some schedulers had difficulties assigning values to constraints. To make this more natural they were advised to fill out their preferences on a 1-10 ordinal integer scale. When comparing the answers given by both hospitals' residents, we could conclude that:

Integrity Constraints	
Minimum number of residents for each day	100
Maximum of one shift assigned to a person at the same time	N.A.
One day-type assigned to a person at the same time	N.A.
No more than the maximum number of shifts	100
Respect resting cycles after night shift	100
Assign sufficient resident to wards	100
No shifts outside of skills category	100

Internal ward constraints	
Resident with more seniority work less weekend shifts	1
Proportionate dispersion of days among residents	7
Proportionate dispersion of total shifts among residents	7
Avoid shift before congress	3
Avoid shift after congress	1
Avoid 2 short resting cycles per period	8

Legal Constraints	
12.08 Shifts at ward	
Respect requested days off	2
Respect congress days	10
Respect holidays	10
Allow 48-hour shift per resident	4
Take into account preferences	2
12.15 Weekends	
No more than two weekends per scheduling period	6
No more than two consecutive weekends	4
No more than two Saturdays	6
12.17 Holidays	
No weekend shift before holidays	8
No weekend shift after holidays	8
12.19 Undercoverage	
Doctors cover for residents	2

Table 4.1 Penalty values defined by residents for objective function

1. Only a slight variation in the legal constraints weights existed between both hospitals and all legal constraints were always included in the optimization model;
2. Between both hospitals different internal wards constraints were included and in shared constraints an amount of variation was visible.

Conclusion 2 concerned four internal ward constraints. The constraint *«Residents with more seniority work less weekend shifts»* was only valid in CHUL optimization model. This was also the case for constraint *«Avoid 2 short resting cycles per period»*. According to schedulers from the Enfant-Jésus it had never occurred that a resident was required to work two 48-hours shifts. Nevertheless, they validated the constraint and added a high weight to this constraint to prevent it from being violated. The schedulers at the Enfant-Jésus were persuaded that the total number of shifts and the days shifts worked had to be proportionaley divided among residents, so that all residents were treated with the same level of fairness. We therefore defined two constraints to ensure this in the model. These were defined as the the *proportionate dispersion of days among residents* (constraint 3.28), and *proportionate dispersion of total shifts among residents* (constraint 3.30). These

constraints were later on proposed to the CHUL's schedulers and were validated and were accorded the same weight as that given by the Enfant-Jésus's schedulers.

Since schedulers expressed an interest in using software to simplify their work it was possible to continue the cooperation with both hospitals. At the start of stage *II* different alternatives were considered for the software design. These alternatives are illustrated in Figure 4.2. A choice had to be made between a system with user interface and a simpler but unfriendly input/output system. Since scheduling tasks required several interactions with the scheduler, we favoured the use of an interface-based application. The choice of this type of system was furthermore motivated by the fact that human-interactivity behavioural observation was an active part of stage *II* to ellicitate the scheduling process from schedulers. Hence, the choice fell on a system that allowed a task-based process capable of handling a synchronous transactional process.

It was observed at this stage that schedulers use specific performance measures when creating schedules manually. Table 4.2 illustrates the performance measures that were promoted in stage *II*. These performance measures have different objectives. For example, the *total number of shifts* is used to distribute shifts fairly among residents. The *total number of weekend shifts* is used for the fair distribution of weekend shifts among

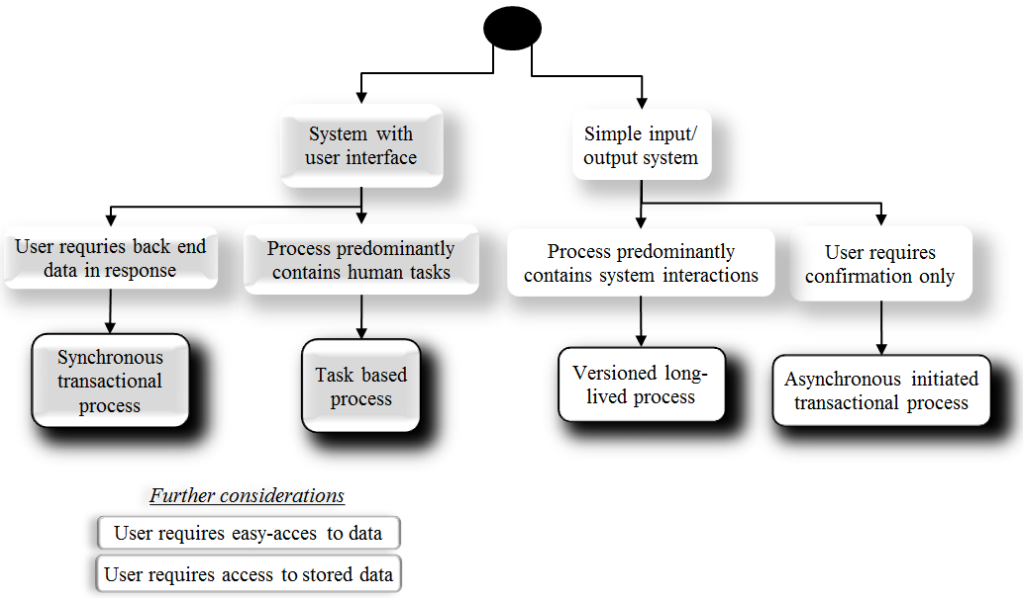


Figure 4.2 Decision tree for the design of the prototype

- *Total number of shifts*
- *Total number of weekend shifts*
- *Total number of Mondays*
- *Total number of Tuesdays*
- *Total number of Wednesdays*
- *Total number of Thursdays*
- *Total number of Fridays*
- *Total number of Saturdays*
- *Total number of Sundays*

Table 4.2 Performance measures used by schedulers

residents. Another example is the *total number of Tuesdays* and the *total number of Thursdays*. Since not all weekdays are considered to be of equal importance schedulers exchange Thursdays and Tuesdays to obtain a fairer distribution

The use of performance measures inspired the development of the construction algorithm. In the construction algorithm the performance measures were used to establish a priority-list of the days to be assigned. Together with the performance measures, a list of other factors that could be taken into account was also suggested to schedulers. The scheduler's scores for the importance of these parameters are reported in Table 4.3. It can be noted that schedulers gave more importance to factors that directly had an impact on the availability of residents to work shifts. For example, *Availability* (conference), and *Availability* (holiday) received the highest scores. Furthermore, *Weekend previous period* was also among the most important factors to users.

Initialisation parameters			
Difficulty Score		**Resident Score**	
Monday	4	Availability (Congres)	20
Tuesday	4	Availability (Holiday)	20
Wednesday	4	Weekend Previous Period	8
Thursday	3	Department	0
Friday	7	Seniority	0
Saturday	9	Category	0
Sunday	6		

Table 4.3 Construction parameters
values defined by residents

4.3 Prototype data structure

The design process led to a final model for the prototype which was implemented using VB.Net. The framework of the prototype encompasses a database and an interface. All data used in the prototype is stored in the database and the user has access to the information through the prototype's interface. Figure 4.4 illustrates the data structure and interdependent entities of the prototype. The entities are sollicitated in different ways at run-time. First of all, the user is only allowed access to the *Problem data* and *Constraint* entities thru the interface. The *neighbourhood structure* entity is defined as the specific solution obtained return by the *algorithm* entity, or search method. The *neighbourhood structure* is connected to the *problem data* and *constraints* since the data held in these two entities respectively determines the possibility of constraints being violated and the constraints that apply to the problem. The *algorithm* entity is responsible for managing all the data held within the entities *Constraints* and *Problem data* and uses this data to find new solutions. This description results in the following task description for the entities:

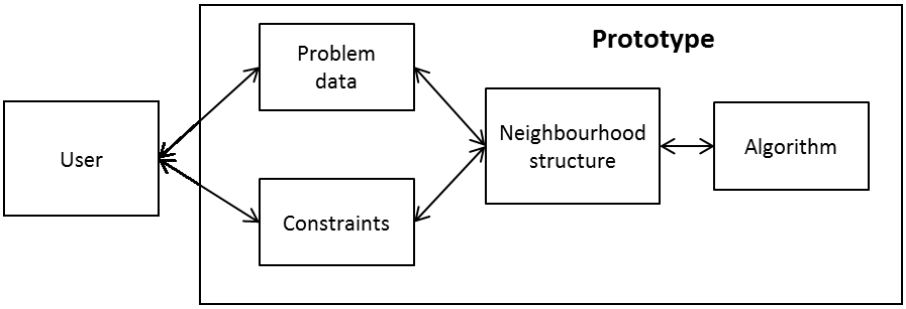


Figure 4.3 Influence of user-computer interactivity on different elements of the prototype

The entity *Constraints* is responsible for executing two tasks:

- Evaluate the penalty cost differential applying to a specific constraint resulting from a change in the neighbourhood of the current solution
- Count and list all violations of all different constraints

The entity *Neighbourhood structure* is responsible for accomplishing the following tasks

- Conserve the list of valid constraints within the data structure
- Conserve the penalty cost differential resulting from a change in the neighbourhood of the current solution and the penalty score of the current solution

The entity *Algorithm* executes a number of tasks:

- Choose the best improvement for a given exchange given the current solution and the list of all constraints
- Update and maintain the Neighbourhood structure used throughout the search
- Select the next operation to be performed
- Perform the next operation using the settings of Constraints and the Neighbourhood structure
- Continue the optimization process until the stopping criterion (*MI*, *TAST)* are met.

The design of the prototype data structure took into account different considerations. To make use of the expertise of the user, we allowed the user to access the different entities of the prototype. For the schedule creation, the user configures the penalty values of the optimisation algorithm and modifies the problem data at run-time. For this reason, the interface has been designed so that the user has access to two entities, the *Constraints* entity and the *Problem data* entity.

To close this chapter we will briefly discuss the insights that we gained from the overall experience of designing the prototype with the schedulers. The total time of creating a new schedule is not completely spent on assigning the shifts to residents. A large part of the job consists of adding the initial data, i.e. the names and time-off requests. In the two hospitals the time it took to create a new schedule was very different. In the CHUL this took approximately 8 hours, whereas this would only take 2 hours at the Enfant-Jésus. At both hospitals collecting and adding the initial data took approximately 1 hour. With a software program the time spent assigning shifts can be brought back to 1 hour for the CHUL. Here, if we suppose that the CHUL would use the prototype to create a schedule, the total time would therefore be situated at 2 hours, a 6-hours time saving. At the Enfant-Jésus assigning shifts could take as little as 30 minutes. A schedule could therefore be created in 1 ½ hours, a ½-hour gain. The hospital management of the *Enfant-Jésus* did not consider this to be a very large improvement.

On an overall basis the resident schedulers were very glad to discuss the efforts they had to make to create schedules. They found that the use of a program was an interesting option. In both hospitals they found the prototype user-friendly and expressed that it would be useful in their activities. They also found it interesting to learn more about the scheduling practices that were used in their respective fellow hospitals.

Chapter V – Experimental results

5.1 Introduction

This chapter presents the experimental results in four different parts. We begin by giving a formal description of the data samples' (or instances) characteristics that the prototype was tested on. Next, the parameters of the tabu search are calibrated using a partial set of these data samples. We furthermore present the results of comparative tests that have been performed between different algorithms. In the comparative tests the two distinct elements of the algorithm – the construction mechanism and the tabu search – have been connected to different mechanisms. The construction mechanism has been hybridized with a steepest descent search, whereas the tabu search has been simplified using random initialization. The resulting algorithms have been compared to the manual schedules, which served as a benchmark set. Lastly, a few manual schedules will be compared to the schedules provided by the prototype to highlight the differences and similarities between the constraint violations of both solutions.

The quality of the manual schedules has to be mentioned. Despite the fact that these schedules were elaborated without optimization tools, not all algorithms are successful in returning schedules of better quality. Since these manual schedules have been implemented in reality, their data is a good benchmark to determine whether the presented prototype could be implemented in real-life situations. If results can be obtained that are similar to the benchmark it can be concluded that the quality of the prototype's schedules is sufficient to meet the requirements of real-life situations. The use of the manual schedules as benchmark is furthermore motivated by the absence of known optimal solutions.

The benchmark set was split in two subsets for testing purposes. The first dataset was used for calibrating the tabu search parameters by cross-testing the parameters, changing the value of a single parameter while all other parameters remained fixed. The second set

of instances was used to compare the different algorithms and obtain more detailed conclusions about the efficiency of the algorithm.

The tests were performed on an *Intel i5, M430, 2.27 GHz* PC under a Windows 7 operating system. For the calibration tests, or preliminary tests, the total available search time (TAST) was 500 seconds, approximately 8.3 minutes. During final analysis TAST = 900 seconds, or 15 minutes. This time limit has been choosen to simulate real-life circumstances. In a hospital environment an acceptable search time is preferably short, for example 15 minutes. As mentioned for example by Warner (1976) and Burke et al. (2004b), schedulers usually consider it unnecessary to use a longer search time because the increments in solution quality become smaller and smaller over time. For example, a scheduler would accept a schedule obtained after a 15-minutes search knowing that a 30-minutes search could provide a 1% improvement in quality. In both the calibration and final tests the solution returned was the best solution found throughout the entire search. In section 5.3 we will mention the times the prototype needed to find a solution. These times corresponds to the time within which the best solution was found.

The quality of the schedule is defined by the penalty score of a solution returned by the algorithm. Since minimization problems are treated, the penalty score is considered to be a cost and a smaller cost is better.

5.2 Description of instances

The instance's characteristics are mentioned in Table 5.1. The instances were divided into the following categories based on their characteristics:

- *PC1: Problems with 1 ward;*
- *PC2: Problems with 2 wards.*

For example, in instance 2 a daily coverage of 2 residents on ward 1 was required. Put differently, this means that enough residents should be available to work 56 shifts

	PC1					PC2				
Instance	Ward 1	Available staff	Monthly needs	Total available shifts	Instance	Ward 1	Ward 2	Available staff	Monthly needs	Total available shifts
1	1	9	28	54	**4**	3	2	25	140	150
2	2	10	56	60	**5**	4	2	31	168	186
3	2	12	56	72	**-**	-	-	-	-	-
6	1	9	28	54	**7**	3	1	23	112	138
8	1	11	28	66	**9**	4	1	23	140	138
10	2	13	56	78	**11**	3	1	26	112	156

Table 5.1 Characteristics of instances used for calibration and computational tests

throughout the planning period. Instances 1-5 were used for the calibration tests, whereas instances 6-11 were used for final testing.

5.3 Parameter calibration

During preliminary testing a number of tests were performed to calibrate the tabu search parameters. The parameters concerned are the TL, the length of the tabu list, and MNI, the maximum number of iterations without improvement. Table 5.2 contains the nine pairs of combinations tested and the range of values used for each parameter. The length of the tabu list was tested in a range of 10%-30% of the total number of residents. The values of the maximum number of iterations without improvement were situated between 10 and 50. For each pair 5 runs were repeated at four run lengths: 30 s., 60 s., 180s. and 300s. The results presented in this section were obtained for TAST (total available search time) = 300 s. and are averaged over 5 runs.

Preliminary tests were also conducted for larger values of parameter pairs TL and MNI on several instances in an exploratory testing phase. For example, a pair of parameters

Parameter	Values
TL (length)	{0.1; 0.2; 0.3}
MNI (iterations)	{10; 25; 50}
{1} = {0.1;10}, {2} = {0.1;25}, {3} = {0.1;50}, {4} = {0.2;10}, {5} = {0.2;25}, {6} = {0.2;50}, {7} = {0.3;10}, {8} = {0.3;25}, {9} = {0.3;50}	

Table 5.2 Test settings for tabu search calibration

including TL = 0.6 and MNI= 100 was tested. However, these pairs never really provided better results than the range of values used for the calibration of the parameters. Hence, the range of values presents a reasonable dispersion of the values that represent efficient settings for the parameters.

Because there were three values for each parameter, nine pairs had to be tested. For example for setting {1}, a test was performed for a TL = 10% with a maximum of 10 iterations without improvement. Another test was performed for a TL = 20% with a maximum of 10 iterations.

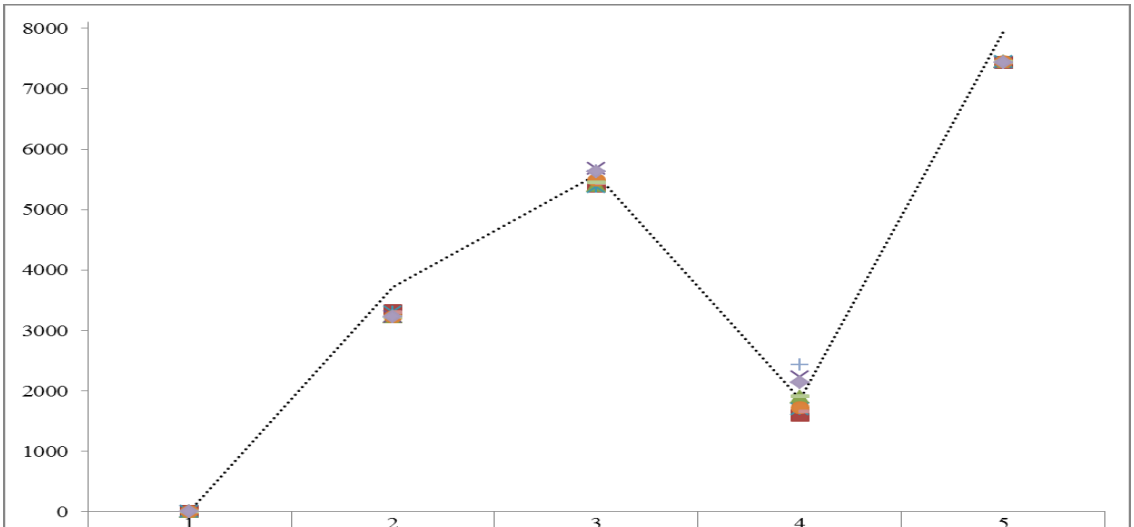
The following paragraphs will present the results obtained for the parameter calibration. First, the average results (over 5 runs of 300 s.) of each pair are presented to provide a macro-view of the results. During preliminary testing a constant behaviour was observed throughout all instances. An instance will therefore be described in more detail to provide a detailed understanding of the most important factors. To conclude this section the normalized difference of the pairs in respect to the benchmark data will be presented.

Figure 5.1 shows the average results for the different pairs of the construction tabu search and compares them to our benchmark set. The benchmark data set is illustrated by the dotted line. Instance 1 shows a very low cost (close to 0), which is significantly different from the costs of the other solutions. Instance 1 presents the same characteristics as instance 6. This is due to the fact that none of the residents had any absence (due to holidays or conference attendance) that could have resulted in potential complications. Hence, there was a very low probability that planning any staff member would lead to a penalty cost. For example, in instance 1 the results obtained show a variation of 3.2 to 8 points for the pairs {1} to {9} and the cost of the manual solution was 10 points.

The first observation that can be deduced from figure 5.1 is that instances 1 to 5 show a homogenous behaviour for the pairs {1}-{9}, i.e. there is a lack of variation between pairs. This illustrates that the results are not significantly influenced by the different parameters settings. Based on the fact that homogenous behaviour is observed between all pairs it can be suggested that none of the pairs dominate the solutions.

The second observation is that the overall performance of the pairs is comparable to the quality of the benchmark data. For example, in instance 3 the results obtained show a variation in the range of 5377.2-5673.6 for the pairs {1}-{9} and the cost of the manual solution is 5580. It can be implied that a solution obtained using no matter which pair should provide a similar solution quality as the manual schedule.

It is useful to illustrate the first observation with an example, for further discussions. A variation in the solution quality due to different parameter settings is visible in instance 4. This data is visible in the table accompanying the graph of Figure 5.1 and plots them out against the penalty cost of the manual solution, which is illustrated as the dotted line. The



	1	2	3	4	5
M	10	3716	5580	1858	7958
{1}	8	3324,4	5389,2	1600,4	7441,2
{2}	6,4	3231,6	5407,2	1895,6	7442,4
{3}	8	3232	5673,6	2226,4	7440
{4}	8	3302,4	5377,2	1698,4	7442,8
{5}	3,2	3231,6	5494,8	1722,8	7449,2
{6}	8	3232	5645,2	2427,2	7431,6
{7}	8	3304,4	5434,8	1656,8	7441,2
{8}	3,2	3231,6	5450,4	1919,6	7447,6
{9}	8	3232	5630	2135,6	7440

Figure 5.1: Average scores found by tabu search for the pairs {1}-{9} for instances 1-5 with TAST = 500 and n=5

benchmark penalty cost of the manual solution is indicated in the first line (M) of this Figure. A number of pairs provide solutions with lower penalty costs than the manual solution, although some pairs provide solutions of lesser quality (higher costs). When comparing the average cost of the solutions found by {3} and {4} – respectively 2226.4 and 1698.4 – it appears that MNI contributes to the quality of good solutions, leaving other parameters unchanged. Nevertheless, the contrary can be concluded by comparing {7} and {8} – respectively 1656.8 and 1919.6 – where it appears that MNI contributes negatively to the quality of a good solution.

The absence of variation is also the main factor observed in all instances. To illustrate this factor in more detail Figure 5.2 shows the average convergence over time towards the final solutions of instance 3. The lowest three curves in this Figure belong to the pairs {1}-{3} and show lower costs for the final solutions than the pairs {4}-{9}. At first sight, this instance seems to be a good example where the variation in the solution quality can be credited to different parameter settings. For the pairs {1}-{3} the value TL is constant and the value of MNI is variable. Hence, when considering the running time in this instance, the variation in the values of MNI has less influence of the performance of the tabu search than the length of the tabu list.

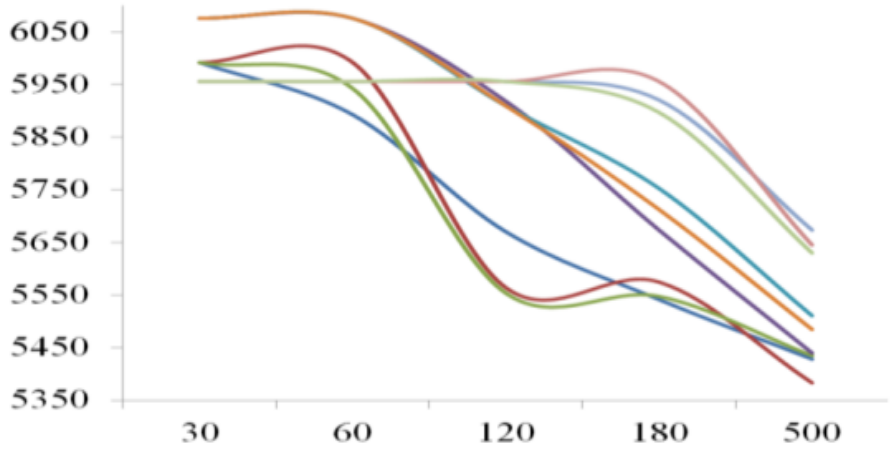


Figure 5.2 Solutions found for instance 3 as a function of computation time (s.) for the pairs {1}-{9}

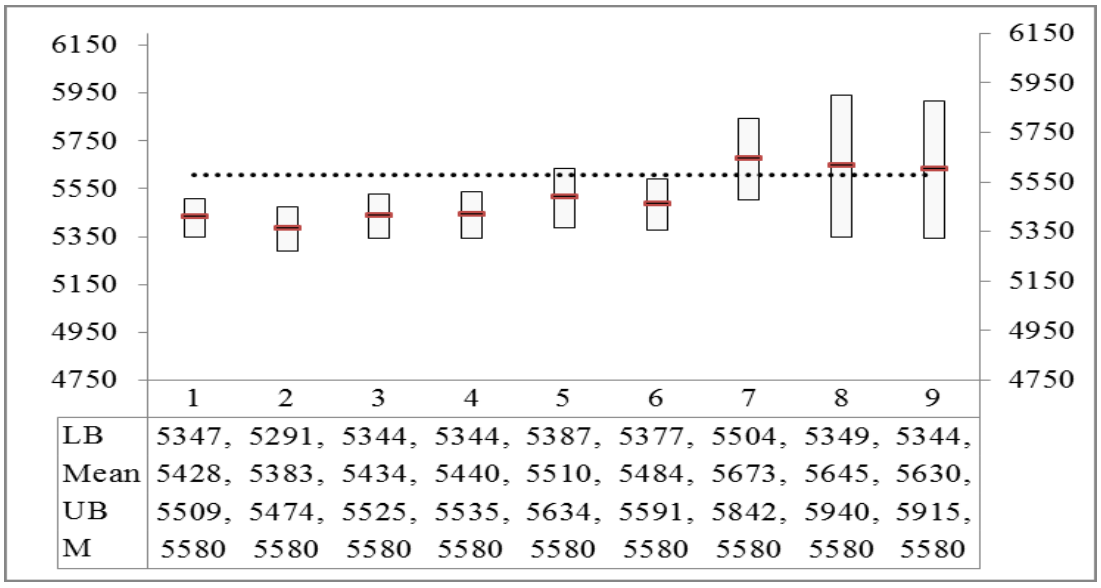


Figure 5.3 Average scores returned for instance 3 with TAST = 500s and 95% mean confidence interval (n=5, t=2.776)

Figure 5.3 shows the 95% mean confidence intervals (α= 0.05, n=5, t= 2,776, for each pair, after 300 s. search time) for the final results of instance 3 and plots them out against the penalty cost of the manual solution, which is illustrated as the dotted line. The upper and lower bounds as well as the mean – respectively *UB* and *LB* and *Mean* - of the pairs are indicated in the table accompanying Figure 5.3. We can conclude that none of the pairs dominate because the confidence intervals of pairs {1}-{9} overlap each other. For example, the means of pairs {1}, {3} and {4} – respectively 5428, 5434, and 5440 – are very close although the parameters are different. In this instance it can therefore be asserted that no significant relationship appears between the pairs. Even if Figure 5.2 displays a better performance over time in instance 3 for the pairs {1}-{3}, Figures 5.3 illustrates that no pair demonstrates a dominant behaviour. The conclusion is therefore that changing parameters values (for the range of retained values) does not have a strong influence on the cost of the solutions.

The results for instances 1 to 5 are illustrated in detail in Appendix B. Figure B.I illustrates the time graph for instances 1-5. The 95% mean confidence intervals (α= 0.05; n=5, t=2.776, for each pair: TAST= 300 s. search time) as well as the results for the different run-times (30 s., 60 s., 180s. and 300s) as a function of computation time (s.) are illustrated in Figures B.II-B.VI.

To conclude the parameter calibration we will consider the normalized difference for each pair, illustrated in Table 5.3. The first column mentions the instances. The other nine columns are the average normalized differences of the tested heuristic against the manually created schedule. The normalized difference is given by the following equation:

$$Value = \frac{Z_{MC} - Z_{IT}}{Z_{MC}} 100\% \quad (5.1)$$

Where Z_{mc} is the penalty score returned by the manually created schedule and Z_{it} is the score found by tabu search with construction. Each of the rows shows the average results of the pair on the instance (on 5 runs, after 300 s.). For example, in instance 5 pair {1} perfomed 6.4% better than the manually created schedule for the real-life period. The average overall performance of the pairs was between 1.8-15.6% better than the manually created schedules.

Instance	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}	{9}
1	20,0%	20,0%	-12,0%	36,0%	52,0%	20,0%	20,0%	20,0%	20,0%
2	12,3%	12,3%	11,7%	15,2%	15,3%	15,2%	15,2%	15,2%	15,2%
3	2,7%	3,5%	2,6%	2,5%	1,2%	1,7%	-1,7%	-1,2%	-0,9%
4	11,3%	6,2%	6,2%	-10,7%	3,1%	-7,7%	-24,8%	-31,7%	-20,5%
5	6,4%	6,5%	6,6%	6,5%	6,5%	6,5%	6,5%	6,4%	6,4%

Table 5.3 Average scores for normalized difference from manual schedule for pairs {1}-{9} (n=5)

Pair {1} shows the best average performance. Instance 1 was a very small problem with a low final cost making it possible to find improvements in the range of 20-52%. For pair {3} the results of instance 1 are 12% worse than the final solution. The tabu search with construction was able to find better schedules for all instances. As demonstrated in the previous example, the overall performance for the different parameter pairs does not suggest a strong dominance of any of these pairs. This is also visible in the normalized differences among pairs.

For the computational results, in the following section, a TL= 10% and MNI= 25 was choosen. This was motivated by the fact that a low standard deviation of the penalty cost

was obtained for these settings. On the overall performance these settings presented a robust performance.

5.4 Computational results of heuristics

The construction tabu search is an algorithm that selects an initial reference solution and then examines subsequent solutions by moving from one neighbourhood to another. The two procedures in the algorithm were decoupled to test their independent efficiency. To verify if the tabu search was capable of providing efficient solutions without starting its search in an initial solution that exploits the problem structure, it was coupled to a random initialization. To see if the construction mechanism did indeed exploit the problem structure and resulted in an interesting neighbourhood it was connected to a steepest descent heuristic. This resulted in the following algorithms:

TBI: Tabu Search with construction method
TBR: Tabu Search with random initialization
SDI: Steepest descent with construction method

All of these algorithms were compared to the benchmark of manual schedules. We therefore also have:
M: Manually created schedule

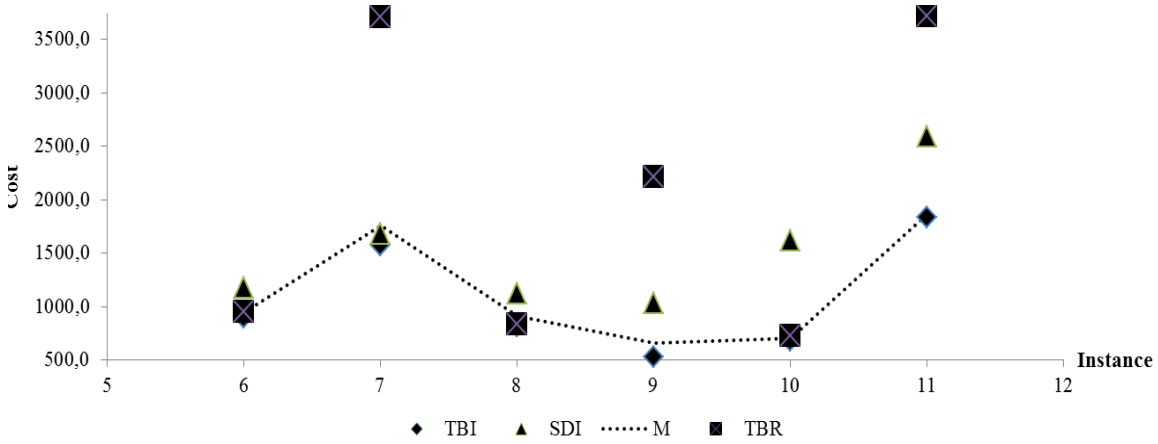


Figure 5.4 Average scores found by the three heuristics in instances 6-11 (n=20)

Instance	M	TBI	time	TBR	time	SDI	time
6	948,0	900,3	422,8	948,6	281,2	1180,0	1,0
7	1760,0	1571,3	404,1	3711,3	818,4	1684,0	90,2
8	916,0	814,2	757,4	835,2	264,9	1128,0	1,0
9	656,0	526,0	277,7	2215,5	619,1	1034,0	1,0
10	700,0	684,0	12,8	728,2	229,0	1620,0	1,7
11	1864,0	1834,0	452,4	3719,5	873,9	2592,0	801,0

Table 5.4 Average costs of the three algorithms and average time within which the best solution was found (n=20)

The following paragraphs first provide a macro-view of the performance of all methods. The conclusions drawn from these first results lead to a further comparison of the results between the TBI and the manual benchmark. For the tabu search, the maximum search time was set to 900 seconds (15 min.). For all algorithms each test was repeated a total of 20 times and the results presented were averaged over 20 runs.

Figure 5.4 shows the average results for the three heuristics compared to the manually created schedules. Table 5.4 shows the numerical values for the average results as well as the average time necessary to find a solution. If the average time is situated below the *TAST* (Total available search time) the search was stopped because MNI (Maximum number of iterations without improvement) was reached.

The TBI - illustrated as ♦ - returns average results that are of the same or better quality as the benchmark data set. For example, in instances 10 and 11 the gap between TBI and M

does not seem to be very large, which is best seen in Table 5.4. On the other hand, this gap is more significant in instance 7. In most instances, the results found by SDI – illustrated as▲- are of lesser quality then the benchmark data. The only exception is instance 7 where SDI returns a cost of 1684 and M a cost of 1760. Finally, the TBR – illustrated as ■ - returns different results. In some instances (6, 8, 10) its performance was better or slightly worse than the cost of the solutions in M. In larger instances (7, 9, 11) the cost of the proposed solutions was much higher than the solutions in M.

Table 5.4 shows that there is a relationship between the performance of TBR and the search time. Solutions that were close to the benchmark schedules were found in a short amount of time. On the other hand, bad solutions returned by the TBR show a large average search time. This is the case for instances 7, 9 and 11. After further investigation of these instances it was found that the solution returned by the TBR was the solution obtained once stopping rule of the total available search time (TAST) had been reached, and that the stopping rule of MNI never resulted in a search termination. This provides an explanation to the low overall performance by the TBR. The SDI provided solutions in a short time frame in most instances. The overall quality of these solutions can be considered low. The conclusion can be drawn that the SDI does not provide a constant solution quality when tested on a set of problems.

Instance	TBI	TBR	SDI
6	5,03%	-0,06%	-24,47%
7	10,72%	-110,87%	4,32%
8	11,11%	8,82%	-23,14%
9	19,82%	-237,73%	-57,62%
10	2,29%	-4,03%	-131,43%
11	1,61%	-99,55%	-39,06%

Table 5.5 Average scores for normalized difference from manual schedule for instances 6-11 (n=20)

Instance	6	7	8	9	10	11
TBI ≤ M	100,0%	100,0%	95,2%	100,0%	100,0%	100,0%
TBR≤M	57,1%	0,0%	76,2%	0,0%	66,7%	0,0%

Table 5.6 Percentage of runs where TBI and TBR outperformed M for instances 6-11 (n=20)

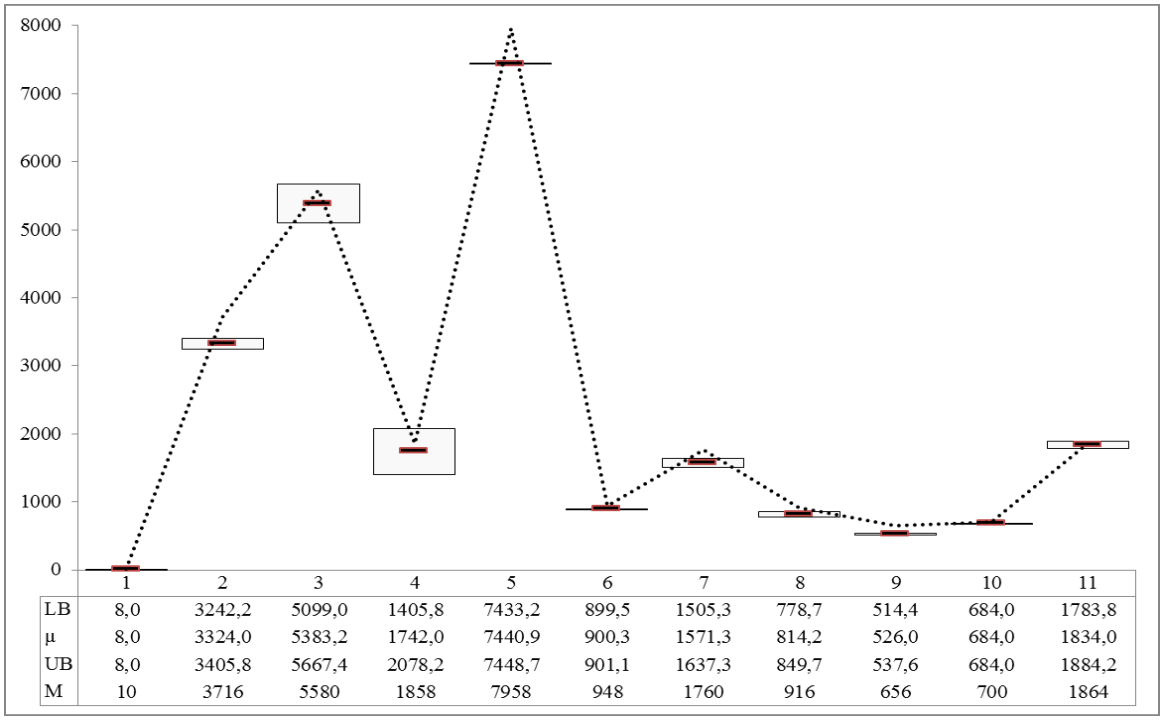
The normalized difference for each algorithm compared to the manually created schedules is given in Table 5.5, based on equation 5.1. The rows show the average results for each instance. In some instances the gap between the cost of TBI and M was fairly small. For example, in instance 11 there was only a 1.61% improvement with the use of TBI over M. The TBR returned results that were of lower quality in. For example, in instance 9 this was 237.73% higher than the cost of M.

Table 5.6 shows the percentage of runs where the TBI and TBR performed better than M. For the TBI this happened on almost every run. The only exception is instance 8 where there is a chance that M returns a better result than TBI aprroximately once every 20 runs. With the TBR this result is variable. For example, in instance 8 the TBR would return a solution with a lower cost than M about every 3 out of 4 runs, or more exactly 76.2% of the time.

It can be asserted that the TBI returns the best overall results out of the three algorithms. The efficiency of the random tabu search shows inconsistencies; the performance of the TBR seems to be about 5% less good than the results found by the TBI in a number of instances. Where this does not appear it seems that the fixed time limit of 900 seconds prevented the TBR from obtaining good results. It can be concluded that the tabu search has the potential of finding good quality solutions with random initial solutions. To obtain consistent results however it can be supposed that more optimization time is necessary. The SDI does not perform well in a general fashion, leading to the conclusion that the construction method is not efficient. The contribution of this method is merely that the search time can be reduced in most cases.

Based on these conclusions it is pertinent to provide a more detailed overview of the TBI's results and the benchmark. The following section will therefore deal with further results of the TBI.

Figure 5.5 plots the 95% mean confidence intervals of TBI against the benchmark data set for the instances used both for the parameter calibration and final testing. The interval parameters used for calibration (instances 1 to 5) have been mentioned previously. For final testing we can mention that n=20, α=0.05, t=2.0930 (instance 7 to 11). In instances 1, 2 and 5-10 the confidence interval is situated below the cost of the manual schedules. In



	1	2	3	4	5	6	7	8	9	10	11
LB	8,0	3242,2	5099,0	1405,8	7433,2	899,5	1505,3	778,7	514,4	684,0	1783,8
μ	8,0	3324,0	5383,2	1742,0	7440,9	900,3	1571,3	814,2	526,0	684,0	1834,0
UB	8,0	3405,8	5667,4	2078,2	7448,7	901,1	1637,3	849,7	537,6	684,0	1884,2
M	10	3716	5580	1858	7958	948	1760	916	656	700	1864

Figure 5.5 Mean 95%-confidence intervals, UB and LB for TBI for all tested instances

these instances it should be asserted that the means returned by the TBI are significantly different with a confidence level of α = 0.05. In instance 3, 4 and 11 the manual cost is situated in the upper half of the confidence interval. For example, in instance 4 the confidence interval is defined as the set of integer solutions situated in the range 1405.8-2078.2. The mean results of the manual schedules and the TBI are therefore not necessarily always significantly different.

In a number of instances the variation in the results was fairly low resulting in a short confidence interval. An example of this is instance 6 where the variation is low. This suggests that the schedule returned by the TBI was the same or very similar on every run.

For comparison Figure 5.6 illustrates the 95%-confidence intervals (α=0.05) for the TBI and TBR. The dark-coloured intervals belong to the TBR and the light-coloured intervals belong to the TBI. In some instances the confidence intervals partially overlap each other. In instances 6, 8 and 10 the TBI confidence interval is situated in the lower half of TBR's confidence interval. The large degree of variation in the results of the TBR for the instances 7, 9 and 11 suggests that better results could have been obtained with longer running times.

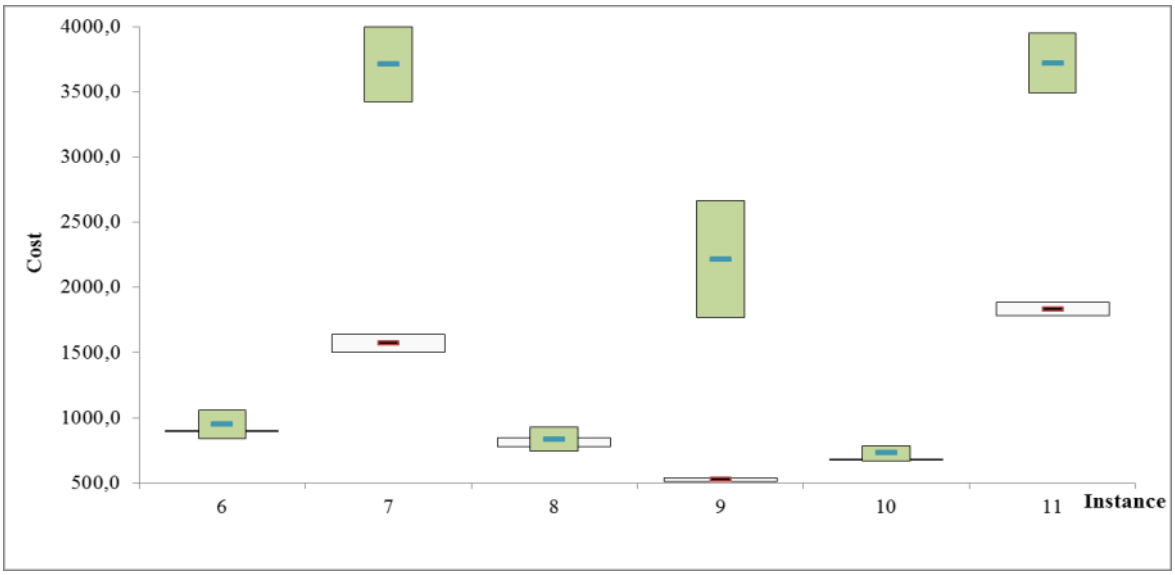
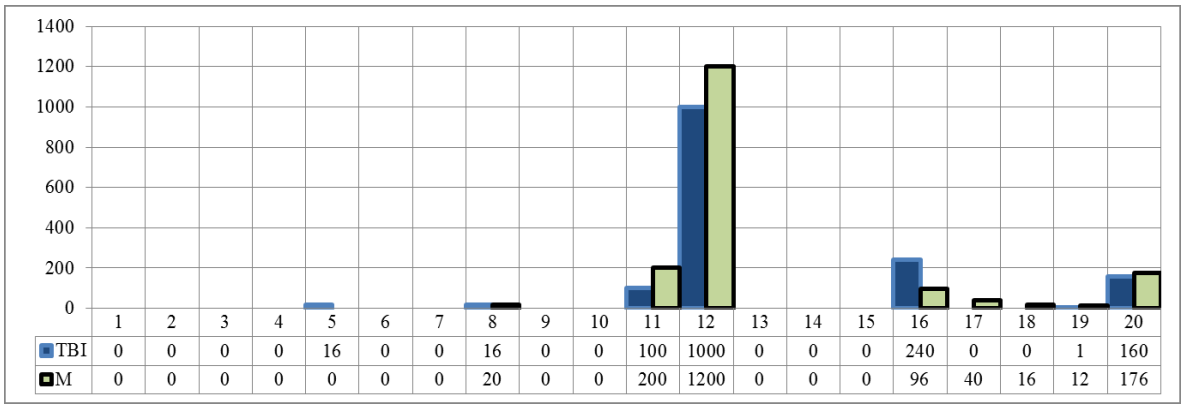


Figure 5.6 Comparison of 95%- mean Confidence intervals of TBI and TBR for all instances

5.5 Behaviour of manual process and prototype

Figure 5.7 shows a bar chart comparing a schedule returned by the TBI of instance 7 to the manual schedule used in real-life. It is interesting to note that the solutions are similar in appearance. There were 23 residents to be scheduled throughout this scheduling period. The total cost of the manual schedule was 1760, whereas this was 1533 for the TBI-schedule. In both cases a large number of constraints were fully satisfied. All



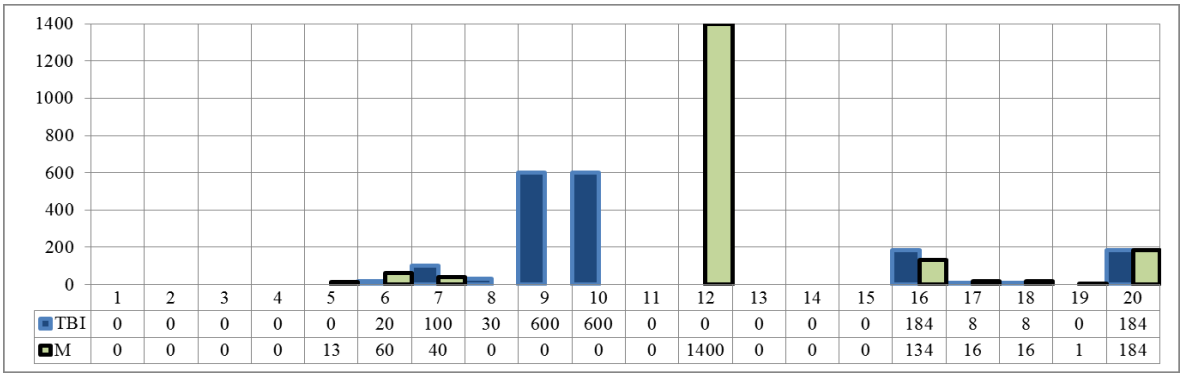
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TBI	0	0	0	0	16	0	0	16	0	0	100	1000	0	0	0	240	0	0	1	160
M	0	0	0	0	0	0	0	20	0	0	200	1200	0	0	0	96	40	16	12	176

Figure 5.7 Comparison of instance between M and TBI (total staff 23 residents, instance 7)

constraints mentioned in this section are described in Chapter 3. For example, constraints 1-4 did not pose any difficulties for schedulers or the TBI.

In neither schedules it was possible to fully satisfy the constraints 11, 12, 16 and 20. These constraints stand out in comparison to the other constraints by their high values. Constraint 11 reflects the constraint 3.9 (*No more than the maximum number of shifts allowed per resident*), constraint 12 reflects constraints 3.17-3.21 (*Respect resting cycles after night shift)*, constraint 16 reflects 3.28 (*Proportionate dispersion of days among residents*), constraint 20 reflects 3.3- (*Proportionate dispersion of total shifts among residents)*. In both solutions it was difficult to find a fair dispersion of days and total shifts. The hard constraint 11 was not fully satisfied in both cases. Ideally, a change would have to occur in the solutions in order to avoid such circumstances. In both solutions the resting cycles after night shifts had been shortened. The weight of this constraint was 100. Hence, in the manual solution this happened 12 times, and in the TBI's schedule 10 times

In Figure 5.8 an example is illustrated where two different schedules were returned by the TBI and the manual schedule for a scheduling period (instance 11) where 26 residents were available. For the manual schedule the total cost was 1864, whereas this was 1734 for the TBI-schedule. As in the previous example, a large number of constraints were fully satisfied. For example, constraints 1-5 did not lead to any penalty costs in the manual schedule nor in the TBI's schedule. In neither schedules constraints 16 and 20 were fully satisfied. Column 16 in the figure reflects constraint 3.28, the *Proportionate dispersion of days among residents*. Column 20 in the figure reflects constraint 3.30, the *Proportionate dispersion of total shifts among residents.* It is interesting to note that the proportionate



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TBI	0	0	0	0	0	20	100	30	600	600	0	0	0	0	0	184	8	8	0	184
M	0	0	0	0	13	60	40	0	0	0	0	1400	0	0	0	134	16	16	1	184

Figure 5.8 Comparison of instance between M and TBI
(total staff 26 residents, instance 11)

dispersion of total shifts does not seem to differ between both schedules.

The main difference between both schedules is situated in the satisfaction of constraints 9, 10 and 12. Constraints 9 and 10 respectively represent the score for doctors replacing resident and the score for undercoverage. Constraint 12 corresponds to Constraint 3.17-3.21, the *Respect resting cycles after night shift* – constraints. Schedulers show a stronger aversion to scheduling doctors to avoid undercoverage. Instead, they prefer to shorten the 72-hour cycle after a night shift to ensure sufficient coverage thus leading to a violation of constraint 20. In the TBI's schedule doctors replace residents on a number of occasions. In the prototype the weight of constraint 9 was 100, meaning that 6 shifts were performed by doctors. The weight of constraint 12 was 100. Hence, there were 14 occurences of shortened 72-hours resting cycles. For constraint 10 the weight was fixed at 100, meaning that there was undercoverage 6 days during the scheduling period. We can conclude that in this instance the penalty weights of the algorithm did not fully reflect the schedulers' preferences.

Considering the two different results it can be concluded that the same configuration can lead to different solutions. The solution presented in Figure 5.7 is more likely to be implemented in a hospital than that in Figure 5.8.

5.6 Summary of results

Comparing the different algorithms to the manually created schedules one gets a good impression of the quality of the schedules. It should be mentioned that the manual schedules are of good quality since not all algorithms are able to improve the schedule. Figure 5.7 shows that the TBI's and the manual solutions can share a number of characteristics. This, together with the computational results of the tests seems to suggest that the TBI should be capable of providing schedules that can be implemented in a real-life environment. Not all of these solutions were discussed with residents however, so that it is not possible to say that this really would be the case. The example of Figure 5.8 shows that the results obtained can still be different from the solutions that a scheduler

would want to implement, because the preferences of schedulers' should be adjusted in the penalty weights from one situation to another.

It is possible that the TBI can be outperformed by other methods because no proof has been given that the solutions are optimal. In our tests with the other algorithms the TBI provided the best overall results. When comparing the TBI with TBR it is shown that both overlap each other on a number of cases but that the TBI has an average performance of slightly better quality. The TBI performs better than SDI. This was the case in all different instances. Manual schedules are often of very good quality and should therefore not be underrated. Given the fact that the cost gaps of the TBI versus manual schedules can vary in the range of 1.61%-19.82% the TBI seems to provide results that are more consistent in quality.

Chapter VI - Conclusion

6.1 On the manual scheduling process framework

The contribution of this thesis to scheduling research was to clarify how manual scheduling techniques impact *schedule quality*. As has been mentioned it is difficult to directly associate this thesis to former works of researchers because, to our knowledge, such a description has not yet been made within the field of scheduling.

In optimisation methods the quality of a schedule is measured by the relative violation of soft constraints. The formulation of the objective function is aimed at satisfying all hard/feasibility constraints and accepting a relative violation of soft constraints whenever this is needed. The constraints are divided in different categories such as was done by Cheang et al. (2003) (§ 2.2.6) for example, who divided them into coverage, work regulations, time related and internal ward constraints. In a general sense, the quality of a schedule therefore reflects the degree of satisfaction of all the different constraint categories.

To analyse the techniques that influence a schedule's quality and see when schedulers use these techniques to influence the schedule quality - we presented a formal framework for the Manual Scheduling Process (MSP). This framework was translated into a scheduling algorithm, the Manually Restricted Space (MRP). The construction heuristic (decoupled from the tabu search) was a good (although not necessarily efficient) way to describe these activities. For example, determining the coverage level in the form of a knapsack formulation of a min-knapsack problem corresponds to constraints 3.2 and 3.9 (*Required number of residents for each period, No more than the maximum number of shifts allowed per resident*). The use of a best-fit decreasing heuristic to assign shifts shows that a scheduler does not assign shifts at random but tries to produce schedules of good quality. Last of all, we considered the conflict resolution in phase II which corresponds to

constraints 3.17-3.21 *(Respect resting cycles after night shift)*. These steps show that a scheduler has a highly developed notion of schedule quality and that he has integrated the satisfaction of constraints in the performance of his activities.

The fact that a scheduler's approach is efficient became clear when the heuristics were compared to the manual solutions. It turned out that manual schedules are often of very high quality and that it was sometimes difficult to find better solutions. From a theoretical perspective, it was suggested that the MSP can be described as a space restriction approach. However, we did not provide a formal proof for such a suggestion and we can therefore only use this term to explain what a scheduler does; by choosing the set of constraints, which he knows are difficult to satisfy, he restricts the search space. Such a set of constraints should include not only those constraints described by the MSP but also constraints for a fair dispersion of shifts (constraints 3.28 and 3.30)

The comparison of manual schedules and solutions by the Tabu Search with construction method (TBI) shows that a number of constraints were hard to satisfy, whereas the majority of constraints had a low cost or were not violated at all. This seems to suggest that the restriction approach could indeed be valid, although no empirical research has been provided, apart from this thesis that only proposes an analytical method, to describe such a method. The most difficult constraints were the constraints 3.9 (*No more than the maximum number of shifts allowed per resident*), 3.17-3.21 (*Respect resting cycles after night shift),* 3.28 (*Proportionate dispersion of days among residents*) and 3.30. (*Proportionate dispersion of total shifts among residents).* If we consider hospital residents as ressources, we could formulate a reduced set of constraints that fix a limit on the ressources. The optimal solution that respects this reduced set provides an optimal combination of ressources, while respecting an inactivity time after each task, and ensuring a proportionate use of all ressources.

We consider that schedule quality can be measured by the degree of violation of the existing constraint categories. However, when implementing the prototype resident schedulers often mentioned that the optimization method did not necessarily result in a fair distribution of shifts. They considered a schedule fair if all residents worked a proportionate number of total shifts and if days were divided proportionately among residents. We therefore defined the constraints 3.28 and 3.30 (*Proportionate dispersion of*

days among residents, Proportionate dispersion of total shifts among residents) to determine the *fairness* of individual resident's schedules. During the literature review different authors (consult Blöchliger (2004) for examples) suggested constraints to balance the workload over a long term period (6 to 12 months), or to take into account preferences. To our knowledge it is the first time that a model includes constraints that directly measure the short term fairness of individual schedules. This suggests that the term schedule quality could be extended to include a category that incorporates constraints on *fairness*. We believe that we were able to define these new constraints because we tried to question resident schedulers directly on how they perform their activities and not necessarily focused our work on the development of the optimization method.

6.2 On the prototype development and implementation

The prototype's development and implementation showed that an interface-based application is suitable for scheduling software. A clear interface also proved to be the key tool to establish a formal description of the manual scheduling process. For example, we noticed that schedulers use performance measures for evaluating a schedule. This observation led to the formulation of the so-called fairness constraints (3.28 and 3.30).

Another important point that came forward was that the implementation of software does not always have an advantage. This appeared clearly in instance 1, whose manual schedule measured a cost of 8 which was close to optimal (the lowest possible score being 0). This instance concerned a small department and the solution returned by the TBI was of almost equivalent quality. If we consider the total time it would have taken to develop this solution with the prototype (1 ½ hours) versus the manual schedule (2 hours) the savings in time would have been approximately half an hour. The potential time-saving could be interesting for larger hospital departments but not for smaller departments.

6.3 On the performance of the heuristics

The performance of the separate elements of the TBI, the construction mechanism of the initial solution and the tabu search, was illustrated by coupling these elements to a steepest descent heuristic and a random initialization. We saw that the TBI obtained the best overall results and that this was generally in a shorter time span than the other heuristics. Separated from the construction method, the tabu search (with random initialization) needed a long search time to obtain good quality solutions and avoid inconsistent solutions. The construction method helped the tabu search find better solutions in a shorter search time. The construction method is not an efficient individual method. However, it provides an upper bound that helps reduce the overall search time.

The results of the manual schedules and the TBI were not necessarily found to be significantly different. However the performance found in the computational results of the tests seems to suggest that the TBI should be capable of providing schedules that can be implemented in a real-life environment. Not all of these solutions were discussed with residents however, so that it is not possible to say if this really would be the case.

6.4 Future paths of development

The described phases of the manual scheduling framework occur in all different categories of scheduling problems and the framework in this thesis can therefore be extended to other fields of scheduling. Because scheduling problems are considered as overconstrained problems this framework suggests that problems can be simplified and thus allow for a better use of search time and more efficient approaches. In such an approach it is beneficial to consider employees as ressources to limit the search to the reduced set of constraints that allow the optimal combination of ressources, respecting inactivity time and ensuring a proportionate use of all ressources. An interesting research avenue is the use of a strongly reduced set of constraints in Artificial Intelligence, where

agents negotiate and exchange a few vital informations. A limited set of constraints could be used directly by the agents.

The resident scheduling problem discussed in this thesis has been defined for two hospital departments. However, the constraints encountered, apart from specific hospital constraints, are uniform across the province of Québec. An interesting research direction would be the implementation of the developed prototype in a large number of different hospitals. However, during the writing of this thesis, there were already reports that a revision of the collective agreement of residents was anticipated in a delay of two to three years. The implementation of a software program would therefore have to be postponed by at least several years.

This thesis only proposes a possible method for the analysis of a reduced framework and shows that some constraints are more likely to be contained in such a framework. Nevertheless, it does not define the valid range of values for these constraints. The comparison of the performance of the framework against different established methods (Mathematical programming, heuristics) would be an interesting research avenue, because it would establish a clearer definition of the constraints and assist in proving the performance and efficiency of the MSP framework.

The manner in which the preferences of decision makers are modelled also is an interesting research subject. For example, is it true that preferences can be modelled by an objective function that is defined by a weighted sum of penalties for the violations of the constraints? Would a scheduler use such an approach? If so, on what scale should these preferences be measured?

References

1. **Aickelin, U., K.A. Dowsland** (2000). "Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem" - *Journal of Scheduling,* Vol. 3, No. 3, pp.139–153.
2. **Arthur, J.L., A. Ravindran** (1981). "A multiple objective nurse scheduling model"*, IIE Transactions*, Vol. 13, No. 1, pp. 55- 60.
3. **Azaiez, M.N., S.S. Al Sharif**, (2005). "A 0-1 goal programming model for nurse scheduling" - *Computers & Operations Research*, Vol. 32, No. 3, pp. 491- 507.
4. **Bailey, I.** (1985). "Integrated days and shift personnel scheduling*"* - *Computer & Industrial Engineering*, Vol. 9, No. 4, pp. 539-544.
5. **Balintfy, J. L., C. R., Blackburn** (1969). "Generalized Multiple Choice Programming with Truncated Block Enumeration*"* - *Operations Research 17*, B222.
6. **Bard, J.F., H.W. Purnomo** (2005a). "Preference scheduling for nurses using column generation*"* - *European Journal of Operational Research*, Vol. 164, No. 2, pp. 510-534.
7. **Bard, J.F., H.W. Purnomo** (2005b). "A column generation-based approach to solve the preference scheduling problem for nurses with downgrading*"* - *Socio-Economic Planning Sciences*, Vol. 39, No. 3, pp. 193-213.
8. **Bard, J.F., H.W. Purnomo** (2007). "Cyclic preference scheduling of nurses using a Lagrangian-based heuristic*"* - *Journal of Scheduling*, Vol. 10, No. 1, pp. 5-23.
9. **Beliën, J., E. Demeulemeester** (2006). "Scheduling trainees at a hospital department using a branch-and-price approach" - *European Journal of Operational Research*, Vol. 175, pp 258-278.
10. **Berrada, I., J.A. Ferland, P. Michelon** (1996). "A multi-objective approach to nurse scheduling with both hard and soft constraints*"* - *Socio-Economic Planning Sciences*, Vol. 30, No. 3, pp. 183-193.

11. **Bilgin, B., P. de Causmaecker, B. Rossie, G. vanden Berghe** (2008). "Local search neighbourhoods to deal with a novel nurse rostering model*"* - *Proceedings of Practice and Theory of Automated Timetabling VII*.
12. **Blöchliger, I.** (2004), "Modelling staff scheduling problems. A tutorial*"* - *European Journal of Operational Research*, Vol. 158, No. 3, pp. 533-542.
13. **Boldy, D., C. O'Kane** (1982). "Health operational research – A selective overview" - *European Journal of Operational Research*, Vol. 10, No. 1, pp. 1-9.
14. **Bradley, D., J. Martin** (1991). "Continuous personnel scheduling algorithms: A literature review" - *Journal of the Society of Health Systems*, Vol. 2, No. 2, pp. 8–23.
15. **Burke, E.K., P. De Causmaecker, G. vanden Berghe** (1998). "A hybrid tabu search algorithm for the nurse rostering problem" - *Lecture Notes in Computer Science,* Vol. 1585, pp. 187–194.
16. **Burke, E.K., P. Cowling, P. De Causmaecker, G. vanden Berghe** (2001). "A memetic approach to the nurse rostering problem" - *Applied Intelligence*, Vol. 15, No. 3, pp. 199- 214.
17. **Burke E.K., P. De Causmaecker, G. vanden Berghe, H. van Landeghem** (2004b). "The state of the art of nurse rostering" - *Journal of Scheduling*, Vol. 7, No. 6, pp. 441–499.
18. **de Causmaecker, P. , G. Vanden Berghe** (2010). "Towards a reference model for timetabling and rostering" – *Annals of Operations Research*, Vol. 63, pp. 105–128.
19. **Chan, P., G. Weil** (2001). "Cyclical staff scheduling using constraint logic programming" - E. K. Burke, W. Erben (eds.), *Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, Lecture Notes in Computer Science*, Vol. 2079, pp. 159–175.
20. **Cheang, B., H. Li, A. Lim, B. Rodrigues** (2003). "Nurse rostering problems - a bibliographic survey" - *European Journal of Operational Research,* Vol. 151, No. 3, pp. 447-460.
21. **Chiaramonte M. V., L. M. Chiaramonte** (2008). "An agent-based nurse rostering system under minimal staffing conditions" - *International Journal of Production Economics*, Vol. 114, No. 2, pp. 697–713.

22. **Chiarandini, M., A. Schaerf, and F. Tiozzo** (2000). "Solving employee timetabling problems with flexible workload using tabu search" - E. K. Burke, W. Erben (eds.), *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling*, pp. 298–302.
23. **Cohn A., S. Root, C. Kymissis, J. Esses, N. Westmoreland** (2009). "Scheduling Medical Residents at Boston University School of Medicine"- *Interfaces,* Vol. 39, No. 3, pp. 186–195.
24. **Darmoni S.J., A. Fajner., N. Mahe, A. Leforestier, M. Vondracek** (1994). "Horoplan: computer-assisted nurse scheduling using constrained-based programming" - *Journal of the Society for Health Systems*, Vol. 5, pp.41-54.
25. **Dowsland, K.A.** (1998). "Nurse scheduling with tabu search and strategic oscillation", *European Journal of Operational Research*, Vol. 106, No. 2-3, pp. 393-407. (Strategic oscillation based on originally: Glover and Laguna, (1993)).
26. **Ernst A.T., H. Jiang, M. Krishnamoorthy, B. Owens, D. Sier** (2004). "An annotated bibliography of personnel scheduling and rostering", *Annals of Operations Research* – Vol. 127, No. 1-4, pp. 21–144.
27. **Ferland J. A., I. Berrada, I. Nabli, B. Ahiod, P. Michelon, V. Gascon, E. Gagne** (2001). "Generalized assignment type goal programming problem: application to nurse scheduling" - *Journal of Heuristics*, Vol. 7, No. 4, pp. 391–413.
28. **Fries, B**. (1976). "Bibliography of operations research in health-care systems" - *Operations Research*, Vol. 24, No. 5, pp. 801–804.
29. **Gaspero, L.D., J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, W. Slany**, (2003) "The minimum shift design problem: theory and practice". G. Di Battista, U. Zwick, *Proceedings. of the 11th Annual European Symposium on Algorithms (ESA 2003)*, No. 2832, Lecture Notes in Computer Science, Springer-Verlag, Berlin-Heidelberg, pp. 593–604. ISBN 3-540-20064-9.
30. **Glover, F., M. Laguna** (1993). "Tabu Search" - *Modern Heuristic Techniques for Combinatorial Problems* – Blackwell Scientific Publications. C. Reeves (Ed.), Oxford pp. 70- 150.

31. **Goodman M. D., K. A. Dowsland, J. M. Thompson** (2007). "A grasp-knapsack hybrid for a nurse-scheduling problem" - *Journal of Heuristics*, Vol. 15, No. 4, pp. 351-379.
32. **Hung, R.** (1995). "Hospital nurse scheduling" - *Journal of Nursing Administration*, Vol. 25, No. (7/8), pp. 21–23.
33. **Jaumard, B., F. Semet, T. Vovor** (1998). "A generalized linear programming model for nurse scheduling" - *European Journal of Operational Research*, Vol. 107, No. 1, pp. 1–18.
34. **Jelinek, R., J. Kavois** (1992). "Nurse staffing and scheduling: Past solutions and future directions" - *Journal of the Society for Health Systems*, Vol. 3, No. 4, pp. 75–82.
35. **Kreeft, D., A. Ruiz and B. Lamond** (2010), "Allowing user interaction in timetable scheduling software", *Actes de la 5e Conférence Francophone en Gestion et Ingénierie des Systèmes Hospitaliers* (GISEH 2010), Clermont-Ferrand, France.
36. **Maenhout, B., M. Vanhoucke** (2005). "An electromagnetism meta-heuristic for the nurse scheduling problem" - *Journal of Heuristics*. Vol. 13, No.4, pp. 359-385,
37. **Maenhout, B., M. Vanhoucke** (2006a). "New computation results for the nurse scheduling problem: A Scatter Search Algorithm" - *Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science*, Springer-Verlag, Berlin-Heidelberg, Vol. 3906, pp. 159- 170.
38. **Maenhout, B., M. Vanhoucke** (2006b). "A comparison and hybridization of crossover operators for the nurse scheduling problem" - *Special Issue on Multidisciplinary Scheduling: Theory and Applications (MISTA); Guest Editors: Graham Kendall, Lei Lei and Michael Pinedo* – Vol. 159, No. 1, pp. 333-353, Annals of Operations Research.
39. **Maenhout, B., M. Vanhoucke** (2007). "Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem", *Journal of Scheduling,* Vol. 13, No. 1, pp. 77-93.
40. **Maenhout, B., M. Vanhoucke** (2009). "Branching Strategies in a Branch- and- Price Approach for a Multiple Objective Nurse Scheduling Problem" - *Journal of Scheduling,* Vol. 13, No 1, pp. 77-93.

41. **Meyer auf'm Hofe, H.** (1997), "ConPlan/SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction" - *Proceedings of the Third International Conference on the Practical Application of Constraint Technology*, London, pp. 257–271.
42. **Mietus D.G.** (1994). "Understanding planning for effective decision support: a cognitive task analysis of nurse scheduling" – Doctoral thesis, Rijksuniversiteit Groningen, Groningen.
43. **Miller H.E., P. William, J.R. Gustave** (1976). "Nurse scheduling using mathematical programming" - *Operations Research*, Vol. 24, No. 5, pp. 857–870.
44. **Musa, A.A., U. Saxena** (1984). "Scheduling nurses using goal programming techniques" - *IIE Transactions*, Vol. 16, No. 3, pp. 216-221.
45. **Okada M., M. Okada** (1988). "Prolog-based system for nursing staff scheduling implemented on a personal computer" - *Computers and Biomedical Research*, Vol. 21, No. 1, pp. 53–63.
46. **Okada, M.** (1992). "An approach to the generalised nurse scheduling problem - Generation of a declarative program to represent institution-specific knowledge" - *Computers and Biomedical Research*, Vol. 25, No. 5, pp. 417–434.
47. **Oldenkamp, J.H., J.L. Simons** (1995). "Quality factors of nursing scheduling" - *Planning and Evaluation, Proceedings AMICE*, Vol. 95, pp. 69–74.
48. **Osogami, T., H. Imai** (2000). "Classification of various neighbourhood operations for the nurse scheduling problem", *Lecture Notes in Computer Science*, Vol. 1969, pp. 72-83.
49. **Ozkarahan, I., J.E. Bailey** (1988). "Goal programming model subsystem of a flexible nurse scheduling support system" - *IIE Transactions*, Vol. 20, No. 3, pp. 306 - 316.
50. **Ozkarahan, I.** (1989). "A flexible nurse scheduling support system" - *Computer Methods and Programs in Biomedicine* – Vol. 30, No. 2/3, pp.145–53.
51. **Ozkarahan, I.** (1994). "A scheduling model for hospital residents" - *Journal of Medical Systems*, Vol.18, No. 5, pp. 251-265.
52. **Ozkarahan, I., S. Topaloglu (2010).** "Integration of OR and AI: Medical Residency Scheduling Application", *International Journal of Computers, Information Technology and Engineering*, Vol. 4, No. 1, pp. 1-17.

53. **Pecora, J.E.** (2008). "Iterative restricted space search: a solving approach based on hybridization" – Université Laval, Québec, Doctoral thesis.
54. **Price, E. M.** (1970). "Techniques to improve staffing" - *The American Journal of Nursing*, Vol. 70, No. 10, pp. 2112-2115.
55. **Randhawa, S.U., D. Sitompul** (1993). "A heuristic-based computerized nurse scheduling system" - *Computers & Operations Research and their Application to Problems of World Concern: an international journal*, Vol. 20, pp. 837- 844.
56. **Roth E.M., D.D. Woods** (1989). "Cognitive task analysis: an approach to knowledge acquisition for intelligent system design" - *G.Guida and C. Tasso (eds.): Topics in Expert System Design*, Elsevier Science Publ. B.V. (North-Holland).
57. **Sherali H. D., M. H. Ramahi, Q. J. Saifee** (2002). "Hospital resident scheduling problem" - *Production Planning & Control*, Vol. 13, No. 2, pp. 220-233.
58. **Sitompul, D., S. U. Radhawa** (1990). "Nurse scheduling: a state-of-the-art review" – *Journal of the Society Health Systems*, Vol. 2, No. 1, pp. 62- 72.
59. **Warner D. M., J. Prawda** (1972). "A mathematical programming model for scheduling nursing personnel in a hospital" - *Management Science*, Vol. 19, No. 4, Application Series, Part 1, pp. 411- 422.
60. **Warner, D.M.** (1976). "Scheduling nursing personnel according to nursing preferences: A mathematical programming approach", *Operations Research*, Vol. 24, No. 5, pp. 842–856.
61. **Warner, M., B.J. Keller, S.H. Martel** (1991). "Automated nurse scheduling" - *Journal of the Society for Health Systems*, Vol. 2, No. 2, pp. 66–80.
62. **Weil, G., K. Heus, P. François** (1994). "Informatisation de l'unité de soins du futur, Gymnaste: Aide à l'élaboration des roulements infirmiers. Du traitement des absences au management participatif" - Springer Verlag, France, *Informatique et Santé Collection*, Vol. 7.
63. **Weil, G., K. Heus, P. Francois, M. Poujade** (1995). "Constraint programming for nurse scheduling" - *Engineering in Medicine and Biology Magazine,* IEEE, Vol. 14, No. 4, pp. 417–422.
64. **de Werra, D.** (1985). "An introduction to timetabling" - *European Journal of Operational Research*, Vol. 19, No. 2, pp. 151-162.

65. **Wolfe, H.** (1964). "A multiple assignment model for staffing nursing units" - Johns Hopkins University, Baltimore, Doctoral thesis.
66. **Wolfe, H., J. P. Young** (1965a). "Staffing the nursing unit, part 1: Controlled variable staffing" - *Nursing Research*, pp. 237- 243.
67. **Wolfe, H., J. P. Young** (1965b). "Staffing the nursing unit, Part 2: The multiple assignment technique" – *Nursing Research*, Vol. 14, No. 4, pp. 299- 303.

Appendix A: Formulation for the Resident Scheduling Problem - Chapter 3

The mathematical model discussed in this appendix is also discussed in Chapter 3 where the focus was maintained on the main constraints essential to resident schedules by excluding several constraints from the description. The model discussed in this Appendix differs from the initial model by including a penalty for unfeasibility. Unfeasibility occurs whenever there is undercoverage, a shortage in available residents which has to be overcome by doctors in real-life. Further constraints included in this Appendix are derived from the wishes that resident scheduler's made but were not described in the collective agreement. The model described in Chapter 3 can be extended to include all of these constraints. The prototype included all the constraints described in this Appendix.

Objective function

$Minimize$: (3.1)

$$PEN\_MC \sum_{t=1}^{n}\sum_{k=1}^{W} d\_MC_{kt}^{-} + PEN\_MAX \sum_{i=1}^{m} d\_max_i^{+} + PEN\_RDO \frac{\varrho}{\sum_{i=1}^{m} d_{rdo_i}^{+}} + PEN\_C \sum_{i=1}^{m} d\_c_i^{+}$$
$$+ PEN\_H \sum_{i=1}^{m} d\_h_i^{+} + PEN\_EXC\_72AFT \sum_{t=1}^{n} d\_exc\_72aft_i^{+} + PEN\_WKD \sum_{i=1}^{m} d\_WKD_i^{+}$$
$$+ PEN\_CWKD \sum_{i=1}^{m} d\_CWKD_i^{+} + PEN\_TSD^{-} \sum_{i=1}^{m} d\_TSD_i^{-} + PEN\_TSD^{+} \sum_{i=1}^{m} d\_TSD_i^{+}$$
$$+ PEN\_DISP^{-} \sum_{i=1}^{m} d\_DISP_i^{-} + PEN\_DISP^{+} \sum_{i=1}^{m} d\_DISP_i^{+}$$
$$+ PEN\_SINGSAT \sum_{i=1}^{m} d\_singsat_i^{+} + PEN\_BCON \sum_{t=1}^{m}\sum_{i=1}^{m} d\_bcon_i^{+}$$
$$+ PEN\_ACON \sum_{t=1}^{n}\sum_{i=1}^{m} d\_acon_i^{+} + PEN\_DOC \sum_{i=1}^{m} x\_doc_t$$

The objective function defines the penalty values over the slack variables. It includes the same penalty values as in Chapter 3. New penalty values are *PEN_SINGSAT, PEN_BCON, PEN_ACON* and *PEN_DOC*.

A.1.1 Integrity constraints

Integrity constraints define the constraints among shifts and are used by the algorithm during the search process. Constraints (3.2)-(3.8) are fully described in Chapter 3.

Required number of residents each day

$$\sum_{i \in W(k)} x_{it} + d\_MC_{kt}^{-} - d\_MC_{kt}^{+} = MC_{kt} \qquad \forall k; \forall t \qquad (3.2)$$

Maximum of one shift assigned to a person at the same time

$$\sum_{k}^{K} x_{i \in W(k)t} \leq 1 \qquad \forall i; \forall t \qquad (3.3)$$

One day-type assigned to a person at the same time

$$x_{it} + c_{it} + h_{it} \leq 1 \qquad \forall i; \forall t \qquad (3.4)$$

$$rdo_{it} + c_{it} + h_{it} \leq 1 \qquad \forall i; \forall t \qquad (3.5)$$

$$rdo_{it} = RDO_{it}\,,\ c_{it} = C_{it}\,,\ h_{it} = H_{it} \qquad \forall i; \forall t \qquad (3.6) - (3.8)$$

Where the following parameters apply

Parameters

RDO_{it} *1* *If resident i requested a day off at day t*

0 *Otherwise*

C_{it} *1* *If resident i is at conference at day t*

	0	*Otherwise*
H_{it}	*1*	*If resident i is on holiday at day t*
	0	*Otherwise.*

Where the parameters are set values of either 0 or 1, the latter meaning in simpler terms that scheduling a resident on that day will results in a penalty.

And the following decision variables

Decision variables

x_{it}	*1*	*If resident i is working at day t*
	1	*Otherwise*
rdo_{it}	*1*	*If resident i requested a day off at day t*
	0	*Otherwise*
c_{it}	*1*	*If resident i is in conference at day t*
	4	*Otherwise*
h_{it}	*1*	*If resident i is on holiday at day t*
	0	*Otherwise*

A.1.2 Legal constraints

Constraints (3.9)-(3.27) are fully described in Chapter 3.

No more than the maximum number of shifts

$$\sum_{t=1}^{n} x_{i\in \mathrm{W(k)}t} + d\_max_i^- - d\_max_i^+ = \ MAX_i \qquad \forall \text{ i} \qquad (3.9)$$

where

$$MAX_i = \left\lceil \frac{(28 - \ legal\ off\ days)}{28} * 6 \right\rceil \quad (3.10)$$

Respect requested days off

$$x_{it} + \ rdo_{it} \ - rdo\_exc_{it} \ \leq 1 \qquad \forall \text{ i; } \forall \text{ t} \qquad (3.11)$$

$$\sum_{t=1}^{n} rdo\_exc_{it} \ + \ d\_rdo_i^- - d\_rdo_i^+ \ = 0 \qquad \forall \text{ i} \qquad (3.12)$$

Respect conference days

$$x_{it} + \ c_{it} \ - c\_exc_{it} \ \leq 1 \qquad \forall \text{ i; } \forall \text{ t} \qquad (3.13)$$

$$\sum_{t=1}^{n} c\_exc_{it} \ + d\_c_i^- - \ d_{c_i}^+ = 0 \qquad \forall \text{ i} \qquad (3.14)$$

Respect holidays

$$x_{it} + \ h_{it} \ - h\_exc_{it} \ \leq 1 \qquad \forall \text{ i, } \forall \text{ t} \qquad (3.15)$$

$$\sum_{t=1}^{n} h\_exc_{it} \ + \ d\_h_{it}^- - \ d_{h_{it}}^+ \ = 0 \qquad \forall \text{ i} \qquad (3.16)$$

Respect resting cycles after night shift

$$x_{i(t-2)} + \ x_{i(t-1)} + x_{it} + x_{i(t+1)} + x_{i(t+2)} \ \leq \ 1 \qquad \forall \text{ i ; } \forall \text{ t} \qquad (3.17)$$

$$x_{i(t-3)} + \ x_{it} - \ exc\_72bef_{it} \leq 1 \qquad \forall \text{ i ; } \forall \text{ t} \qquad (3.18)$$

$$x_{i(t+3)} + \ x_{it} - exc\_72aft_{it} \ \leq 1 \qquad \forall \text{ i; } \forall \text{ t} \qquad (3.19)$$

$$\sum_{t=1}^{n} exc\_72aft_{it} \ + \ d\_exc\_72aft_i^- - \ d\_exc\_72aft_i^+ = \mathbb{Q} \qquad \forall \text{ i} \qquad (3.20)$$

$$\sum_{t=1}^{n} exc\_72bef_{it} \ + \ d\_exc\_72bef_{it}^- - \ d\_exc\_72bef_{it}^+ = \mathbb{Q} \qquad \forall \text{ i} \qquad (3.21)$$

No more than two weekends per scheduling period

$$\sum_{t \in WKD} x_{it} + d\_WKD_{it}^- - d\_WKD_{it}^+ \ = 2 \qquad \forall \text{ i} \qquad (3.22)$$

No more than two consecutive weekends

$$\sum_{t\in PW} x_{it} + \sum_{t\in WKDSET1} x_{it} - CWKD_{ia} = 2 \qquad \forall i \qquad (3.23)$$

$$\sum_{t=26\in PW}^{28} x_{it} + \sum_{t\in WKDSET1} x_{it} + \sum_{t\in WKDSET2} x_{it} - CWKD_{ib} = 2 \qquad \forall i \qquad (3.24)$$

$$\sum_{t\in WKDSET1} x_{it} + \sum_{t\in WKDSET2} x_{it} + \sum_{t\in WKDSET3} x_{it} - CWKD_{ic} = 2 \qquad \forall i \qquad (3.25)$$

$$\sum_{t\in WKDSET2} x_{it} + \sum_{t\in WKDSET3} x_{it} + \sum_{t\in WKDSET4} x_{it} - CWKD_{id} = 2 \qquad \forall i \qquad (3.26)$$

$$CWKD_{ia} + CWKD_{ib} + CWKD_{ic} + CWKD_{id} + d\_CWKD_i^- - d\_CWKD_i^+ = 0 \qquad \forall i \qquad (3.27)$$

A.1.3 Hospital constraints

The hospital defined a few constraints that were not legally defined but were still considered to be a very important factor in the fairness of each schedule. This third set of constraints reflects the hospital's concern that all residents are treated fairly. Constraints (3.28)-(3.30) are described in Chapter 3.

Proportionate dispersion of days among residents

$$disp\_score_i + d\_disp_i^- - d_{disp}{}_i^+ = \frac{\left(\sum_{i\in PW} disp_{score_i}\right)}{m} \qquad \forall i \qquad (3.28)$$

Each resident has his own dispersion score, defined as:

$$disp\_score_i = \qquad (3.29)$$
$$MO\_PEN \sum_{t\in MONDAY} x_{it} + TU\_PEN \sum_{t\in TUESDAY} x_{it} + WE\_PEN \sum_{t\in WEDNESDAY} x_{it} + TH\_PEN \sum_{t\in THURSDAY} x_{it}$$
$$+ FR\_PEN \sum_{t\in FRIDAY} x_{it} + SA\_PEN \sum_{t\in SATURDAY} x_{it} + SU\_PEN \sum_{t\in SUNDAY} x_{it}$$

Proportionate dispersion of total shifts among residents

$$\sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{i \in W_k} x_{it} + d\_tsd_i^- - d_{tsd}{}_i^+ = \frac{\sum_{i=1}^{m} \sum_{t=1}^{n} x_{it}}{m} \qquad \forall \text{ i} \qquad (3.30)$$

Junior and senior residents are assigned to the same shift

$$\sum_{i \in JUNIOR} x_{it} - \sum_{i \in SENIOR} x_{it} \leq 0 \qquad \forall \text{ i} ; \forall \text{ t} \qquad (3.31)$$

No more than one Saturday per scheduling period

$$\sum_{k=1}^{W} \sum_{t \in SATURDAY} x_{it} + d\_singsat_i^- - d\_singsat_i^+ \leq 1 \qquad \forall \text{ i} \qquad (3.32)$$

No weekend shift before a week of holidays

$$\sum_{t=1}^{n} h_{i \in WEEK,t} - 4\, h_{bef_{exc}\ i,WEEK} \leq 0 \qquad \forall \text{ i} ; \forall \text{ WEEK} \qquad (3.33)$$

$$WEEK = \{WEEK1, \ldots, WEEK4\}$$

$$\sum_{t \in WKDSET} x_{i,t} + 3\, h_{bef_{exc}\ i,WEEK} \leq 3 \qquad \forall \text{ i} ; \forall \text{ WKDSET} \qquad (3.34)$$

$$WDKSET = \{WKDSET1, .., WKDSET4\}$$

No weekend shift after a week of holidays

$$\sum_{t=1}^{n} h_{i \in WEEK,t} - 4\, h_{aft_{exc}\ i,WEEK} \leq 0 \qquad \forall \text{ i} ; \forall \text{ WEEK} \qquad (3.35)$$

$$WEEK = \{WEEK1, \ldots, WEEK4\}$$

$$\sum_{t \in WKDSET} x_{i,t} + 3\, h_{aft_{exc}\ i,WEEK} \leq 3 \qquad \forall \text{ i} ; \forall \text{ WKDSET} \qquad (3.36)$$

$$WDKSET = \{WKDSET1, .., WKDSET4\}$$

Avoid night shift before conference days

$$x_{it} + c_{it+1} + d\_bcon_i^- - d\_bcon_i^+ = 1 \qquad \forall \text{ i} ; \forall \text{ t} \qquad (3.37)$$

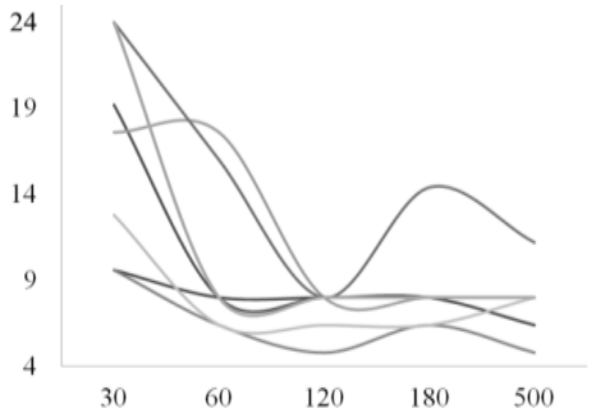
Avoid night shift after conference days

$$x_{it} + c_{it-1} + d\_acon_i^- - d\_acon_i^+ = 1 \qquad \forall i ; \forall t \qquad (3.38)$$

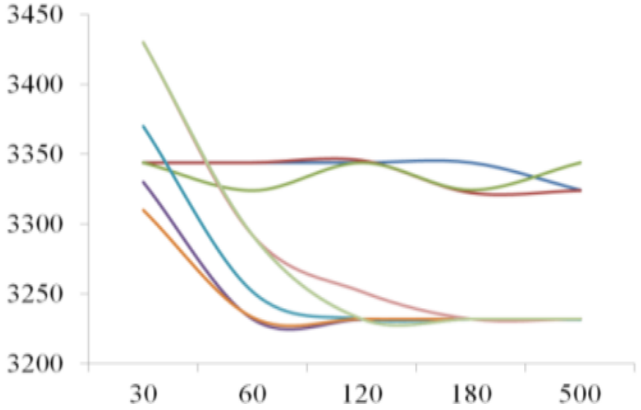
Doctors replace residents

$$x\_doc_t - d\_MC_{kt}^- = 0 \qquad \forall t ; \forall k \qquad (3.39)$$

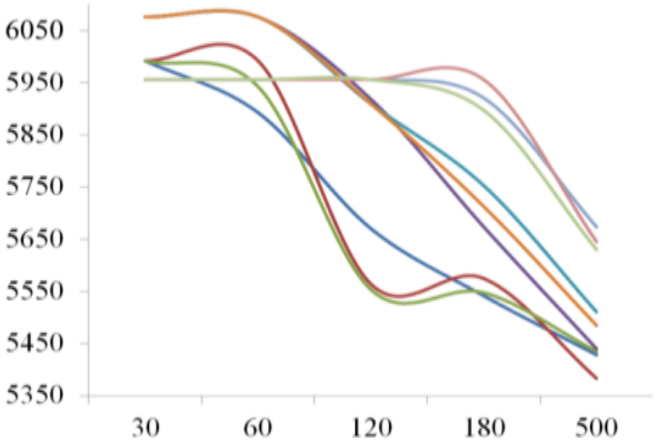
Appendix B: Preliminary test results – Chapter 5



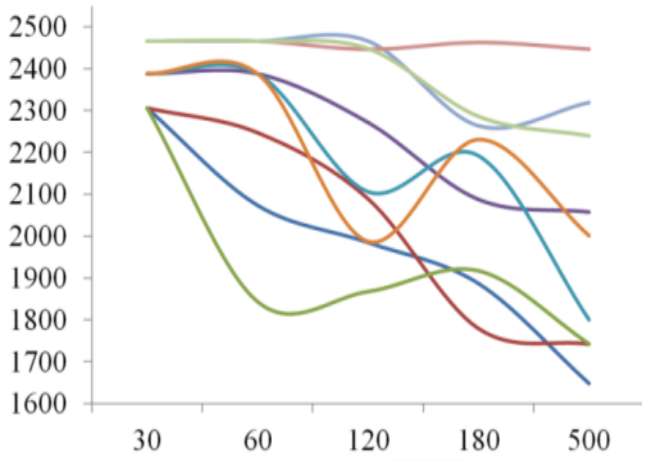
(a) Results for PC1: Instance 1



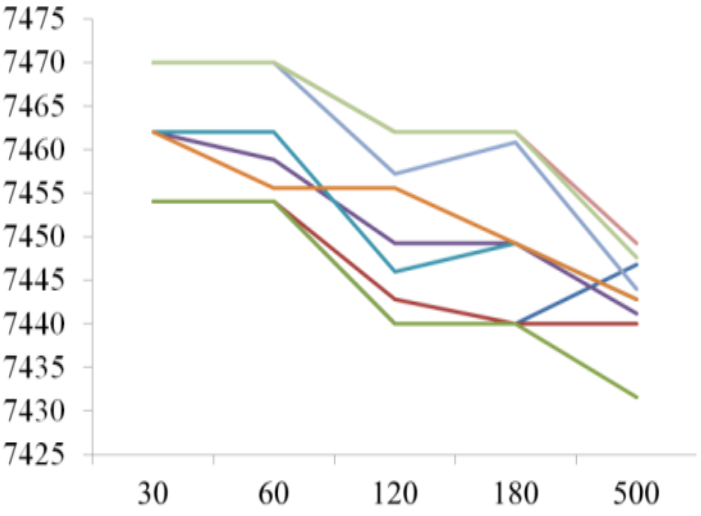
(b) Results for PC1: Instance 2



(c) Results for PC1: Instance 3



(d) Results for PC2: Instance 4



(e) Results for PC2: Instance 5

Figure B.I: Comparing performance of tabu search over a range of parameters

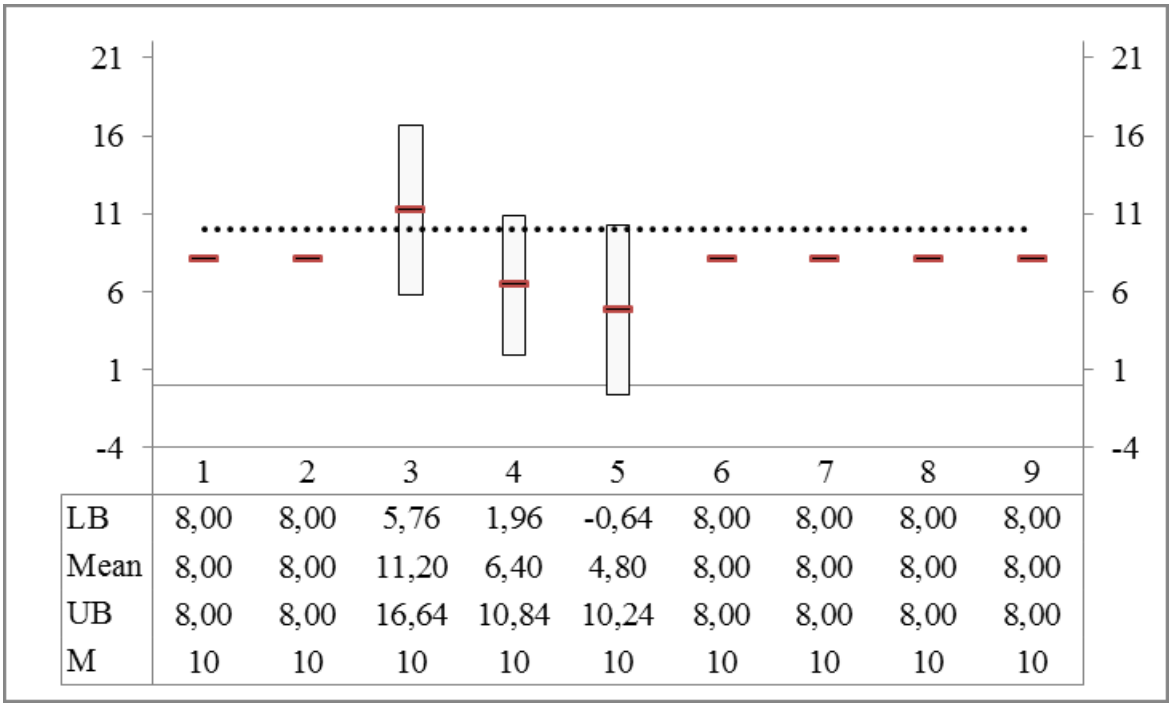


Figure B.II 95% confidence interval PC1: Instance 1

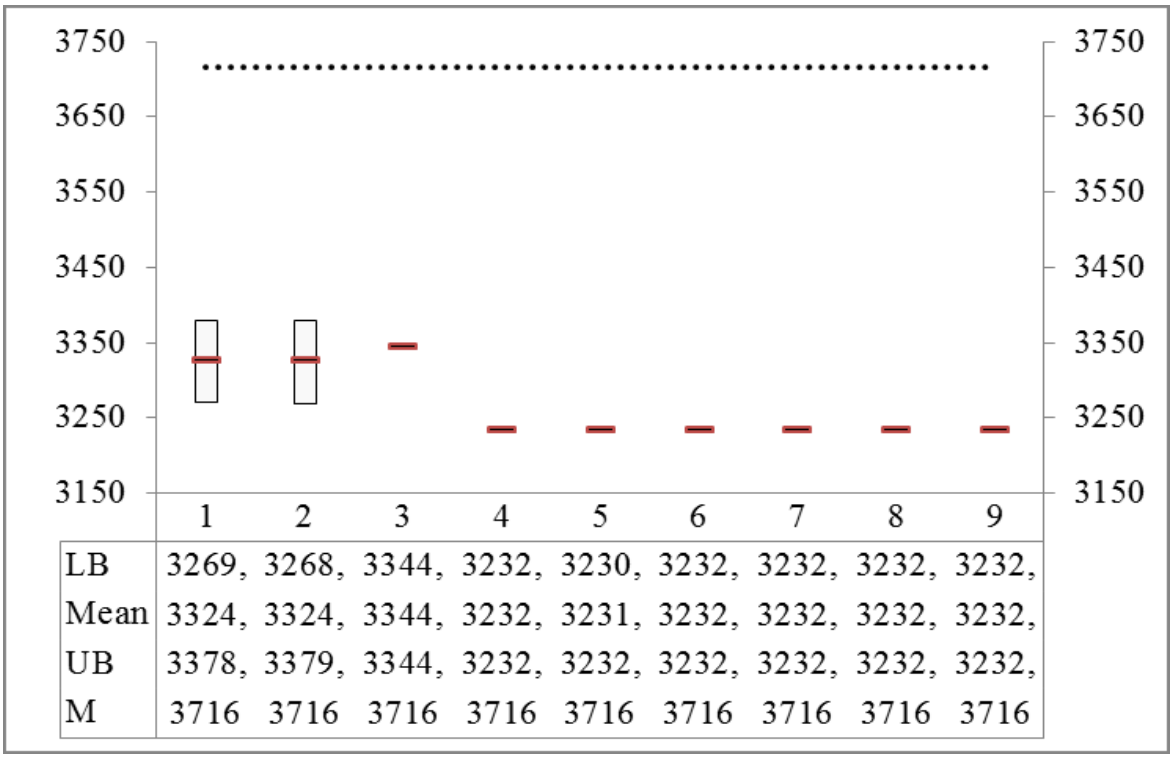


Figure B.III 95% confidence interval PC1: Instance 2

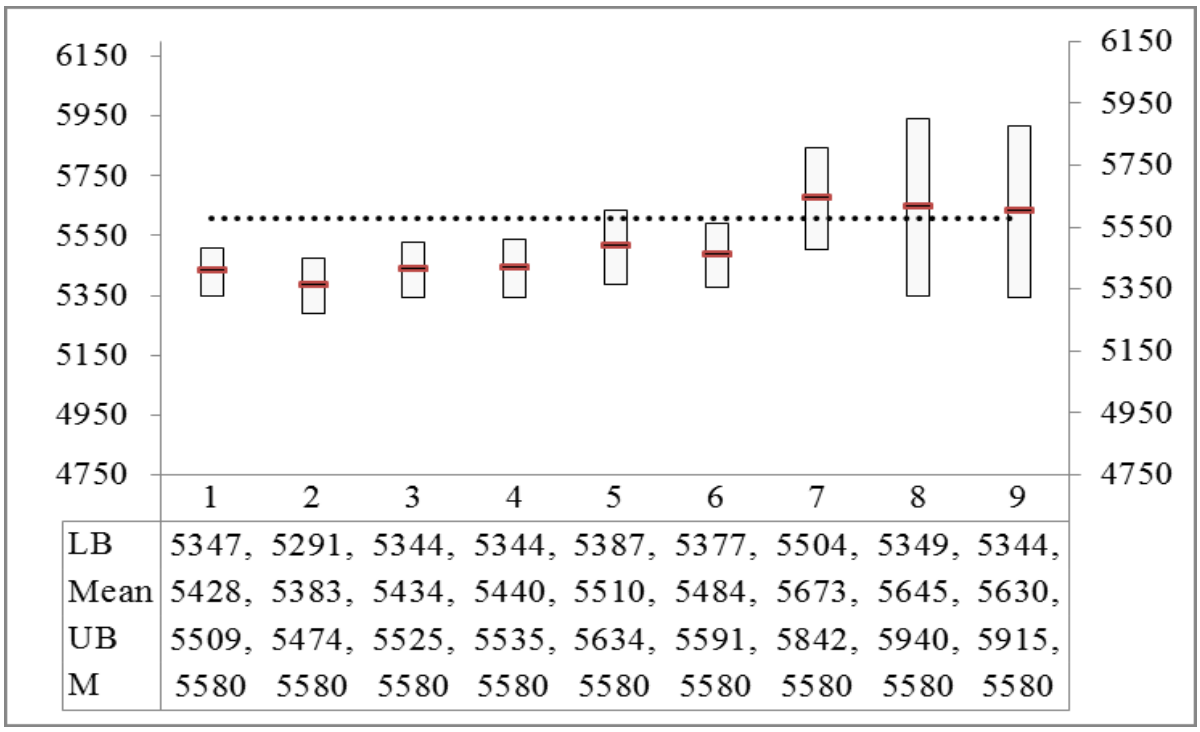


Figure B.IV 95% confidence interval PC2: Instance 3

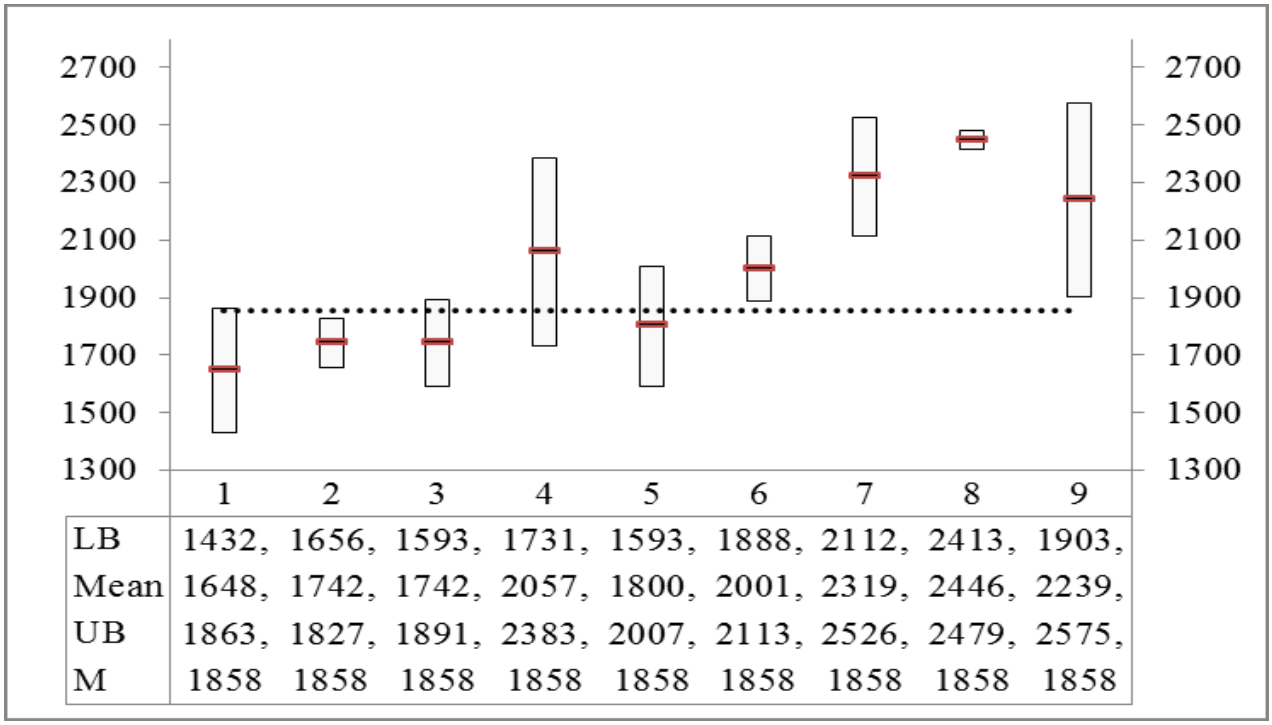


Figure B.V 95% confidence interval PC2: Instance 4

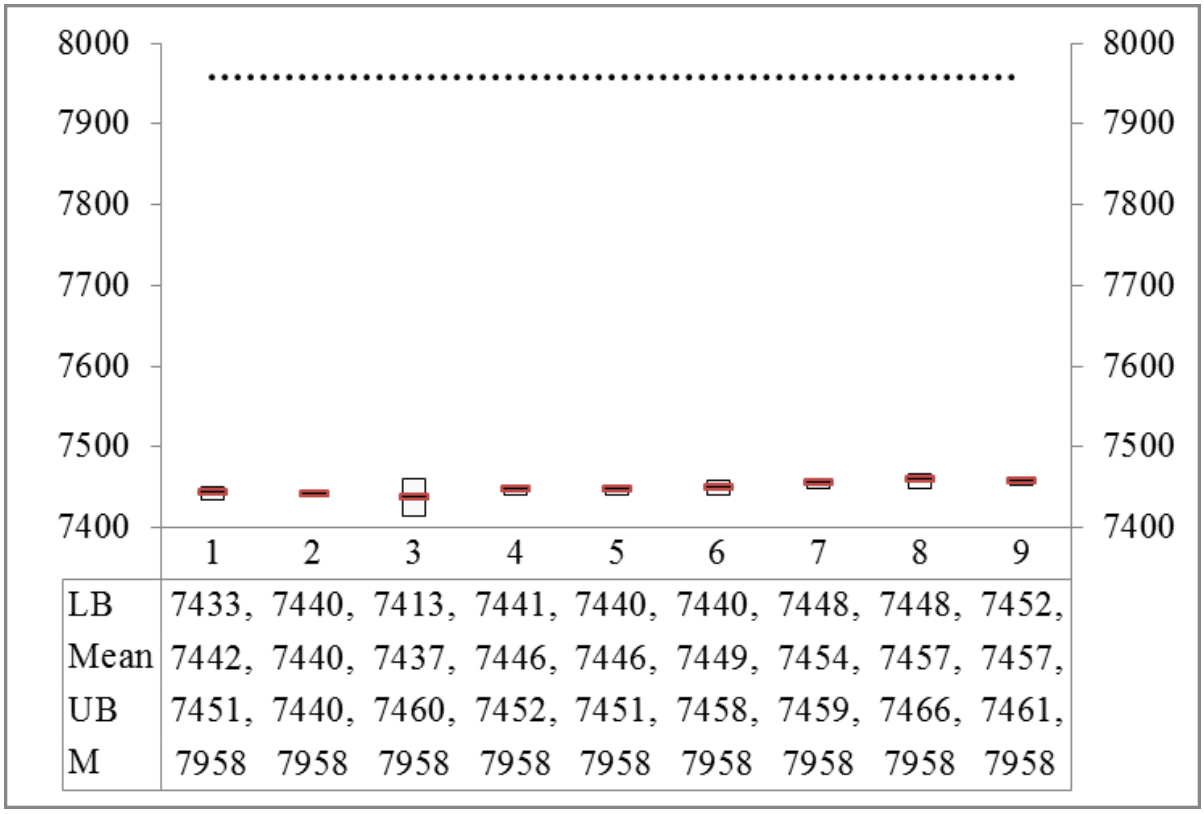


Figure B.VI 95% confidence interval PC2: Instance 5

Appendix C: Interface description – Chapter 4

C.1 Introduction

The following section presents an overview of the prototype's interface. First, the main screens of the interface will be defined. Next, the different buttons in the main screen are described with the help of sceenshots. Finally, the prototype's submenus will be described.

C.2 Main screens interface

The prototype main interface has two different screens that are associated to the design stage of the schedule. The main interface can be set to either the *Day editor* or *Shift editor*. If no schedule has been created by the construction tabu search, the main work screen will be the *Day editor*. When the heuristic has generated a schedule the main work screen will be the *Shift editor*. However, both screens can be used in parallel even if a schedule has been created. Within the *Day editor* the user enter the data for the availability of each resident by specifying for each resident when he will be available or cannot be scheduled.

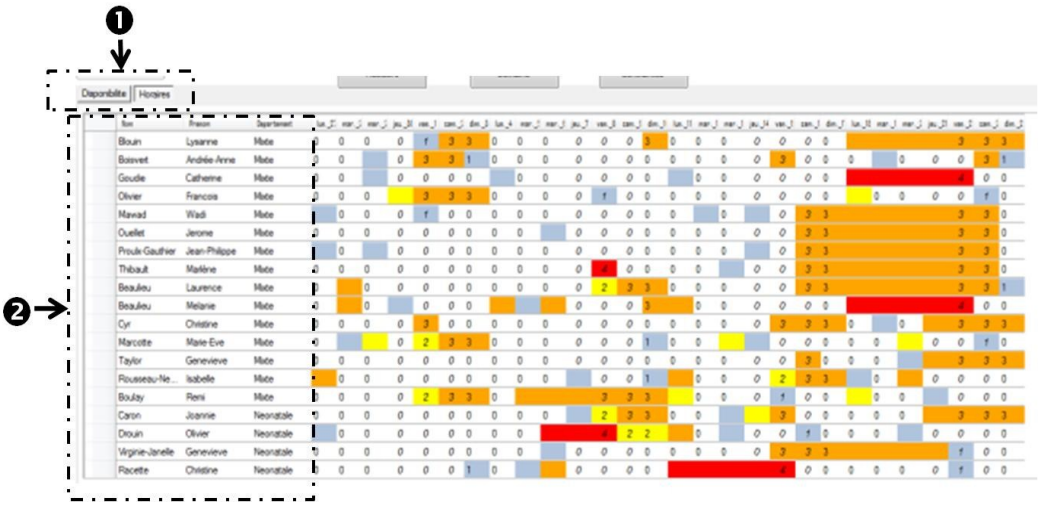


Figure C.I Example of a schedule

The specificity of the *Day editor* is that the shifts are not displayed. The *Shift editor* on the other hand shows the shifts. The schedule is created from the schedule displayed in the *Day editor*.

Figure C.I illustrates an example of a schedule in the *Shift editor* screen. Field ❶ shows the prototype's button used for switching between the *Day editor* and *Shift editor* screens. Field ❷ illustrates the line's that contain the resident's name, surname and category. Within the schedule, each line contains the individual's schedule and each column shows the day of the planning period. The schedule is colour-coded. All worked shifts are indicated in blue-grey, days-off in white, requested days-off in yellow, conference days in orange, and holidays in red.

C.2.1 Day editor screen

Figure C.II shows the *Day editor screen*. Group ❶ contains the start up buttons that allow the user to start working on a calendar. This can be a new (*nouvel horaire* button) or existing calendar (*ouvrir horaire* button). Group ❷ contains the pre-generation edit and optimization model edit buttons, which are *Resident* (❸), *Semaine* (❹), and *Constraints* (❺).The menus that are accessed by the buttons of groups ❶ and ❷ will be described further in this section. Both groups of buttons are available in the *Day editor* and *Shift editor* screens. The group ❻ buttons are pre-generation edit buttons that allow the user to edit the days in the schedule. Finally there are the buttons of group ❼. These buttons are pre-optimisation buttons, referring to the fact that these buttons can only be used before a schedule is generated. The user can generate an initial schedule by clicking on the button *Déterminer horaire (❽).* To save the pre-optimisation schedule the user can save the schedule by clicking on *Sauvegarder (❾)*.

The group ❻ buttons allow the user to edit the colour-codes that were described in Figure C.I. For example, the ■ button allows the user to label a staff member's requested days-off in the schedule. The user can select one or several cells and click on the ■ button. All selected cells will turn yellow, meaning that a staff member requested to be off. The ♦ button is to indicate the availability of staff. The ★ button is for indicating conference-days. Finally the ● button is for indicating holidays.

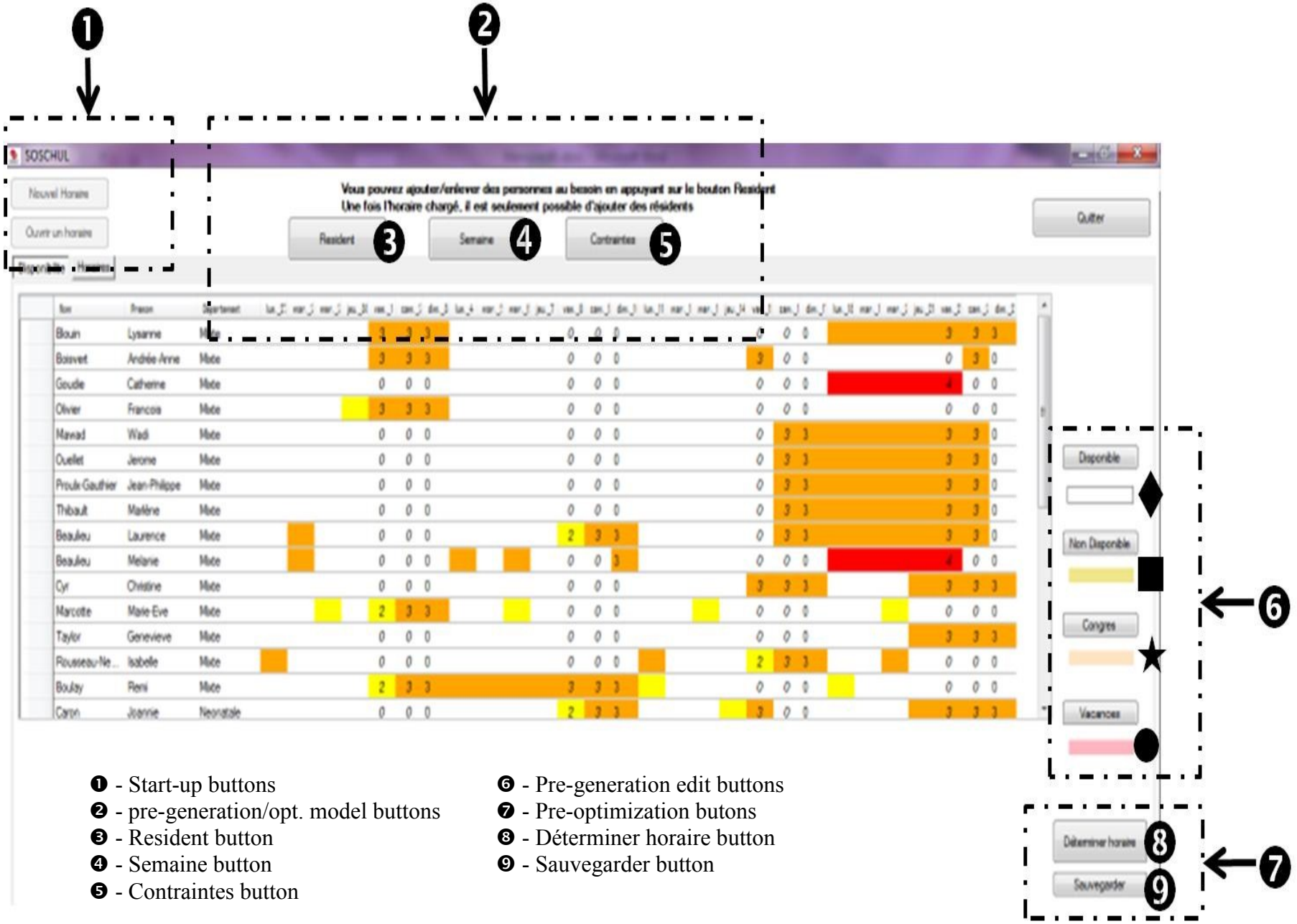


Figure C.II Prototype main window in schedule days editor view

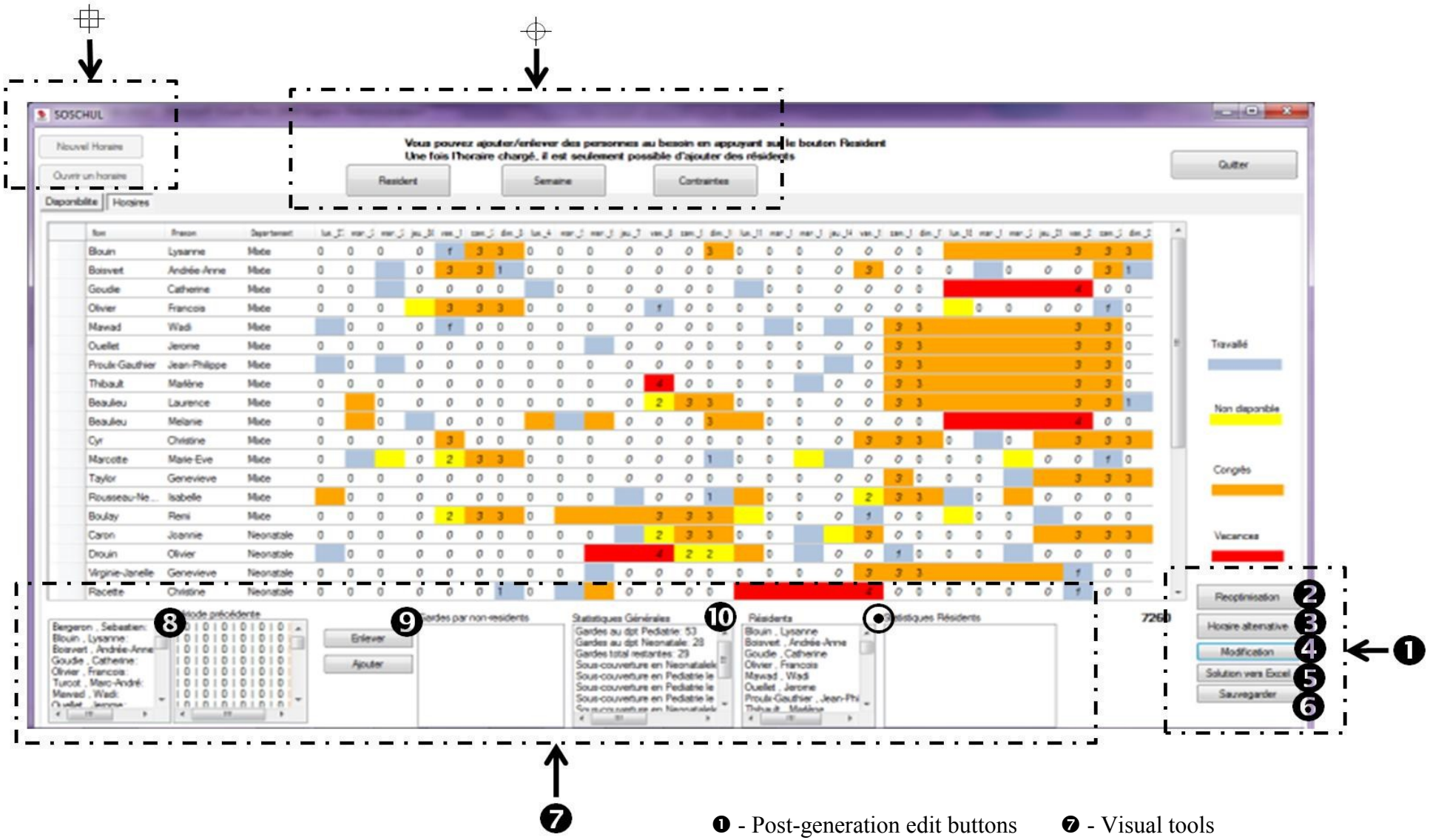


Figure C.III Schedule shift editor view

C.2.1 Shift editor screen

The *Shift editor* screen is shown in Figure C.III. This screen contains the same group of buttons - as the *Days editor* screen. In Figure C.II these are indicated as groups ❶ and ❷, whereas in Figure C.III these are indicated as ⌗ and ⌖ respectively.

Group ❶ contains the post-generation edit buttons. The button *Reoptimisation* (❷) relaunches the optimization process using the on-screen schedule in the *Shift editor* as starting point for the search. The button *Horaire alternative* (❸) relaunches the search to return a schedule of the same relative quality as the on-screen schedule but with an alternative assignment of shifts to residents. The button *Modification* (❹) allows the user to make changes in the assignment of shifts and will have for consequence that the two buttons presented in Figure C.IV appear on-screen. The button *Solution vers Excel* (❺) exports the schedule in the *Shift editor* to an excel file. Finally *Sauvegarder* (❻) saves the schedule from the *Shift editor* screen for future reference.

Group ❼ presents a number of visual tools to provide the user with a quick information overview. The first visual tool is the *previous period tool* (❽) whichs show the individual schedules that residents worked during the last two weekends preceding the planning period. The second tool is the *doctor replacement tool* (❾) that displays all shifts that could not be covered by residents and were assigned to doctors instead. The two buttons to the left of ❾ allow the user to respectively add and remove doctors to/from the schedule. The general information tool (❿) shows general information about the schedule. For example, it provides information about the number of shifts that can still be assigned, the number of shifts assigned per department and the days where overcoverage or undercoverage

Figure C.IV
Schedule shift

occurs. The individual tool (⦿) provides detailed information about each resident. It provides the number of shifts worked by a resident and a breakdown of the number of shifts per weekday.

Button ❹ gives access to the buttons shown in Figure C.IV. The user can add shifts to the schedule with button (Fig C.IV. ❶ - *Travaillé*) selecting the department - from the list in ❸ (Fig. C.IV) - for which he wants to add a shift. He can also remove shifts from the current schedule with button ❷ (Fig. C.IV *Effacer*).

C.3 Submenus

C.2.1 New calendar menu

The first group of buttons that will be discussed are from group ❶ in Figure C.II. By clicking on button *nouvel horaire* the new schedule menu appears. This window is shown in Figure C.V and is used to create a new calendar. The user is asked to provide the start (in ❶) and end (in ❷) date of the scheduling period. He can then accept (❸ - *accepter*) both dates and or *decline* (❹ - *annuler)*. The choice of the user is subject to 2 limitations. First of all, the schedule has to start on a Monday. Next, the schedule has to be at least 7 days long. The implementation of these limitations was necessary because of considerations for the optimization model.

Figure C.V New schedule window

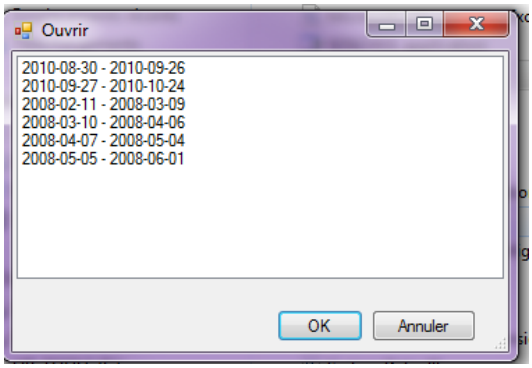


Figure C.VI Open schedule window

C.2.2 Open calendar menu

The button *ouvrir horaire* in the main menu opens up the window shown in Figure C.VI. This window shows all schedules that have been created previously and have been saved by the user. The user can open a schedule by double-clicking on any of the existing schedules or select a schedule and click OK.

C.2.3 Resident menu

Figure C.VII shows the *Active resident editor* window, which is used to edit, add, or delete residents within the prototype. This window is accessed by clicking on the Resident button (Fig C.II ❸). This editor consists of two lists and a submenu. List ❶ (*active resident list*) shows all residents that are available in the prototype. List ❷ (*assigned resident list*) shows all residents that are part of the current schedule and are shown in the schedule. Below the two lists, next to ❸ is the resident description. This provides the first and last name of the resident as well as his seniority, department and category. The button ❸ (*Corriger*) is used to edit the resident's information next to ❸ .

The roll-down menu at ❹ allows the user to add new residents, definitively take them out, or recuperate previously deleted residents of the *active resident list*. By clicking on any of these options in ❹ the *Resident editor* submenu shown in Figure C.VIII appears. The screen

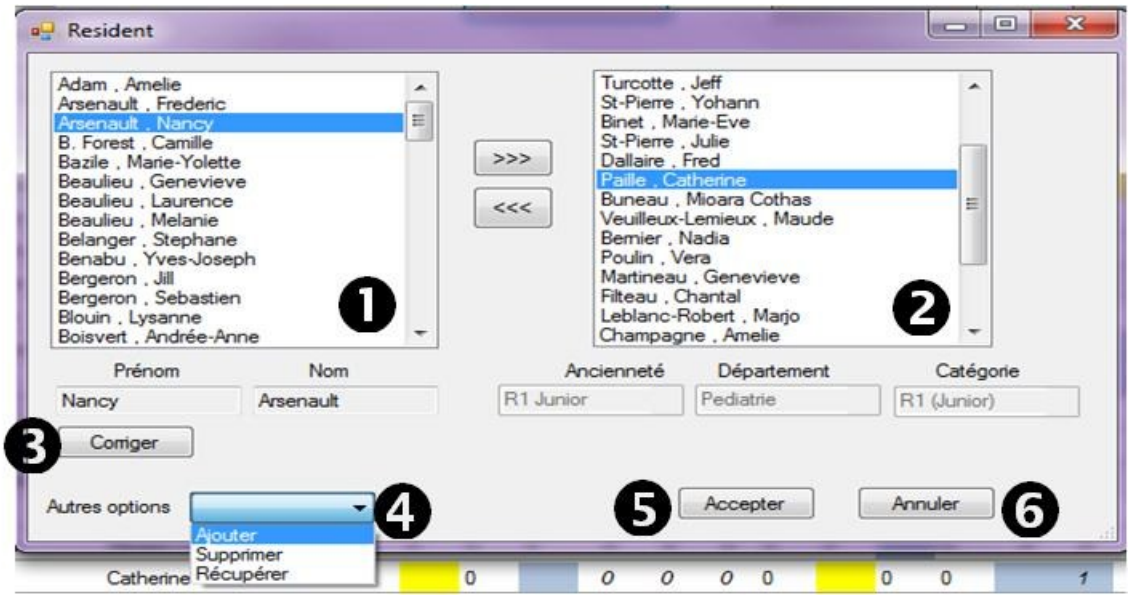


Figure C.VII Active resident editor menu

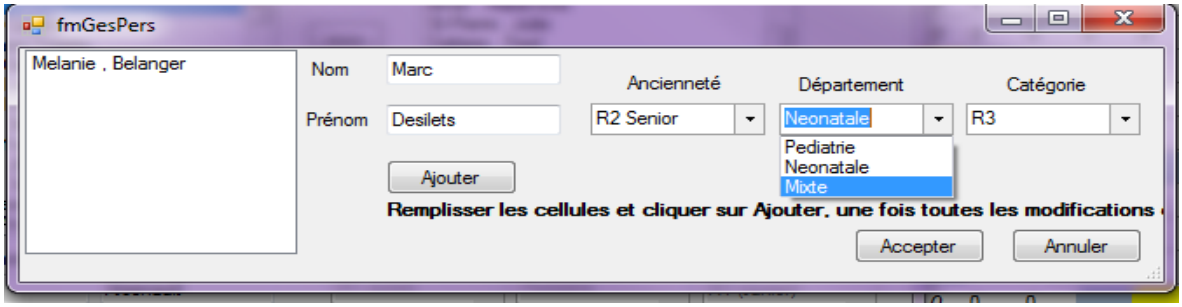


Figure C.VIII Resident editor menu

shows the adding of residents. For each new resident added the user has to indicate the resident's seniority, the department he works on and the category he belongs too. All of this information can be edited in the *Active resident* window.

C.2.4 Week menu

The week menu, illustrated in Figure C.IX shows the settings of the construction algorithm. The user can change the set of values of weekdays under ❶ - *difficulté d'affectation* (assignment difficulty) - or the set of values attributed to the status of residents under ❷ - *paramétrage residents* (resident parameters). The details for the department can be adjusted

in the boxes ❸, and ❺. The user can adjust the department names (❸), choose to include a one or two departments (❹), and their respective coverage levels (❺).

C.2.5 Constraints window

Figure C.X shows the constraints window that contains all various constraints and allows the user to set the search parameters. The set of all legal constraints are shown next to ❶ under *Règles convention collective*. The computer model constraints are illustrated next to ❷ under *Règles pour programme informatique.* These constraints are also considered to be the hard constraints. The set of constraints next to ❸ under *Règles optionnels*, or *Optional constraints*, are optional constraints. The user can set different parameters for the search under *Configuration de résolution* (❹) or the *optimisation configuration* the user can specify the allowed search time, the maximum number of iterations allowed and the number of iterations without improvement

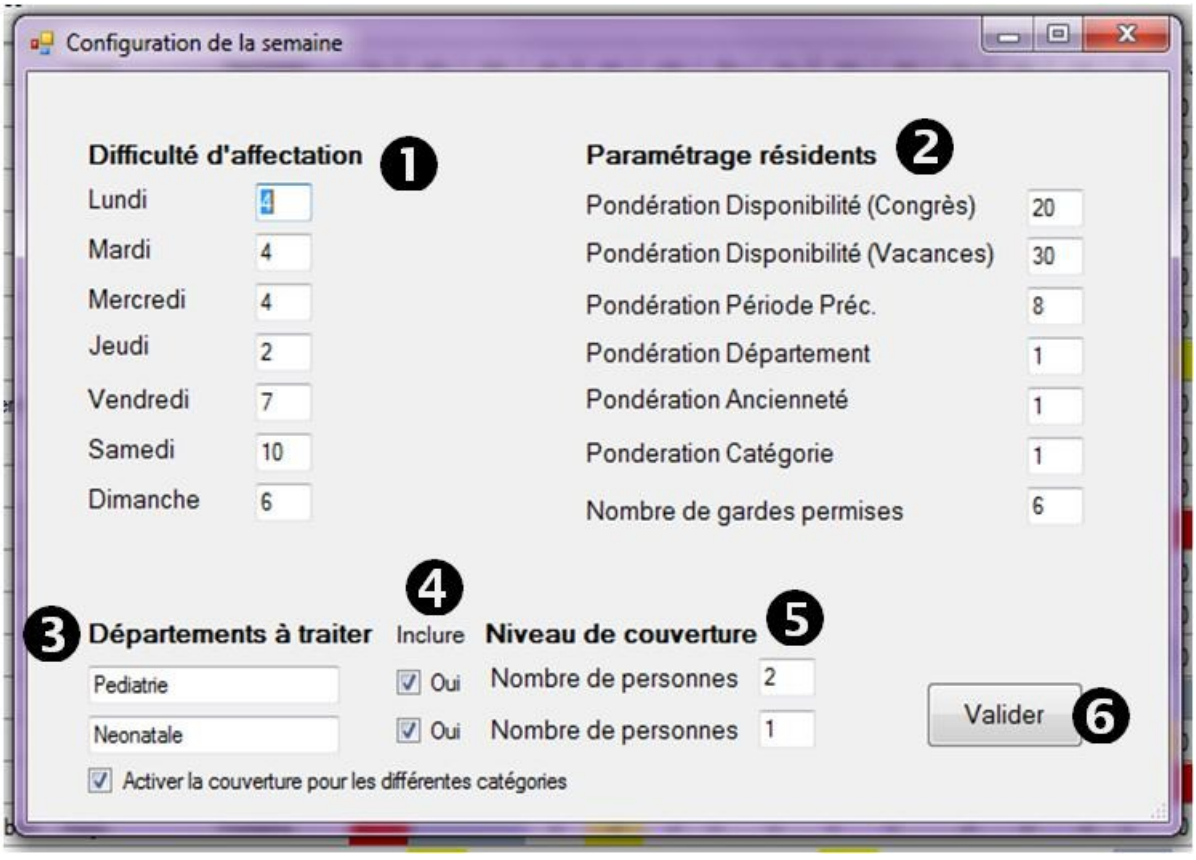


Figure C.IX Week window

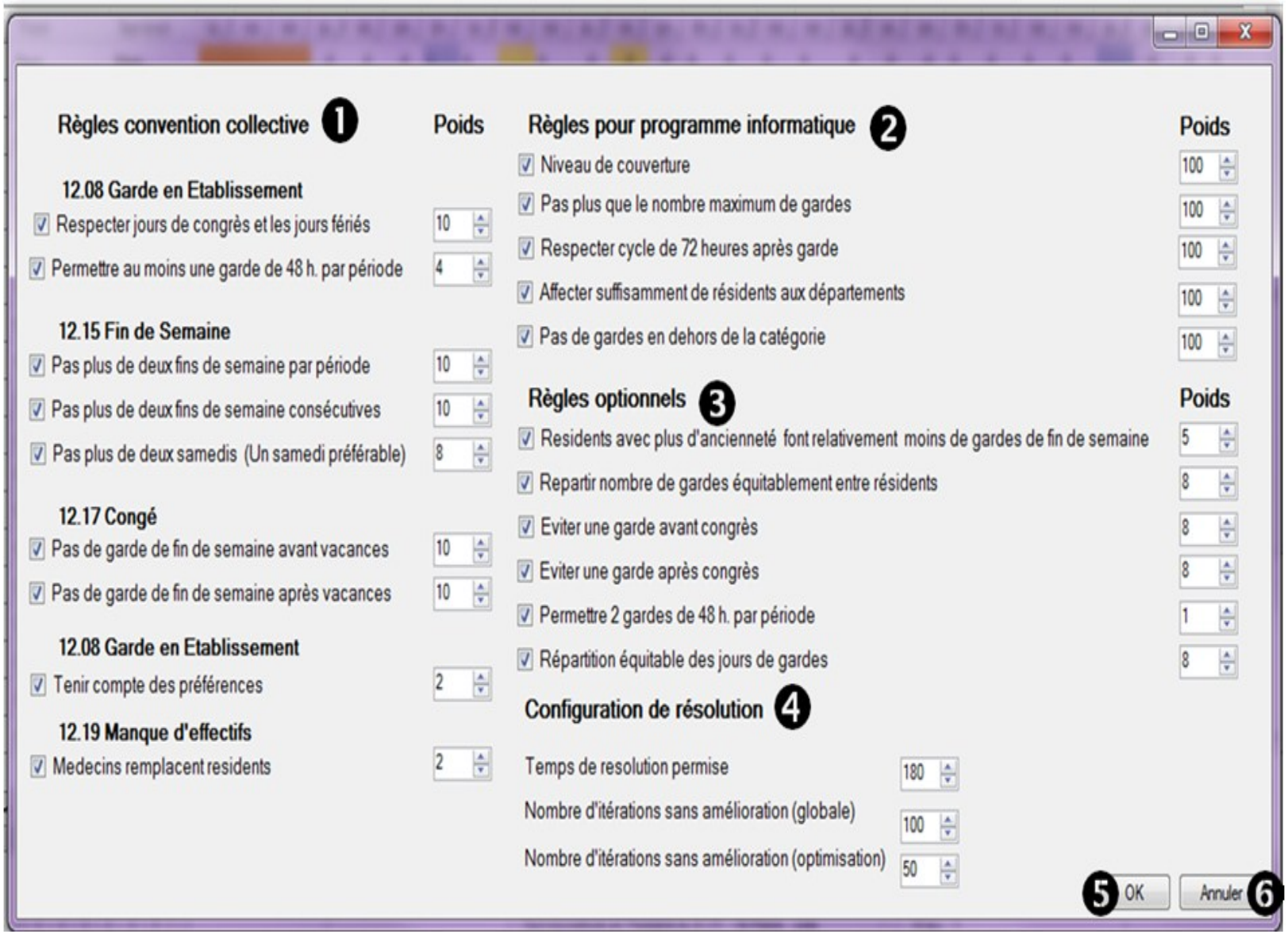


Figure C.X Constraints window