An Advanced Model and Novel Meta-heuristic Solution Methods to Personnel Scheduling in Healthcare

Greet Vanden Berghe

Contents

List of Figures

List of Tables

Glossary

AH_p	the standard performance on absence days for person p
AU	set of assignment units
AU_N	set of assignment units for which the numbering N does not have value U
A_p	number of absence days for person p in the planning period
$CNTR$	number of counters (see Constraint 23)
D	number of days in the planning period
$E_{schedule_p}$	set of events induced by $schedule_p$
M	absolute value of the largest element in a numbering
P	number of personnel members in the ward
PA	serial number of the patterns (Constraint 22)
PIT	set of points in time per day
PS	number of prototype shifts
Q	number of skill categories
QA_p	set of alternative skill categories for p
$QA_{p,d}$	set of alternative skill categories of person p at day d, it equals QA_p when the skill category does not change within the planning period
$RIM_{q,ti,d}$	minimum personnel requirements for qualification q in time interval ti at day d
$RIP_{q,ti,d}$	preferred personnel requirements for qualification q in time interval ti at day d

$RM_{q,s,d}$	minimum personnel requirements for qualification q and shift type s at day d
$RP_{q,s,d}$	preferred personnel requirements for qualification q and shift type s at day d
S	number of different shift types
T	number of assignment units in the planning period
U	represents the assignment units for which a numbering is undefined
W	number of different work regulations
$bank\_holidays_w$	maximum number of assignments on bank holidays for work regulation w (Constraint 20)
$counters_w$	set of counters which are valid in work regulation w (Constraint 23)
$days\_off_{p,d}$	denoting whether person p has a day off on day d (value 1 for day off, otherwise value 0)
$extra\_penalty$	factor to multiply the penalty weight of personal constraints with
$extra\_shift_{p,t}$	denoting whether the personal request for person p at assignment unit t needs a stronger penalty (value 1) or not (value 0)
$max\_day_{w,day}$	maximum number of assignments on the day of the week denoted by *day* for work regulation w (Constraint 10)
max_hours_w	maximum numbers of working hours in a planning period for work regulation w (Constraint 8)
$max\_shift_{w,shift}$	maximum number of assignments for the shift type denoted by *shift* (Constraint 11)
max_w	maximum number of assignments for work regulation w (Constraint 3)
min_hours_w	minimum number of hours to work during the planning period for work regulation w (Constraint 9)
$night$	the set of shift types which start before and end after 00:00
$night\_free_w$	denotes whether the constraint on 2 free days after a night shift (Constraint 14) is valid (value 1 is valid, otherwise value 0)
$night\_weekend_w$	value denoting whether the constraint on night shifts before free weekends (Constraint 16) is valid for work regulation w (value 1 is valid, otherwise value 0)
$not\_together_p$	person who should not work when p is at work (Constraint 28)

pat_w	value of the pattern number which corresponds to work regulation w, 0 denotes that w has no pattern (Constraint 22)
$penalty_{p,C2}$	penalty for shifts which are not performed for the main skill category of person p (Constraint 2)
$penalty_{p,C15}$	penalty for not assigning complete weekends (Constraint 15) for person p
$penalty_{p,C13}$	penalty for forbidden sequences of consecutive shifts (Constraint 13) for person p
$penalty_{p,C18}$	penalty for violating the constraint on the maximum number of consecutive weekends (Constraint 17) for person p
$penalty_{p,C23}$	penalty for violating counters (Constraint 23) for person p
$penalty_{p,C24}$	penalty for violating the personal request on days off (Constraint 24) for person p
$penalty_{p,C20}$	penalty for violating the maximum number of assignments on bank holidays (Constraint 20) for person p
$penalty_{p,C17}$	penalty for not assigning identical shifts during the weekend (Constraint 17) for person p
$penalty_{p,C3}$	penalty for maximum number of assignments (Constraint 3) for person p
$penalty_{p,C4}$	penalty for violating the maximum number of consecutive days (Constraint 4) for person p
$penalty_{p,C6}$	penalty for violating the maximum number of consecutive free days (Constraint 6) for person p
$penalty_{p,C10}$	penalty for violating the maximum number of assignments per day of the week (Constraint 10) for person p
$penalty_{p,C8}$	penalty for overtime (Constraint 8) for person p
$penalty_{p,C12}$	penalty for violating the maximum number of assignments of a shift type per week (Constraint 12) for person p
$penalty_{p,C11}$	penalty for violating the maximum number of assignments for shift types (Constraint 11) for person p
$penalty_{p,C5}$	penalty for violating the minimum number of consecutive assignments (Constraint 5) for person p
$penalty_{p,C7}$	penalty for violating the minimum number of consecutive free days (Constraint 7) for person p
$penalty_{p,C9}$	penalty for undertime (Constraint 9) for person p
$penalty_{p,C1}$	penalty for minimum time between two assignments (Constraint 1) for person p

$penalty_{p,C14}$	penalty for not respecting 2 free days after night shifts (Constraint 14) for person p
$penalty_{p,C16}$	penalty for scheduling a night shift before a free weekend (Constraint 16) for person p
$penalty_{p,C28}$	penalty for assigning someone who should not work together with person p at the same time as p (Constraint 28)
$penalty_{p,C22}$	penalty for violating the patterns (Constraint 22) for person p
$penalty_{p,C26}$	penalty for not respecting requested assignments (Constraint 26) for person p
$penalty_{p,C25}$	penalty for assigning duties on requested shifts off (Constraint 25) for person p
$penalty_{p,C21}$	penalty for assigning forbidden successions of shifts and/or free days (Constraint 21) for person p
$penalty_{p,C27}$	penalty for violating the constraint on tutorship (Constraint 27) for person p
$penalty_{p,C19}$	penalty for violating the maximum number of assigned weekends within periods of 4 weeks (Constraint 19) for person p
$pref$	value added to assignments to show that they are marked
$previous\_hours_p$	balance for working hours for person p, at the end of the previous planning period
$previous\_sat_p$	denotes whether the last Saturday of the previous planning period was planned for person p
$previous\_sun_p$	denotes whether the last Sunday of the previous planning period was planned for person p
$previous_p$	schedule of the previous planning period for person p
$pshift\_duration_s$	duration of prototype shift s
$pshift\_end_s$	end time of prototype shift s
$pshift\_start_s$	start time of prototype shift s
q_p	main skill category of person p
$q_{p,d}$	main skill category of person p at day d, it equals q_p if the skill does not change within the planning period
$requirements$	denotes whether the personnel requirements are formulated as shift types (value *shifts*) or as time intervals (value *floating*)
s_t	shift type corresponding to assignment unit t

$shift\_ps_s$	prototype shift corresponding to shift type s
$shift_{p,s}$	personalised version of shift type s, for person p
$shift\_after_s$	recommended free time after shift type s ends (Constraint 1)
$shift\_before_s$	recommended free time before shift type s starts (Constraint 1)
$shift\_duration_s$	duration of shift type s
$shift\_end_s$	end time of shift type s
$shift\_off_{p,t}$	denotes the off shifts for person p, the value is 1 if p does not want an assignment at assignment unit t, otherwise the value is 0 (Constraint 25)
$shift\_start_s$	start time of shift type s
$succession_w$	two dimensional structure denoting whether shift types and free days can be scheduled immediately before or after others (Constraint 21) for work regulation w
t_s	assignment unit corresponding to shift type s
$threshold\_hours$	threshold value for overtime under which no penalty is generated
$threshold_{cntr}$	threshold value for counter *cntr* which relaxes the evaluation of the constraint on balancing the counter
$tutor_p$	person who should work when p is at work (Constraint 27)
w_p	work regulation for person p
$w_{p,d}$	work regulation of person p at day d, it equals w_p when the work regulation for p does not change within the planning period
$weekend_w$	weekend definition for work regulation w (0: Saturday-Sunday, 1: Friday till Sunday, 2: Friday till Monday and 3: Saturday till Monday)
ANROM	Advanced Nurse ROstering Model: the model developed in this research
$complete\_weekends_w$	value denoting whether the constraint on complete weekends (Constraint 15) is valid for work regulation w (value 1 is valid, otherwise value 0)
$consecutive\_shift_{w,shift}$	set of allowed sequences of consecutive shifts of type *shift* for work regulation w (Constraint 13)
$counter\_actual_{p,cntr}$	value of counter *cntr* for person p

$counter\_balance_{cntr}$	denotes whether the counter has to be balanced among personnel members or not (value 1 for balance, value 0 otherwise)
$counter\_duration_{cntr}$	duration of the evaluation period of counter *cntr*
$counter\_max_{w,cntr}$	maximum value for counter *cntr* in work regulation w
$counter\_previous_{p,cntr}$	the balance for counter *cntr* at the end of the previous planning period for person p
$counter\_start_{cntr}$	start date of counter *cntr*
$counter\_subject_{cntr}$	denoting the subject counter *cntr* counts (Constraint 23)
$counter\_time_{cntr}$	time measure for counter *cntr*
$extra\_requested\_shift_{p,t}$	denotes whether a required assignment for person p at assignment unit t has a higher importance (value 1) or not (value 0)
$identical\_weekend_{w}$	value denoting whether the constraint on identical shift types during weekends (Constraint 17) is valid for work regulation w (value 1 is valid, otherwise value 0)
$max\_cons\_free\_days_{w}$	maximum number of consecutive free days for work regulation w (Constraint 6)
$max\_consecutive\_days_{w}$	maximum number of consecutive working days for work regulation w (Constraint 4)
$max\_consecutive\_weekends_{w}$	maximum number of weekends in which duties are assigned for work regulation w (Constraint 18)
$max\_shift\_week_{w,shift,week}$	maximum number of assignments of shift type *shift* in week *week* for work regulation w (Constraint 12)
$max\_weekends\_4\_weeks_{w}$	maximum number of working weekends in 4 weeks for work regulation w (Constraint 19)
$min\_cons\_free\_days_{w}$	minimum number of consecutive free days for work regulation w (Constraint 7)
$min\_consecutive\_days_{w}$	minimum number of consecutive working days for work regulation w (Constraint 5)
$pattern\_length_{pa}$	length of pattern *pa*
$penalty_{p,C23,balance}$	penalty for violating the balance of counters (Constraint 23) for person p

$penalty_{p,C23,max}$	penalty for violating the maximum value of counters (Constraint 23) for person p
$previous\_free\_after\_night_p$	number of free days after a night shift at the end of the previous planning period for person p
$previous\_bank_p$	number of cumulative assignments on bank holidays for person p
$previous\_consecutive\_days_p$	consecutive scheduled days at the end of the previous planning period
$previous\_consecutive\_shifts_{p,s}$	sequence of consecutive shifts of type s for person p at the end of the previous planning period
$previous\_consecutive\_weekends_p$	number of consecutive scheduled weekends at the end of the previous planning period for person p
$previous\_weekends\_1_p$	number of working weekends in the last week of the previous planning period for person p
$previous\_weekends\_2_p$	number of working weekends in the last 2 weeks of the previous planning period for person p
$previous\_weekends\_3_p$	number of working weekends in the last 3 weeks of the previous planning period for person p
$requested\_assignment_{p,t}$	denotes whether person p has a personal request for an assignment (Constraint 26) at assignment unit t (value 1) or not (value 0)
$start\_pattern_p$	week in which the pattern corresponding to person p's work regulation starts
$sum\_counter\_actual_{cntr}$	the sum of the value of counter *cntr* for all the people belonging to a work regulation which includes the counter

Abstract

Constructing timetables of work for personnel in healthcare institutions is known to be a highly constrained and difficult problem to solve. In this thesis, we introduce a model for the practical rostering problem in Belgian hospitals. It is general enough to cope with the large set of constraints and to meet varying objectives encountered in practice.

We set up a solution framework that consists of a modifiable and explanatory evaluation function, many options for handling initialisation parameters and for formulating various objectives, and meta-heuristics to search solutions. A set of neighbourhoods was designed for organising an effective exploration of the search space. We combined different local search heuristics with these neighbourhoods and managed to find scenarios that produce algorithms for widely varying problem settings. The hybrid tabu search approach deserves special attention because it is applied in practice, as part of a software package based on the model proposed in this thesis. A range of new memetic approaches for rostering is introduced. They use local search improvement heuristics within a genetic framework. We identify the best evolutionary operators of a memetic algorithm for the rostering problem, particularly the nature of effective recombination, and show that these memetic approaches can handle initialisation parameters and a range of instances more effectively than usual tabu search algorithms, at the expense of longer computation times.
Having presented cost function based search heuristics, we finally introduce a new multi criteria approach which overcomes some practical difficulties for automated nurse rostering. The developed multi criteria approach, incorporated in a meta-heuristic, takes into account the fact that some constraints are easier to satisfy than others while allowing schedulers to control compensation of constraints.

By automating the nurse rostering problem, the scheduling effort and time are reduced considerably in comparison with the manual approach that was previously used. The software based on the model developed in this thesis provides an unbiased way of generating the schedules for all the personnel members. It enables simple verification of the constraints, helps redefining unrealistic hard constraints, and thus leads to an overall higher satisfaction of the personnel.

Part I

The Nurse Rostering Problem

Chapter 1

Introduction

The pressure of work in healthcare organisations increases persistently and continues to remain a serious problem in spite of recent significant technological advances. Personnel rosters in hospitals are more than resources to cover the patients' requirements. They affect the organisational structure of the hospital and they directly influence the private life of the personnel members. Unpredictable and strongly fluctuating timetables ruin the atmosphere in the wards and give rise to a high level of absenteeism. To prevent these problems, hospital planners should be able to create efficient timetables and preferably a long time in advance. It is therefore important to provide an interactive system that generates high quality scheduling solutions within a reasonable computing time. Such schedules should cover the hospital requirements while avoiding patterns that are harmful for the nurses' well being. Many unhealthy and unwanted shift patterns were determined, and many possibilities exist for nurses to express their desires. A flexible planning system should incorporate as much knowledge as possible to relieve the personnel manager or head nurse from the unrewarding task of setting up objective schedules which please all the individual nurses.

In order to meet strict quality standards in patient care, the objective of personnel rostering in healthcare is to match the number of skilled people working at given time intervals to the demand for certain nurse services. Timetables are constrained by governmental and hospital rules but also by personal preferences and work regulations. Hospital personnel rostering is a very complex scheduling domain, in which services are provided continuously, i.e. 24 hours per day and 7 days per week.

Real-world nurse rostering problems are difficult combinatorial problems that belong to the domain of scheduling and timetabling. More specifically they are classified as personnel scheduling problems. Obtaining an optimal solution is an impracticable task to undertake due to the large number of alternative schedules. Moreover, optimal solutions in this case may not be suitable since a compromise need to be reached. Therefore, approximate solutions are satisfactory, they can be obtained quickly. Several meta-heuristics have successfully been applied to similar problems. Related problems are security and law enforcement personnel scheduling, bus driver scheduling, crew scheduling, employee scheduling in production environments, etc. In that category, many of the personnel scheduling problems can be solved with cyclical schedules, and thus are less complex than hospital rostering. Compared to most industrial situations (where personnel schedules normally consist of stable periodic morning-day-night cycles) healthcare institutions require more flexibility in terms of hours and shift types, they have a larger set of different skill categories, and more possibilities for defining part time contracts.

Nurse scheduling can be split into sub-problems according to the time horizon in which decisions are made. On long-term strategic level, hospital personnel scheduling considers determining the service level to meet patients' needs, in addition to forecasting large fluctuations. The next phase involves hiring qualified personnel to fulfil the tasks, already considering possible understaffed periods by outlining a policy of exchange of personnel among wards or by hiring 'float' nurses. This problem domain is called 'staffing'. The research of this thesis concentrates mainly on the short-term problem of assigning specific tasks to qualified nurses at every moment within a planning period, referred to as 'rostering' or 'timetabling'. A part of the problem data, such as the number of personnel in a ward, the required qualifications, the definition of shift types, etc are determined at the strategic level. Although these settings are not considered as part of the nurse rostering problem, some decisions on a longer term can affect the solution strategies. The model developed in this thesis therefore provides several possibilities for flexible problem setting. Examples are: shift types can be divided over several nurses, personnel demands can be expressed in terms of shorter intervals than shift length, night shifts can be assigned to a special category of night nurses, possibilities exist for creating part time work, some people can temporarily be assigned to different wards in order to set off emergencies, personnel members can express certain preferences on particular times in the planning period, etc.

The problem of finding a high quality solution for the personnel timetabling problem in a hospital ward has been addressed by many personnel managers and schedulers over a number of years. In recent years, the emergence of larger and more constrained problems has presented a real challenge for researchers. However, finding good quality solutions can lead to a higher level of personnel satisfaction and a very flexible organisation.

The framework designed in this thesis addresses the very complex nurse rostering problem that is typical for Belgian hospitals. Although there is software available dealing with personnel rostering in hospitals, most packages have serious disadvantages in that they either do not support automated scheduling or that they are simply developed for solving very specific local

problems. After an extensive market research effort carried out by GET[1] in 1993, none of the existing packages turned out to be satisfactory for general applicability in hospitals. Especially for the Belgian situation, in which nurses are highly demanding for quality time and for contracts adapted to their private life, the available software proves to be unsuited. Unlike hospitals in the United States, Japan, and some European countries, for which most of the available software has been developed, the flexibility with respect to part time work is in Belgium remarkably large. It is obvious that hospital planners need to encompass these extra opportunities for flexible work time. Furthermore, the situation in Belgian healthcare institutes includes particular possibilities for letting nurses express their personal preferences for shifts and days off, for weekend work, and for certain cyclical patterns to be superposed on variable schedules. It is therefore clear that a program, specifically including all the features of Belgian nurse rostering, is required. The results of the market research corroborate this requirement, Belgian hospital planners indeed desire an easily modifiable software package to facilitate personnel rostering.
Shortly after the market research, GET set up a collaboration with the software developer Impakt N.V.[2], which resulted in the product called *Plane*. Impakt was responsible for the user interface and the database, while all the modelling and scheduling was done within the framework of this thesis. The chapters represent each of the different steps in developing the system: the modelling of the nurse rostering problem (Chapter 2), the coding of the constraints (Chapter 4), the setting up of a solution framework (PART II), and the development of appropriate search techniques (PART III). The model has been developed as a flexible and interactive system, which can cope with different kinds of strategic or staffing inputs. Beta versions were released and as the number of users grew, we were presented with more complex problems. Extended functionalities, new planning procedures, extra constraints, etc were developed in order to meet all the planners' needs (Chapter 5 and 6). The system is now flexible enough to be applied in other personnel scheduling domains.

In Fig. 1.1, the relation between the four parts of this thesis is presented schematically. Apart from the tabu search algorithm, which is implemented in the commercial software, we have carried out experiments with other meta-heuristics and with hybrid approaches in an attempt to meet the requirements better.
A very detailed model for the particular kind of personnel rostering problems tackled in this research is introduced in Chapter 2. We elaborate on the constraints and requirements of healthcare personnel in Belgian institutions and incorporate also functionalities offered by Plane. In Chapter 3, we review the available literature on hospital planning. Although similar problems have been addressed by other researchers, it will become clear that none of the existing approaches were applicable to the current Belgian situation. Therefore, the

[1] GET, General Engineering & Technologie, Antwerpse Steenweg 107, B-2390 Oostmalle
[2] Impakt N.V., Dendermondsesteenweg 42, B-9000 Gent

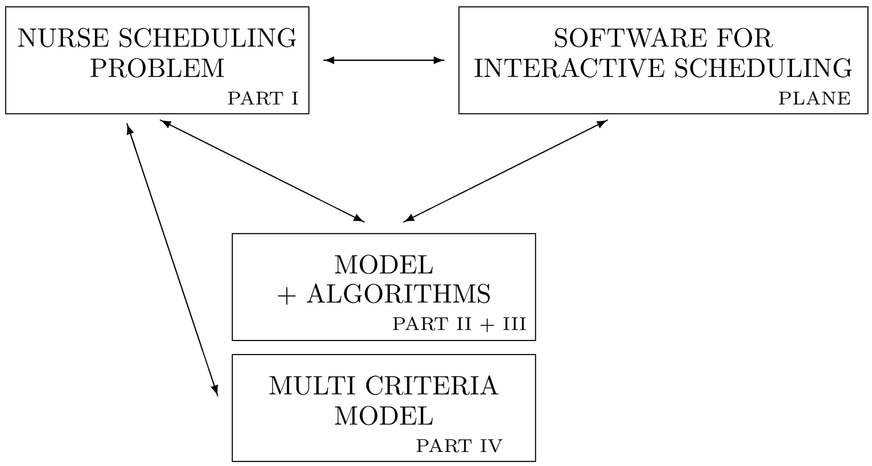

Figure 1.1: Schematic overview of the nurse rostering system

models and the algorithms in this thesis had to be developed specifically for this purpose.
Chapter 4 introduces the evaluation function which is used throughout the algorithms of PART II and III. It enables tackling the large set of constraints in a flexible, modular way while providing feedback for the users. The specific objectives detected in several hospital wards form the basis for particular heuristics to solve optional planning procedures discussed in Chapter 5. These pre-planning and post-planning algorithms keep the real search algorithms separated from particular problem requirements. In Chapter 6, we introduce a new model for an unusual way of formulating the personnel requirements. These 'floating' requirements enable hospital planners to extend the problem formulation and to increase the number of alternative solutions. Thanks to a higher degree of freedom, the time interval based model makes it possible for hospitals to create all kinds of part time work and to obtain better quality solutions.
PART III concentrates on different meta-heuristics for solving combinatorial problems. In Chapter 7, we introduce a set of neighbourhoods for the scheduling problem. Combining the neighbourhoods with steepest descent algorithms and with a tabu search approach leads to a modifiable variable neighbourhood search method. All the developed neighbourhoods will be further applied in the meta-heuristics of Chapter 8-10. The algorithms described in Chapter 8 are hybridisations of a tabu search algorithm that have been developed especially for solving nurse rostering problems in practice. The overall results outperform those obtained by single algorithms. Chapter 9 compares other meta-heuristic approaches to the same problem. The results are better than the previously

obtained results, at the expense of the computation time.

In Part IV, we investigate the applicability of a multi criteria approach for solving the nurse rostering problem. The possibility of assigning weights to certain criteria or conditions guides the search through a different set of solutions and produces interesting results of a very good quality.

We compare the results of the developed algorithms, summarise their benefits and drawbacks and end with a general discussion in Chapter 11.

Chapter 2

Problem Description

Nurse rostering is a complex scheduling problem that daily affects hospital personnel all over the world. The need for quality software solutions is high for a number of reasons. In particular, it is very important to flexibly model real-world constraints, to attempt to satisfy wishes and preferences of personnel members, and to evenly balance the workload among people. A high quality roster can lead to a more contented and thus more effective workforce. Compared to many industrial situations (where personnel schedules normally consist of stable periodic morning-day-night cycles) healthcare institutions often require more flexibility in terms of hours and shift types. The motivation for this research has been provided by hospital administrators/schedulers in Belgian hospitals. Planners who used the early releases of the nurse rostering software *Plane* (Section 2.1) based upon the model and solution framework of this thesis, suggested improvements and extensions. We will refer to the problem description built in this work as ANROM, which stands for Advanced Nurse ROstering Model. The dimensions of the problem are explained in Section 2.2. We present the hard and the soft constraints of ANROM and describe them formally in Section 2.3 and 2.4.

2.1 Plane, nurse rostering system for Belgian hospitals

Plane is a commercial software system developed in a collaboration between Impakt N.V. and GET for assistance in hospital employee scheduling. The author contributed to that rostering program by modelling the problem and the evaluation function, and by developing a solution framework. Plane arose from the need for automated rostering assistance in Belgian healthcare organisations. In such a rostering problem, personnel requirements for every skill category have to be fulfilled over a fixed period in time, while respecting a number of constraints that limit the personal assignments.
The initial version of Plane was first implemented in a hospital in 1995 but the

system is still evolving to cope with the new and more complex real-world problems that keep appearing. So far, over 40 hospitals in Belgium, of which some have about 100 wards, have replaced their time consuming manual scheduling by this system.
Although the problem is user-defined to a large extent, the software has to be efficient in different settings. Every specific hospital ward should be able to formulate its problem within the restrictions of the model described in the following sections. The cost function used in the algorithms is modular and can deal with all the constraints matching the types described in Section 2.4.

2.2 Dimensions

The main goal of our personnel rostering model is to create a schedule by assigning shift types to skilled personnel members, in order to meet the requirements in a certain planning period. Personnel have work regulations limiting their assignments. The concepts of the problem will be explained in detail in this section.

2.2.1 Personnel

Hospitals are organised in wards with fixed activities, usually a settled location, and, for the most part, a permanent team of nurses. Although practical situations often allow people to migrate to another ward whenever a personnel shortage is unsolvable, the personnel rostering problem in this research normally concerns a group of personnel belonging to the same ward. Planning the different hospital wards reduces the complexity of the problem considerably. The algorithms for scheduling single wards should not exclude people from working temporarily elsewhere (see Section 5.3).
In the personnel rostering model of this research, the number of personnel members is user definable. It is not the result of calculations within the planning algorithm. Staffing considerations and decisions on the capacity in terms of beds and patients, are beyond the competence of the short-term timetablers in Belgian hospitals. The number of people P in a ward varies in practice from about 20 to over 100. In this research, the nurse rostering problem concentrates on scheduling the personnel members of a single ward, unless explicitly stated differently.

2.2.2 Skill categories

Personnel members in a ward belong to skill categories. The division into categories is based upon the particular level of qualification, responsibility, job description, and experience of the personnel. Typical categories in hospitals are: *head nurse*, *regular nurse*, *junior nurse*, *ambulance driver*, *caretaker*, *cleaner*, etc. The groups are called 'grades' in some other applications. In our model, there are Q different skill categories. Each personnel member p ($1 \leq p \leq P$)

belongs to one skill category q_p.
The approach in ANROM allows for a personalised organisation of substitution among skill categories. Rather than strictly disjunctive skill categories or hierarchical substitutability (in which higher skilled people can replace less experienced colleagues), we opted for a solution that is closer to the reality in hospitals. For example, a particular skill category might be a class of junior nurses. It might be the case that we could allocate someone in the ward manager's category to a junior shift on a given day. In practice, very senior staff are usually reluctant to stand in for junior staff. It is also the case that, in practice, a regular (not a junior) nurse will temporarily stand in for a head nurse.
The problem of replacing people is solved in ANROM by assigning *alternative* skill categories to certain people. People with more experience or who have taken some exams, can be substitutes for higher skill categories. The nurses who might replace a head nurse, for example, normally are a couple of senior nurses who know everything about the working of the ward. We denote by QA_p the list of alternative skill categories for person p. The number of elements in the list is 0 when the person p has no permission to replace people from other skill categories than q_p and QA_p contains $Q-1$ elements if p can do the work of any other skill category in the ward. A refinement of q_p and QA_p is necessary when a personnel member p does not have the same skill category during the entire planning period. That is explained in Section 4.2.5.

2.2.3 Work regulations

Cyclical schedules obey very strict patterns. They are applied in many personnel rostering environments but are very impractical in real healthcare environments. In our model, hospital personnel have work regulations or contracts with their employer. Most organisations allow several job descriptions such as part time work, night nurses' contracts, weekend workers, etc. These regulations involve different constraints but they can make the schedules much more flexible. Moreover, very personal arrangements like 'free Wednesday afternoons' or 'refresher courses' at regular points in time, etc can easily be formulated. It is not unlikely to have personalised contracts for the majority of personnel members in Belgian hospitals.
The different work regulations are denoted by w $(1 \leq w \leq W)$ in this research. For every personnel member p, the work regulation is denoted by w_p. People can change their contract during the planning period, which, of course, has an influence on the schedule (see Section 4.2.5).

2.2.4 Shift types

A shift type is a predefined period with a fixed start and end time in which personnel members can be on or off duty. Many continuously working organisations schedule three typical shift types called *morning*, *late* and *night shift*. This is the way that manufacturing generally works. However, these working hours cannot cover the personnel requirements of hospitals in practice. More-

		Start	End
M	morning shift	06:45	14:45
L	late shift	14:30	22:00
N	night shift	22:00	07:00

Table 2.1: Example of shift types

over, all the possible kinds of part time work require a variety in start and end times and in shift length. There are S shift types per ward. Each shift type s has a start time $shift\_start_s$, an end time $shift\_end_s$, and a duration $shift\_duration_s$. Table 2.1 presents a simplified example of a set of shift types in a ward. A shift type does not always involve continuous activity between the start and the end time, and hence $shift\_duration_s$ is not necessarily equal to $shift\_end_s - shift\_start_s$. Examples of such shift types are those in which a long break enables personnel to have lunch at home, or guard duties, which require availability from the personnel during a longer period than the actual time which is counted as working time.

It is common to allow hospital schedulers to define shift types according to their needs. ANROM works with prototype shifts, which are defined for the entire hospital organisation. There are PS prototype shifts, each of which have a start time $pshift\_start_s$, end time $pshift\_end_s$, and a duration $pshift\_duration_s$.

The schedulers in a ward copy these prototypes and modify them locally in order to match their activities. A shift type s corresponds to the prototype shift $shift\_ps_s$. One of the major advantages of the general definition of prototype shifts is that personnel members working in different wards can still evaluate their schedule with the locally applicable constraints (see constraints on shift types in Section 2.4.3).

Apart from locally redefining prototype shifts in a ward, exceptional shift characteristics can be permitted for particular nurses. Examples are shifted start and end times, extended breaks, etc which often solve practical private problems. Personalised shift types are denoted by $shift_{p,s}$. In the formal description of the ANROM model, the notation is not used when all the people in a ward work the same set of shifts, which is the most common case. The time related constraints are formulated in terms of the regular shift types (provided the personalised shift types do not differ too much from them). In the evaluation procedure, the regular shift types which hold for the entire ward can replace the personalised shift types.

2.2.5 Planning period

Planning periods for nurse rostering vary from a couple of days to a few months. The length of the period is expressed as a number of days D, or a number of assignment units T (explained in Section 2.2.6). Since cyclical rosters are not common at all, it is important for individual employees to know their schedule

some time in advance. Long term scheduling, on the other hand, should not be too detailed because the personnel demand and personal preferences fluctuate and are not predictable but for the near future. In this model, we always consider planning periods which start on Monday and end on Sunday, no matter what the duration (number of weeks) is.
Short planning periods enable the search algorithms to find good quality results much faster than longer planning periods. However, very short planning periods reduce the possibility of guaranteeing fairness among personnel members. Some situations require shorter planning periods. Examples are unexpected changes in the requirements or the constraints. In such cases, the personnel manager tends to prefer the fewest modifications possible in the people's schedules.
ANROM provides some planning procedures for organising rescheduling processes. Parts of the already existing roster can be 'frozen' during the planning. Both personal schedules and periods in time (for all the members of the ward) can be kept unchanged while the algorithms search for a better solution in the remaining part of the problem domain. Freezing parts of a schedule is explained in more detail in Section 5.3.

2.2.6 Schedule

The roster of the ward, in which the shift assignments to people are stored, is called the *schedule*. It has dimensions P and T, where T stands for the number of *assignment units*. We will refer to a personal schedule for person p by the notation $schedule_p$ and to a particular assignment at unit t by the notation $schedule_{p,t}$.
We define assignment units as entities of minimum allocation in a schedule. They are mainly introduced to handle the soft constraints on the personnel's schedules. For the evaluation model (described in Chapter 4), assignment units play a particularly important role. In the approach of ANROM, where personnel requirements and schedules make use of shift types, each shift type on each day has a corresponding assignment unit. The total number of different assignment units T for a schedule is therefore equal to $S * D$. An assignment unit t_s $(1 \leq s \leq S)$ corresponds to shift type s. For the clarity of notations, s_t is introduced as the shift type that corresponds to the assignment unit t.
We illustrate the meaning of assignment units with a simple example. A fragment of a possible personnel roster is presented in Fig. 2.1. We notice that there are five people in the ward, and that there are three different shift types. Fig. 2.2 presents the *schedule* that corresponds to the roster of Fig. 2.1. Each column in the schedule represents an assignment unit. For every day of the planning period there are S columns, each of them corresponding to a shift type. The assignment units are ordered according to the start times of the shift types they represent. When two shift types have the same start time, the first assignment unit will match the shift type with the earliest end time. The assignment units defined for this approach do not represent consecutive or separate periods but they will very often overlap in time. For the nurse rostering problem considered in this thesis, the number of assignment units equals the number of shift types

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
P1	M	M	L	L	N		
P2	N		N	L	L		
P3	M	M	M	M	M	M	M
P4	M		L	N	N	N	
P5	M L	L	L	L			

Figure 2.1: Roster example for 5 people (P1,..., P5) and 1 week; M, L, and N being the shift types introduced in Table 2.1

	Schedule Example																				
P1	*	_	_	*	_	_	_	*	_	_	*	_	_	_	*	_	_	_	_	_	_
P2	_	_	*	_	_	_	_	_	*	_	*	_	_	*	_	_	_	_	_	_	_
P3	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_
P4	*	_	_	_	_	_	_	*	_	_	_	*	_	_	*	_	_	*	_	_	_
P5	*	*	_	_	*	_	_	*	_	_	*	_	_	_	_	_	_	_	_	_	_

Figure 2.2: Schedule corresponding to the roster in Fig. 2.1: '*' denotes that there is an assignment in the schedule, '-' denotes that the schedule is free, * will be specified in Section 2.3

times the number of days in the planning period ($T = S * D$, where D denotes the number of days in the planning period).

2.3 Hard Constraints

Hard constraints are those that must always be satisfied. We can cover most real-world hospital situations with the hard constraints of the following set.

2.3.1 Maximum one assignment per shift type per day

In ANROM, it is not considered feasible to assign the same shift to a member of the ward more than once per day. A violation of this constraint would involve twice the workload for that person. The constraint can be represented by (2.1), in which *pref* is a number indicating that assignments have been made in a post-planning coverage procedure. A value 0 at a schedule position $schedule_{p,t}$ indicates that there is no assignment for person p at assignment unit t. Any value different from 0 denotes the skill category for which an assignment is made. It is only in exceptional cases that a person will have an assignment for another than his or her main skill category. The values $pref + 1$ to $pref + Q$ represent the skill categories 1 to Q but the added number *pref* indicates how the assignments were made. The importance of this expansion of the model will be explained in Section 5.6.

$$\forall p, (1 \leq p \leq P);\ \forall t, (1 \leq t \leq T):$$
$$schedule_{p,t} \in \{0, 1, \ldots, Q, pref + 1, \ldots, pref + Q\} \tag{2.1}$$

Within a ward, the schedule representation of ANROM thus prevents personnel members from having more than one assignment per shift and per day, be it for their main or for an alternative skill category. It is not the case that this constraint prevents people from being assigned to overlapping shifts, a condition that will be handled by a soft constraint in Section 2.4.

2.3.2 Required skill

In case the number of people required for a skill category is higher than the available number, those who have the skill as an alternative grade can assist (see also Constraint 2 in Section 2.4). It is infeasible to assign tasks to people who are not qualified to carry them out. This is formally presented in (2.2), in which the values that can occur in a personal schedule are restricted to the following set.

$$\forall p, (1 \leq p \leq P);\ \forall t, (1 \leq t \leq T):$$
$$schedule_{p,t} \in \{0, q_p, q_p + pref\} \cup QA_p \cup QA_p + pref \tag{2.2}$$
$$with\ \ A + a = \{x\ \xi\ (x - a) \in A\}$$

In the initial phase of the scheduling algorithm, violations of soft constraints are not taken into account. This is simply done by planning all the required duties at random in the schedule, while ignoring the soft constraints such as limitations on the number of shifts each member of staff may work. However, it is not unlikely that situations occur in which this particular hard constraint cannot be satisfied. A few consistency checks can assist the planner to overcome this problem (see Section 5.2).
We modelled the problem and developed the algorithms in such a way that the hard constraints will never be violated during the search procedures, no matter how bad the evaluation of the current solution is. Once an initial feasible solution has been found, no new assignments can enter the schedule nor can assignments be removed from it (except for the **SWT** heuristic that will be defined in Section 9.4). In the ANROM model, we opted for not shifting assignments from one assignment unit to another. The number of assignments per shift type and per skill category on a particular day remains constant.

2.3.3 Personnel requirements

Personnel requirements, also called ‘coverage constraints’, express the number of personnel of each skill category needed to staff the ward. They are set by management and are usually expressed in terms of the minimum number of personnel required and the preferred number of personnel to meet patients' needs. The minimum number of personnel strictly covers all the unavoidable tasks and the preferred number of personnel ameliorates the atmosphere by reducing the workload of staff members. The requirements can be formulated either in terms of shift types (which is the traditional approach also used in literature) or in terms of begin and end times (called floating requirements). The term *requirements* denotes shift type requirements when it equals *shifts*, and floating requirements when it equals *floating*.

Shift type requirements

$RM_{q,s,d}$ denotes the minimum required number of nurses of skill category q, for shift s at day d. $RP_{q,s,d}$ denotes the preferred number of personnel. A simple example of shift type personnel requirements for three different skill categories (head nurse, regular nurse, and nurse aid), and for one day, is presented in Table 2.2. In this example, 6 different shift types are defined in the ward. Each row in the table corresponds to one of the shift types. Per skill category, there are two columns in the table. They correspond to the minimum and the preferred requirements.

Requirements	Head Nurse		Regular Nurse		Nurse Aid	
Shift s	$RM_{h,s,d}$	$RP_{h,s,d}$	$RM_{r,s,d}$	$RP_{r,s,d}$	$RM_{a,s,d}$	$RP_{a,s,d}$
Short Early			1	2		
Early			1	2		
Day	1	1	0	1	2	2
Late			1	2		
Short Late			1	2		
Night			1	2		

Table 2.2: Example of personnel requirements on day d formulated with shift types

To satisfy the personnel requirements for the head nurse, for example, it is sufficient to have one head nurse available for the day shift. Alternatively, if no head nurse is free, only those people who are (with respect to their alternative skill category) authorised to replace the head nurse can be assigned to the required shift. Creating a feasible schedule is generally fairly easy, provided the personnel requirements do not exceed the total number of skilled personnel for any shift and day. When there are not enough skilled people in the ward, there exists no feasible schedule. In that case, either the personnel requirements have to be relaxed on certain days, float nurses will temporarily assist, or extra personnel needs to be hired.
The shift type personnel requirements are formulated as shown in (2.3).

$$
\begin{aligned}
&\forall s, (1 \le s \le S);\ \forall d, (1 \le d \le D) : \\
&\quad t = (d-1) * S + t_s \\
&\quad RM_{q,s,d} \le \\
&\qquad |\{p \ \xi\ 1 \le p \le P \wedge (schedule_{p,t} = q \vee schedule_{p,t} = pref + q)\}| \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \le RP_{q,s,d}
\end{aligned}
\tag{2.3}
$$

Floating requirements

It is an entrenched habit (within some hospitals) to think in terms of the number of personnel from hour to hour. Manual planners tend to not always defining their personnel needs as a combination of shift types. We broadened the framework for defining the daily staff complement since the formulation used by hospital planners often allows for a higher flexibility in constructing the timetables. Personnel requirements in terms of time intervals are expressed as $RIM_{q,ti,d}$ for minimum and as $RIP_{q,ti,d}$ for preferred personnel requirements. $interval\_start_{ti}$ and $interval\_end_{ti}$, denote the start and end times of the time interval ti ($1 \le ti \le TI$). We introduce a set PIT for all the different points in time during a day. This is formally presented in (2.4). If we deduce the personnel requirements from the shift types it is possible to allocate several kinds of part-time employment over the shift period. Table 2.3 presents a simplified example of the personnel requirements for a single day expressed as floating requirements. The details of this representation are explained in Chapter 6.

$$
\begin{aligned}
PIT = \quad &\{interval\_start_1, interval\_start_2, \ldots, interval\_start_{TI}, \\
&interval\_end_1, interval\_end_2, \ldots, interval\_end_{TI}, \\
&shift\_start_1, shift\_start_2, \ldots, shift\_start_S, \\
&shift\_end_1, shift\_end_2, \ldots, shift\_end_S\}
\end{aligned}
\tag{2.4}
$$

Depending on the chosen planning procedure (see Section 5.6), either the minimum requirements or the preferred requirements are the hard constraints. Consequently, a feasible schedule is one in which there are people scheduled for every required shift or time period. The personnel requirements for the time interval are satisfied when the following criteria, reported in (2.5), are fulfilled.

Head Nurse			
$start_{ti}$	end_{ti}	$RIM_{h,ti,d}$	$RIP_{h,ti,d}$
8:00	17:00	1	1

Regular Nurse			
$start_{ti}$	end_{ti}	$RIM_{r,ti,d}$	$RIP_{r,ti,d}$
0:00	7:00	1	1
7:00	13:00	2	3
13:00	21:00	2	2
21:00	24:00	1	1

Nurse Aid			
$start_{ti}$	end_{ti}	$RIM_{a,ti,d}$	$RIP_{a,ti,d}$
8:00	17:00	1	1

Table 2.3: Example of floating personnel requirements on day d, the indices h, r, and a correspond to the three different skill categories

2.4 Soft Constraints

2.4.1 Introduction

The real-world situation addressed in this research incorporates a high number of soft constraints on the personal schedules. The soft constraints will preferably be satisfied, but violations can be accepted to a certain extent. It is highly exceptional in practice to find a schedule that satisfies all the soft constraints. The aim of the search algorithms is to minimise the real impact of violations of these constraints. The users of the system specify all the constraints.
Some of the constraints strengthen other constraints, while others are adverse factors in the planning of real-world hospital wards. Very often a few constraints are even contradictory in reality. It is sometimes obvious that certain constraints can never be satisfied at all. In all of these cases, the user of the planning software must be informed about the extent to which each type of constraint is violated.

$$\forall pit, (1 \leq pit \leq PIT);\ \forall d, (1 \leq d \leq D):$$

$$\sum_{t_i\ \xi\ interval\_start_{ti} \leq PIT < interval\_end_{ti}} RIM_{q,ti,d} \leq$$

$$|\{(p,s)\ \xi\ 1 \leq p \leq P \wedge shift\_start_s \leq pit < shift\_end_s \wedge (schedule_{p,(d-1)*S+t_s} = q \vee schedule_{p,(d-1)*S+t_s} = pref + q)\}| \tag{2.5}$$

$$\leq \sum_{ti\ \xi\ interval\_start_{ti} \leq PIT < interval\_end_{ti}} RIP_{q,ti,d}$$

2.4.2 Relaxation of constraints

Apart from describing the meaning of every constraint, we also explain some exceptions for the evaluation in addition to certain corrections which are required in holiday periods or periods of illness absence. Boundary constraints at the beginning and end of the planning period have an important impact on the evaluation. In general, the rule holds that a penalty is generated when a violation of a constraint could be avoided in the current planning period by scheduling appropriately. No violation is generated when the constraint, which is not satisfied, can still be satisfied by scheduling appropriate shifts in the next planning period.
The evaluation of an absence and a free day can be very different. Absences are personal constraints such as holidays, illness, etc (Constraint 24 and 25 in Section 2.4.3). Free days are days in a personal schedule on which nothing is scheduled. Manual planners evaluate some constraints in a less strict manner when they are violated by a *day off* than in the case where the violation is caused by a free day. Some constraints are relaxed when an absence prevents scheduling shifts which could satisfy the constraint. Free days cannot allow for a compensation because free days can become scheduled days by assigning a shift. The same ideas hold when distinguishing between scheduled shifts and requested shifts for some particular soft constraint types (Constraint 26), as explained in Section 2.4.3.

2.4.3 Categories of soft constraints

Hospital constraints

Personnel scheduling is organised per hospital ward in this research project. A ward is a group of personnel working together in the same location (e.g. a certain floor in a hospital) or on the basis of their activities (e.g. the ambulance team). A number of general constraints are recommended by the hospital but in certain situations, they may need to be ignored. Next to the rules which hold in the entire hospital, each ward can define their house rules. The general hospital constraints will be listed in turn and explained in detail.

Constraint 1 *Minimum time between two assignments*

There is a legal constraint depicting how many hours personnel should be free between two assignments. In practice, the time between two assignments depends on the shift types. The formulation of this constraint is represented by two extra data fields corresponding to the shift types. In ANROM, planners can decide to augment or diminish the rest time before and after each shift type. Scheduling a morning shift which starts only 7 hours after a late shift has ended is obviously worse than scheduling a short afternoon shift within 7 hours after a short morning shift, for example. Before and after very short duties (like a morning shift from 8 till 12) a shorter break can be acceptable. The terms $shift\_before_s$ and $shift\_after_s$ denote the recommended free time before shift type s starts and after it ends. A penalty will be generated whenever there is an overlapping in time between a shift type and a forbidden zone from another shift. Since this constraint is always very important, the penalty is proportional to the number of overlapping minutes. Table 2.4 presents an example, extracted from a real-world hospital situation, in which the start and end times of the shift types, the recommended idle times before and after them are shown. The names of the shift types are abbreviations of the shift names in the hospital. Fig. 2.3, which is derived from the data in Table 2.4, illustrates the constraint better. Each shift type is presented as a bar in the figure. The start and end time of the shifts are clearly visible and forbidden sequences between shift types can be derived from the periods before and after the shift, in which no other work is allowed. The constraint restricts shift types scheduled on a certain day, in addition to assignments on consecutive days in some particular cases. It makes no sense to check this constraint for assignments that are two days or more apart. The time between them will always suffice. The penalty for Constraint 1 is denoted by $penalty_{p,C1}$ for person p. The formal representation is given in (2.6).

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P),\ \forall t_1, (1 \leq t_1 \leq T),\ \forall t_2, (t_1 < t_2 \leq T) : \\
\\
\quad penalty_{p,C1} = 0 \\
\quad s_1 = s_{t_1}, s_2 = s_{t_2} \\
\quad d_1 = t_1/S, d_2 = t_2/S \\
\quad x = shift\_end_{s_1} + shift\_after_{s_1} - (shift\_start_{s_2} + 24 * 60 * (d_2 - d_1)) \\
\quad y = shift\_end_{s_1} - (shift\_start_{s_2} + 24 * 60 * (d_2 - d_1) - shift\_before_{s_2}) \\
\\
\quad IF\ (d_1 + 2 \geq d_2 \wedge schedule_{p,t_1} \neq 0 \wedge schedule_{p,t_2} \neq 0) \\
\\
\qquad\qquad \Rightarrow penalty_{p,C1} + \left\{ \begin{array}{ll} x & IF\ (x > 0) \\ y & IF\ (y > 0) \end{array} \right.
\end{array}
\tag{2.6}
$$

Constraint 2 *Alternative skill category*

				Unoccupied Time (hrs)	
Shifts		$shift\_start_s$	$shift\_end_s$	$shift\_before_s$	$shift\_after_s$
SE	'Short' Early	6:00	10:00	10	8
EE	'Early' Early	6:00	14:00	10	8
SD	'Short' Day	8:00	12:00	8	8
E	Early	8:00	17:00	8	8
D	Day	10:00	18:00	8	8
SL	'Short' Late	14:00	18:00	8	8
L	Late	14:00	22:00	8	10
LL	'Late' Late	17:00	22:00	8	10
N	Night	22:00	6:00	10	10

Table 2.4: Start and end times of a realistic shift type set; unoccupied periods before and after

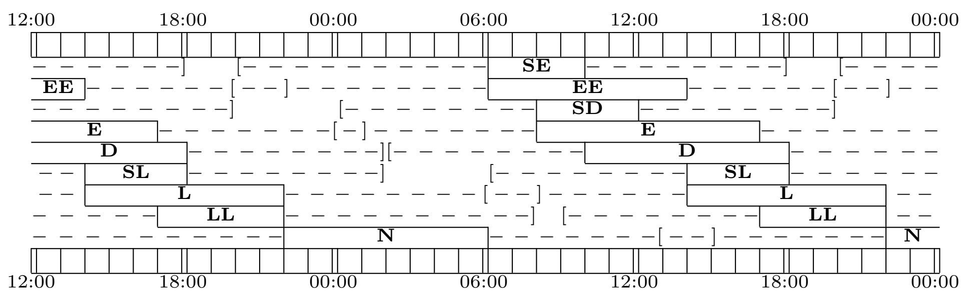

Figure 2.3: Minimum time between two shift types

It is a hard constraint that all the work has to be done by skilled personnel. If a certain duty requires a head nurse, then preferably a head nurse will do the job unless there is none available. The only possibility to still obtain a feasible solution in that case is to find a person from another skill category who is authorised to replace people from the required skill category (see Section 2.2.2). Assigning people from alternative skill categories is sometimes necessary to cater for staff shortages, but it is not desirable and will be penalised in the evaluation function. A penalty is generated each time a shift is performed for a duty other than ones that are covered by the prime skill category: $penalty_{p,C2}$. This is presented in (2.7).

$$\forall p, (1 \leq p \leq P):$$

$$penalty_{p,C2} = |\{t \; \xi \; 1 \leq t \leq T \wedge schedule_{p,t} \in QA_p \cup QA_p + pref\}| \quad (2.7)$$

Constraints defined by the work regulation

Every personnel member has a contract with the hospital. It is called the work regulation or work agreement (see Section 2.2.3). There are different work regulations for full time and half time personnel members, night nurses, etc. Many hospitals in Belgium allow for the definition of a personal work agreement per nurse. This enables them to formulate personal constraints such as *every Wednesday afternoon free*, *work a weekend every two weeks*, *no stand by duty*, etc. When defining the work regulation, either of the following constraints can be defined or made idle.

Constraint 3 *Maximum number of assignments*

This constraint determines the number of shifts a person with the work regulation w can -at most- work during the planning period: max_w. In order to reflect the real-world situation, adaptations to this number are made in cases of illness, holiday, ... When a personal schedule contains an 'absence', the type of leave or absence is given. Depending on the reason for the absence, it will be taken into account for the evaluation of the assignments and hours. We denote by A_p the number of absence days within the planning period. Consider a planning period of 4 weeks (28 days) and a *full time* work agreement with maximum 20 assignments during the planning period. Suppose a full time personnel member is on sickness leave during 10 days. The constraint is adapted to this situation by changing the value of the maximum number of assignments from 20 to $\lceil 20 * (28 - 10)/28 \rceil$. The cost function value for the constraint, $penalty_{p,C3}$ for person p, is calculated per planning period. There is no transfer of excesses to the next planning period. This is represented by (2.8).

$$\forall p, (1 \leq p \leq P):$$

$$\begin{aligned} & penalty_{p,C3} = x \ IF \ (x > 0) \\ & w = w_p \\ & x = |\{t \ \xi \ 1 \leq t \leq T \wedge schedule_{p,t} \neq 0\}| - \tfrac{D - A_p}{D} * max_w \end{aligned} \tag{2.8}$$

Constraint 4 *Maximum number of consecutive days*

This constraint limits the maximum number of consecutive working days by $max\_consecutive\_days_w$. The evaluation involves checking the days which are scheduled at the end of the previous planning period ($previous\_consecutive\_days_p$). In practice, the schedule of the previous planning period is known and the consecutive days at the end of it can be calculated. For the description of the soft constraints in this section, we assume that the required data is given. Suppose, for example, that the maximum number of consecutive days is 7. If 6 consecutive days are scheduled at the end of the previous planning period, the first day of the current planning period can be scheduled without violating the constraint. Scheduling both the first and the second day will generate a penalty. When the last 8 consecutive days at the end of the previous planning period have assignments, no penalty will be generated in the current period when there is nothing scheduled on the first day. Constraint violations made in the past are not penalised in the current planning period. This is formally presented in (2.9).

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\quad w = w_p \\
\quad consecutive\_days = previous\_consecutive\_days_p \\
\quad penalty_{p,C4} = 0 \\
\quad \forall d, (1 \leq d \leq D): \\
\quad \left\{ \begin{array}{ll}
x = |\{t \ \xi \ 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| & \\
consecutive\_days + 1 & IF \quad (x \geq 1) \\
\left. \begin{array}{l}
y = consecutive\_days - max\_consecutive\_days_w \\
penalty_{p,C4} + y, \ IF \quad (y > 0) \\
consecutive\_days = 0
\end{array} \right\} & IF \quad (x = 0)
\end{array} \right.
\end{array}
\tag{2.9}
$$

Constraint 5 *Minimum number of consecutive days*

A working day between two free days is seldom wanted. We denote $min\_consecutive\_days_w$ as the minimum number of consecutive working days for the work regulation w. Also for this constraint, the previous planning period for person p is taken into account by $previous\_consecutive\_days_p$. A penalty will be generated if the constraint could be satisfied by scheduling a duty on the first day(s) of the current planning period. Consequently, it is not considered a violation when the constraint is not satisfied at the end of the planning period. If, for example, at least 3 consecutive days are required, no penalty will be generated if only the last two days of the current planning period are scheduled. The constraint on the minimum number of consecutive days is relaxed when a stretch of working days and requested days off meets the requirement. Consider the case where at least three consecutive days are required. Suppose a person works two consecutive days and ends with a requested day off immediately afterwards. This schedule will not generate a penalty because the requested day off is, unlike free days, considered part of the work stretch. A day off d, for person p is denoted by 1 in $days\_off_{p,d}$ (see also Constraint 24) other days are denoted by 0. This is illustrated by (2.10).

Constraint 6 *Maximum number of consecutive free days*

$$
\begin{aligned}
&\forall p, (1 \leq p \leq P): \\
&\quad w = w_p \\
&\quad consecutive\_days = previous\_consecutive\_days_p \\
&\quad consecutive\_days\_off = previous\_consecutive\_days\_off_p \\
&\quad penalty_{p,C5} = 0 \\
&\quad \forall d, (1 \leq d \leq D): \\
&\quad \begin{cases}
x = |\{t \;\xi\; 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| & \\
consecutive\_days + 1, & IF \;\; (x \geq 1) \\
cons\_days\_off + 1, & IF \;\; (x = 0 \wedge days\_off_{p,d} = 1) \\
\left.\begin{array}{l}
y = min\_consecutive\_days_w - consecutive\_days \\
\qquad -cons\_days\_off \\
penalty_{p,C5} + y, \quad IF \;\; (y > 0) \\
consecutive\_days = 0 \\
cons\_days\_off = 0
\end{array}\right\} & \\
& IF \; (x = 0 \wedge days\_off_{p,d} = 0)
\end{cases}
\end{aligned}
\tag{2.10}
$$

The value $max\_cons\_free\_days_w$ denotes the maximum number of consecutive free days for work regulation w. The number of consecutive free days at the end of the previous planning period is denoted by $previous\_consecutive\_free\_days_p$ for person p. The constraint is presented by (2.11).

Constraint 7 *Minimum number of consecutive free days*

The minimum number of consecutive free days for work regulation w is denoted by $min\_cons\_free\_days_w$. More formal detail is given in (2.12). Constraint 6 and 7 are analogous to the previous two constraints, they limit the consecutive free days instead of the consecutive working days. The same rules hold at the beginning and the end of the planning period. With respect to absences and free days, the most relaxed attitude holds. Absence days are not added to the number of consecutive free days (even if they occur in the sequence) when the maximum number is evaluated. Absence days are added to the number of consecutive free days for the minimum constraint. There is no violation for the minimum number of consecutive free days, when absence days are isolated (flanked by working days).

Constraint 8 *Maximum number of hours worked*

The limit on the maximum number of hours during a planning period is given by max_hours_w for the corresponding work regulation. Unlike most other constraints, the working time is cumulative and this also affects the evaluation of Constraint 9. By adding the real amount of overtime or undertime to the

$$
\begin{aligned}
&\forall p, (1 \leq p \leq P):\\
&\quad w = w_p\\
&\quad cons\_free\_days = previous\_consecutive\_free\_days_p\\
&\quad penalty_{p,C6} = 0\\
&\quad \forall d, (1 \leq d \leq D):\\
&\quad \left\{ \begin{array}{ll}
x = |\{t \ \xi \ 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| & \\
consecutive\_free\_days + 1 & IF\ (x = 0)\\
\left. \begin{array}{l}
y = cons\_free\_days - max\_cons\_free\_days_w\\
penalty_{p,C6} + y,\ IF\ (y > 0)\\
cons\_free\_days = 0
\end{array} \right\} & IF\ (x \geq 1)
\end{array} \right.
\end{aligned}
\tag{2.11}
$$

scheduled time in the next planning period, the system prevents the team from having unfair schedules. The balance of the working hours for personnel member p, which is transferred to the next planning period, ($previous\_hours_p$) is added to the hours carried out in the current planning period. Suppose the starting balance for a person is negative in the current planning period. There is a possibility for compensating undertime without generating a penalty for overtime.

Overtime is very common in hospitals, so an option is available for not penalising overtime unless a certain threshold $threshold\_hours$ is exceeded. The value is the same for all the personnel in the ward. When violations are less than this given number of hours, the penalty is ignored.

Every work regulation has a $standard\_performance_w$, which gives the normal number of hours worked on a standard day. We call $AH_p = standard\_performance_w * A_p$ the number of hours on absence days. In nearly the same way as explained with respect to Constraint 3, a weighted new value for the maximum number of hours is calculated. This is demonstrated in (2.13).

Constraint 9 *Minimum number of hours worked*

The minimum number of hours a person should work during the planning period is min_hours_w. Evaluating the constraint is carried out in the same way as Constraint 8. The balance of hours worked at the end of the previous planning period is added to the hours of the current period. Absence days or illness can be compensated in the evaluation of the constraint. The constraint is illustrated in (2.14). The evaluation of the constraint can be relaxed by setting a threshold value for generating penalties. When violations are less than a given number of hours, it is possible to ignore these violations of the constraint. Since the real amount of overtime or undertime will be added to the scheduled time in the next planning period, the system prevents the team from having unfair schedules. A

$$
\forall p, (1 \leq p \leq P):
$$

$$
\begin{array}{l}
w = w_p \\
cons\_free\_days = previous\_consecutive\_free\_days_p \\
cons\_days\_off = previous\_cons\_days\_off_p \\
penalty_{p,C7} = 0 \\
\\
\forall d, (1 \leq d \leq D): \\
\left\{ \begin{array}{l}
x = |\{t \ \xi \ 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| \\
consecutive\_days + 1, \qquad IF \quad (x \geq 1) \\
cons\_days\_off + 1, \qquad IF \quad (x = 0 \wedge days\_off_{p,d} = 1) \\
\quad \left. \begin{array}{l}
y = min\_cons\_free\_days_w - (cons\_free\_days + cons\_days\_off) \\
penalty_{p,C7} + y, \ IF \quad (y > 0) \\
cons\_free\_days = 0 \\
cons\_days\_off = 0
\end{array} \right\} \\
\qquad\qquad IF \quad (x = 0 \wedge days\_off_{p,d} = 0)
\end{array} \right.
\end{array}
\tag{2.12}
$$

$$
\forall p, (1 \leq p \leq P):
$$

$$
\begin{array}{ll}
w = w_p & \\
penalty_{p,C_8} = 0 & \\
x = max\_hours_w - AH_p & \\
y = previous\_hours_p + \sum_{t=1}^{T} shift\_duration_{s_t} * (1 - \delta_{schedule_{p,t},0}) & \\
penalty_{p,C_8} = y - x & IF \quad (y - x > threshold\_hours)
\end{array}
\tag{2.13}
$$

correction is calculated when a person is absent due to illness or a holiday. In exactly the same way as explained with respect to Constraint 3, a weighted new value for the maximum and minimum number of hours is calculated.

Constraint 10 *Maximum number of assignments per day of the week*

This constraint limits the number of assignments on certain days of the week by $max\_day_{w,day}$, in which w denotes the work regulation and day is the day of the week (Monday till Sunday). It is, for example, possible to provide at least one free Monday during the planning period or to restrict the number of working weekends with this constraint. There is no transfer between planning periods. The constraint is formally presented in (2.15).

Constraint 11 *Maximum number of assignments for each shift type*

This constraint provides the possibility of forbidding and/or restricting the assignment of certain shift types $shift$ for the work agreement by $max\_shift_{w,shift}$.

$$\forall p, (1 \leq p \leq P):$$

$$\begin{array}{ll} w = w_p & \\ penalty_{p,C_9} = 0 & \\ x = min\_hours_w - AH_p & \\ y = previous\_hours_p + & \\ \sum_{t=1}^{T} shift\_duration_{s_t} * (1 - \delta_{schedule_{p,t},0}) & \\ penalty_{p,C_9} = x - y & IF \ \ (x - y > threshold\_hours) \end{array} \tag{2.14}$$

$$\forall p, (1 \leq p \leq P):$$

$$w = w_p$$
$$penalty_{p,C_{10}} = 0$$

$$\forall day, (1 \leq day \leq 7):$$

$$\begin{cases} x_{day} = max\_day_{w,day} & \\ y_{day} = |\{(week, t) \ \xi \ 1 \leq week \leq D/7 \wedge 1 \leq t \leq S & \\ \qquad \wedge schedule_{p,(week-1)*7*S+(day-1)*S+t} \neq 0\}| & \\ & \\ penalty_{p,C_{10}} + y_{day} - x_{day} & IF \ \ (y_{day} > x_{day}) \end{cases} \tag{2.15}$$

The planner can, when for example defining a work agreement for night nurses, set to 0 the number of allowed shift types for every shift that differs from the night shift. Other work agreements, like cleaner for example, will never work a night shift. Very often, the maximum number for each shift type is set to a rather low number in order to enable shift type variation in the schedules. The constraint is demonstrated formally in (2.16).

Constraint 12 *Maximum number of a shift type per week*

For every week *week* in the planning period, the user can limit the number of assignments in a personal schedule for every shift type *shift* by $max\_shift\_week_{w,shift,week}$. This constraint can, for example, prevent the assignment of seven night duties in one week. Since the system allows different constraint values for different weeks, it can also allow for the definition of shift type cycles like one 'early week followed by a late week'. It can be formally illustrated as in (2.17).

Constraint 13 *Number of consecutive shift types*

For each shift type *shift*, a series of allowed sequences can be defined. In $consecutive\_shift_{w,shift}[i]$, i can take values from 1 to 10 and

$$
\forall p, (1 \leq p \leq P) :
$$

$$
\begin{aligned}
& w = w_p \\
& penalty_{p,C_{11}} = 0 \\
& \forall shift, (1 \leq shift \leq S) : \\
& \begin{cases}
x_{shift} = max\_shift_{w,shift} \\
y_{shift} = |\{d \; \xi \; 1 \leq d \leq D \wedge schedule_{p,(d-1)*S+t_{shift}} \neq 0\}| \\
\\
penalty_{p,C_{11}} + y_{shift} - x_{shift} \qquad\qquad IF \;\; (y_{shift} > x_{shift})
\end{cases}
\end{aligned}
\tag{2.16}
$$

$$
\forall p, (1 \leq p \leq P) :
$$

$$
\begin{aligned}
& w = w_p \\
& penalty_{p,C_{12}} = 0 \\
& \forall week, (1 \leq week \leq D/7), \forall shift, (1 \leq shift \leq S) : \\
& \begin{cases}
x_{shift,week} = max\_shift_{w,shift,week} \\
y_{shift,week} = |\{d \; \xi \; 1 \leq d \leq 7 \wedge \\
\quad schedule_{p,(week-1)*7*S+(d-1)*S+t_{shift}} \neq 0\}| \\
\\
penalty_{p,C_{12}} + y_{shift,week} - x_{shift,week}, IF \; (y_{shift,week} > x_{shift,week})
\end{cases}
\end{aligned}
\tag{2.17}
$$

if $consecutive\_shift_{w,shift}[i] = 1$ the sequence i is allowed. The model, for example, supplies the possibility of defining 2, 4, and 6 as allowed sequences when $consecutive\_shift_{w,shift}[2] = consecutive\_shift_{w,shift}[4] = consecutive\_shift_{w,shift}[6] = 1$, and for all the other sequences the value is 0. Sequences of consecutive shifts at the end of the previous planning period can influence this constraint. They are denoted by $previous\_consecutive\_shifts_{p,s}$. The occurrence of an absence or illness day relaxes this constraint. When the result of adding the absence day(s) to a sequence satisfies the constraint, no penalty will be charged. Also, when the addition of absence days is not necessary to satisfy the constraint, the absences are ignored. It resembles the evaluation of Constraint 4, 5, 6, and 7 and is formally presented in (2.18).

Constraint 14 *Assign 2 free days after night shifts*

Night shifts are all the shift types which begin before and end after 00:00. The set *night* contains the serial numbers of the corresponding shift types. When this constraint is valid ($night\_free_w = 1$), a night shift must be followed by

$$
\boxed{
\begin{array}{l}
\forall p, (1 \le p \le P): \\
\qquad w = w_p \\
\qquad penalty_{p,C5} = 0 \\
\qquad \forall shift, (1 \le shift \le S): \\
\qquad \left\{
\begin{array}{ll}
cons\_days = previous\_consecutive\_shifts_{p,shift} & \\
cons\_days\_off = previous\_consecutive\_days\_off_p & \\
x = |\{d \ \xi \ 1 \le d \le D \wedge schedule_{p,(d-1)*S+t_{shift}} \ne 0\}| & \\
cons\_days + 1 & IF\ (x \ne 0) \\
cons\_days\_off + 1 & IF\ (x = 0 \wedge days\_off_{p,d} = 1) \\
\left.
\begin{array}{l}
y = consecutive\_shift_{w,shift}[cons\_days] \\
z_i = consecutive\_shift_{w,shift}[cons\_days + i] \\
\qquad\qquad and\ 0 \le i \le cons\_days\_off \\
penalty_{p,C13} + 1,\ IF\ (y = 0 \wedge \sum_i z_i = 0) \\
consecutive\_days = 0 \\
cons\_days\_off = 0
\end{array}
\right\} & \\
& IF\ (x = 0 \wedge days\_off_{p,d} = 0)
\end{array}
\right.
\end{array}
}
\tag{2.18}
$$

another night shift or by two consecutive free days. An absence day counts for a free day. The constraint depends on daily sequences and the values will thus be transferred at the border of different planning periods. The number of free days after a night shift at the end of the previous planning period is given by $previous\_free\_after\_night_p$. The constraint is presented in (2.19).

Constraint 15 *Assign complete weekends*

Setting this constraint does not allow a shift on Saturday without one on the next Sunday or vice versa. It is denoted by $complete\_weekends_w$ for the entire work regulation. There is a possibility for redefining weekends, by either considering Friday and/or Monday as part of the weekend. The weekend definition is given by $weekend_w$, for which value 0 means a Saturday-Sunday weekend; 1 denotes a Friday-till-Sunday weekend; 2 stands for a Friday-till-Monday weekend, and 3 is a Saturday-till-Monday weekend. The complete weekend constraint will impose a shift to be planned on all the other weekend days as soon as there is an assignment on Saturday or Sunday. However, a scheduled shift on Friday or Monday does not require assignments on Saturday and Sunday. Again, absence days will be considered as free days or as working days whenever they relax the constraint most.
The schedule of the previous planning period can play a role in the evaluation of the constraint if the switch between planning periods happens in a weekend. In ANROM, it will only occur when the value of $weekend_w$ equals 2 or 3. We de-

$$
\begin{aligned}
&\forall p, (1 \le p \le P): \\
&\quad w = w_p \\
&\quad penalty_{p,C14} = 0 \\
&\quad cons\_days = previous\_free\_after\_night_p \\
&\quad IF\ (night\_free_w = 1) \\
&\quad \forall d, (1 \le d \le D): \\
&\quad \begin{cases}
n = |\{t \ \xi\ 1 \le t \le S \wedge schedule_{p,(d-1)*S+t} \ne 0 \\
\quad \wedge s_t \in night\}| \\
m = |\{t \ \xi\ 1 \le t \le S \wedge schedule_{p,(d-1)*S+t} \ne 0 \\
\quad \wedge s_t \notin night\}| \\
\\
IF\ (last\_shift \in night) \\
\begin{cases}
cons\_days + 1 & IF\ (n = 0 \wedge m = 0) \\
cons\_days = 0 & IF\ (n \ne 0) \\
penalty_{p,C14} + (2 - cons\_days) & \\
 & IF\ (n + m \ne 0 \wedge 2 - cons\_days > 0)
\end{cases} \\
\\
ELSE\ IF\ (n \ne 0 \vee m \ne 0) \\
\begin{cases}
last\_shift = s_t \\
cons\_days = 0
\end{cases}
\end{cases}
\end{aligned}
\tag{2.19}
$$

note the weekend days which are transferred from the previous planning period by $previous\_sat_p$ and $previous\_sun_p$. These values are equal to 1 if there is at least one assignment on the corresponding day; it is 0 if the day is empty.The full constraint can be seen in (2.20).

Constraint 16 *No night shift before a free weekend*

Shifts which end after midnight cannot be scheduled before a free weekend when this constraint is valid, i.e. when $night\_weekend_w$ equals 1. For ordinary Saturday-Sunday weekends, this constraint requires that night shifts are not scheduled on Fridays when the entire weekend is free. Other definitions of weekends (e.g. Friday-Saturday-Sunday) restrict the schedule similarly. The constraint is illustrated in (2.21).

Constraint 17 *Assign identical shift types during the weekend*

If this constraint (denoted by $identical\_weekend_w = 1$) is active, it creates a penalty when the shift types during the weekend days are not the same. No matter what the weekend definition is, whether it includes Friday and/or Monday, this constraint only looks at the shifts which are assigned on Saturday

$$\forall p, (1 \le p \le P):$$

$$w = w_p$$
$$penalty_{p,C15} = 0$$
$$weekend\_days = previous\_weekend\_days$$

$$IF\ (complete\_weekends_w \ne 0):\ \forall wk, (1 \le wk \le D/7):$$

$$\begin{cases}
absence_fri = absence_sat = absence_sun = absence_mon = 0 \\
fri = sat = sun = mon = 0 \\
fri = 1 \quad IF\ |\{t\ \xi\ 1 \le t \le S \wedge schedule_{p,((wk-1)*7+4)*S+t} \ne 0\}| \ne 0 \\
sat = 1 \quad IF\ |\{t\ \xi\ 1 \le t \le S \wedge schedule_{p,((wk-1)*7+5)*S+t} \ne 0\}| \ne 0 \\
sun = 1 \quad IF\ |\{t\ \xi\ 1 \le t \le S \wedge schedule_{p,((wk-1)*7+6)*S+t} \ne 0\}| \ne 0 \\
mon = 1\ IF\ |\{t\ \xi\ 1 \le t \le S \wedge schedule_{p,(wk*7*S+t} \ne 0\}| \ne 0 \\
\begin{cases} fri = 1 \\ sat = previous_sat_p \quad\quad IF\ (wk = 1) \\ sun = previous_sun_p \end{cases} \\
mon = 1 \quad\quad\quad IF\ (wk = D/7) \\
absence_fri = days_off_{p,(wk-1)*7+4}\ IF\ (fri = 0) \\
absence_sat = days_off_{p,(wk-1)*7+5}\ IF\ (sat = 0) \\
absence_sun = days_off_{p,(wk-1)*7+6}\ IF\ (sun = 0) \\
absence_mon = days_off_{p,wk*7}\ IF\ (mon = 0) \\
absence = absence_sat + absence_sun \\
\\
IF\ (weekend_w = 0): \\
penalty_{p,C15} + 1,\ IF\ (sat \ne sun \wedge absence = 0) \\
\\
IF\ (weekend_w = 1): \\
penalty_{p,C15} + 1,\ IF\ ((sat \ne sun \wedge absence = 0) \\
\quad \vee ((fri = 0 \wedge absence_fri \ne 0) \wedge sat + sun = 2)) \\
\\
IF\ (weekend_w = 2): \\
penalty_{p,C15} + 1, \\
\quad IF\ ((sat \ne sun \wedge absence = 0) \vee (((fri = 0 \wedge absence_fri \ne 0) \\
\quad \vee (mon = 0 \wedge absence_mon \ne 0)) \wedge sat + sun = 2)) \\
\\
IF\ (weekend_w = 3): \\
penalty_{p,C15} + 1,\ IF\ ((sat \ne sun \wedge absence = 0) \\
\quad \vee ((mon = 0 \wedge absence_mon \ne 0) \wedge sat + sun = 2))
\end{cases} \tag{2.20}$$

$$
\forall p, (1 \leq p \leq P):
$$

$$
\begin{array}{l}
w = w_p \\
penalty_{p,C16} = 0 \\
\\
IF\ (night\_weekend_w \neq 0):\ \forall week, (1 \leq week \leq D/7): \\
\\
\left\{
\begin{array}{l}
\begin{array}{rcl}
fri & = & |\{t\ \xi\ 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+4)*S+t} \neq 0\}| \\
sat & = & |\{t\ \xi\ 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+5)*S+t} \neq 0\}| \\
sun & = & |\{t\ \xi\ 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+6)*S+t} \neq 0\}|
\end{array} \\
\\
IF\ (weekend_w = 0 \vee weekend_w = 3) \\
\left\{
\begin{array}{l}
n = |\{t\ \xi\ 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+4)*S+t} \neq 0 \\
\qquad\qquad\qquad \wedge\ s_t \in night\}| \\
penalty_{p,C16} + 1, \quad IF\ (sat + sun = 0 \wedge n = 1)
\end{array}
\right. \\
\\
IF\ (weekend_w = 1 \vee weekend_w = 2) \\
\left\{
\begin{array}{l}
n = |\{t\ \xi\ 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+3)*S+t} \neq 0 \\
\qquad\qquad\qquad \wedge\ s_t \in night\}| \\
penalty_{p,C16} + 1, \quad IF\ (fri + sat + sun = 0 \wedge n = 1)
\end{array}
\right.
\end{array}
\right.
\end{array}
\tag{2.21}
$$

and Sunday. An absence will take a dummy value for this constraint, in order to generate the lowest possible penalty. The constraint is presented in (2.22).

Constraint 18 *Maximum number of consecutive working weekends*

This constraint limits the number of weekends in which duties are assigned with $max\_consecutive\_weekends_w$. It does not matter if the weekends are not completely scheduled. Only Saturdays and Sundays contribute to this constraint, even if Friday and/or Monday are considered part of the weekend. As is explained for other constraints on the order in which assignments may or may not be made (consecutiveness constraints), values are transferred to the next planning period. The number of consecutive weekends in person p's schedule at the end of the previous planning period is $previous\_consecutive\_weekends_p$. A formal representation can be seen in (2.23).

Constraint 19 *Maximum number of working weekends in 4 weeks*

The constraint is a restriction on weekend work during periods of 4 consecutive weeks, provided that at least one of the 4 weeks belongs to the planning period. The maximum number of weekends is given by $max\_weekends\_4\_weeks_w$. Suppose, for example, that we have a planning period of x weeks. The constraint will be evaluated in x overlapping periods, from the period which starts 3 weeks before the current one, up until the period which ends with the last week of the

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\\
\quad w = w_p \\
\quad penalty_{p,C17} = 0 \\
\\
\quad IF\ (identical\_weekend_w = 1):\ \forall week, (1 \leq week \leq D/7): \\
\\
\quad \left\{ \begin{array}{l}
\forall t, (1 \leq t \leq S): \\
\left\{ \begin{array}{ll}
sat = t\ \ IF\ (schedule_{p,((week-1)*7+5)*S+t} \neq 0) & \\
sun = t\ \ IF\ (schedule_{p,((week-1)*7+6)*S+t} \neq 0) & \\
absence\_sat = days\_off_{p,(week-1)*7+5} & IF\ (sat = 0) \\
absence\_sun = days\_off_{p,(week-1)*7+6} & IF\ (sun = 0)
\end{array} \right. \\
penalty_{p,C17} + 1,\ IF\ (sat \neq sun \wedge absence\_sat + absence\_sun = 0)
\end{array} \right.
\end{array}
\tag{2.22}
$$

current period. The number of working weekends for person p in the previous planning period is given by $previous\_weekends\_3_p$, $previous\_weekends\_2_p$, and $previous\_weekends\_1_p$. It is a very specific request which was implemented to satisfy the needs of particular users of the software based on an earlier version of ANROM and is formally illustrated in (2.24).

Constraint 20 *Maximum number of assignments on bank holidays*

Unlike most of the other constraints this constraint is cumulative. Bank holidays are recorded over a longer period than the planning period only. For each person, the number of cumulative assignments on bank holidays is denoted by $previous\_bank_p$. The maximum number of assignments on bank holidays per work regulation is given by $bank\_holidays_w$. A structure *bank* is an array of length D and it has value 1 for bank holidays and value 0 for other days. Usually hospitals prefer to limit the number of assignments on bank holidays during an entire year with this value. The constraint is formally demonstrated in (2.25).

Constraint 21 *Restriction on the succession of shift types*

The constraint on the minimum time between shift types already restricts some sequences of constraints. However, the current constraint can explicitly forbid particular combinations of shift types. Unlike the constraint on minimum time between assignments, this constraint evaluates shifts which are scheduled on consecutive days. A scheduled shift is connected to the day at which the shift starts. The succession constraint also provides the possibility of forbidding certain shifts after a free day or even free days after certain shifts. The restrictions are denoted by $succession_w$, a two dimensional structure with a column and row for each shift type in addition to one for an empty day. The elements in $succession_w$ are 0 when the column shift cannot be scheduled after the row shift, and 1 when there is no restriction on the succession. Only the last day of the previous planning period can influence the evaluation of the constraint

$$\forall p, (1 \leq p \leq P):$$

$$w = w_p$$
$$consecutive\_weekends = previous\_consecutive\_days_p$$
$$penalty_{p,C18} = 0$$

$$\forall week, (1 \leq week \leq D/7):$$

$$\begin{cases} sat = |\{t \xi 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+5)*S+t} \neq 0\}| \\ sun = |\{t \xi 1 \leq t \leq S \wedge schedule_{p,((week-1)*7+6)*S+t} \neq 0\}| \\ consecutive\_weekends + 1, \qquad IF\ (sat + sun > 0) \\ \left. \begin{array}{l} x = consecutive\_weekends - max\_consecutive\_weekends_w \\ penalty_{p,C18} + x, \quad IF\ (x > 0) \\ consecutive\_weekends = 0 \end{array} \right\} \\ \qquad IF\ (sat + sun = 0) \end{cases} \tag{2.23}$$

in the current period. The parameter $last\_day_{p,t}$, has value 1 when the corresponding assignment unit t on the last day of person p's previous planning period is occupied. A real-world example demonstrating this constraint is given in Table 2.5. The combinations of letters in the rows and columns (SE, EE, etc) are abbreviations of shift types in a practical hospital application. The corresponding shift types have been presented in Table 2.4.

Constraint 22 *Patterns*

A pattern is a very complex constraint which is built with a combination of different pattern types. Patterns guide the schedule to follow certain predefined cyclic arrangements. Every work regulation is subject to at most one pattern pat_w. The value is the serial number of the corresponding pattern, or is 0 when no pattern is assigned to the work regulation. Each pattern pa, $(1 \leq pa \leq PA)$ is defined by an array of length $pattern\_length_{pa}$, which is restricted to a whole number of weeks. The start date for the pattern can differ for every personnel member, but it is confined to Mondays in our approach. At the end of a pattern, the pattern will start all over again (it is independent of the planning period). We denote it by the number of the week in which the pattern for person p starts in the current planning period: $start\_pattern_p$. This enables taking weekends and particular days of the week into account in the definition.
The software provides 7 types of building blocks for the patterns, some of which are still modifiable. In order to define a pattern, one of these building blocks has to be assigned to every day of the pattern period. The 7 types of block are:

PAT–1 Obligatory assignment of a shift. This building block does not specify the shift type but prevents the corresponding day from being a free day.

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\\
\quad w = w_p \\
\quad penalty_{p,C19} = 0 \\
\\
\quad \forall period, (1 \leq period \leq D/7): \\
\\
\quad \left\{ \begin{array}{l}
n = 4 - period \\
weekends = \left\{ \begin{array}{ll} previous\_weekends\_n_p & IF\ (n > 0) \\ 0 & ELSE \end{array} \right. \\
\forall week, (-n \leq week \leq -n + 4 \wedge week > 0): \\
\\
\left\{ \begin{array}{l}
sat = |\{t \ \xi \ 1 \leq t \leq T \wedge schedule_{p,((week-1)*7+5)*S+t} \neq 0\}| \\
sun = |\{t \ \xi \ 1 \leq t \leq T \wedge schedule_{p,((week-1)*7+6)*S+t} \neq 0\}| \\
weekends + 1, \qquad\qquad\qquad IF\ (sat + sun > 0)
\end{array} \right. \\
x = weekends - max\_weekends\_4\_weeks_w \\
penalty_{p,C19} + x, \qquad IF\ (x > 0)
\end{array} \right.
\end{array}
\tag{2.24}
$$

PAT–2 Obligatory assignment of a certain shift type. The user has to set the shift type.

PAT–3 Obligatory assignment of a shift type of a certain duration, to be preset. The algorithm allows for a small deviation from that preset time, generally 15 minutes.

PAT–4 No restriction on this day.

PAT–5 Free day.

PAT–6 Day off; the day off type has to be indicated, this can be compensation, holiday, refresher courses, family reasons, etc. Depending on the type, the day off can be important for some other constraints (see also Constraint 6 and 7).

PAT–7 Forbidden shift types. Any combination of shift types can be forbidden. This building block provides the possibility of keeping Wednesday afternoons free, for example. All shift types that last longer than 12:00 or start before 19:00 will cause a violation.

Users are free to assemble these building blocks in any combination in the pattern. A pattern, built with pattern blocks and their corresponding pattern details, can schematically be presented as follows:

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\quad w = w_p \\
\quad bank\_days = previous\_bank_p \\
\quad penalty_{p,C20} = 0 \\
\quad \forall d, (1 \leq d \leq D \wedge bank[d] = 1): \\
\quad \left\{ \begin{array}{ll} x = |\{t \ \xi \ 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| & \\ bank\_days + 1 & IF \ (x > 0) \end{array} \right. \\
\quad y = bank\_days - bank\_holidays_w \\
\quad penalty_{p,C20} = y \qquad IF \ (y > 0)
\end{array} \tag{2.25}
$$

Succession	-	SE	EE	SD	E	D	SL	L	LL	N
-	v	v	v	v	v	v	v	v	v	v
SE	v	v	v	v	v	v	v	v	v	v
EE	v	v	v	v	v	v	v	v	v	v
SD	v			v	v	v	v	v	v	v
E	v			v	v	v	v	v	v	v
D	v					v	v	v	v	v
SL	v						v	v	v	v
L	v						v	v	v	v
LL	v								v	v
N	v									v

Table 2.5: Allowed successions of shift types on consecutive days are represented by ‘v’, ‘-’ denotes a day on which nothing is scheduled

$$
\begin{array}{l}
\forall pa, (1 \leq pa \leq PA): \\
\quad \forall d, (1 \leq d \leq pattern\_length_p): \\
\quad pattern\_day_{pa,d} \in \{PAT\text{-}1,\ PAT\text{-}2,\ \ldots,\ PAT\text{-}7\} \\
\quad \left\{ \begin{array}{ll} 1 \leq pattern\_detail_{pa,d} \leq S & IF \ \ (pattern\_day_{pa,d} = PAT\text{-}2) \\ 1 \leq pattern\_detail_{pa,d} \leq 24 * 60 & IF \ \ (pattern\_day_{pa,d} = PAT\text{-}3) \\ pattern\_detail_{pa,d} = day\_off\_type & IF \ \ (pattern\_day_{pa,d} = PAT\text{-}6) \\ pattern\_detail_{pa,d} = \{f_1, f_2, \ldots, f_n\} & IF \ \ (pattern\_day_{pa,d} = PAT\text{-}7) \end{array} \right. \\
\quad \textit{with } n = \textit{number of forbidden shift types;} \\
\qquad\qquad f_i \textit{ corresponds to a shift}
\end{array}
$$

$$
\begin{aligned}
&\forall p, (1 \le p \le P):\\
&\quad w = w_p\\
&\quad succession = succession_w\\
&\quad last\_day = last\_day_p\\
&\quad penalty_{p,C21}\\
&\quad\; = \; |\{(t,u)\ \xi\ (1 \le t \le S \wedge 1 \le u \le t) \wedge\\
&\quad\qquad (last\_day_t \ne 0 \wedge schedule_{p,u} \ne 0) \wedge succession_{t,u} = 0\}|\\
&\quad\; + \; |\{(t,u)\ \xi\ (1 \le t \le T - S \wedge t < u \le t + S) \wedge\\
&\quad\qquad (schedule_{p,t} \ne 0 \wedge schedule_{p,u} \ne 0) \wedge succession_{s_t,s_u} = 0\}| \qquad (2.26)\\
&\quad\; + \; |\{t\ \xi\ (1 \le t \le S) \wedge succession_{t,-} = 0 \wedge\\
&\quad\qquad last\_day_t \ne 0 \wedge |\{u\ \xi\ 1 \le u \le S \wedge schedule_{p,u} \ne 0\}| = 0\}|\\
&\quad\; + \; |\{t\ \xi\ (1 \le t \le T - S) \wedge succession_{s_t,-} = 0 \wedge schedule_{p,t} \ne 0\\
&\quad\qquad \wedge\ |\{u\ \xi\ 1 \le u \le S \wedge schedule_{p,(t/S+1)*S+u} \ne 0\}| = 0\}|\\
&\quad\; + \; |\{t\ \xi\ (1 \le t \le S) \wedge succession_{-,t} = 0 \wedge\\
&\quad\qquad schedule_{p,t} \ne 0 \wedge |\{u\ \xi\ 1 \le u \le S \wedge last\_day_u \ne 0\}| = 0\}|\\
&\quad\; + \; |\{t\ \xi\ (S \le t \le T) \wedge succession_{-,s_t} = 0 \wedge schedule_{p,t} \ne 0\\
&\quad\qquad \wedge\ |\{u\ \xi\ 1 \le u \le S \wedge last\_day_{(t/S-1)*S+u} \ne 0\}| = 0\}|
\end{aligned}
$$

Since a pattern can be incompatible with some of the other constraints of the work regulation, a method was developed to make conflicting soft constraints idle. Patterns in which the activities are set for every day (work days and free days are known) require no evaluation of the day type constraints (e.g. Constraint 3, 4, 5, 6, 7, etc). The pattern constraint has priority over these day type constraints. Other constraints, such as overtime and undertime (Constraint 8 and 9), and personal constraints (Constraint 24, 25, and 26) remain valid no matter what is in the pattern.

Details on the constraints which are made idle by certain pattern types can be found in Table 2.6, which presents a selection of possible pattern type combinations. In practice, the table includes all the possible pattern types which can be obtained by combining the pattern blocks of the above list. The first column of the table presents the combination of building blocks in the pattern. Every other column represents a soft constraint of the problem. Active constraints are denoted by a v in the table. All the constraints of the counter type (Constraint 23) are grouped in one column in the simplified representation of Table 2.6. Some of them are idle and others are active for the same pattern type combination, which is denoted by a (v) in the table. In order not to overload the notations, we do not explain further details about the validity of the different possible counter constraints combined with a pattern. The formal definition is given in (2.27).

Constraint 23 *Counters*

blocks	Constraints																										
	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
none	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v
PAT-1	v	v						v	v		v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-2	v	v						v	v		v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-3	v	v						v	v		v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-4	v	v						v	v		v	v		v		v	v			v	v	v	v	v	v	v		
PAT-6	v	v						v	v	v	v	v		v		v			v	v	v	v	(v)	v	v	v		
PAT-7	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			v	v	v	v	(v)	v	v	v	v	v
PAT-12	v	v						v	v											v		v	(v)	v	v	v	v	v
PAT-13	v	v						v	v							v			v	v		v	(v)	v	v	v		
PAT-14	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-15	v	v						v	v		v	v	v	v		v	v			v	v	v	v	v	v	v	v	v
PAT-16	v	v						v	v	v	v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-17	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			v	v	v	(v)	v	v	v	v	v
PAT-123	v	v						v	v		v	v		v		v				v	v	v	(v)	v	v	v		
PAT-124	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-125	v	v						v	v		v	v	v	v		v	v			v	v	v	v	v	v	v	v	v
PAT-126	v	v						v	v	v	v	v		v		v				v	v	v	(v)	v	v	v		
PAT-127	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v				v	v	v	(v)	v	v	v	v	v
PAT-134	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-135	v	v						v	v		v	v	v	v		v	v			v	v	v	v	v	v	v	v	v
PAT-136	v	v						v	v	v	v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-137	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			v	v	v	(v)	v	v	v	v	v
PAT-146	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-1236	v	v						v	v	v	v	v		v		v				v	v	v	(v)	v	v	v		
PAT-1246	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-1346	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-12345	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-123456	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-23	v	v						v	v		v	v		v		v				v	v	v	v	v	v	v		
PAT-24	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	(v)	v	v	v	v	v
PAT-26	v	v						v	v	v	v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-27	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			v	v	v	(v)	v	v	v	v	v
PAT-234	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-236	v	v						v	v	v	v	v		v		v				v	v	v	(v)	v	v	v		
PAT-237	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v				v	v	v	(v)	v	v	v	v	v
PAT-246	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-2346	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-34	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-36	v	v						v	v	v	v	v		v		v	v			v	v	v	(v)	v	v	v		
PAT-37	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v			v	v	v	(v)	v	v	v	v	v
PAT-346	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		v	v	v	v	v	v	v	v	v	v
PAT-46	v	v						v	v	v	v	v		v		v	v			v	v	v	v	v	v	v		

Table 2.6: Constraints conflicting with a selection of the total set of pattern types; v denotes which constraints have to be evaluated with the corresponding pattern types; according to the type of counter (v) can either mean evaluate or not

$$
\begin{array}{l}
\forall p, (1 \le p \le P): \\
\qquad w = w_p \\
\qquad IF \;\; (pat_w \neq 0) \\
\\
\qquad pa = pat_w \\
\qquad st = start\_pattern_p \\
\qquad l = pattern\_length_{pa} \\
\\
\qquad \forall d, (1 \le d \le D): \\
\\
\qquad pa\_day_d = pattern\_day_{pa,(1+l-st)/l*7+d} \\
\qquad pa\_detail_d = pattern\_detail_{pa,(1+l-st)/l*7+d} \\
\\
\qquad penalty_{p,C22} \\
\qquad \begin{array}{cl}
= & |\{d \;\xi\; 1 \le d \le D \wedge pa\_day_d = PAT\text{-}1 \wedge \\
 & |\{t \;\xi\; 1 \le t \le S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| = 0\}| \\
+ & |\{d \;\xi\; 1 \le d \le D \wedge pa\_day_d = PAT\text{-}2 \wedge \\
 & |\{t \;\xi\; 1 \le t \le S \wedge \\
 & ((schedule_{p,(d-1)*S+t} \neq 0 \wedge s_t \neq pa\_detail_d) \\
 & \vee\; (schedule_{p,(d-1)*S+t} = 0 \wedge s_t = pa\_detail_d))\}| = 0\}| \\
+ & |\{d \;\xi\; 1 \le d \le D \wedge pa\_day_d = PAT\text{-}3 \wedge x_d\}| \\
+ & |\{d \;\xi\; 1 \le d \le D \wedge pa\_day_d = (PAT\text{-}5 \vee PAT\text{-}6) \wedge \\
 & |\{t \;\xi\; 1 \le t \le S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| \neq 0\}| \\
+ & |\{d \;\xi\; 1 \le d \le D \wedge pa\_day_d = PAT\text{-}7 \wedge \\
 & |\{t \;\xi\; 1 \le t \le S \wedge \\
 & (schedule_{p,(d-1)*S+t} \neq 0 \wedge s_t \in pa\_detail_d)\}| = 0\}|
\end{array} \\
\\
\qquad with\ x_d = \begin{cases} y_d & IF\ (|y_d| > 15) \\ 0 & ELSE \end{cases} \\
\qquad and\ y_d = \sum_{t=1}^{S} (schedule_{p,(d-1)*S+t} \neq 0) * shift\_duration_{s_t} \\
\qquad\qquad - pa\_detail_d
\end{array}
\tag{2.27}
$$

For personnel members belonging to the same work regulation, there is a possibility of defining a maximum value for the workload and to balance the workload over a period different from the planning period. Users can define a set of counters ($1 \leq cntr \leq CNTR$), which become available for every work regulation in the ward. The defined counters can be related to work regulations but not all the work regulations need to be related to all the counters. We denote by $counters_w$ the list of counters which are valid for work regulation w. Every counter consists of the following fields:

- Subject to count: $counter\_subject_{cntr}$. This can be anything like *hours*, *assignments*, *days off*, *shift types* s ($1 \leq s \leq S$). A different counter can be defined per shift type.
- Time measure: $counter\_time_{cntr}$. Each of the subjects can be counted on certain time measures, as there are: *weekdays*, *weekends*, *all days*, specified days of the week day ($1 \leq day \leq 7$), *bank holidays*, ...
- Start date: $counter\_start_{cntr}$
- Duration: $counter\_duration_{cntr}$. The counter restarts each time a period as long as the start date plus the duration is finished.
- Balance: $counter\_balance_{cntr}$ can take the value 1 for balanced, 0 for not balanced.

When a counter is used in a work regulation, a maximum value has to be set. The value is denoted by $counter\_max_{w,cntr}$. Different work regulations can make use of the same counter but define their own maximum value (for example, 6 night shifts during weekends in a period of 3 months for full time contracts; 3 night shifts in weekends for the half time work regulation in the same period). When the maximum appearance of the subject to count on the time measure is violated during the counter period to which the planning period belongs, the constraint will generate a penalty. This is the contribution of the capacity part of the counter constraint to the penalty. Several counter periods can be valid during the planning period, each one requiring an evaluation. In order not to make the formulation too complex, we assume in the formulas that the counter period covers the planning period completely. For every personnel member p, the balance for each counter *cntr*, at the end of the previous planning period, is denoted by $counter\_previous_{p,cntr}$.
The second constraint type which can be evaluated with counters is more complex. Balancing the counters requires information about all the personal schedules of people for whom the work regulation is related to the counter. The evaluation of the balancing is organised as follows. In a first step, the sum $sum\_counter\_max_{cntr}$ of all the maximum values for the counter ($counter\_max_{w_p,cntr}$, for every personnel member p in the schedule) are made. The maximum values are those which belong to the work agreement corresponding to the personnel member. In case the person has absences or holidays in the counter period, the maximum number is corrected correspondingly (see also Constraint 3 and 8). The number of occurrences in the schedule of the subject to count on that particular time measure is summed for all the people who belong to a work regulation which includes the counter

in $sum\_counter\_actual_{cntr}$ $(= \sum_{p=1}^{P} counter\_actual_{p,cntr})$. Note that the period is not the current planning period but the period corresponding to the counter (which is the counter's start date plus its duration) and thus, for all the personnel who work with the counter, the value at the end of the previous planning period is added. For every person, the balanced value for the counter can be calculated by multiplying the maximum value with the ratio $counter\_actual_{cntr}/counter\_max_{cntr}$. The result rarely is a whole number and therefore, the value is replaced by an interval of length 1 whose lower limit is the integer smaller than the value ($small\_value_{cntr}$) and whose upper limit is the integer larger or equal ($large\_value_{cntr}$). To make this constraint less restrictive, the user can set a threshold value $threshold_{cntr}$ for each counter. If a value belongs to the interval whose lower limit is $small\_value_{cntr} - threshold_{cntr}$ and whose upper limit is $large\_value_{cntr} + threshold_{cntr}$, no violation is generated. Counters evaluating hours require a different threshold than daily counters. The threshold value for hours is the same as in Constraint 8. A detailed description of the constraint can be seen in (2.28).

Personal constraints

It is often possible for individual personnel members to make agreements with the personnel manager or head nurse. External or private obligations do not fall under the category of hard constraints. They can theoretically be cancelled in emergency situations. However, there are several possibilities of giving extra weight to a personal obligation. The reason for the absence can be taken into account in addition to the importance of the external commitment. Such situations are represented by the following constraints.

Constraint 24 *Day off*

Anything that prevents the personnel member from being at work can be handled as a day off in the cost function. Depending on the reason for the day off, some types of requests for absence will affect the value which is set for some of the other constraints (see Constraint 3, 7, 8, 9, etc). Every person has the right to take holiday during the working year. We do not want to generate penalties for undertime when a person takes holidays in the same period.
Illness, refresher courses, compensation, and occasional family reasons are all examples of day off types which can be placed in the schedule. The requested days off for person p at day d are denoted by 1 in $days\_off_{p,d}$. The program foresees a possibility of using an extra weight (to multiply the weight factor with) for imperative needs. When the extra weight is valid, we find a 1 in the structure $extra_{p,d}$. The value of the extra weight is the same for the entire ward: $extra\_penalty$. The system will thus distinguish between strong and weak day off requests and penalise correspondingly. This constraint is formally defined in (2.29).

Constraint 25 *Shifts off*

$$
\begin{aligned}
&\forall cntr, (1 \leq cntr \leq CNTR):\\
&\quad sum\_counter\_max_{cntr} = 0\\
&\quad sum\_counter\_actual_{cntr} = 0\\
&\quad \forall p, (1 \leq p \leq P):\\
&\quad \left\{\begin{array}{l}
w = w_p\\
penalty_{p,C23} = 0\\
penalty_{p,C23,max} = 0\\
penalty_{p,C23,balance} = 0\\
\\
\forall cntr \in counters_w :\\
\left\{\begin{array}{l}
counter\_current_{p,cntr} =\\
\quad |\{t \;\xi\; t \in counter\_time_{cntr}\\
\quad\quad \wedge \{t, schedule_{p,t}\} \sim counter\_subject_{cntr}\}|\\
counter\_actual_{p,cntr} =\\
\quad counter\_previous_{p,cntr} + counter\_current_{p,cntr}\\
sum\_counter\_actual_{cntr} + counter\_actual_{p,cntr}\\
sum\_counter\_max_{cntr} + counter\_max_{w,cntr}\\
\\
x = counter\_actual_{p,cntr} - counter\_max_{w,cntr}\\
(\textit{in case of absence, an adaptation to counter\_max is made}\\
\textit{corresponding to the correction in Constraint 3})\\
\\
penalty_{p,C23,max} + x \qquad\qquad IF\ (x > 0)
\end{array}\right.
\end{array}\right.\\
&\quad \forall p, (1 \leq p \leq P):\\
&\quad \left\{\begin{array}{l}
w = w_p\\
\forall cntr \in counters_w :\\
\left\{\begin{array}{l}
balance_{p,cntr} = counter\_max_{w,cntr}\\
\quad\quad *sum\_counter\_actual_{cntr}/sum\_counter\_max_{cntr}\\
large\_value_{cntr} = \lceil balance_{p,cntr} \rceil\\
small\_value_{cntr} = \lfloor balance_{p,cntr} \rfloor\\
y = counter\_actual_{p,cntr} - (large\_value_{cntr} + threshold_{cntr})\\
z = (small\_value_{cntr} - threshold_{cntr}) - counter\_actual_{p,cntr}\\
penalty_{p,C23,balance} + \left\{\begin{array}{ll} y & \qquad\qquad IF\ (y > 0)\\ z & \qquad\qquad IF\ (z > 0)\end{array}\right.
\end{array}\right.\\
penalty_{p,C23} = penalty_{p,C23,max} + penalty_{p,C23,balance}
\end{array}\right.
\end{aligned}
\tag{2.28}
$$

$$
\forall p, (1 \leq p \leq P) :
$$

$$
\begin{array}{l}
penalty_{p,C24} = 0 \\
\forall d, (1 \leq d \leq D) : \\
\left\{ \begin{array}{l}
x = |\{t \ \xi \ 1 \leq t \leq S \wedge schedule_{p,(d-1)*S+t} \neq 0\}| \\
IF \ (days\_off_{p,d} = 1 \wedge x > 0) \\
\qquad penalty_{p,C24} + \left\{ \begin{array}{ll} 1 & IF \ (extra_{p,d} = 0) \\ extra\_penalty & ELSE \end{array} \right.
\end{array} \right.
\end{array} \tag{2.29}
$$

$$
\forall p, (1 \leq p \leq P) :
$$

$$
\begin{array}{l}
penalty_{p,C25} = 0 \\
\forall t, (1 \leq t \leq T) : \\
\left\{ \begin{array}{l}
IF \ (shift\_off_{p,t} = 1 \wedge schedule_{p,t} \neq 0) \\
\qquad penalty_{p,C25} + \left\{ \begin{array}{ll} 1 & IF \ (extra\_shift_{p,t} = 0) \\ extra\_penalty & ELSE \end{array} \right.
\end{array} \right.
\end{array} \tag{2.30}
$$

People can avoid certain shifts on a particular day of the planning period and then $shift\_off_{p,t}$ equals 1. For the rest of the assignment units of the planning period, the value is 0. It is recommended to avoid conflicts with certain activities in the personal agenda by blocking small parts of the planning period. The idea is the same as in patterns, but this constraint is not cyclic. Also, the feature to attach a stronger weight to some requests (as explained with respect to Constraint 24), exists for this constraint. Those requests which require a stronger penalty have 1 in $extra\_shift_{p,t}$ The formal definition can be seen in (2.30).

Constraint 26 *Requested assignments*

There are cases in which a person wants to be assigned to a specific shift type on a certain day. The set of required assignments (corresponding to assignment unit t) for person p is denoted by 1 in $requested\_assignment_{p,t}$. For the other assignment units, the value is 0. As explained with respect to Constraint 25, there is a possibility for giving a higher or lower importance to each requested assignment. For the required assignments with a higher importance, the structure $extra\_requested\_shift_{p,t}$ has the value 1. The system applies the same weight factor and multiplication factor for violations on personal constraints as explained in the day off constraint (Constraint 24). A formal illustration is presented in (2.31).

Constraint 27 *Tutorship*

There exists a possibility of defining a tutor for a personnel member who cannot work alone: $tutor_p$. This constraint implies that the tutor has to be working

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\\
\quad penalty_{p,C26} = 0 \\
\quad \forall t, (1 \leq t \leq T): \\
\quad \left\{ \begin{array}{l} IF\ (requested\_assignment_{p,t} = 1 \wedge schedule_{p,t} \neq 0) \\ \qquad penalty_{p,C26} + \left\{ \begin{array}{ll} 1 & IF\ (extra\_requested\_shift_{p,t} = 0) \\ extra\_penalty & ELSE \end{array} \right. \end{array} \right.
\end{array}
\tag{2.31}
$$

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\\
\quad tutor = tutor_p \\
\quad penalty_{p,C27} = |\{t\ \xi\ 1 \leq t \leq T \wedge schedule_{p,t} \neq 0 \wedge cover_{tutor,t} = 0\}| \\
\\
\quad with \\
\\
\quad cover_{tutor,t} = |\{u\ \xi\ (t/S) * S \leq u \leq (t/S + 1) * S \\
\qquad \wedge\ shift\_start_{s_u} \leq shift\_start_{s_t} \wedge shift\_end_{s_u} \geq shift\_end_{s_t}\}|
\end{array}
\tag{2.32}
$$

whenever the other person is. The same concept can be used for any set of people who want to work at the same time (e.g. tutees, car-poolers, etc). ANROM does not generate a penalty when the tutor is working during a free moment of the tutee, and neither when the tutor's shift overlaps completely with the tutee's shift. When two people are required at the same time all the time, they can be set as each other's tutor. The constraint is presented in (2.32).

Constraint 28 *People not allowed to work together*

This constraint applies the same idea as the above. It only prevents the two people involved from being present in the ward at the same time. The person $not\_together_p$ should not work when p is at work. The constraint is often used in order to provide a maximal availability of people with equal skills. Other applications are those in which family members prefer to alternate their working time in order to take care of the children. All the assignments which are overlapping in time in their schedules violate this constraint which is formally presented in (2.33).

2.5 Summary

The nurse scheduling model tackled in this thesis is derived from the real-world problem of short-term personnel rostering in Belgian hospitals. In this chapter, we introduced a novel personnel rostering terminology and developed formal

$$
\begin{aligned}
&\forall p, (1 \leq p \leq P): \\
&\quad not = not\_together_p \\
&\quad penalty_{p,C28} = |\{t \;\xi\; 1 \leq t \leq T \wedge schedule_{p,t} \neq 0 \wedge overlap_{not,t} \neq 0\}| \\
&\quad with \\
&\quad overlap_{not,t} = |\{u \;\xi\; (t/S) * S \leq u \leq (t/S) * S + S \\
&\qquad\qquad\qquad \wedge (shift\_start_{s_t} \leq shift\_start_{s_u} < shift\_end_{s_t} \\
&\qquad\qquad\qquad \vee shift\_start_{s_t} < shift\_end_{s_u} \leq shift\_end_{s_t})\}|
\end{aligned}
\tag{2.33}
$$

model components, upon which the general nurse rostering framework of this thesis is based.

Hospital personnel belong to various skill categories that can be defined by the planners. There is even a possibility for certain personnel members to carry out jobs for other skills. Substitutability of personnel is much more flexible than in hierarchical organisations. This will be explained in detail in the literature overview of Chapter 3. Hard constraints, which should never be violated, make sure a sufficient number of people with the requested skills is scheduled at any time during the planning period. A very extended list of soft constraints enables the search algorithms to take personal requests, balanced workload, etc into consideration. The user definable work regulations enable a very high flexibility towards defining constraints. They group constraints on the schedules of people with the same contracts. Some hospitals in practice work with personalised work regulations in order to cope with varying preferences related to the personal life of the nurses. The cost parameters per constraint are the same for all the employees of a ward (see Chapter 4) and the evaluation of a schedule is carried out in an impartial way. Long term fairness can be obtained by an appropriate definition of the counter constraints (Constraint 23). For the evaluation of most of the other constraints, a fair consideration at the transition of consecutive planning periods is provided.

PART II, III and IV of this work focus on solving the advanced nurse rostering model introduced in this chapter.

Chapter 3

The State of the Art

3.1 Introduction

Employee scheduling has occupied personnel managers, operations researchers and computer scientists for more than 40 years. The domain often covers staffing, budgeting and short-term scheduling problems. Although these fields have variable time horizons, they are strongly interrelated. Scheduling of hospital personnel is particularly challenging because of different staffing needs on different days and shifts. Unlike most other facilities, healthcare institutions work around the clock.
Until recently, nearly all personnel scheduling problems in Belgian hospitals were solved manually. This type of planning process is now referred to as self-scheduling (Section 3.2.4). Scheduling by hand used to be a very time consuming and unrewarding task for a head nurse in hospitals or a personnel manager or foreman in other work situations. Planners had no automatic tool to test the quality of a constructed schedule. They made use of very straightforward constraints on working time and idle time in the recurring process.
The importance of a systematic approach to create good timetables is very high, especially in healthcare, where it is unacceptable not to fully support patient care needs. The jobs in hospitals are hard to automate, difficult to value, and very labour intensive. There are still great possibilities for automated nurse scheduling.

In this literature overview, we distinguish between hospital personnel scheduling and general employee scheduling (Section 3.4). Although the main purpose is the same, the problems often differ considerably and so do the approaches to solve them. The overview on hospital personnel presents the problem both from problem formulation (Section 3.2) and from solution method perspectives (Section 3.3).

In the general introduction (Chapter 1), the different scheduling approaches

for various time horizons are mentioned. Since this research concentrates on the short-term timetabling part of the problem, which is often called ‘nurse rostering’, we will only briefly discuss the management decision part. Section 3.2.2 especially stresses the difference between the topic of this research and the long-term management decision called ‘staffing’. In the next sections, ‘centralised or unit scheduling’, ‘tour scheduling’, and ‘self scheduling’ are briefly introduced. Before going into the details of approaches to solve nurse rostering problems (Section 3.3), the often very attractive cyclical schedules are presented in Section 3.2.5. Section 3.2.6 gives a comparison between the problem tackled in this thesis and other problems, with respect to particular constraints and dimensional criteria.
It is much easier for personnel members to accept an automatically obtained schedule because it is obviously unbiased. Mathematical or heuristic approaches can easily produce a number of solutions, they can report on the quality of schedules, they can try to divide the work evenly among workers, etc. One of the largest benefits of automating the personnel scheduling process is undoubtedly a very considerable time-saving for the administrative staff involved.
When discussing nurse scheduling approaches in this literature overview, we compare the complexity of the tackled problems to the problems addressed in this project.

3.2 Nurse Scheduling - Classifications

Nurse rostering or nurse scheduling covers several types of personnel scheduling. In the literature [26, 193, 204, 214], staffing and rostering tackle different decision levels. Manual or automated scheduling, cyclical or non-cyclical scheduling, are important decisions which can lead to completely different solutions. In this classification, we do not concentrate on the scheduling procedures but merely on the strategic decisions.

3.2.1 Literature overviews

Some literature reviews on personnel scheduling are discussed in this section. Most of them specifically tackle personnel scheduling in healthcare.

In his overview, **Warner** [214], (1976), distinguishes 3 major areas of research: staffing, scheduling and reallocation of nurses. 5 different criteria are defined for the scheduling part of the problem.

- coverage: the difference between the required and the scheduled number of people for a task
- quality: including fairness, work stretch length, etc (comparable to some soft constraints considered in Section 2.4)
- stability: in terms of the perception of nurses that schedules are generated consistently and that they can predict on/off days and weekends
- flexibility: how well the system can adapt to changes in the environment

- cost: resources consumed in making the decision: e.g. personnel manager's time, computer time, etc.

It is very interesting to combine these criteria for evaluating schedules since they address more than computable standards. From a general hospital scheduling point of view, it makes sense to take such a broad interpretation of cost (to generate the schedule) into account. However, it would also make sense to add other criteria (like 'personnel cost', for example) to the list. Nearly all the criteria are very hard to measure. Coverage can only be calculated provided the required number of personnel has reliably been derived. 'Stability' is a very interesting criterion, which is not evaluated in the research of this thesis. At the time Warner did his research, it was much harder to make trustworthy schedules long time in advance. This is less a problem today, although stability is not easy to calculate. The pattern constraint (Constraint 22 in Section 2.4.3), however, was introduced in ANROM exactly to give in to the personnel's request for stability.

Warner compares three scheduling approaches against these 5 criteria:

- The Traditional Approach, in which the schedules are generated by hand. The only advantage with respect to the criteria is that this policy is flexible.
- Cyclical Scheduling generally provides good schedules but it cannot easily address personal requests. The cost of such schedules is low but the method is not flexible enough to cope with changing environments.
- Computer Aided Traditional Scheduling enables a fast and more complete search for good schedules. The advantages of this approach are high with respect to all the criteria considered.

Compared to the research in this thesis, Warner's overview is more oriented towards the staffing decisions, which are also briefly discussed in Section 3.2.2.

Fries [98], (1976), presents a bibliography of early methods for personnel rostering in healthcare institutions. Many of these early approaches rely on manual procedures, following a set of arbitrary rules. They are too restricted to be applicable for problems with the complexity of the situations in Belgian hospitals.

Tien and Kamiyama [204], (1982), present a list of personnel scheduling algorithms, not restricted to healthcare. Many of them are based on arbitrary trial and error methods. Tien and Kamiyama concentrate on the hospital scheduling situation in the United States.

In their literature overview, they decompose the manpower scheduling problem into five separated stages: determination of temporal manpower requirements, total manpower requirement, recreation blocks, recreation/work schedule, and shift schedule.

Stage 1 and stage 2 are management decisions (also called the 'manpower allocation problem') which belong to the long-term staffing part of the problem (see again Section 3.2.2). Both stages consider defining hospital requirements

and selecting resources. The problem tackled in this thesis expects the results of stage 1 and 2 as input data for the scheduling problem. However, ANROM is very flexible in that it provides the possibility to define minimum and preferred personnel requirements for certain shifts or time intervals; it also allows the possibility of setting different personnel requirements for every moment of the day (not mentioned in Tien and Kamiyama's overview).
In the classification of [204] stages 3 to 5 include the entire short-term timetabling part of the problem, taking preferences and constraints on personal schedules into account. The research of this thesis tackles the three latter stages simultaneously, whereas some other methods solve one phase after another. No automatic interaction between the manpower allocation and the automated scheduling engine exists in ANROM. However, the interactive program allows for re-allocating people and rescheduling resources in a very simple way. Most papers referred to in the overview treat constraints, which are soft in ANROM, as hard constraints, e.g. precedence constraints (no changeover from later to earlier shifts on consecutive days, consecutive working days, number of free days, number of consecutive free days, free weekends per number of weekends,etc).
Tien and Kamiyama were able to classify a large number of papers in their 5 stage model, some covering a number of stages simultaneously. However, we believe that this division is often too simplified to capture all the problem specific features of certain problem areas and applications.

Sitompul and Randhawa [194], (1990) define the objective of manpower scheduling in hospitals as the objective to develop a 'systematic procedure' for allocating nurses to work shifts and work days to ensure continuous high quality service. The organisation has to provide a variable mix of nursing skills and specialisations, and to satisfy organisational scheduling policies such as work patterns. Finances are central in this paper; the goal is to reduce the personnel cost.
Sitompul and Randhawa divide the nurse scheduling approaches according to three different models: heuristic, optimising and AI-based methods. Characteristics of manpower scheduling in hospitals are:

- fluctuating demand
- human effort (which cannot be inventoried)
- critical customer convenience.

while the schedules are subject to different kinds of constraints.
It is a very interesting idea to base decisions on the analysis of human effort and on the measurement of patient satisfaction. So far, ANROM only provides an evaluation of mathematically measurable constraints in terms of working time and skill classes.
Sitompul and Randhawa define four stages in nurse scheduling:

- determine a set of feasible schedules that satisfy the constraints
- select the best schedule in terms of cost, coverage, and/or other criteria
- fine tune to accommodate changes

- make specific shift assignments.

For real-world problems tackled by other researchers, this division into four categories is often very arbitrary. It would be very difficult to categorise the problem tackled in this thesis with respect to these categories. Instead, ANROM covers the four stages simultaneously, respecting the coverage constraints. In real-world situations, the problem complexity makes it nearly impossible to satisfy the constraints (stage 1). In this research, we do not evaluate schedules in terms of personnel cost and coverage (stage 2). The personnel are (usually) hired for a longer period than the planning period; pure personnel management decisions do not belong to the short-term rostering field. Minimum coverage is a hard constraint in our approach. It is an interesting idea, however, to compare schedules with respect to the perceived quality by the personnel members (a criterion of Sitompul and Randhawa) instead of using the violation of constraints as a criterion. ANROM provides plenty of possibilities to address changes (stage 3), for example freezing parts of the schedule (see Section 5.3), interactive scheduling with immediate diagnosis of the violated soft constraints, etc. Moreover, fluctuating daily personnel demands are part of ANROM. Most of the constraints are directly related to the personnel members personally. Therefore it makes no sense to unlink specific shift assignments (stage 4) and the schedule design because assignment to different people really influences the quality of the schedule.

In the overview, Sitompul and Randhawa distinguish between optimal and heuristic scheduling techniques. They remark that people using optimising techniques work with a simplified version of real-world problems. When the problems are complex, planners apply heuristic techniques and accept non-optimal solutions. Heuristic scheduling techniques are applied to generate cyclical schedules in most cases. Other mentioned approaches are rule-based decision support systems. Since hospitals and even wards within a hospital differ largely, it is very difficult to integrate experts' knowledge in a workable system.

Sitompul and Randhawa advocate the approach of tackling staffing and rostering at the same time. They argue that separating the rostering from management decisions leads to sub-optimal schedules. In the case of Belgian hospitals, however, we believe that a general scheduling procedure would not work without significant changes in working practices, for several reasons:

- even though there is a high fluctuation in patient needs, shifting personnel around the hospital all the time is not recommended
- it would not be acceptable to take on additional staff or to lay off workers each time the personnel request does not match the available staff
- people prefer to express personal preferences with respect to work and free time, these preferences differ from month to month. Planners only grant personnel wishes if they do not deteriorate the quality of the work
- the problems are nearly all over-constrained and too complex to find an optimal solution in a reasonable amount of time, hospital people accept non-optimal schedules.

Sitompul and Randhawa believe that decision support systems can address many problems which seem unsolvable with the previously discussed techniques. A decision support system can incorporate the objectives of the hospital and the personnel by adding an interactive component to automatic schedule generation. It can make use of a database with solutions for specific problems. The characteristics of the problems in many hospitals are undefined at the outset. A drawback of many analytical formulations is that they are rigid and incomplete. They cannot be easily adapted to the changing needs of the sector. Users are also reluctant to enter a lot of data in an analytical system. Sitompul and Randhawa realise, however, that a lot of obstacles have to be taken into account in order to develop a workable decision support system.

Bradley and Martin [26], (1990), distinguish three basic manpower decisions in hospital personnel scheduling: staffing, personnel scheduling and allocation (as introduced by Warner [214]).
The first problem consists of determining the long-term number of personnel which have to be employed. People with different skills are hired to be assigned to certain wards or teams. The number of personnel is expressed in terms of FTE (full time equivalents) and is supposed to be sufficient to cover holiday periods (annual leave), training and further education. Hiring part-time nurses, allowing flexible work (or permitting the definition of different work agreements like in this work), etc permits a closer match between the personnel demands and the effective hours worked. Staffing decisions are influenced by the stochastic nature of personnel requirements and personnel capabilities. Personnel managers often deal with the problem to weigh up the pros and cons of overstaffing and understaffing. The work in this thesis starts from the results of this staffing phase. The examples of flexible work, personal education, different skills, etc can all be handled in ANROM.
The second phase in Bradley and Martin's manpower decision scheme is the conversion of expected daily work force into precise assignments: personnel rostering. It consists of determining which person works which shift on which day, matching the minimum service requirements and taking some constraints on personal schedules into account. These constraints can be of a legal nature, as a result of negotiated labour agreements. Better schedules can be generated if the problem allows for differentiation between days of the week, seasonal variations, etc. Patients needs are very hard to predict. Since schedules are generated before the actual patient needs are known, the personnel manager or scheduler has to anticipate the personnel requirements. This phase corresponds strongly to the work presented in this thesis for the Belgian hospital market. ANROM does not provide a tool for the calculation of personnel requirements, however. The daily personnel requirements (per shift or per hour) are expected as input to the system. Thanks to the high flexibility in defining minimum and preferred personnel requirements, and thanks to the possibility to handle overstaffing (by adding shifts and by choosing different planning procedures, see Section 5.6) and understaffing (by allowing overtime and reporting it), it is possible to check certain scenario's (see Section 5.2). Considering the precise shift assignment to personnel, ANROM provides, in comparison with other approaches, a very large

range of soft constraints for the personal schedules (Section 2.4).
The third phase, the allocation phase, refers to assigning people to actual work sites. It is remarkable that many researchers consider this as a separate phase (see also [194, 204], etc). Somehow, most Belgian nurses differ so much from each other (in skills, work regulation and preferences) that the allocation is better not postponed until the end of the process.
Bradley and Martin make a classification of schedules both formally and from a solution method viewpoint, just like Sitompul and Randhawa [194] do:

- exact cyclical
- heuristic cyclical
- exact non-cyclical
- heuristic non-cyclical.

Siferd and Benton [193], (1992), presented a review of factors influencing hospital staffing and scheduling in the United States. A survey among hospital managers reveals the complexity of the problem. The work first discusses the staffing history in which cost reduction became more and more important. In the second part of the work short-term personnel scheduling is discussed, in which various constraints on the nurses' schedules are taken into account. Many operations management researchers and hospital managers understand the linkages between decision making in staffing and scheduling. Hiring part time nurses and making use of 'pool' nurses, in addition to making use of overtime became more customary in this respect. It allows for more flexibility in short notice personnel demand increases. Hospital managers have to understand the stochastic and variable nature of the demand for service. Patient care can vary over a very wide range but is often arbitrarily based on the number of beds. Scheduling issues such as length of shifts, number of weekends worked, days on and off, and general flexibility of the nursing staff are intertwined with issues of the qualifications of nurses from temporary nursing services and in-house nursing pools. Many cyclical approaches are mentioned, some of them take preferences into account and others are less flexible. The researchers collected data from 31 different hospitals, and, in total, 348 wards. Decentralised manual scheduling was the most common approach, often performed in co-operation with a large number of people per ward. The questioned hospitals work with different skill classes for personnel. Personnel shortage is often solved by allowing overtime (sometimes leading to 12 or 16 hours per day services) and by using personnel from other wards. Full time work seems to be still more popular than any kind of part time work. It is also rather rare to have nurses doing both day and night shifts. In many cases the night work is done by a special group of personnel (this is not generally the case in Belgium). A large number of personnel is assigned to a set shift in practice. Most shifts have fixed start and end times (unlike the approach in ANROM), 50% of the hospitals work with three start times for day shifts on weekdays and 30% have 5 different start times. A small number of constraints, which are comparable to the constraints handled in this research can be extracted from the survey. Most hospitals seem to work with stricter rules (e.g. in 93% of the cases, there are no 'split' shifts, or, people working the same days every week). We note that ANROM allows for a much more flexible definition of constraints, work regulations, etc than any of the approaches mentioned. Some of the 'soft' constraints from this US review are:

- limit on the number of weekends worked (comparable but less flexible than Constraint 19 in Section 2.4.3, which limits the number of weekends per 4 weeks)
- set patterns for days on and off (less flexible than ANROM's Constraint 22, in which the user can define his own pattern items).

The personnel rostering problem addressed by ANROM corresponds strongly to the second scheduling phase of Siferd and Benton. However, the constraints and features in the US hospitals are less complex than the requirements we identified in Belgian hospitals.

Hung [120], (1995), collected 128 articles on nurse scheduling, from the 60's up until 1994, and presents the references as an overview. Most papers study the experience of new work week arrangements. Hung's main interest is to meet the requirements by patients and to find work arrangements that lead to high personnel satisfaction. It is mainly management and constraint issues that are taken into account, to improve the schedules. Some examples are: experiments with shifts of different length, 3-days work schedule, 4-weeks work schedule, 7-days work every two weeks, self-scheduling, etc.

There are a few **PhD dissertations** on the topic of hospital scheduling, most of them belong to the staffing domain:

- D.M. Warner: A Two Phase Model for Scheduling Nursing Personnel in a Hospital, Tulane University, New Orleans, LA, (unpublished), 1971
- D. Schneider: A Systems Analysis of Optimal Manpower Utilization in Health Maintenance Organizations, University of Florida, Gainesville, Florida, 1973
- V.M. Trivedi: Optimum Allocation of Float Nurses Using Head Nurses' Perspectives, University of Michigan, Ann Arbor, Michigan, (unpublished), 1974
- M.V. Tobon Perez: An Integrated Methodology to Solve Staffing, Scheduling and Budgetting Problems in a Nursing Department, University of Pittsburgh, 1984
- D. Lukman: An Hierarchical Approach in Schedule Formulation and Maintenance under Uncertainty, University of Pittsburgh, 1986
  In the developed rule based decision support system there is no possibility to change or add rules and the number of required personnel is not calculated. The system allows for qualitative considerations without quantitative values.
- I. Ozkarahan: A Flexible Nurse Scheduling Support System, Arizona State University, 1987
  A goal programming formulation, including both the determination of possible schedules and the assignment of individual nurses to these sched-

ules is presented in this work. Ozkarahan realises that her formulation requires a zero-one integer program much larger than anything available at that time. The work is considered a part of a large decision support system which can incorporate artificial intelligence techniques in the nurse scheduling process.

- J.M.H. Vissers: Patient flow based allocation of hospital resources, Eindhoven University of Technology, 1994
  The research focuses on the analysis, design and control of operational health care processes and systems. Special interest areas are the development of the process concept and the allocation of shared resources within a hospital setting and beyond. The personnel scheduling part of this works belongs to the staffing domain.
- J.H. Oldenkamp: Quality in Fives: On the Analysis, Operationalization and Application of Nursing Schedule Quality, Rijksuniversiteit Groningen, 1996
  The thesis describes a study of the support of scheduling nurses, in which it focusses on the consequences of nursing schedules on the performance of the nursing unit. Three parts are distinguished in this performance: the effectiveness in providing nursing care, the efficiency of a nursing unit and, the influence of a nursing schedule on the nursing unit's performance.
- U. Aickelin [4]: Genetic Algorithms for Multiple-Choice Optimisation Algorithms, European Business Management School University of Swansea, 1999
  One multiple choice problem of Aickelin's work is strongly related to the subject of this thesis. The corresponding part of [4] is called 'A Direct Genetic Algorithm Approach for Nurse Scheduling'. It is the main aim to balance feasibility and solution cost or quality within a genetic algorithm framework.
- H. Meyer auf 'm Hofe: Kombinatorische Optimiering mit Constraintverfahren - Problemlösung ohne anwendungsspezifische Suchstrategien, University of Kaiserslautern, 2000
  The work builds on the development of a constraint library for a commercial personnel planning system. Soft constraints represent different kinds of restrictions on personal schedules, going from rather strict general conditions to flexible personal requests and costs. Search algorithms for the combinatorial problems make use of special propagation procedures from the constraint library.

3.2.2 Staffing

Hospital staffing involves determining the number of personnel of the required skills in order to meet predicted requirements. In practice, several interrelated considerations make the task very complex. Factors are the organisational structure and characteristics, personnel recruitment, skill classes of the personnel, working preferences, patient needs, circumstances in particular nursing units, etc. Another significant staffing decision is to define work agreements

for part time workers, to decide whether substitution of skill classes is allowed and for which people, etc. In the real-world problems studied in this thesis, staffing, budgeting and personnel rostering takes place at different levels and for completely different time horizons. Many researchers have therefore decomposed the nurse rostering problem in phases (3 phases in [26, 214], 4 in [194], and 5 phases in [204]). Interaction between the levels is certainly necessary but in practice it would be unworkable to handle the problems simultaneously all the time, although sub-optimal short-term decisions could theoretically be avoided. Personnel are usually hired for longer periods than the short-term rostering period. Although staffing and hospital management decisions are beyond the scope of this project, a brief summary of some work is presented. This section is mainly presented to discover different kinds of input data for the short-term timetabling problem. The system discussed in this thesis will preferably tackle the most general and complete staffing decisions.
The literature overviews from Section 3.2.1, nearly all mention some of the staffing stages [26, 120, 193, 194, 204, 214].

From the 1960's on, hospital staffing has fascinated many researchers from varying fields: pure mathematics, operations research, artificial intelligence, social and life sciences. **Wolfe and Young** [221, 222], for example, presented in 1963 a model to minimise the cost for assigning nurses of different skill classes to various tasks.

Schneider and Kilpatrick [183], (1975), developed mathematical programming models to determine optimal manpower utilisation in health maintenance organisations. The problem corresponds very well to that of group practices and outpatient settings and thus differs from the nurse scheduling problem in hospitals. Three different healthcare team configurations are considered, having people with different medical skills. The analytical models combine medical care aspects and financial considerations to search for an optimal solution. The developed methods produce very good results when applied in the very early stages of setting up a health maintenance organisation.

Warner [214], (1976), presented an overview of three stages: nurse staffing, nurse scheduling (see also Section 3.2.1) and nurse reallocation. The staffing problem in this work is defined as an annual decision in which seasonal variation can be considered. It consists of determining an appropriate number of full time equivalent nurses for each skill. A methodology for the staffing decision is proposed by Warner and many hospitals accept it (subject to small adaptations). After the scheduling phase comes the third step: the reallocation of nurses. This phase is a fine-tuning of staffing and scheduling. It involves determining how float nurses are assigned to units based on nonforecastable changes or absenteeism. Among hospital schedulers, the potential benefits of this reallocation step in the process are still uncertain. However, Warner is convinced that the combination of the three stages in the end leads to a better scheduling policy.

de Vries [80], (1987), developed a framework to balance the supply and the demand for nursing care. There seems to be a range of balance between them instead of a strict equilibrium. He counts the actual capacity utilisation by dividing the actual workload per hour by the available staff per hour. Uniform criteria can be handled for all wards in the hospital. However, differences in workload between wards can be registered and result in a mechanism for co-ordination between wards.

Smith-Daniels et al. [198], (1988), present a literature overview on capacity planning in healthcare. They distinguish between capacity decisions on facility resources and on work-force resources. In these categories, two decision levels are selected: acquisition decisions and allocation decisions.
The acquisition decisions for work-force resources match the meaning of 'staffing' as it is defined in this section. The research domain of this thesis only considers the allocation decisions for work-force resources, namely the assignment of workers to days and shifts. This part is not deeply studied in [198]. Two other decisions in the group are the assignment of workers to units and to tasks. Many different strategies and approaches have been collected. Smith-Daniels et al. predict that the strict staffing and timetabling of people and other resources will all be combined in an objective for the new large scale health organisations.

Easton et al. [86], (1992), compare 12 different staffing policies during a one month period in a large hospital in the United States. They are attempting to provide adequate staffing levels to meet the patients needs and attractive work schedules to satisfy the personnel. The research is carried out at the management level, considering costs and the annual percentage of personnel turnover, reflecting dissatisfaction.
It is a common problem in hospital environments that unplanned capacity adjustements have to be made from time to time. In busy periods, unscheduled nurses will be expected to work, and in slack periods, people will work too few hours to earn their full wages.
Restrictions on shift rotation and work stretches, distribution of unattractive work, higher wages for weekend and night work, 12-hour shifts during the weekend, etc are considered. Alternative scheduling patterns (called ALTOURs in [86]) are getting more and more common in nurse scheduling environments. The patterns involve 8, 10, 12 or 16 hours shifts, combined with days off patterns and compensation days.
Easton et al. also discuss the possibility of working with 'float' nurses. Float nurses can easily solve temporarily occurring under- and overstaffing in different wards. It is not recommendable however, to ask float nurses to undertake high risk tasks that require a lot of experience, such as working in intensive care, assisting in an operating theatre, etc.
Finally, the overview presents the results of 12 different methods, it compares:

- scheduled hour utilisation

- paid hours
- workforce distribution
- the number of different 'tours' (see also Section 3.2.5)
- ...

for both unit scheduling and centralised scheduling (see Section 3.2.3). They conclude that the expected nursing expenses decrease as the scheduling alternatives increase. In order to obtain this result, the nursing requirements have to obey some rules. The research also excludes overtime, part-time work, understaffing, etc because it is very hard to formalise them.
Although this thesis provides no staffing policies, the algorithms can handle the results of any of the management decisions discussed in this section.

3.2.3 Centralised and unit scheduling

Centralised scheduling [86, 193, 198, 214] relieves head nurses from the time consuming task of constructing schedules on a very regular basis. Two major advantages of this approach are fairness to employees through consistent, objective, and impartial application of policies and opportunities for cost containment through better use of resources.
When head nurses or unit managers are given the responsibility to generate the schedules locally [5, 26, 84, 138, 142, 194], etc, it is considered an advantage that nurses get more personalised attention. Consequently, personnel members might see their schedule as a punishment or suspect the head nurse to give preferential treatments to the same people.
ANROM allows for unit scheduling, but thanks to the handling of constraints and the automatic search procedure, the major drawbacks of the method are avoided. There are plenty of possibilities to impose constraints and policies on a centralised level as well.

3.2.4 Self-scheduling

As opposed to computer decision support systems, discussed in Section 3.3, self-scheduling is a manual process. The technique is more time consuming than automatic scheduling but it has the advantage that the nurses co-operate and are asked for advice.

Generally it is performed by the personnel members themselves and coordinated by the head nurse of a ward. Nurses and other personnel collectively develop their schedules, taking coverage and time-related constraints into account. While the individual personnel members express their preferences for schedules and help setting the number of people required at any time, the personnel manager ensures that the hospital requirements are met. It is a very labour intensive procedure in which the nurses indicate their preferences and negotiate during breaks and before and after a shift. The effort for the personnel manager is reduced, his task is more supervisory. The term 'interactive scheduling' is also used in this respect (Miller [146] and Ringl and Dotson [178]).

Manual scheduling has been generally adopted in hospital wards. It is not recommended to eliminate it completely. ANROM provides algorithms to generate the schedules automatically but it also enables users to set specific preferences and adapt parameters. Resulting schedules can always manually be adapted to better satisfy people.

3.2.5 Cyclical scheduling

Cyclical scheduling concerns organisations in which each person works a cycle of n weeks. This type of schedule is common if the day is partitioned in distinct shifts and if the personnel requirements per day and per shift obey a cyclical pattern.

Cyclical schedules offer several advantages (**Warner** [214], 1976). Personnel members know their schedule a long time in advance, the same blocks are used repeatedly, the work is divided evenly, unhealthy work rotations are avoided because it is common to apply 'forward' rotation, etc (Forward rotation is met when a schedule includes no shift starting at an earlier time than a shift on the day before.) The benefits are plenty but cyclical schedules unfortunately lack generality. They cannot address (without big changes) flexible work regulations, fluctuating personnel demands and personal preferences. Also, cyclical scheduling requires a higher level decision to provide a precise number of skilled personnel members and strict personnel tasks. Working according to cyclical schedules is impossible if the problem is not very correctly stated.

Hung [119], (1991), presents a cyclical pattern for short-term nurse scheduling. He introduces 4-day workweeks with 10-hour shifts. Long shifts have plenty of benefits if the overlaps are strategically timed. Hospitals can cope with daily peak overloads, the communication between consecutive shifts is improved, and overtime is reduced. Hung allows 'downward' substitution in order to fill shortages for certain skill classes. The approach provides a permanent-shift system; this is a schedule in which nurses do not rotate. Advantages are that the people who work at the same time form a real team. There are also benefits for the social activities of the personnel members. For the scheduling problem, it implies the consideration of days on and off only, which reduces the complexity of the problem considerably.
The constraints on the algorithm are the workforce, and some constraints correspond to the soft constraints in ANROM: three free days per week (strict version of Constraint 12) and at least a number of free weekends per set of weeks (Constraint 19). The algorithm is not complex at all and can be implemented by hand. Some results are presented for problems in which the daily personnel requirements are not constant. The schedules match perfect cyclical schedules to a high extent.

Cyclical personnel rostering problems are generated using constraint satisfaction by **Muslija et al.** [154], (2000), and applied on real-world examples (see Section 3.3.2).

Tour scheduling

Tour scheduling is a special case of cyclical scheduling. The tour scheduling problem is one of simultaneously determining optimal levels for nursing resources and deploying these people among a set of feasible schedules or tours. By redefining traditional work weeks for nurses, many hospitals implemented new nurse schedules (or tours). As an example, alternative scheduling patterns (ALTOURs) were introduced in the work of Easton et al. [86] in Section 3.2.2.

Bechtold and Showalter, [16], (1987) combine the problem of staffing and scheduling personnel in a tour scheduling model. A similar example of a tour scheduling approach is presented in [14].

Although they can easily be generated and cover the personnel requirements, cyclical schedules are not flexible at all where it comes to addressing slight changes in personnel demands or in expressing personal preferences. For specific problems encountered in Belgian hospitals, cyclical scheduling is only applicable in very rare cases. Moreover, personnel seem to prefer 'ad hoc' schedules. Such schedules address fluctuating hospital demands in addition to flexibility with respect to private preferences of the personnel.

3.2.6 Problem dimensions and complexity

Tables 3.1 to 3.20 help to place the problem tackled in this thesis within the context of the group of problems studied in the literature. They present a systematic overview of some dimensional parameters and objectives collected from a range of publications.
In this thesis we are particularly concerned with the real-world nature of the problems tackled: the flexibility of defining shift types, work regulations, skill categories, the applicability in practice, etc.

When only considering the short-term rostering problem, two main goals are distinguished: coverage and time related constraints for personnel (Table 3.1). It is mandatory to provide enough assigned personnel at any time of the planning period in ANROM at the expense of violations on time related constraints. The approaches in which personnel requirements are not hard allow decisions of management level in the short-term planning. In ANROM, however, we have not let the algorithm make coverage decisions. The model is interactive enough to change the personnel requirements when necessary.
All the constraints in ANROM are modifiable and extendable but violations are allowed and explained. Those who have hard time related constraints all have fewer and less strict constraint types. In ANROM, schedules satisfying them all are not realistic.

	Hard Constraints	Soft Constraints
Coverage	ANROM: Burke et al. [36, 39], De Causmaecker and Vanden Berghe [74]	Meyer auf'm Hofe [142, 144]
	Kawanaka et al. [126]	Chen and Yeung [56]
	Warner and Prawda [216]: minimum coverage is obligatory	Warner [215]: minimum coverage can be violated on predefined days
	Meisels et al. [138]	Miller et al. [147]
	Schaerf and Meisels [182]	Okada [158], Okada and Okada [159]
	Aickelin and Dowsland [5]	
Time Related Constraints for Personnel	Berrada et al. [21]	Warner [215]
	Miller et al. [147]: feasibility set (3 constraints)	Miller et al. [147]: non-binding constraints
		ANROM: Burke et al. [34, 36, 37, 39]
		Meisels et al. [138]
		Meyer auf'm Hofe [142, 144]

Table 3.1: Hard and Soft Constraints

The objectives differ from approach to approach, which is clear from the straightforward categories in Table 3.1. In Table 3.2, a list of possible goals is presented in two different categories: the optimising and the heuristic approaches. Some of these examples include decisions of a higher level.

Examples in which all the constraints and parameters are set are quite rare and are very often pure theoretical implementations of one single problem. Most systems allow the user to adapt some predefined constraints and penalty values to their own needs (Table 3.3). Flexible software systems, which are extendible with new constraints, are much more complex to generate solutions.

Generally, the design of cyclical schedules requires more than short-term rostering decisions only (see also Section 3.2.5). However, once the requirements are set, cyclical schedules are much easier to generate than others because the search space is considerably smaller.

Most researchers allow small violations of the coverage constraints (see Table 3.5), and penalise them in a cost function. In the context of ANROM, personnel demands per shift or per time interval are expected to be satisfied. If they are not carefully defined by the users, a consistency check will indicate infeasibilities (Section 5.2). ANROM also provides several planning options to

	Optimising	Heuristic
Minimise violations on time related constraints	Warner [215]: schedules constructed with predefined patterns, the objective is to minimise $\sum_{people}$ ('aversion' for the pattern)	ANROM: Burke et al. [36], minimise $\sum_{people}$ (violations on soft constraints)
		Arthur and Ravindran [8]: minimise staff dissatisfaction by minimising the number of staff with ungranted requests
Combined coverage and time related constraints	Miller et al. [147]: nearly optimal solution generated with a mathematical algorithm	Okada [158], Okada and Okada [159]
Minimise number of employees	Alfares [6]	Easton and Mansour [85]
		Arthur and Ravindran [8]
Minimise personnel cost	Tanomaru [201]	Meyer auf'm Hofe [142] takes personnel costs into account in addition to the cost for expenditure of work
Minimise non-negative coverage	Warner and Prawda [216]: the cost for 'nursing care shortage' is minimised	
Uniform distribution of shortages and surpluses over weekdays	Berrada et al. [21]	
Minimise deviation between scheduled nurses and demand		Ozkarahan [162]: minimise nurse shortages and surpluses
		Arthur and Ravindran [8]
Minimise deviation between scheduled people and the total work capacity from the work regulations	Ozkarahan and Bailey [166]	

Table 3.2: Objectives

	Constraints	Costs and Weights
Fixed	Aickelin [4], Aickelin and Dowsland [5], Dowsland [84]	Berrada et al. [21]: weights are fixed
	Warner [215]	
Adaptable	Musa and Saxena [153]	Musa and Saxena [153]
	Warner and Prawda [216]: a few organisational constraints	Warner [215]: personal and unit wide 'aversion' for patterns
	Miller et al. [147]	Miller et al. [147]: personal 'aversion' for non-binding constraints
	Okada [158]	
User Definable	ANROM, cost parameters: Burke et al. [36]	ANROM: Burke et al. [36]
	Weil et al. [218]: generic model can cope with different legal regulations	ANROM, weights in Burke et al. [35]
	Meyer auf'm Hofe [144]	Meyer auf'm Hofe [144]
	Meisels et al. [138]	Meisels et al. [138]

Table 3.3: Flexibility

find the best coverage in every situation (Section 5.6).

Most authors restrict the applicability of their models to some simplified examples of nurse rostering, with, for example, three different shifts, short planning horizons, a limited number of possible patterns for personnel members with an identical work regulation, etc.
Skill classes are hierarchically substitutable when higher skill classes can do jobs replacing lower skilled people (see Table 3.6). In other problems, people from different skill classes can substitute each other in a user defined way. The latter approach reflects the real-world situation as it occurs in Belgian hospitals best. Among the group of people with the same skill class, some are more experienced or have better management skills to replace the head of their department.

In simplified research examples, the problems are often defined with equal constraints for all the personnel members. The assignment of schedules to people is then very arbitrary. More realistic examples take part time contracts into account and provide flexibility to define personal work agreements.
It is also shown in Table 3.7 that in case of personnel shortage, many hospitals make use of a group of 'float' nurses, to assist temporarily. In the current version of ANROM, people from other wards can assist in very busy wards. There is a procedure to evaluate time related constraints over the different wards.

Cyclical	Semi-Cyclical	Non-cyclical
Chan and Weil [55]: but flexible with respect to annual leave and unexpected events	ANROM provides the possibility to define cyclical patterns (Constraint 22) which can be superimposed on non-cyclical schedules	ANROM: Burke et al. [34, 36, 39]
Muslija et al. [154]	Warner [215]: manual preprocessing of the number of people who rotate day and night weeks	Aickelin [4], Aickelin and Dowsland [5]
Alfares [6]	Smith [195]: not all the personnel members have a rotating schedule	Meyer auf'm Hofe [142, 144]
	Chan and Weil [55]	Miller et al. [147]
		Dowsland [84]
		Kawanaka et al. [126]
		Okada [158], Okada and Okada [159]
		Schaerf and Meisels [182]

Table 3.4: Cyclical and Non-cyclical Approaches

Some approaches generate schedules which consist of days off and on. The next step in the process, the assignment of actual shifts to people is left for a head nurse to do manually. Algorithms which are developed for use in practical healthcare environments do not work with three strictly distinct shift types (see Table 3.8). The activities in hospitals are so varied that a large number of user-definable shifts is allowed. In ANROM, start and end times can even be personal as a result of a negotiation with the hospital manager for practical reasons (see Section 2.2.4). The higher the number of shift types, and the more flexible they are, the larger the search space is.

Most researchers are aware of regular changes in personnel demands (Table 3.9). This is one of the reasons why pure cyclical schedules are generally not workable. It is a part of Warner and Prawda's scheduling work [216] to predict the personnel requirements for the next few days. The personnel requirements are nearly always expressed as a number of people required per shift type or even per day. ANROM tackles the problem in a much more flexible way as a result of feedback from the users of this system in several Belgian hospitals. Not only is the number of possible shift types higher than in most problems encountered, but also the approach to compose a schedule with different combinations of shift types is exceptional.

Understaffing		Overstaffing		More Options
Allowed	Not Allowed	Allowed	Not Allowed	
Miller et al. [147]	ANROM: unless certain circumstances occur (see Section 5.2)	Miller et al. [147]	ANROM: unless certain circumstances occur (see Section 5.2)	ANROM: minimum, preferred, compromise, add hours, etc
Warner [215]	Warner and Prawda [216]	Warner and Prawda [216]		
Ozkarahan [162]	Kawanaka et al. [126]	Ozkarahan [162]		
Isken and Hancock [121]	Meyer auf'm Hofe [144]	Isken and Hancock [121]		Meyer auf'm Hofe [144] defines minimum and standard staffing levels which are treated as fuzzy constraints, there is a considerably larger penalty for understaffing than for overstaffing
	Ahmad et al. [3]	Ahmad et al. [3] only for day shifts		
	Schaerf and Meisels [182]	Okada [158], Okada and Okada [159]	Schaerf and Meisels [182]	

Table 3.5: Coverage

	Number of Skill Categories			
	1	**2**	**3**	**User Definable Number**
Skill Classes Scheduled Separately	Weil et al. [218]	Ozkarahan [162]	Musa and Saxena [153]	Schaerf and Meisels [182]
	Chen and Yeung [56]	Okada and Okada [159]	Arthur and Ravindran [8]	
	Isken and Hancock [121]		Kawanaka et al. [126]	
			Warner [215]	
			Okada [158]	
Hierarchical Substitutability			Aickelin [4], Aickelin and Dowsland [5]	Meisels et al. [138]
			Dowsland [84]	
User Definable Sustitutability			Miller et al. [147]: a subgroup of the regular nurses might be the group of those who can perform as head nurses	Warner and Prawda [216]: small overlap (substitution) between skill classes allowed, not related to people individually
				ANROM: Burke et al. [39, 34], see also Section 2.2.2

Table 3.6: Skill Categories

Short planning periods are much easier to generate schedules for. It can be an option to split the period into smaller intervals and to combine the schedules afterwards. In nearly all the cases this will lead to sub-optimal solutions. Table 3.10 gives examples of some of the realistic and theoretical approaches studied in this literature overview.

The number of personnel members in a hospital ward can vary from less than 10 to far over 100. In cases where the problem cannot be split into sub-problems, the algorithms must be powerful enough to solve problems with widely varying dimensions (Table 3.11).

Identical for all People	Miller et al. [147]: full time nurses only
	Arthur and Ravindran [8]: full time nurses only
	Weil et al. [218]: full time nurses only
	Warner and Prawda [216]: no distinction between people
	Chen and Yeung [56]
Mixed Workforce: FT & HT	Ozkarahan and Bailey [166]: different work regulations
	Musa and Saxena [153]: various part time options are possible
User Definable	ANROM; see 'work regulations' in Section 2.4.3
	Meyer auf'm Hofe [142, 144]
	Chiarandini et al. [58]
	Schaerf and Meisels [182]
	Warner [215]: people can have different 'contracted workloads'
Float Nurses	*in ANROM it is possible to let people work in more than one ward, see Section 5.3*
	Warner and Prawda [216]: generally, nurses are assigned to a unit and do not move around at zero cost; a few 'float' nurses do move around
	Meyer auf'm Hofe [142, 144] constrains the expenditure of work
	Trivedi and Warner [208]

Table 3.7: Work Regulation

Table 3.12 presents purely theoretical models in addition to algorithms which are implemented in software packages for practical use. The generic systems belong to the most flexible and complex problems for nearly all the dimensional parameters presented in this section. In fact, the software systems in the right column of Table 3.12, are among the very few that offer automatic procedures for solving widely varying nurse rostering problems.

Tables 3.13 to 3.20 present a list of time related constraints, which belong to the category of soft constraints in ANROM (Section 2.4). Some researchers set strict values for the constraints, while others let them be user definable. If we compare the table with Table 3.12, it is clear that the most flexible definitions exist in the approaches which are applicable in real scheduling situations.

3.3 Nurse Rostering Approaches

This section will present some scheduling approaches and models. The methods are grouped according to an arbitrary categorisation into optimising, heuristic, AI, etc methods.

	Strictly Distinct	**Overlapping Allowed**
1 Single Shift or no Shifts Defined: Days	Miller et al. [147]	
	Musa and Saxena [153]	
	Narasimhan [157]	
3 Different Shifts	Berrada et al. [21]; there is no rotation: the problem can be split into 3 single-shift problems	Weil et al. [218]: 8-hour day and evening shifts and 10-hour night shift
	Ahmad [3] overlap control-eren	Hung [119]: 3 slightly overlapping 10-hour shifts
	Aickelin [4]; Aickelin and Dowsland [5]; Dowsland [84]: the night shifts are scheduled separately to a certain extent so the complexity is reduced to a 2-shift problem	Okada [158], Okada and Okada [159]: the shifts have very strict start and end times, on Saturdays, a different morning shift (same start time, half the duration) is accepted
	Warner [215]	
	Trivedi and Warner [208]	
	Warner and Prawda [216]: 3 8-hour shifts per day	
	Strict Start-End Times	**Floating Intervals**
Defined Length		Bailey and Field [10]: 12-hour shifts instead of 8-hour shifts; the shifts can start at any time of the day
User Definable Shifts	Meyer auf'm Hofe [142, 144]	Tanomaru [201]
	ANROM: Burke et al. [34, 36, 39]	ANROM: Burke et al. [37], see Chapter 6
	Kawanaka [126]	Isken and Hancock [121]: variable starting times instead of 3 fixed shifts per day
	Meisels and Lusternik [140]	Brusco and Jacobs [30]
	Meisels et al. [138]	
	Schaerf and Meisels [182]	
	Chiarandini et al. [58]	

Table 3.8: Shift Types

	Days	Shifts	Hours	Minimum-preferred
Constant		Ahmad et al. [3]		
Weekdays-weekends		Kawanaka et al. [126]		
Fluctuating	Alfares [6]	ANROM: Burke et al. [34, 39]	ANROM: Burke et al. [37]	ANROM: Burke et al. [34, 37, 39]
	Miller et al. [147]	Warner [215]	Miller et al. [147]	
	Meyer auf'm Hofe [142, 144]	Warner and Prawda [216] (4 days ahead)	Meyer auf'm Hofe [142, 144]	Warner and Prawda [216]
		Aickelin [4], Aickelin and Dowsland [5], and Dowsland [84]		

Table 3.9: Personnel Demand

Since the 1960's many papers have been published on various aspects of health-care personnel scheduling. Most mathematical scheduling approaches make use of an objective function which is optimised subject to constraints. Earlier papers [147, 208, 215, 216] are nearly all examples of optimising scheduling algorithms. Researchers attempted to develop linear models for the problems. When it comes to real-world applications, pure mathematical algorithms are rarely applied. Most real-world problems are NP-complete (Tien and Kamiyama [204], for example, call it more complex than the travelling salesman problem, which is NP-complete) thus too complex for optimising algorithms. For real-world problems, in which the optimal solution is not really required, several heuristic methods have been developed [7, 24, 25, 128, 196, 197].
Although cyclical schedules are generally considered less difficult to generate, most of them are constructed with heuristic techniques. In the 1980's and later, artificial intelligence techniques for nurse scheduling (declarative approaches, constraint programming, expert systems, etc) were investigated. Some of these approaches are still relevant to today's research issues [55, 58, 144].
Many of the most recent papers (1990's and later) tackle the problem with meta-heuristic approaches.

4 days	Warner and Prawda [216]: longer periods are not trustworthy with respect to personnel demand forecast
1 week	Aickelin [4], Aickelin and Dowsland [5]
2 weeks	Musa and Saxena [153]
	Blau and Sear [25]
3 weeks	Alfares [6]
2-4 weeks	Berrada et al. [21]
	Ahmad et al. [3]: maximum 30 days
2-6 weeks	Warner [215]
	Miller et al. [147]
1 month	Smith and Wiggins [197]
1 year	Chan and Weil [55]
User Defined	ANROM: Burke et al. [34, 36, 39] usually 4 weeks

Table 3.10: Planning Period

User Definable Fixed Number	Ozkarahan and Bailey[166]
	Warner [215]: examples with 19-47 nurses
	Miller et al. [147]
	Musa and Saxena [153]: example with 11 people
	Aickelin [4], Aickelin and Dowsland [5], and Dowsland [84]
	Chan and Weil [55]: 150 people
	Ahmad et al. [3]: maximum 30 people
	ANROM: Burke et al. [34, 36, 39]
To Be Minimised	Arthur and Ravindran [8]; the model relies on an even number of nurses to function properly
	Alfares [6]
	Easton and Mansour [85]

Table 3.11: Staff Size

3.3.1 Optimising approaches: mathematical programming

These methods are especially interesting for finding optimal solutions but they are not useful to solve complex real-world problems because of their combinatorial complexity. Most researchers restrict the problem dimensions and consider a small set of constraints in their models.

Most of the mathematical approaches are based on optimising the value of a single objective function. However, a number of experiments have been carried out with goal programming or multi objective decision making.

Theoretical Approaches	Applied in Practice	
	One Single Problem	**Generic**
Ozkarahan [162]	Aickelin and Dowsland [5], Dowsland [84]	Plane (ANROM)
		Meyer auf'm Hofe [142]: ORBIS Dienstplan
Meisels and Lusternik [140]		Meisels et al. [138]: TORANIT

Table 3.12: Applicability of the approach

Linear and integer programming

Abernathy et al. [2], (1973), isolated nurse scheduling from the general staffing problem and solved it using mathematical (stochastic) programming techniques. They divide the staffing of hospitals into three decision levels: policy decisions (including the operating procedures for service centers and for the staff-control process), staff planning (including hiring, discharge, training and reallocation), and short-term scheduling of available personnel subject to the constraints imposed by the first two stages. Even the short-term scheduling in Abernathy et al.'s work involves more management decisions than the timetabling problem which is subject of this thesis. The number of people required to fulfil the -stochastically varying- personnel demands is not yet determined. Specific skills have to be considered, unlike in other work environments where the quality of the work is less dependent on the person. The solution has an iterative and a non-iterative part. It is applied to an example application.

Warner and Prawda [216], (1972), present a mixed-integer quadratic programming formulation to calculate the number of nurses from a certain skill class to do a number of shifts per day. Three non-overlapping shift types of 8 hours each are used. The goal function aims at minimising the difference between a given lower limit for the number of nurses and the variables which are the number of nurses. By adding nursing time (i.e. employing more people), the cost for personnel shortage can be reduced (never under zero, however). The minimum staffing requirements should consider the possibility to replace personnel members with different skills and the organisation's established standards (comparable to the hospital and work regulation categories in Section 2.4.3). There is no possibility to include personal preferences in the model of Warner and Prawda, all the nurses are anonymous. An excess of nursing supply for a particular skill class can absorb (at some suitable rate) the shortage of other skills. The implementation in this research (Section 2.2.2) actually builds upon the same ideas inspired by solutions in practice. A drawback of the approach is that an accurate forecast of personnel demand cannot be trustworthy for a period longer than four days. Due to the complexity of the non-linear formulation, Warner and Prawda advice to apply a

	User Definable	Set Values
Finite Capacity Constraints		
Maximum Number of Assignments	ANROM: Burke et al. [36] (Constraint 3)	Kawanaka [126]: restrict the number of free days to the total number of Saturdays, Sundays and bank holidays
		Miller et al. [147]: exact number of assignments: 10 per 14 days; hard constraint
		Meyer auf'm Hofe [144]: 10 per 14 days, a legal constraint
Overtime	ANROM: Burke et al. [36] (Constraint 8) and also Constraint 23 for other time intervals than the planning period	Meyer auf'm Hofe [142]: minimise overtime
	Meisels et al. [138]	Chen and Yeung [56]
Maximum Number of a Shift Type	ANROM: Burke et al. [36] (Constraint 11): a general constraint which can be adjusted for every shift type	
Maximum Number of Shifts per Week	ANROM: Burke et al. [36] (Constraint 12): can be set per shift type	
	Meisels et al. [138]	
	Berrada et al. [21]: hard constraint; restricted version limiting the total number of assignments per week, can also be organised with Constraint 23	

Table 3.13: Soft Constraints: Capacity

Personal Preferences	
General (unspecified)	Meyer auf'm Hofe [142]
	Warner [215]
Days Off	ANROM: Burke et al. [36] (Constraint 24): with extra information about the type of absence (holiday, illness, educational, etc)
	Miller et al. [147]: special requests can even overrule some of the feasibility constraints
	Chen and Yeung [56]
	Berrada et al. [21]
	Kawanaka [126]
	Okada [158], Okada and Okada [159]
	Meisels et al. [138]
Shifts Off	ANROM: Burke et al. [36] (Constraint 25)
Days On	ANROM: Burke et al. [36] (Constraint 22): possibility to define working days without specifying a particular shift type
	Berrada et al. [21] allow to ask for specific working days
Shift On	ANROM: Burke et al. [36] (Constraint 26)
	Meisels et al. [138]

Table 3.14: Soft Constraints: Personal Preferences

linear programming formulation with post-optimality analysis could be applied instead. Since the model does not include anything personal, they suggest that the results should be considered to be a good quality input for the problem of assigning individual nurses to specific wards and shifts.

Warner [215], (1976), elaborates on his previous formulation [216] by introducing weights or fairness levels. Warner works with shift patterns of 2 weeks length, with a fixed day and night rotation but which respond differently to some flexible constraints. The degree of freedom to construct a schedule is considerably smaller than in ANROM because the implemented constraints in Warner's approach are predefined and not to be violated, thus reducing the number of possible patterns. The constraints in Warner's work belong to the category of soft and modifiable constraints in our approach (e.g. Constraint 4, Constraint 7, Constraint 18, etc in Section 2.4.3). Some work stretches allow for extended weekends (3 or 4-days weekends), similar to our flexible weekend length (see Constraint 15).
Nurses and entire wards distribute a number of 'penalty weights' to constraints and thus to patterns and schedules. The sum of the weights for a schedule is called the 'cost'. This approach allows for a very fair evaluation of obtained schedules. Moreover, full time nurses, for example, have a higher amount of penalties to apportion than half time nurses. The sum of the costs for each personal schedule are part of the objective function. This global 'aversion' to

	User Definable	Set Values
Consecutiveness Constraints (1)		
Maximum Number of Consecutive Days	Miller et al. [147]: non-binding constraint; non-binding maximum $<$ feasibility maximum	Miller et al. [147]: feasibility constraint: feasibility maximum
	ANROM: Burke et al. [36] (Constraint 4)	Warner [215]
	Berrada et al. [21]	Chen and Yeung [56]: restricted to 6
	Okada [158], Okada and Okada [159]	Arthur and Ravindran [8]: implicitly restricted to 12 because every other weekend is free
	Weil et al. [218]	
	Warner [215]: some workstretches allow for a 3 or 4 day weekend	
Minimum Number of Consecutive Days	ANROM: Burke et al. [36] (Constraint 5)	Warner [215]: no isolated working days
	Miller et al. [147]: non-binding constraint: non-binding minimum $>$ feasibility minimum	Miller et al. [147]: feasibility constraint: feasibility minimum
		Weil et al. [218]: minimum 2; no isolated days on
	Chen and Yeung [56]: minimum 2; on/off patterns are avoided	Berrada et al. [21]: minimum 2; no on/off patterns
	Warner [215]: off/on/off days are avoided	Jaszkiewicz [122] grouping working days
Number of Consecutive Days		Aickelin [4], Aickelin and Dowsland [5]: a set number, incorporated in the shift pattern cost
Maximum Number of Consecutive Free Days	ANROM: Burke et al. [36] (Constraint 6)	Miller et al. [147]: no patterns containing 4 consecutive days off
		Jaszkiewicz [122] grouping free days

Table 3.15: Soft Constraints: Consecutiveness Constraints (1)

	User Definable	Set Values
Consecutiveness Constraints (2)		
Minimum Number of Consecutive Free Days	ANROM: Burke et al. [36] (Constraint 7)	Weil et al. [218]: minimum 2; no isolated free days
	Warner [215]: minimum 2; no isolated free days	Aickelin [4], Aickelin and Dowsland [5]: incorporated in the shift pattern cost
		Chen and Yeung [56]: minimum 2; on/off patterns are avoided
		Miller et al. [147]: no on/off/on patterns, non-binding constraint
		Berrada et al. [21]: minimum 2, another aim is to group days off
Patterns	ANROM: Burke et al. [36] (Constraint 22)	Aickelin [4], Aickelin and Dowsland [5]: rotate night shifts and weekend work
	Warner [215]: restricted number of 2 week patterns for day/night weeks and alternating free weekends; nurse specify their 'aversion' to certain patterns	Berrada et al. [21]: hard constraint; restricted constraint on weekend working patterns
		Miller et al. [147]: days on/off patterns only; less possible patterns than [215]
		Musa and Saxena [153]: nurses chose which one of 2 alternative weekends to be free
		Meyer auf'm Hofe [142]: preference of working time models (common sequences of working shifts); less flexible than Constraint 22, usually 2 weeks long
		Arthur and Ravindran [8]: 5 possible shift patterns

Table 3.16: Soft Constraints: Consecutiveness Constraints (2)

	User Definable	Set Values
Consecutiveness Constraints (3)		
Free Days after Night Shifts	ANROM: Burke et al. [36], a minimum number of consecutive free days after each overnight shift type (Constraint 14)	Okada [158], Okada and Okada [159]: appropriate interval between series of night attendances
Time Between Assignments	ANROM: Burke et al. [36] (Constraint 1): defines a minimum rest time which depends on the shift types	Kawanaka [126]: no early or day shift after a night shift; can also be formulated with Constraint 13
	Meyer auf'm Hofe [142, 144]: obligatory and preferred breaks between consecutive assignments	Warner [215]: no 'doubling back': not less than 16 hours between two assignments
Consecutive Shifts	ANROM: Burke et al. [36] (Constraint 13): to allow or forbid shift sequences on consecutive days	Kawanaka [126]: no early or day shift after a night
	Meyer auf'm Hofe [144]: defines a minimum and preferred rest time between shifts	
Sequences of Shift Types	ANROM: Burke et al. [36] (Constraint 13): flexible definition for each shift type	Meyer auf'm Hofe [144]: undesired sequences of work and free time
Mixture of Day and Night Shifts in 1 Week		Aickelin [4], Aickelin and Dowsland [5]: could be formulated with Constraint 12 in ANROM: Burke et al. [36]

Table 3.17: Soft Constraints: Consecutiveness Constraints (3)

	User Definable	Set Values
Balance the Workload		
Balance	ANROM: Burke et al. [36] (Constraint 23)	Kawanaka [126]: balance of shifts
	Meyer auf'm Hofe [144]: balance working time accounts	Okada [158], Okada and Okada [159]: balance night shifts, work on Sundays and work on bank holidays per month and per year
	Chen and Yeung [56]: some fairness measures	Okada [158], Okada and Okada [159]: night shifts evenly distributed per day of the week (per person)
	Warner [215]: nurses and entire wards distribute a limited amount of penalty weights to constraints	Chiarandini et al. [58]: long-term fair distribution of undesired shifts as well as a distribution among personnel members of violations on the number of assignments per week
	Miller et al. [147]: individual 'aversion' coefficient for violations of non-binding constraints	Jaszkiewicz [122]: distribute the shifts evenly among nurses

Table 3.18: Soft Constraints: Balance the Workload

the schedule is compensated by scheduling more nurses than strictly required.
Certain parts of the scheduling are done manually before the optimisation starts: weekends are assigned by hand and there is also a manual determination of people who will rotate. This simplifies the model. The mathematical programming algorithm consists of 2 phases: a search for feasibility and an improvement of the objective. In 20-30 % of the time the personal constraints do not allow a feasible solution (considering the number of constraints ANROM provides, a solution satisfying all the soft constraints is non-existent in nearly all cases). Therefore, the personnel are shifted around in order to move the shortages to some specified days on which the minimum coverage can be violated.
At the time of publication, the algorithm was implemented in several hospitals in the United States. A few hospitals which were used to schedule cyclical rosters did not see the benefits of the approach. Other users perceived a slight increase of the quality of the schedules and a very considerable reduction of scheduling time. Automatically generated schedules can still be fine-tuned manually, just like in the approach presented in this thesis.

	User Definable	Set Values
Weekends		
Weekends in x Weeks	ANROM: Burke et al. [36] (Constraint 19): per 4 weeks	
	Miller et al. [147]: per 4 or 6 weeks	
Complete Weekends and Extended Weekends	ANROM: Burke et al. [36] (Constraint 15); weekends can be extended with Friday and/or Monday	Berrada et al. [21]: encourage to extend the weekends
	Miller et al. [147]	
	Warner [215]: some work stretches allow for a 3 or 4 days weekend	
Compensate Weekend Work	Meyer auf'm Hofe [144]	Okada [158], Okada and Okada [159]: days off to compensate for work on Sunday and half days off to compensate for 8-hours Saturday work; preferably within that week
Number of Consecutive Weekends	Warner [215]: alternate free weekends (e.g. one every 2 or 3 weeks)	Arthur and Ravindran [8]: strictly schedule every other weekend off
	ANROM: Burke et al. [36] (Constraint 18)	
	Musa and Saxena [153]: alternate free and working weekends, individual nurses can choose which weekends are free	
	Miller et al. [147]	

Table 3.19: Soft Constraints: Weekends

	User Definable	Set Values
Other Constraints		
Preference to Work Days or Nights		Aickelin [4], Aickelin and Dowsland [5]; can be formulated with maximum number of shift types (per week (Constraint 12) and in the planning period (Constraint 11) or by defining a corresponding work regulation
Working History	ANROM: Burke et al. [36], influence on most constraint evaluations; see Section 2.4.2	Aickelin [4], Aickelin and Dowsland [5] (cost of the previous period is added to the current one, with a maximum of 100)
	Miller et al. [147]: individual 'aversion' index (historical schedule quality versus personal preferences)	
Changes in Working Shifts on Consecutive Days		Jaszkiewicz [122]: changes should be minimised
Maximum Number of Consecutive on/off/on Patterns		Miller et al. [147]: feasibility constraint
People Working Together or Not	ANROM: Burke et al. [36], (Constraint 27 and 28)	Okada [158], Okada and Okada [159]: work with as many different personnel members as possible for weekend and night work, this constraint is not as such implemented in ANROM
	Ahmad et al. [3]: nurses have the right to select preferred partners	

Table 3.20: Soft Constraints: Others

Trivedi and Warner [208], (1976), describe a branch and bound algorithm to arrange the short-term assignment of nurses from different units (called 'float' nurses) whenever there is a shortage of personnel. However, the system does not deal with time intervals but with 3 shift types, usually referred to as Early, Late and Night shift. These mathematical approaches cope with small-scale problems only.

Miller et al. [147], (1976), formulated the personnel requirements in terms of minimum and preferred number of personnel per day, just like in ANROM (Section 5.6). Everything is scheduled in terms of days on and off, however, without specifying shifts. Many other characteristics of the problem are also quite similar to the nurse rostering tackled in this thesis. It includes fewer constraints and fewer modifiable features, however. All the personnel members have full time contracts and they can belong to different skill classes. A certain amount of substitutability amongst skill classes is organised by defining subgroups which can, for example, perform as head nurses.
Staffing coverage and time related constraints with individual preferences are weighted against each other. Compared to Warner's approach [215], the number of unwanted shift patterns is much higher, thus reducing the complexity of the problem. The time related constraints (much fewer than soft constraints in ANROM) are divided in two groups: the feasibility set and the non-binding constraints. Most of the non-binding constraints are stricter versions of already existing feasibility constraints. Feasibility constraints are: the maximum number of assignments per person (Constraint 3 in Section 2.4.3), maximum/minimum number of consecutive working days, taking the previous planning period into account (Constraint 4 and 5). Non-binding constraints can be defined individually per nurse: consecutive days work (Constraint 4 and 5), consecutive free days (Constraint 6 and 7), the number of working weekends (Constraint 19), working complete weekends (Constraint 15). One constraint of Miller et al.'s approach is not implemented in this thesis. It limits the number of consecutive patterns of the type (working day - free day - working day). Apart from the objective penalty assigned to a violated constraint, an extra weight is added reflecting the nurse's personal perception of that violation. In ANROM, the only way to treat constraints individually is by defining personal work agreements; the weight factors per constraint are set for the entire ward. Miller et al. introduce even an 'aversion' index, which is a measure of how good or bad this particular nurse's schedules have been in the past. Unless for overtime (Constraint 8) and undertime (Constraint 9), this historical 'aversion' feature is not part of ANROM.
Personal requests, such as the demand for a day off (comparable to Constraint 24) are permitted. When granted, these personal preferences cancel the violated binding constraints out. They reduce the number of possible patterns, and thus the search space, even further.
A cyclic coordinate descent algorithm is applied to look for a nearly optimal solution. A comparison with a branch and bound algorithm demonstrates that the algorithm by Miller et al. requires a much lower computation time.

The obtained solutions are not always feasible. Some schedules are under- or overstaffed on certain days of the planning period.

Bailey and Field [10], (1985), formulated a general mathematical model for the nurse scheduling problem. The cost function in their definition is the sum of the cost for utilising a shift type times the number of occurances of that shift type in the schedule. A number of constraints are imposed on the schedule. Schedules minimising cumulative costs are identified, they do not take personal preferences into account. Choosing one schedule out of the set can be done manually or by a linear program. Bailey and Field reduced idle time in schedules, they propose a 12-hour scheduling period instead of a traditional 8-hour period. Their method allows shifts to begin at any time during the day.

Fuller [99], (1998), solves the same problem as Aickelin and Dowsland. The problem is described in detail in Section 3.3.4. She used XPRESS MP, a commercial integer programming software package. It was necessary to develop extra software in order to obtain results in a reasonable amount of calculation time.

Goal programming

Mathematical programming techniques are not always flexible enough to cope with relative ranking assigned to various goals. They are often restricted to optimising one single goal or criterion. Goal programming defines a target level for each criterion and relative priorities to achieve these goals. The method aims at finding a solution as close as possible to each of the targets in the order of the priorities given. The approach is also called a 'multi criteria' method (see Chapter 10, in which this approach is investigated for the problem of this thesis).

Arthur and Ravindran [8], (1981), propose a two phase goal programming heuristic for the nurse scheduling problem. They aim simultaneously at minimising (in priority sequence) staff size, the number of staff with ungranted requests or preferences, staff dissatisfaction, and the deviation between scheduled and desired staffing levels. A zero-one goal programming approach is used to assign days on and off to nurses. The schifts are heuristically assigned to the personnel members at the end of the scheduling process.
The overall problem definition is rather simple compared to the real-world problems encountered in this thesis. To reduce the complexity, Arthur and Ravindran recommend scheduling periods of one week. All the nurses have full time contracts and they belong to three -independently scheduled- skill classes. For the model to function properly, an even number of nurses is recommended. Time related constraints are restricted due to a limit of 5 shift patterns for each nurse. There is no limit on the work stretch length, however, but the system strictly schedules every other weekend off.

Musa and Saxena [153], (1984), propose an interactive heuristic procedure for solving the nurse rostering problem. Users can change the relative weights given to the goals during the scheduling process in order to take special temporal conditions into account. This is certainly a very interesting feature for real-world problems in which scheduling circumstances change regularly and are very hard to model mathematically. Within a planning period, the method introduced in Chapter 10 for the problem of this thesis, no weight changes are allowed.
In the problem tackled by Musa and Saxena, there are 11 people belonging to 3 different skill classes. Various part time options can be defined but individual preferences are not incorporated in the model. Free and working weekends alternate but the nurses can choose which one out of two weekends to work. With a two week planning period and one single shift to be scheduled, the complexity of the tackled problem remains rather low.

Ozkarahan and Bailey [166], (1988), defined three basic objective functions for their goal programming approach. The first goal is to minimise the deviation between the number of nurses scheduled and the demand, for each period of the day (called time-of-day scheduling). Just like in ANROM, the staff size is fixed and thus the hourly coverage will be maximised by this approach. The second goal minimises the deviations between the sum of days on work patterns and the size of the work force (called day-of-week scheduling). With this goal, the system tries to schedule people according to their contract or work agreement. The third goal combines the day-of-week and time-of-day scheduling problems. Ozkarahan and Bailey demonstrate the flexibility of the method by modifying the three basic objective functions.
Since the computational size of the studied problems is very large, they suggest to divide the work into two phases: one to determine schedules for the day-of-week and time-of-day schedules and one to assign people to the proposed schedules. It is a heuristic assignment of schedules, the algorithm does not assign shift times and days to individual nurses. Compared to ANROM, the latter signifies an important simplification. All the soft constraints in ANROM are related to individual personnel members.

Ozkarahan [162], (1991), presents a goal programming approach for a decision support system. The model aims at maximising the utilisation of full time personnel, minimising over- and understaffing, and minimising several kinds of personnel costs. It provides support for staffing decisions and for nurses' preferences. The problem dimensions are also very small compared to those studied in this thesis. There exist only two strictly distinct skill classes. A brief overview of one of the scheduling models is presented.
For her PhD (1987), Ozkarahan presented a formulation which would require a very large 0-1 integer program, unsolvable at that time. Later publications still present nurse scheduling models but are decomposed [166] and simplified [162, 163]. In 1995 [164] and 2000 [165], Ozkarahan has publications on allocating surgical procedures to operating rooms.

Franz et al. [96], (1989), developed a multi-objective integer linear program for health care staff working at different locations, called multi-clinic health regions. The problem involves staffing of personnel with varying skills, minimising travel costs, maximising the quality of service by considering personal preferences in addition to personnel requirements. Compared to the problem of this thesis, Franz et al. developed a much more general optimisation approach, covering decisions at higher levels than short-term personnel rostering only. Purely looking at the timetabling problem, they do not investigate it in as much detail as we do in ANROM.

Chen and Yeung [56], (1993), combine goal programming with expert systems. Assigning shift types to personnel members is carried out by the expert system part of the approach (see also Section 3.3.2). Goal programming assists in satisfying the time related constraints on personal schedules (which are comparable to the soft constraints in this project) and attempts to cover personnel demands in the meantime.

Berrada et al. [21], (1996), combine a multi-criteria approach for the nurse scheduling problem with tabu search (see also Section 3.3.3) in a very flexible tool. In order to obtain a feasible solution, a set of hard constraints, differing slightly from the hard constraints in this thesis, must be satisfied. The hard constraints are related to administrative and union contract specifications:

- weekend working patterns (comparable to Constraint 18 and to Constraint 22 in a simplified manner)
- the number of weekly working days (comparable to Constraint 12 but the constraint in ANROM [36] can distinguish between shift types which is not necessary in [21])
- uniform distribution of the surplus or shortage of nurses over weekdays (it is a hard constraint in ANROM not to have surpluses or shortages on the number of personnel at any time).

The soft constraints are treated as goals to be reached and the overall objective is to get as close as possible to these goals:

- limited number of consecutive working days to prevent long stretches (Constraint 4)
- no off/on/off situations (Constraint 7 in ANROM [36])
- the daily requirement for supervising personnel (in ANROM, users can define appropriate personnel requirements for the particular situation, personnel requirements have to be satisfied in all circumstances)
- grouping days off and extending the weekend. Days off can be grouped in ANROM with constraint 7 and weekends can be extended by Constraint 15 and the specification of longer weekends, i.e. weekends which include Friday and/or Monday (Burke et al. [36]).

The method can be extended to define additional constraints:

- specific day off (personal requests, Constraint 24)
- specific working day (corresponds to Constraint 26)
- satisfying the daily demand exactly on specific days (this constraint is not applicable to ANROM since the daily demands are hard constraints).

There is one single skill class and not more than the three regular (non-overlapping, 8-hour) shift types. Every nurse works the same shift all the time. There is no rotation. This implies that the problem can be split into 3 single-shift problems of reduced complexity. For several reasons, the planning horizon is limited to 2-4 weeks. Short-term requests of individual nurses are easier to implement and the scheduling problem remains simple.
For some real-world examples, the approach produces satisfactory results, compared with those of a commercial software package (CPLEX).
In order to really assist head nurses and save time, a user friendly software system should be developed in which it is possible to modify the weights for the different objectives.

Jaszkiewicz [122], (1997), introduces a decision support system for the nurse scheduling problem in Polish hospitals. The requirements on personal schedules are considerably less extensive than the soft constraints in this thesis and the results are much more rigid than what is required in ANROM. Working days and free days are preferably grouped, the number of shift changes on consecutive working days should be minimised, and shifts have to be divided evenly among nurses.
The problem is solved in two stages. In the first stage, a simulated annealing approach is applied in combination with a multi-objective algorithm (called Pareto-Simulated Annealing) in order to generate a set of good quality solutions. The samples are work stretches which meet the objectives in a satisfactory way. A hospital planner evaluates these results in an interactive way in the second phase.

In the mathematical approaches, exact methods are used to find a feasible set of schedules. However, the real-world problem is so complex that almost all the publications mention heuristic methods to assign work patterns to people and to take preferences into account. In order to make the optimising algorithms work properly, other researchers simplify a large number of problem characteristics.

3.3.2 AI Methods

Declarative and constraint programming

Okada and Okada [159], (1988), present a formal core method in Prolog which assists in the assignment of shifts to nurses. Skilled and unskilled nurses are the only two considered skill classes. There are three very strictly defined shift types among which two are overlapping by half an hour. On Saturdays, an extra morning shift can be scheduled, with precisely the same start time as the regular morning shift but with only half the duration.

The relative significance of various requirements can change during the planning period. Not all of the constraints have to be strictly satisfied, Okada and Okada find it even hard to define what an optimal schedule is. They distinguish between the scheduling task and the general requirements that must be fulfilled. The scheduling tasks are:

- On weekdays, assign the required number of personnel to night and evening shifts. The remaining personnel members are assigned to day shifts unless otherwise specified.
- On Saturdays, assign members to 8-hour shifts in addition to night and evening shifts. The remaining staff serve only in the morning, unless otherwise specified.
- On Sundays and bank holidays, assign members to all 3 shifts. The rest of the personnel are free unless otherwise specified.
- Give a compensatory half day off to those who worked 8 hours on Saturday.

  Give a compensatory day off for those who worked on Sunday, etc.
- Assign an evening shift to the nurses who worked a night shift the day before (respecting one of the most encountered constraints: no back-ordering).

The other goals are general requirements:

- Night shifts and Sunday/bank holiday assignments should be equally divided among staff members each month, each year.
- An interval of an appropriate number of days must be placed between night attendances.
- A specified number of skilled nurses must be in service at any shift of a given day.
- For any given nurse, night shift assignments should preferably be distributed over each day of the week.
- As for night and Sunday/bank holiday attendances, each staff member should preferably gain the experience of working with as many different members as possible.
- Individual preferences for days off or assignments to particular shifts should be taken into account as much as possible.
- Days off to compensate for Sunday attendance and half days off for Saturday are preferably given within that week.

The approach is much stricter than ANROM. There is a very systematic method to assign shifts whereas in ANROM, nearly anything can be scheduled and a penalty cost will be generated for the violated constraints. Assignments are done in a manual-like manner, following a strict procedure which is visible in the 'general requirements' above.

Okada [158], (1992), elaborated on the general scheduling procedure presented in [159] in order to develop a system which can handle varying institution-specific requirements. A declarative program, which is generated

through an interview with the user, models institution-specific information. The method represents a set of 'role sequences' as a language, in which the constraints are presented as a grammar, individual preferences are constraints on strings, etc. There are multiple criteria to evaluate the possible schedules for personnel members. By taking them all into account, the system tries to discover the 'best' schedules.
The problem dimensions are comparable to those of [159] but there is an extra skill class, consisting of two leaders. Like in ANROM, Okada's system allows for a very flexible definition of the soft constraints by the users of different types of hospitals.

Weil et al. [218], (1995), reduce the complexity of a constraint satisfaction problem by merging some constraints and by eliminating interchangeable values and thus reducing the domains. They consider problems with three shift types, a day and an evening shift of 8 hours and a 10-hour night shift. The model is generic and can cope with different legal regulations. Some time related soft constraints in ANROM are also present in this approach, be it in a very strict manner: no isolated days on (Constraint 5 with value 2), no isolated days off (Constraint 7 with value 2), a maximum work stretch (Constraint 4). However, no individual preferences are included in the model.

Darmoni et al. [69], (1995), describe a software system called Horoplan for scheduling nurses in a large hospital. They make use of knowledge based rules for the assignments, although they could also be handled with the applied constraint logic programming tool CHARME.

Meisels et al. [138], (1995), combine constraint networks and knowledge-based rules to solve employee timetabling problems. The described approach is implemented in a commercial software package, called TORANIT, which is particularly flexible with respect to defining constraints and shifts. It cannot guarantee optimal timetables because of the complexity of allowed formulations. For the constraint programming approach, constraints fall into 3 groups:

- mutual exclusion constraints: a nurse can be assigned to one job at one time (introduced as a hard constraint in ANROM in Section 2.3, although ANROM allows for flexibility, see Section 6.3)
- finite capacity of employees: a limited number of daily/weekly/monthly working hours, a limited number of night shifts per employee, etc. These constraints are represented by a counter, which is comparable to Constraint 23 in this work.
- objectives: constraint the distribution over time of employee assignments per shift (a minimum number of people of a certain skill class must be present).

The rule-based part of the system combines assignment rules and constraint rules, which are representations of human knowledge. Personal preferences for certain shifts are tackled by the assignment rules. This constraint orders

assignments by preference, e.g. rather morning shifts than late shifts. It is a constraint which is not defined in ANROM. In order to model preferences such as this, we would need to adapt several constraints. The constraint rules handle the demand for certain types of nurses or for individual nurses, in addition to personal constraints.
As a real-world example, Meisels et al. describe a team of 14-18 nurses of 3 hierarchical skill classes, which have to be assigned to 3 different shift types in weekly planning periods. They conclude that generic non-binary constraints in constraint networks and the ordering of constraints in line with their preference are very efficient to solve the constraint networks.

Meisels and Lusternik [140], (1997) also investigate constraint networks for employee timetabling problems. Just like in [138], a very flexible problem formulation should be possible. The approach consists mainly of standard constraint processing techniques, which solve randomly generated test problems. Experiments show that the domain size of the variables in the constraint networks is a critical factor to have solvable problems. Meisels and Lusternik also tested a genetic algorithm and found the same results.

Meyer auf'm Hofe [142], (1997), presents the nurse rostering problem as a hierarchical constraint satisfaction problem. His research resulted in the development of a library containing various search algorithms and constraint propagation techniques. All of this is part of a nurse scheduling system (ORBIS Dienstplan), which is tested on complex real-world problems in hospitals and fire departments.
The model provides the possibility of flexibly defining personnel requirements, provided that they are expressed in terms of shift types. It enables the use of arbitrary sets of shifts, by adding additional overlapping shift types to the tradional Early, Day and Night shift. Like in ANROM, the personnel requirements can differ from day to day and they are specified as a minimum and as a preferred size of crew attendance. Generated schedules also have to meet requirements like: legal regulations, personnel costs, flexibility with respect to the actual expenditure of work, and the consideration of special qualities. Some of the previous considerations belong to a higher decision level than the pure short time rostering which is the subject of this thesis. It is not clear, however how staffing decisions are implemented in the model. They might be also just input data, like in ANROM. Employees' contracts (work regulations in this thesis) determine to which extent requirements can be fulfilled, taking different skill categories into account. The software enables users to adapt it to their own needs. It enables the definition of work regulations and Meyer auf'm Hofe mentions qualified and experiences personnel but it is not clear how the system deals with them.
The time related constraints are less general than in ANROM, but they certainly belong to the most elaborate of the published approaches. Some specific constraints are:

- ensure minimum crew (a hard constraint in ANROM)
- prefer standard crew (scheduling more people than strictly needed to meet the requirements is considered better, this is comparable to the ‘Planning Procedures’ in Section 5.6)
- compensate work on weekends (not explicitly defined as a constraint in ANROM, but easy to simulate with the counter on weekends: Constraint 23)
- keep working time accounts in balance (can be done with the counters, Constraint 23 in ANROM)
- minimisation of overtime (Constraint 8)
- management of vacation and absence (Constraint 23)
- a minimum and preferred rest time is defined, these are constant values for all the shift types (ANROM permits rest times which depend on the shift type, there is only a minimum time defined: Constraint 1)
- working time models: preferred sequences of shifts, usually of two weeks length (comparable to Constraint 22 in ANROM but less flexible)
- undesired sequences of shifts, e.g. long chains of night shifts, work/free/work sequences, etc (instead of defining them strictly, ANROM allows for a flexible definition related to work regulations for any of the constraints)

All the above requirements can be represented by constraints in the constraint satisfaction approach. Just like in this thesis, it is impossible in practice to satisfy all the constraints; Meyer auf'm Hofe therefore mentions ‘partial’ constraints satisfaction. Consequently, requirements are treated in order of importance. It is a very complex task to generate a satisfactory schedule in practical personnel planning situations, but like in ANROM, the method is interactive and the user can alter the result of the algorithms by hand.

In [144], (2000), **Meyer auf'm Hofe**, builds on his previous research and on experiences of the software system [142], which is used in practice. The generic model is developed for use in different real-world personnel rostering settings and it corresponds therefore better to the approach in ANROM than any of the other methods discussed in this section.
Hierarchy levels and constraint weights are defined, it is not possible to satisfy all the constraints anyway. Instead of tackling it with constraint satisfaction, nurse rostering is rather constraint optimisation. Fuzzy or ‘non-crisp’ constraints are introduced in this approach as constraints which can be partially violated and partially satisfied.
A hybridisation of iterative improvement and branch and bound are used in a constraint propagation algorithm that deals with the fuzzy constraints. It is robust enough to handle varying real-world formulations of the nurse rostering problem.

Chan and Weil [55], (2000), construct timetables for a continuous work environment, not in the healthcare sector. They provide a cyclical pattern for 150 people during an entire year but the general problem characteristics are less complex than in ANROM. The requirement constraint is comparable to the hard coverage constraint in the problem tackled in this thesis. The rosters are constructed with 3 different shift types and they are flexible with respect to annual leave and other unforeseen events breaking the work cycles. When assigning cyclical timetables, some constraints have to be relaxed in order to obtain feasible schedules. Chan and Weil propose a unified model by increasing the time units from days to weeks and they solve it with constraint logic programming.

Muslija et al. [154], (2000), generate cyclical solutions for a simplified version of general workforce scheduling problems. Rotating workforce schedules are beneficial for the employees' health and satisfaction, and thus increase the work performance of the personnel.
Muslija et al. generate allowed (satisfying legal constraints) shift sequences in a one week planning period. Certain coverage levels must also be guaranteed. Important characteristics of schedules are the length of work blocks and 'optimal' weekend characteristics. Even when generating this type of rotating schedule, personal preferences and extra constraints can be implemented. The proposed method can assist in calculating good schedules very quickly but is probably too simplified to be of use in large scale healthcare environments.

Expert systems - Decision support systems

Decision support systems provide the possibility of developing user-interactive, integrated (staffing, rostering, etc) approaches to nurse scheduling problems.

Smith et al. [196], (1979), developed a 'what-if' decision support system for various sets of weights instead of providing optimal solutions. The software is interactive and allows users to assign weights to different objectives and to take personal preferences into account. The time related constraints are very basic compared to those of ANROM.

The decision support system introduced by **Ozkarahan** [162], (1991), makes use of a goal programming model (see Section 3.3.1). The problem dimensions are kept very small in order to make the approach function well. The problem of this thesis would be far too complex to be tackled with the proposed method.

Ozkarahan and Bailey [166], (1988) describe more or less the same work, but they define three objectives in the goal programming approach (see also Section 3.3.1).

Chen and Yeung [56], (1993), schedule full time nurses of a single skill class with a hybrid expert system approach. The system handles constraints

such as requested days off, maximum consecutive working days (restricted to 6), minimum consecutive free days, avoiding on/off patterns and minimising overtime. Some other fairness measures are also taken into consideration. In the meantime, the program attempts to meet minimum staff levels by applying a goal programming module. Unlike in ANROM, where minimum coverage is a hard constraint, Chen and Yeung define aspiration levels for each goal. Minimum staffing requirements on particular days can thus be relaxed.
The expert system itself is involved in assigning early, late and night shifts. Problem dimensions are very small compared to the real-world situation, all personnel members have the same work regulation and the same skills.

3.3.3 Heuristics

Real-world nurse rostering problems have a quasi uncountable number of alternative solutions. Finding the optimal solution, in terms of the value of the cost function, cannot be attained in a polynomial calculation time (= NP hard). In practice, the problem size, and the lack of knowledge about the structure of most nurse rostering problems hinders the applicability of exact optimisation methods. Many heuristics have been developed to obtain high-quality schedules for real-world problems in an acceptable computation time.
The applicability of heuristic scheduling algorithms requires a clear formulation of the hospital requirements. It is necessary to quantify the quality of different schedules in an unambiguous way. Heuristic schedulers outline a number of steps in order to generate a schedule which respects the constraints. Most heuristics are developed to generate cyclical schedules and very often they emulate the trial-and-error manner in which the planner used to construct the schedule by hand. In this section, most of the researchers apply meta-heuristics, which are more realistic attempts to solve complex real-world scheduling problems.

Smith [195], (1976), presents an interactive algorithm which helps the scheduler to construct a cyclical schedule. The algorithm takes coverage constraints and days off policies into account and it determines the number of personnel members, which is a staffing decision (Section 3.2.2). Not all the staff members can have rotating schedules, however.

Smith and Wiggins [197], (1977), developed a software system, using list-processing techniques generating non-cyclical monthly schedules for several skill categories, which allows for different kinds of part time work. Schedules are developed per person, meeting the staffing requirements by alternating days off. This is more similar to the work done in this thesis than Smith's earlier work [195]. The model incorporates a considerable number of constraints corresponding to some soft constraints in this research: patterns (Constraint 22), days off (Constraint 24), etc. It also allows the specification of the type of leave (see Section 2.4.3). Just like ANROM, the system is interactive, users can make manual changes to the generated schedules.

Blau and Sear [25], (1983), generate all possible shift patterns in a two week period and evaluate them with respect to the nurses' preferences in a first step. A cyclic descent algorithm is used in the second step in order to find an optimal overall schedule with one of the 60 best patterns for each nurse, taking over- and understaffing into account. The approach is developed for wards with three skill classes in which substitutability is hierarchical.

Blau [24], (1985), tries to equalise the distribution of unpopular work in addition to the frequency with which employees are granted requests for shifts or days. In ANROM, the latter feature does not exist.

Anzai and Miura [7], (1987), present a cyclic descent algorithm for a ward in which the personnel members are identical (with respect to skills and work regulations). They admit that their model is too simplified for practical applications.

Kostreva and Jennings [128], (1991), solve the nurse scheduling problem in two phases. Groups of feasible schedules are calculated in a first step. The groups respect the minimum staffing requirements and each individual schedule fulfils all major working constraints. In the second phase, the best possible 'aversion score', which is based on the preferences of the individual nurses, is calculated. The tackled problems are not complex, all the skill classes are scheduled independently, for example.

Schaerf and Meisels [182], (1999), present a general definition of employee timetabling problems. They define employee timetabling as the problem of assigning employees to tasks in shifts. The shifts are predefined time periods that can reside anywhere on the time axis. A general problem definition is given, with strict coverage constraints but with flexibility in time related constraints. The problem involves exactly meeting the coverage and a set of time related constraints, while trying to meet preferences in assignments.
A general local search is introduced that allows partial assignments and thus makes use of a larger search space. The paper concentrates on hill climbing algorithms for the local search. Each technique concentrates on a different part of the search space, denoting their 'steepness'. In the approach, the neighbourhood functions can include 'insert', 'delete', and 'replace' moves. The approach has been tested in theoretical environments: a hospital and a production environment.

3.3.4 Meta-heuristic scheduling

Simulated Annealing

Isken and Hancock [121], (1990), belong to the rare group of researchers who allow variable starting times instead of 3 fixed shifts per day. They formulate the problem, which is (in other respects) rather simplified (1 skill class, for example) as an integer program. Unlike in the research of this thesis (Chapter 6) under- and overstaffing are allowed but penalised. Isken and Hancock thus solve a different problem, thanks to the flexibility in personnel coverage.

Brusco and Jacobs [30], (1995), combine simulated annealing and local search to generate cyclical schedules for continuously operating organisations. Apart from hospitals, many other organisations (including telecommunications, public safety, and transportation organisations ...) face demands for labour on a continuous basis - 24 hours per day, 7 days per week. Commonly, organisations that must service continuous demand allow their workers' schedules to begin at any hour of the week. This problem, comparable to the 'floating personnel demands' of Chapter 6, is rather exceptional and makes the scheduling process far more complex than the shift type rostering.
The work concentrates merely on staffing by comparing the cost of alternative personnel scheduling options.
Brusco and Jacobs call their problem a tour scheduling problem; it determines daily shift schedules and weekly days-off assignments for employees across a specified planning horizon. One of the most common flexibility alternatives for pure tour scheduling encountered in practical applications is the use of a mixture of both full-time and part-time workers (mixed workforce). One such approach involves a problem reduction that prohibits the use of daily shift schedules that would overlap from one 24-hour period to the next (comparable to Constraint 1 but less general). The mathematical problem associated with this reduction is referred to as the 'discontinuous tour-scheduling' formulation.

Tabu Search

Berrada et al. [21], (1996) combine tabu search with a multi-objective approach (see also goal programming in Section 3.3.1). The tabu search moves are very similar to the moves applied in this research (see Section 7.3.1). The moves switch days off and working days for different people, while in ANROM, an assigned shift is moved to a person without assignment for the shift on the particular day. Since the problem dimensions of Berrada's work are much smaller than in ANROM, it is not possible to compare the results.

Dowsland [84], (1998), makes use of different neighbourhood search strategies in a tabu search algorithm. The heuristic oscillates between feasible solutions meeting the personnel requirements and schedules concentrating on the nurses' preferences. At any time of the planning period, the algorithm must

provide enough personnel with the requested qualities, while satisfying the people by granting personal requests in a fair manner. The attractiveness of work patterns differs from person to person in this work. Three different skill classes are hierarchically substitutable and the problem dimensions are rather low compared to those of this thesis. The planning horizon is one week, there are three different shifts, of which the night shifts are scheduled separately.
Rather than designing a generic, widely applicable algorithm, this work was developed to solve the personnel scheduling problem in one particular hospital (see also Aickelin and Dowsland [5] and Aickelin [4]). This explains the very good quality of the results.

Genetic Algorithms

Easton and Mansour [85], (1991), developed a distributed genetic algorithm for an employee staffing and scheduling problem called ‘tour scheduling’. The algorithm aims at minimising the number of personnel members to fulfil the demands. The fitness function represents violations of constraints and individual solutions are improved with local hill climbing operators. Personal preferences are not implemented in this work. The genetic algorithm works very well for a set of test problems.

Tanomaru [201], (1995), developed a genetic algorithm to solve a staff scheduling problem. The objective is to minimise the total wage cost, in a situation where the number of personnel is not fixed. Solutions have to meet the total workforce requirements while respecting the maximum number of individual working shifts. Overtime is allowed, however. Although the problem dimensions are very basic (one week planning horizon, low number of constraints, etc), this is one of the very few researchers who allows flexible starting times for the shifts. Solutions for the personnel are represented by 7 pairs of integers, giving the start and stop times per day.
For real-life problems, Tanomaru concludes that his heuristic mutation operators might be too time consuming.

Aickelin [4], (1999) wrote a PhD thesis on Genetic Algorithms for Multiple-Choice Optimisation Problems. One of the two problems he introduces to present his method is a nurse scheduling problem.
The same problem is tackled by **Aickelin and Dowsland** in [5], (2000), where the evolutionary approach is a complex ‘co-operative genetic algorithm’. Problem specific knowledge is used both to guide the crossover operator and a hill-climbing operator within the evolutionary algorithm.
Separate soft constraints on the personnel schedules are not evaluated. Aickelin and Dowsland determine the value or penalty of weekly schedules beforehand (this is similar to what Warner [215] does). Only a limited number of such patterns exists and instead of evaluating constraints, the values per pattern are determined and saved. Constraints which are taken into account in the

evaluation of patterns are:

- days off together or separate (Constraint 7)
- mixture of day and night shifts in a week (can be organised with Constraint 12)
- preference to work days or nights (Constraint 11 and again Constraint 12)
- preference to work certain shifts (Constraint 26)
- number of consecutive days: a set number (Constraint 4)
- rotating night shifts (Constraint 12 or Constraint 22)
- rotating weekend work (Constraint 18)
- the working history is taken into consideration: the cost of the previous schedule is added to this one, with a maximum of 100.

All, but the last constraint are also applied in ANROM. The previous planning period is taken into account in some constraints (consecutive days, rotating nights, consecutive days, etc). This corresponds to the approach in this thesis (see Section 2.4.2). Since the planning period in Aickelin's work is only one week, this constraint is really useful to provide a minimum fairness level among employees. In ANROM, which has a longer planning period, the value of the cost function is assigned only once. However, for many constraints which are not taken into account in Aickelin's work, our system takes the working history into account, for example, Constraint 8 (overtime), Constraint 9 (undertime), Constraint 23 (all kinds of different counters), ...
The work looks at a planning period of one week, schedules three different shift types (morning, late and night shift) for three different skill classes (called grades). Nurses' preferences can change, so a cyclic schedule cannot be generated to satisfy the requirements.
In the presented approach, Aickelin tries to decompose the problem in 'easier to solve' sub-problems. Night and day shifts are preferably not combined in a personnel member's weekly schedule. Night shifts can be scheduled separately to a certain extent. The skill classes are handled in a hierarchical manner in that higher qualified people can replace lower qualified people. This approach works very well for the personnel scheduling problem of a particular hospital. However, it would not be applicable to the situation in Belgian hospitals, in which it is rather unthinkable to allocate highly qualified personnel to tasks for junior nurses (see Section 2.2.2). Substitutability among skill categories is personalised in practice. Scheduling night shifts separately is not in accordance with most Belgian hospital customs.

An evolutionary approach called a population-less co-operative genetic algorithm is applied to solve another 3-shift problem by **Ahmad et al.** [3], (2000). They distinguish between hard and soft constraints. Feasible schedules satisfy the hard constraints, which are coverage constraints and personal requests for days off. The soft constraints are time-related constraints on personal schedules (a subset of the soft constraints in this thesis). A 15-days history of personal schedules is taken into account for the evaluation. It is not clear how a feasible initial schedule is created. After the initialisation, the

genetic algorithm searches solutions in the feasible region only. New schedules are generated by applying a two-point crossover on two personal schedules: the worst schedule and a randomly selected one. The search stops when a predefined number of generations is reached. Some optimisation methods have been explored: increasing the number of mates for crossover, diversification and the application of mutation and 'escape' operators.
More realistic individual cost functions are required, in addition to an evaluation procedure for the hospital planner to estimate the quality of a schedule.

Kawanaka et al. [126], (2001), developed a genetic algorithm for scheduling nurses under various constraints. Three skill classes are defined, the personnel requirements in the weekends differ from those on weekdays. In the approach, a distinction is made between 'absolute' and 'desirable' constraints. Among the absolute constraints are the minimum coverage per skill class for the constraints which are equal to the hard constraints in this work (Section 2.3). The other constraints of this category are treated as soft constraints in ANROM, they are:

- maximum number of night shifts (a particular example to be formulated by Constraint 11)
- at least one free day per week (maximum number of assignments per week in ANROM is more flexible, Constraint 12)
- the total number of free days equals the total number of Saturdays, Sundays and bank holidays (can be implemented by setting the maximum total number of assignments to the total number of days minus this value, Constraint 3)
- after a night shift, no early or late shift is allowed (can be done by strictly setting the constraint on time between shifts, Constraint 1, or by Constraint 14 if two free days have to be scheduled after a night shift).

The objective function considers weights for the desirable criteria: the balance of shifts, the granting of requested holidays, the number of night shifts assigned to unskilled, new nurses, etc.
When crossover is applied to strings by genetic operators, many absolute constraints are violated (see also Section 9.3). Shifts are exchanged in order to overcome this problem while attempting to maintain the characteristics of the parents. Compared to a conventional method, which only implements the absolute constraints in the evaluation function, the presented approach generates considerably better results.

3.4 General Personnel Scheduling

Apart from hospital and healthcare personnel scheduling, there are several other organisations which require personnel attendance 24 hours per day. Glover and McMillan [105], for example, define a general employee scheduling problem and tackle the design of shifts and the assignment of these shifts to workers as one problem.

Burns and Carter [45], (1985), present an operations research approach for employee scheduling. The main aim of the work is to design work cycles which minimise workforce requirements and fulfil a set of constraints. Some of the presented heuristics for solving the problem are proven to be optimal. The constraints treated by Burns and Carter are a subset of the constraints presented in Section 2.4, only the simplest constraints are considered. **Emmons and Fuh** [90], (1997), developed a method for optimising the workforce, and for scheduling full-time and half-time personnel while considering some soft constraints. Millar and Kiragu [145], (1998), combine all the possible shift patterns of 4 days length. They construct a network in which each node represents a feasible pattern and solve it with an algorithm based on CPLEX software.

Although doctors and surgeons are hospital personnel, they are normally not scheduled in the way nurses are. In some cases, a separate algorithm is developed for this category of healthcare personnel (e.g. Graff and Radford [110]). Locations (operating theatres, patients' addresses, for example) or available equipment

such as cars (De Causmaecker et al. [74]), specialised medical equipment (Schreuder [185]) can play an important role in the healtcare sector. Similar problems are those of magistrates in justice courts (Schreuder [184]).

Many of the employee scheduling problems are less complex than hospital scheduling, e.g. the bank sector, customs personnel, call centers, postal centers, etc. They are often solved with cyclical schedules, which are also very common in production environments. Media personnel, for example in broadcasting stations and publishing environments, cannot work according to perfect cyclical patterns either.

All kinds of commercial activities require employee scheduling: sales assistants, cash registers in stores, telephone sales, etc. Fast food restaurants form another group, in which duties are composed of very short tasks. Personnel requirements per task depend strongly on the time of the day. Good schedules are those in which people can work continuously, even though they may be undertaking different tasks all the time.

The personnel demands in many other personnel scheduling, are fluctuating less from day to day and the number of different skill categories for personnel is generally lower. However, some other personnel timetabling domains include constraints which are normally not considered in hospitals. Examples are law enforcement (Tien and Kamiyama [204]), prison staff, police (Taylor and Huxley [202]), security personnel, fire departments, service personnel (Collins and Sisley [59]), etc in which locations can play a role. Such schedules must address emergency situations in addition to routine personnel demands. In healthcare, emergencies are often covered by assigning the typical 'keep guard' duties. The personnel members which are on guard duty are called when a sudden personnel shortage occurs. Military manpower scheduling problems form a special group within this category.

Scheduling courses (Carter and Laporte [51]), exams (Burke et al. [40], Carter

and Laporte [50]) and teaching personnel in schools or universities is another related problem (Bardadym [11]). Constraints on groups of students, on the programme and on rooms with special facilities make the problems very hard to solve.
Other related problems which involve locations are audit personnel scheduling, generating preachers' timetables (Corne and Ogden [62]), etc. Courier services, telephone engineers, transportation (Bürckert et al. [32]), personnel at airline stations [31], sanitation (Tien and Kamiyama [204]), etc incorporate constraints on vehicles and routes. The problem definition differs strongly from the subject of this thesis.
There is a group of problems called 'crew scheduling' (Beasley and Cao [13], Bianco et al. [22], Caprara et al. [47], Morgado and Martins [151]): scheduling buses (Wren and Rousseau [225]), with e.g. partial driver shifts, trains, boats, airports and aircrafts (Dowling et al. [82], Gopalan and Talluri [109]), etc. A team of people, with the required skills, has to take a vehicle from one place to another. In many situations, locations cause extra difficulties when the crew stays at the destination point until another vehicle is scheduled back.

The personnel rostering techniques which are developed for this research are to a certain extent applicable to other personnel scheduling domains but in many cases, a lot of extra information is required.

Some scheduling approaches for non-healthcare personnel have been introduced in Section 3.2 and Section 3.3 already [30, 55, 85, 138, 140, 154]. A few other problems, which differ even more from the nurse rostering problem, are presented in this section.

Bailey [9], (1985), models days on and off patterns for service organisations with hourly fluctuating demands. He distinguishes 'shift scheduling' and 'days off' scheduling; the first determining the number of 8-hour shifts needed to satisfy fluctuating demands during the day and the second one to determine the weekly work patterns for personnel members. Bailey presents a decomposable linear programming approach. Constraints on the daily level, such as consecutive days off, staff size, overtime, etc belong to the shift scheduling part of the problem while hourly demand variations, changes in the cost of understaffing and overstaffing are reflected in the days off problem. The goal is to minimise the number of work patterns with the highest difference between start times over the week.

Alfares [6], (2000), developed a method to solve a real-world -non healthcare- employee timetabling problem with days on and off, which is especially suitable for personnel working in remote areas. He minimises the number of assigned employees while respecting a very strict cyclical schedule. The presented method is optimal, it makes use of the dual LP formulation but avoids the inefficient use of integer and linear programming. The application is used by an oil company to schedule workers in remote areas.

Chiarandini et al. [58], (2000), apply tabu search to generate solutions for general employee timetabling problems. Personnel members are assigned to tasks and locations in the form of shifts under flexible workload conditions. The method aims at a long-term fair distribution of undesired shifts. Different contracts involve different values for the constraints. A planning period of one week is common. The qualifications of employees enable them to fulfil certain types of tasks.
In the objective function, the following soft constraints are implemented as a weighted sum:

- preferential ability and availability: personnel members express their preferences for certain shifts and for work requiring particular skills
- flexible workload: violations on the number of assignments per week (both positive and negative) have to be evenly distributed among personnel members and over time
- fairness for special shifts (can be achieved with counters, Constraint 23)
- shift and location stability: people prefer to be assigned to the same shift and the same place in a week time (in ANROM, locations are not a subject to consider; the shift stability could be attained by setting the minimum value for consecutiveness of certain shifts, Constraint 5).

Special shifts can be defined by the user, typical examples are weekend shifts and night shifts. For test examples with 50 and 100 employees, the method generates good solutions and requires very few calculation time.

Cowling et al. [64], (2000), test several hyperheuristics on a sales summit scheduling problem. Hyperheuristics do not require much problem specific knowledge and therefore represent a very promising research direction for building more general scheduling systems. A choice function determines which low-level (local search) heuristic to choose from a set of problem-independent algorithms, under the given circumstances.
The problem consists of organising meetings between 'suppliers' and 'delegates', subject to constraints. Although this problem does not resemble the nurse scheduling problem at all, the hyperheuristic approach certainly has potential to be applied for different types of personnel scheduling problems.

3.5 Conclusions

Nurse rostering belongs to the general domain of personnel scheduling. In the literature about personnel scheduling, several planning levels are distinguished: 'staffing', which covers a long-term planning horizon, intermediate levels such as 'cyclical scheduling', in which long-term patterns are set up, and 'rostering', the short-term timetabling part.
Compared to the planning and scheduling literature for personnel, the problem tackled in this thesis is situated among the short-term rostering problems which

require management decisions as an input. ANROM does not deal with staffing nor human resources problems but provides a tool for generating personal schedules which fulfil the hospital, patient, and personal needs without altering the local customs in a ward.

The review in this chapter first compared the dimensions and the complexity of related problems with the description of ANROM. By setting up tables of comparison between several nurse rostering issues, we could clearly state that ANROM is the most flexible and complete model encountered. It provides more options for setting the goal of the planning algorithms, in addition to flexible personnel requirement formulations in any kind of planning period, user definable shift types, work regulations, skill categories, and most importantly, it has the most extensive set of modifiable time related constraints.

The second part of the review presented a discussion on different nurse rostering approaches: mathematical programming, artificial intelligence, heuristic and meta-heuristic approaches, etc. Manual schedulers tend to defend cyclical schedules, which are easier to construct but not very flexible. Automatic scheduling generally involves a decrease in time to construct the schedule and also a higher satisfaction level for the personnel. Although many advanced techniques were developed to tackle real nurse rostering problems, none of the approaches is directly suitable for solving a problem equally difficult as ANROM. The problem definitions vary widely, not uncommonly because of varying rules and habits in different countries. Most researchers develop a solution method which directly relates to their particular problem structure.

Other employee scheduling approaches were also briefly mentioned. The main goal and most of the constraints are comparable to those of the nurse rostering problem, yet solution methods which work for general employee timetabling are not flexible enough to cope with the problem defined in this thesis. Nevertheless, a set of employee timetabling problems provides restrictions that are not considered in ANROM. Examples are locations, equipment, etc.

Compared to what the research community has done, strong arguments for the importance of the work presented in this thesis are the flexibility of the approach, the applicability in practice, and the generic problem formulation.

Part II

Solution Framework

Chapter 4

The Evaluation of Solutions

4.1 Introduction

The search heuristics for solving the ANROM model introduced in Chapter 2 are driven by an evaluation function which estimates the quality of schedules. In this chapter we introduce a new method to model and evaluate the soft constraints. It is implemented as a series of modules, each corresponding to a time related constraint on personal schedules. For all the personnel members in the problem, the evaluation function sums all violations of these constraints. Users fix the parameters and set the penalty weight per unit violation of a soft constraint.

We consider a particular cost function approach that allows for a quick evaluation and is general enough to address other kinds of resource planning problems with time related constraints. It can tackle a high number of specific and modifiable constraints of a very different nature such as the constraints of Section 2.4. Simple constraints (e.g. those affecting the personal wishes of employees) and global constraints (e.g. balancing the workload among people) can be formulated easily using this method. The addition of new constraints is relatively straightforward. A major benefit of the presented approach lies in the simple evaluation of solutions, considering all the soft constraints with the same evaluation function.

Our approach deals with very complex time-related constraints as well as conditions that are related to previously planned work. Moreover, it provides clear feedback about violation of constraints.

A slightly modified version of this chapter was published as *E.K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe: Fitness Evaluation for Nurse Scheduling Problems, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, 2001, IEEE Press, 1139-1146* (Burke et al. [36]).

		From	Till
M	morning shift	06:45	14:45
L	late shift	14:30	22:00
N	night shift	22:00	07:00

Table 4.1: The shift types

4.2 The Evaluation Approach

Our method evaluates the group of very different constraints that is outlined in Section 2.4. More generally, the method also provides a technique to calculate the extent to which constraints on the schedule are violated. The main ideas of the approach, as well as some guidelines to translate real-world constraints into the model, are explained in this section.
The approach allows for the evaluation of the solution per resource (in this nurse rostering example, a resource is a person). The solution of every resource will be evaluated against a schematic representation of the constraints in order to determine the value of the evaluation function for the solution.
To facilitate the explanation of the evaluation method, we consider again the simple example consisting of a one-week planning period for a ward with 5 people, which was introduced already in Section 2.2.4 and 2.2.6. The number of shift types in use is restricted to the morning (M), late (L), and night (N) shifts presented in Table 4.1. All personnel members have the same work agreement. This implies that their personal schedules are all subject to the same set of soft constraints. Each requested shift can be assigned to any of the nurses because they all belong to the same skill category.

A schedule or solution is depicted as a two dimensional matrix, in which the rows represent the personal schedules (see Section 2.2.6). For each shift on each day of the planning period, there is a column in the matrix. Fig. 4.1 shows a personnel schedule especially constructed to demonstrate the solution representation and the evaluation approach. The schedule for the previous planning period is also presented in the table. The previous solution is used for defining the start values of the constraints to be evaluated.

4.2.1 Formal description of the evaluation method

The basic ideas of the evaluation method are presented formally in this section. The assignment units introduced in Section 2.2.6 are basic concepts in the description. Suppose there are D days in the planning period and that the problem consists of S shift types, then $D * S$ assignment units are used. The set of assignment units $(1, 2, \ldots, T)$ is denoted by AU. Fig. 4.1 is translated into an assignment unit schedule in Fig. 4.2. In this example, $D = 7$ and $S = 3$. Since the example consists of only 3 possible shift types, every day in the real-world

	Previous planning period						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
P1						M	M
P2	M	L	L				N
P3			L	L			
P4						L	L
P5	N	N	N	N	N		

	Current planning period						
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
P1	M	M	L	L	N		
P2	N		N	L	L		
P3	M	M	M	M	M	M	M
P4	M		L	N	N	N	
P5	M L	L	L	L			

Figure 4.1: Solution for the basic problem with 5 people (P1,. . . , P5) and 1 week; M, L, and N being the shift types introduced in Table 4.1

planning is represented by 3 columns in the assignment unit schedule.
We introduce *numberings* as templates that are put on each personal schedule in order to evaluate constraints in a uniform way. Instead of writing a separate algorithm for the evaluation of each constraint, we designed the numberings so that all constraints can be evaluated using a single algorithm. The evaluation of every personal schedule is performed in one go, starting from the first assignment unit for which the person is scheduled and ending at the last. For some easy constraints (e.g. Constraint 3, 4, 5, etc), very simple numberings suffice. When we come to complications involving weekends (e.g. Constraint 15, 17, 19, etc), night work (e.g. Constraint 14, 16, etc), the numberings are constructed in order to allow for sufficient abstractions from the real-world details of the problem.

Definition 1 *A* **numbering** N_i *is a mapping of the set of assignment units* AU *to a set of numbers i.e.*

$$N_i : AU \rightarrow \{-M, -M+1, \ldots, 0, 1, \ldots, M-1, U\}$$
where i=1, . . . , I and I is the total number of numberings.

M *is a positive integer and* U *is a symbol introduced to represent the assignment units for which the numbering is undefined.*

The mapping need not be into or onto, nor need it conserve the sequence. A set of 9 constraints of different nature are selected from the real-world constraints to explain how the approach covers the personnel rostering problem (see Table 4.2). Fig. 4.3 presents 3 numberings denoted by N_1, N_2, and N_3, created for the schedule presented in Fig. 4.1. The value for M in both N_1 and N_2 is

	Previous planning period																				
P1	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	*	_	_	*	_	_
P2	*	_	_	_	*	_	_	*	_	_	_	_	_	_	_	_	_	_	_	_	*
P3	_	_	_	_	_	_	_	*	_	_	*	_	_	_	_	_	_	_	_	_	_
P4	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	*	_	_	*	_
P5	_	_	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_	_	_	_	_

	Current planning period																				
P1	*	_	_	*	_	_	_	*	_	_	*	_	_	_	*	_	_	_	_	_	_
P2	_	_	*	_	_	_	_	_	*	_	*	_	_	*	_	_	_	_	_	_	_
P3	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_	*	_	_
P4	*	_	_	_	_	_	_	*	_	_	_	*	_	_	*	_	_	*	_	_	_
P5	*	*	_	_	*	_	_	*	_	_	*	_	_	_	_	_	_	_	_	_	_

Figure 4.2: Assignment unit representation of the solution in Fig. 4.1: '*' denotes that the schedule position differs from 0 (it is 'assigned') while '-' denotes that the schedule position has value 0 (it is free)

	Previous planning period																				
N_1	-7	-7	-7	-6	-6	-6	-5	-5	-5	-4	-4	-4	-3	-3	-3	-2	-2	-2	-1	-1	-1
N_2	U	U	-7	U	U	-6	U	U	-5	U	U	-4	U	U	-3	U	U	-2	U	U	-1
N_3	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	-3	-3	-3	-2	-2	-2

	Current planning period																				
N_1	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6
N_2	U	U	0	U	U	1	U	U	2	U	U	3	U	U	4	U	U	5	U	U	6
N_3	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	0	0	0	1	1	1

Figure 4.3: Numberings used for the real-world constraints of Table 4.2

7 and M is 2 in N_3. Each numbering is assigned to one or more constraints. When constraints are related to days, for example, the numbering will consist of increasing numbers for the assignment units corresponding to the days (as in numbering N_1). In Fig. 4.3, the appropriate numbers identifying the previous planning period are also shown. These numbers will be used for the initialisation of the evaluation method. In fact, the 3 presented numberings are sufficient to evaluate the 9 selected real-world soft constraints given in Table 4.2. The values in the numbering depend on the nature of the real-world constraints. Numberings provide the possibility of implementing irregular concepts such as days off, bank holidays, etc. All numberings are potentially susceptible to the same set of numbering constraints, introduced later on in Section 4.2.2. One of the main aims of the approach presented in this chapter is the reduction of the effort of implementing new real-world constraints to designing a proper

	Soft Constraints	V	C	N
3	maximum assignments	6	1	N_1
4	maximum consecutive days	4	1	N_1
5	minimum consecutive days	2	1	N_1
6	maximum consecutive free days	8	1	N_1
7	minimum consecutive free days	2	1	N_1
10	maximum assignments per day of the week	1	1	N_1
11	maximum night shifts	3	1	N_2
13	minimum consecutive night shifts	2,3,4	1	N_2
15	work full weekends	1	1	N_3

Table 4.2: Some soft constraints; the column V denotes the value, C denotes the cost parameter and N denotes the numbering associated with the constraint

numbering. The following definitions allow us to be more specific.

Definition 2 *A* **personal schedule** *for person p is a mapping*

$$schedule_p : AU \rightarrow \{assigned, free\}.$$

In the personal schedule, an *event* occurs at every assignment unit when the person is assigned to a shift (or when $schedule_p \neq 0$). At an event, each numbering associated with the personal schedule will be checked against its *constraints*. The events are generated following the order of the assignment units and will be evaluated in that sequence in the algorithm (see Fig. 4.4).

Definition 3 *For a given personal schedule $schedule_p$, an* **event** *is an assignment unit e for which $schedule_{p,e} \neq 0$. Denote by $E_{schedule_p}$ the set of all events that are induced by $schedule_p$.*

Denote by AU_N the set of assignment units for which the numbering N does not have value U (undefined). Denote by $E_{N,schedule_p}$ the set of events of $schedule_p$ which are defined for the numbering N. In other words, $E_{N,schedule_p} = AU_N \cap E_{schedule_p}$.

The basic idea of the evaluation method is to go through the set $E_{N,schedule_p}$ for each personal schedule of person p and consider the values $N(e)$ of each event in $E_{N,schedule_p}$. The number of constraint types per numbering is limited to 8 and we will now explain their meaning.

4.2.2 Numbering constraints and values

A formal description of the numbering constraints and their values is given here. A numbering constraint is a condition, which is checked against its value during or at the end of the evaluation. Numbering constraint values (between brackets in Table 4.3) are derived from the real-world constraints' values (denoted by V in Table 4.2) as presented in the left part of the columns.

Max_total is an upper limit for the number of events

	N_1	N_2	N_3	
Constraints	**(Real-world constraint)**			**Penalty costs**
max_total	6 (3) 1	3 (11) 1		cost_max_total
min_total				cost_min_total
max_pert	1 (10) 1			cost_max_pert
min_pert				cost_min_pert
max_between	8 (6) 1			cost_max_between
min_between	2 (1) 1			cost_min_between
max_consecutive	4 (4) 1			cost_max_consecutive
min_consecutive	2 (5) 1	2 (13) 1	2 (15) 1	cost_min_consecutive

Table 4.3: Constraint values and penalty costs for each of the 3 numberings in Fig. 4.3; the numbers between brackets refer to the corresponding real-world constraints of Table 4.2

$$\#E_{N,schedule_p} \leq max\_total$$

The real-world constraints given in Table 4.2 are translated into 8 numbering constraints presented in Table 4.3. In the real-world constraints 3 and 11 presented in Table 4.2 max_total has value 6 and 3 (see Table 4.3). The other real-world constraints in Table 4.2 are not evaluated with the max_total constraint.

Min_total is a lower limit for the number of events

$$\#E_{N,schedule_p} \geq min\_total$$

In fact, an evaluation of the numbering constraint min_total is not required for the selected elementary set of constraints in Table 4.2. The real-world constraints on undertime (Constraint 9), patterns (Constraint 22), and required assignments (Constraint 26) in Section 2.4 make use of min_total for their evaluation.

Max_pert is an array of size M representing for each number in the numbering the maximum number of events that can be mapped to it.

Min_pert is an array of size M which is similar to max_pert except that it represents a minimum instead of a maximum. None of the constraints in Table 4.2 makes use of the min_pert constraint. In more realistic rostering problems, however, the constraint is used to evaluate real-world constraints such as patterns (Constraint 22), identical shifts in weekends (Constraint 17), and balancing the workload (Constraint 23) in Section 2.4.

For convenience, we introduce a new operator:

Definition 4 *Two numbers a and b (where $a \leq b$) are said to be* **consecutive** *with respect to a numbering N if and only if for every number m in $\{a, \ldots, b\}$ the numbering N maps an event in $E_{N,schedule_p}$ to m.*

This allows us to introduce four additional constraints:

Max_consecutive is the maximum number of consecutive events. The con-

straint max_consecutive is used in numbering N_1 for the real-world constraint 4 of Table 4.2 and its value is 4.
Min_consecutive is the minimum number of consecutive events. 3 different real-world constraints are evaluated with this constraint, and so 3 different numberings are required. Constraints 5, 13, and 15 use numberings N_1, N_2 and N_3 respectively. In the example, the value of min_consecutive is 2 for all the numberings.
Max_between is the maximum gap between two non-consecutive events a and b i.e. $b - a \leq max\_between$. Constraint 6 is evaluated with max_between. It has value 8.
Min_between is the minimum gap between two non-consecutive events a and b i.e. $b-a \geq min\_between$. For Constraint 7 in Table 4.2 the value of min_between is 2 and the numbering is N_1. In Table 4.4 a list of all the numbering constraints for each of the soft constraints introduced in Section 2.4, is presented.

4.2.3 Counters

A counter is a variable that is initiated at the beginning of the evaluation and changes during the procedure in order to calculate the constraint violations. Some real-world constraints can be handled with a single counter, for others a counter array is required. The counters will be adjusted during the course of the evaluation and checked against the values of the constraints. The real-world constraints described in Section 2.4 can be evaluated using not more than 8 different constraint types. The counters are: total, consecutive, pert, and last; respectively representing the total number of events for the numbering, the number of consecutive events, the number of events per value in the numbering and the number of the last evaluated event. The *pert* counters are used to count certain scheduling features for different time periods (e.g. count night shifts in weekends). Among the real-world constraints in Table 4.2, only the 7th uses max_pert as a constraint. For every value of the numbering N_1, max_pert is set to 1.
The constraints introduced above can all be evaluated by one single algorithm. In Section 4.3 we demonstrate, using the schedule of Fig. 4.1 and the real-world constraints presented in Table 4.3 as an example, how the evaluation approach is implemented.

4.2.4 Cost parameters

The evaluation function is completely modifiable. The approach allows for the establishment of weight factors adapted to the needs of the schedulers. Any violation of a constraint will contribute to the overall value of the cost function in proportion to the weight factor.
In the evaluation approach, weight factors are denoted by the term *cost_* followed by the type of the numbering constraint as defined in Section 4.2.1. For the demonstration, all cost parameters for the real-world constraints of Table 4.2 (denoted by C) are set to 1, as presented in the right part of Table 4.3.

RWC	max_total	min_total	max_pert	min_pert	max_between	min_between	max_cons	min_cons
1						x		
2	x							
3	x							
4							x	
5								x
6					x			
7						x		
8	x							
9		x						
10			x					
11			x					
12			x					
13							x	x
14						x		
15								x
16						x		
17								x
18							x	
19	x							
20	x							
21						x		
22	x	x	x	x				
23	x	x						
24	x							
25	x							
26		x						
27	x							
28	x							

Table 4.4: Numbering constraints used for the evaluation of the soft constraints in ANROM (in column RWC)

4.2.5 Changing work regulation or skill categories

During the planning period, the work regulation or skill category of personnel members is not necessarily constant. People can, for example, change from full time to part time contracts, at a certain date in the middle of the planning period. We introduce $w_{p,d}$ for the work regulation of person p at day d ($1 \leq d \leq D$). The possible change of work regulations implies that the constraints to be evaluated change abruptly. The user of ANROM still wants to know exactly when violations of constraints happen and therefore we have introduced a 'split' evaluation, in which the constraint values at the end of the first work contract provide the input for the constraints of the second one. The idea is similar to the procedure presented in Fig. 4.5, where the impact of a previously scheduled planning period on the current evaluation is explained. When considering the previous planning period, however, the initial values have to be calculated only once, while in this case, the schedule changes constantly and so do the constraint values.

The case in which a person receives promotion or changes his/her skill category can be treated equally. Also, the set of alternative skill categories can be updated within the planning period. The simplest situation occurs when a personnel member acquires or looses an alternative skill class because this will only affect one constraint. For every person p, $q_{p,d}$ denotes the skill category, and $QA_{p,d}$ denotes the set of alternative skill categories at day d ($1 \leq d \leq D$). As is explained in Section 2.2, the subscript d is dropped in the problem description whenever a person has an unchanged work regulation, skill category and set of alternative skills during the entire planning period.

4.2.6 Evaluation mechanism

Every personal schedule is evaluated separately. The procedure can be presented schematically as in Fig. 4.4. An evaluation starts with the initialisation of the numbering counters, for which the schedule of the previous planning period is used. We call $previous_p$ the personal schedule of the previous planning period for person p.

The initialisation procedure is described in Fig. 4.5. The start values are only important for the constraint types max_consecutive, min_consecutive, max_between and min_between. After the initialisation phase, the evaluation will go from one event to another adjusting the counters for all the numberings. The method is schematically presented in Fig. 4.6. Suppose the number corresponding to the event is n (different from U) then the value of total will be increased by 1, as will the value of pert[n]. Depending on the relationship between n and the number of the last event encountered, either consecutive will be increased by 1 or an intermediate evaluation of the 'between' and 'consecutive' counters will be performed. The details of this intermediate evaluation are presented in Fig. 4.6. When the evaluation has reached the last event in the planning period, a final evaluation of the constraints is required (Fig. 4.7). This provides the values of all the violations of the constraints for the schedule. Since

Copy the start values (Fig. 4.5) into the numberings.

Find first element e of AU belonging to $E_{schedule_p}$.

DO

- *Update all the numbering counters in the intermediate evaluation (Fig. 4.6).*
- *Find next element e of AU belonging to* $E_{schedule_p}$.

WHILE the end of the solution is not reached.

Perform a final evaluation of all numbering constraints (Fig. 4.7).

Communicate the results to an output device.

Figure 4.4: Overview of the evaluation of a schedule

the violation values are stored in appropriate data structures, called *penalty_x* where x is one of the numbering constraints (see Fig. 4.7), the quality of the schedule in terms of each particular constraint can easily be traced back. This approach reduces the difficulty of defining proper cost parameters considerably because the impact of changes to the parameters is immediately visible in the value of all the constraints' violations. In Section 4.3 the whole procedure is executed on a particular personal schedule from the example in Fig. 4.1.

4.3 Example

4.3.1 Demonstration of the method

In this section, we demonstrate how the soft constraints from Section 2.4 are formulated and evaluated using this approach. All the data is presented in Tables 4.1 to 4.3 in Section 4.2. We will follow the entire evaluation procedure of Fig. 4.4 for two different personnel members. The results for these people are presented in Tables 4.5 and 4.6. The left part of the table shows the initial values for the counters. The values can be stored in memory since they will not change when evaluating new solutions. Some counters in the numbering do not reflect any real-world constraints in this particular example (see empty fields in Table 4.3). They hardly affect the evaluation method; their impact on calculation time and memory is very low indeed. For each numbering, only one value for each counter and penalty is stored in the memory.
In the left part of the tables, the initial values for the counters are given. They need not be recalculated when the evaluation of another than the current schedule is required. From left to right in the table, the chronological updating of the counters is illustrated. Due to the different number of assignments to the personnel members, there is a different number of events in the numbering evaluation. Note that the values of the counters can change each time a new event

Initial			**P1**	**Event 1**			**Event 2**			**Event 3**			**Event 4**			**Event 5**		
N_1	N_2	N_3		N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3	N_1	N_2	N_3
-1	U	-2	last	0	U	-2	1	U	-2	2	U	-2	3	U	-2	4	4	-2
0	0	0	total	1	0	0	2	0	0	3	0	0	4	0	0	5	1	0
2	0	2	consecutive	3	0	2	4	0	2	5	0	2	6	0	2	7	1	2
0	0	0	pert[0]	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[1]	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[2]	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[3]	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	pert[4]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	pert[5]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	pert[6]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_between				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_between				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
penalty_min_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 4.5: Evaluation procedure for 1st person's schedule (P1)

```
FOR i=1,...,I (I is the total number of numberings)
    numbering_initialised=False
    consecutive=0
    last_nr=N_i(t); t: the assignment unit for the smallest value of N_i
    max_nr=N_i(t); t: the assignment unit for the highest value of N_i
    Find last element e of AU belonging to E_previous_p
    DO
        nr = N_i(e)
        nr = nr − max_nr − 1
        IF (nr ≠ U)
            IF (nr = last_nr − 1)
                consecutive = consecutive + 1
            ELSE IF (nr < last_nr − 1)
                numbering_initialised=True
            last_nr=nr
        Find previous element e of AU belonging to E_previous_p.
    WHILE (¬numbering_initialised)
i=i+1
Save the results.
```

Figure 4.5: Pseudo code for the initialisation algorithm

is found. The 2 personal solutions we choose for this explanation (P1 and P4) consist of 5 assignments (events) in the planning period. For P1, no penalty is created during the intermediate evaluation phase (see Fig. 4.6). After the last event was found, the evaluation goes to the final evaluation phase of the algorithm (see Fig. 4.7). In this part of the algorithm, a penalty is created for the max_consecutive constraint in numbering N_1 and one for min_consecutive in N_2. Translating the results back into feedback for the planners, the penalties for P1 indicate that the maximum number of assignments on consecutive days is violated with three. In addition, the number of consecutive night shifts is one less than required.
Following the evaluation for the P4 solution step by step, the second event already creates a penalty in the intermediate evaluation phase of Fig. 4.6. The min_between constraint of numbering N_1 is violated when going from event 1 (where the corresponding number is 0) to event 2 (the corresponding number is 2). Since the ($nr = last\_nr + 1$) condition of Fig. 4.6 is not fulfilled, the intermediate evaluations are executed. One extra violation occurs during the final evaluation because the min_consecutive constraint of N_2 is violated. In terms of real-world constraints, this schedule violates the minimum number of consecutive free days with one and the constraint on scheduling complete week-

```
FOR i=1,...,I
    nr = N_i(e)
    IF (nr ≠ U)
        total = total + 1
        IF (nr = last_nr + 1)
            consecutive = consecutive + 1
        ELSE IF (nr > last_nr + 1)
            IF (consecutive < min_consecutive)
                penalty_min_consecutive = penalty_min_consecutive +
                cost_min_consecutive * (min_consecutive − consecutive)
            IF (consecutive > max_consecutive)
                penalty_max_consecutive = penalty_max_consecutive +
                cost_max_consecutive * (max_consecutive − consecutive)
            IF (nr − last_nr − 1 < min_between)
                penalty_min_between = penalty_min_between +
                cost_min_between * (min_between − (nr − last_nr − 1))
            IF (nr − last_nr − 1 > max_between)
                penalty_max_between = penalty_max_between +
                cost_max_between * ((nr − last_nr − 1) − max_between)
            consecutive = 1
        pert[nr] = pert[nr] + 1
        last_nr = nr
i=i+1
```

Figure 4.6: Pseudo code for the intermediate evaluation

ends.
This explanatory feedback is especially interesting for practical use when the result of calculations is presented to the planners in practice. Considering the high number of implemented soft constraints in ANROM (Section 2.4), it is nearly impossible to have an estimation of the quality of a schedule on sight. The detailed interpretation enabled by this modular evaluation approach is essential for the practical use of the model.

4.3.2 Real-world issues

The following figures give an idea of the importance of a quick evaluation scheme for the solutions of the nurse rostering problem tackled with ANROM. In a hospital, all wards (the number of wards can be hundreds) have access to the planning system. A modest ward consists of 20 people, has 6 different shift types and about 30 different soft constraints per personal schedule. The length of the most encountered planning period is 4 weeks. One iteration in the evolutionary

Initial			**P4**	**Event 1**			**Event 2**			**Event 3**			**Event 4**			**Event 5**		
N1	N2	N3		N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3
-1	U	-2	last	0	U	-2	2	U	-2	3	3	-2	4	4	-2	5	5	0
0	0	0	total	1	0	0	2	0	0	3	1	0	4	2	0	5	3	1
2	0	2	consecutive	3	0	2	1	0	2	2	1	2	3	2	2	4	3	1
0	0	0	pert[0]	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	pert[2]	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0
0	0	0	pert[3]	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0
0	0	0	pert[4]	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
0	0	0	pert[5]	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	pert[6]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_total				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_pert				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_max_between				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_between				0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
penalty_max_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
penalty_min_consecutive				0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 4.6: Evaluation procedure for 4th person's schedule (P4)

```
FOR i=1,...,I
    IF (total > max_total)
        penalty_max_total =
        penalty_max_total + cost_max_total * (total − max_total)
    IF (total < min_total)
        penalty_min_total =
        penalty_min_total + cost_min_total * (min_total − total)
    IF (consecutive > max_consecutive)
        penalty_max_consecutive = penalty_max_consecutive +
        cost_max_consecutive * (consecutive − max_consecutive)
    IF (consecutive < min_consecutive)
        penalty_min_consecutive = penalty_min_consecutive +
        cost_min_consecutive * (min_consecutive − consecutive)
    ∀t ξ N_i maps an event to t
        IF (pert[t] > max_pert[t])
            penalty_max_pert =
            penalty_max_pert + cost_max_pert * (pert[t] − max_pert[t])
        IF (pert[t] < min_pert[t])
            penalty_min_pert =
            penalty_min_pert + cost_min_pert * (min_pert[t] − pert[t])
    IF (N_i(1) + N_i(AU_{N_i}) − N_i(last_event) > max_between
        penalty_max_between = penalty_max_between + cost_max_between *
        (N_i(1) + N_i(AU_{N_i}) − N_i(last_event) − max_between)
i=i+1
```

Figure 4.7: Pseudo code for the final evaluation of the algorithm

algorithms described in Chapter 7-9 requires approximately 100 evaluations of the cost function. On an IBM RS6000, it takes about one minute to perform 300 iterations.

The evaluation approach introduced in this chapter is perfectly suitable for personnel scheduling problems (such as those presented in Section 3.4) and even more generally, for other timetabling and scheduling problems such as (Burke et al. [42]) and (Paechter et al. [167]), especially when evolutionary algorithms are being applied.

4.4 Conclusion

The new evaluation approach for the nurse rostering problem has proven to be very effective. It makes use of a simple evaluation function, independent of the number and character of the constraints imposed on the system. Although the problem is complex, the current approach enables a fast evaluation of solutions. This is crucial for the high number of cycles that is typical for meta-heuristic algorithms.

We introduced the concept of numberings, which form a generic way of evaluating constraints. They also provide a very structural technique for modifying existing constraints and for handling new constraints. The modular nature of the approach allows the evaluation method to provide some feedback for the planners and thus assist with the interpretation of the quality of a schedule.

This novel evaluation approach is also applicable to other application domains.

Chapter 5

Planning Procedures

5.1 Introduction

Before describing meta-heuristic approaches to nurse rostering problems, we introduce a few pre- and post-planning heuristics, which can be combined with the algorithms of Chapter 7-10. They are presented as separate planning options in order to isolate the meta-heuristics from typical objectives when the ANROM model is applied in practice. Fig. 5.1 schematically demonstrates the order in which these planning procedures (presented in bold) appear in the total planning process. Experiments with real-world data revealed the benefit of dividing the rostering problem in sub-problems and thus, we opted in ANROM for executing all the planning procedures per skill category. The large box in Fig. 5.1 indicates which part of the solution framework is executed per group of people belonging to the same skill category. The meta-heuristics are not specified in this figure but any search algorithm maintaining the coverage can be plugged in. Boxes presented on the same level denote exclusive options, e.g. 'accept' and 'repair' when the consistency check discovers infeasibilities.

Nearly all the data in the nurse scheduling model ANROM are modifiable. This enables a large number of very diverse teams to cover the hospital needs, while the system offers a general approach to solve the problems. However, a drawback in practice is the danger of defining the problem in such a way that it causes difficulties for the algorithms to find good quality or even feasible solutions. A few modules that have been built to especially overcome particular problems, are also introduced in this chapter.
The relaxation procedures introduced in this chapter are the subject of the following paper: *P. De Causmaecker and G. Vanden Berghe: Relaxation of Coverage Constraints in Hospital Personnel Rostering, accepted for publication in Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, Gent, 2002* (De Causmaecker and Vanden Berghe [78]).
In Section 5.2, a consistency check algorithm for assisting the planner in repairing some infeasibilities in the data is introduced. Certain circumstances require

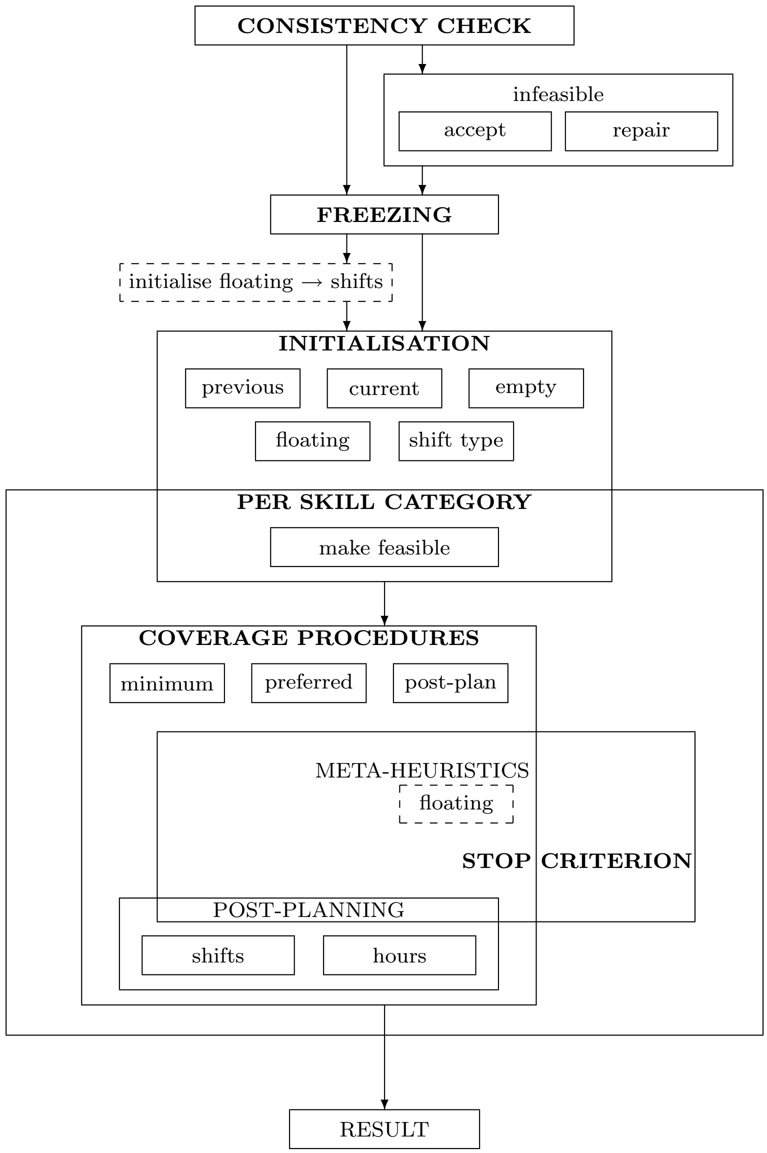

Figure 5.1: Overview of the solution framework

a reduced search space (Section 5.3). It is the case when a previously generated schedule can be partly re-used. The possible options for creating an initial schedule are explained in Section 5.4. In Section 5.5, we explain how the problem can be divided into sub problems which are related to the skill categories. The large solution space can be split into smaller regions which are not necessarily completely disjunctive. A number of strategies of which some are applied after the search heuristics have ended, are introduced in Section 5.6. Finally, in Section 5.7, we indicate how the stop criteria are adjustable to widely varying problems.

5.2 Consistency check on available people

When the hard constraints are so strict that no feasible solution exists, planners can opt to relax them. The planning system is not developed for handling infeasible problems. In most cases, the hard constraints are so strong that it is obvious, after a preliminary check, that some of the soft constraints cannot be satisfied.

The following situation, in which a ward consists of 10 people, is a simple example of how we can calculate that the number of people will not be sufficient to satisfy the personnel required, where we might add $inconsistent_{q,t}$ (see lower) to the number instead. Suppose that 10 people are available on a particular day, and that the staff who are not working according to a predefined pattern have special requests:

- 2 ask for a day off (Constraint 24)
- 1 person asks to work the late shift (Constraint 26).

The patterns for the other staff on that day are as follows:

- 2 people have a day off (Constraint 22, PAT-6)
- 2 people work an early shift (Constraint 22, PAT-2)
- 2 people work a night shift (Constraint 22, PAT-2).

Suppose 4 early shifts are required on the day in question. Since, from the 10 people available, 7 are already planned for other things than an early shift, the soft constraints above will certainly be violated when satisfying the hard constraints. However, when accepting the suggested changes for the hard constraints by the consistency check algorithm ('repair' in Fig. 5.1), the hard constraint for 4 early shifts that day can be relaxed to 3, allowing all these soft constraints to be satisfied.

In order not to expect too much insight from the hospital planners who set up the data, a simple consistency check is performed before the planning starts. In this pre-planning process, the number of available people of a certain skill category is compared with the number of requested people for that skill category at any time during the planning period. Depending on the formulation of the personnel demands (see Section 2.3.3), the algorithm checks shift types or time intervals.

Apart from an obvious check on the hard constraints, the users demanded an extra check on some 'precedence' soft constraints, namely on Constraint 22 (patterns), 24, 25 (personal requests for days and shifts off), and 26 (requested shift assignments). Initially, time slots which are not available for assignments of shifts for a skill category are presented in Fig. 5.2. A clear distinction between hard and soft constraints is made in the figure. The algorithm for shift type personnel requirements is schematically presented in Fig. 5.3. The consistency check respects the planning order of the skill categories. For every personnel member, a list of available time slots (shift types or time intervals) is constructed. The personnel requirements are reduced with the number of requested assignments (Constraint 26) or requested shifts in the patterns (building blocks PAT-1 and PAT-2 of Constraint 22).

In Fig. 5.3, the term requirements is used instead of the more specific options minimum or preferred personnel requirements. We explain in Section 5.6 how the hard coverage constraints for the scheduling algorithm are derived from the personnel requirements and the planning options.

Some assignments cannot be removed by the scheduling algorithms (see Section 5.3). The variable *occupied* has been introduced in the algorithm to keep track of the areas in the schedule that are not free for assignments. All the free days in patterns (type PAT-5) render the corresponding days in the schedule occupied. When estimating the number of assignments in a schedule, the algorithm starts with calculating all the requested shifts (Constraint 26 for the considered assignment unit). Patterns (Constraint 22), in which a type PAT-2 for the shift type corresponding with the day and assignment unit, are also added to the assignments. In all these cases, we set $occupied_{p,t}$ equal to 1, for all the assignment units t of the day on which the imaginary assignment was made. If, for a certain assignment unit t, the assignments already exceed the requirements, the value $inconsistent_{q,t}$ is set equal to the excess.

In the other case, the algorithm searches positions in the schedule where extra assignments can be made. For all the time units on which the assignments are less than the requirements, we try to add shifts to unoccupied schedule positions with a pattern type PAT-1, or with PAT-2 if it corresponds to the duration of s_t. All the assignments in this step set the value of $occupied_{p,t}$ equal to 1, on the corresponding days. When there is still an excess of requirements, the algorithm continues by assigning the shifts to personal schedules that are still unoccupied on the particular day. If there are not enough such schedules, the value $inconsistent_{q,t}$ equals the number of assignments minus the number of requirements.

Blocking the entire day on which an assignment is made is often more strict than necessary. However, in real-world situations, it rarely occurs that a personnel member in healthcare is assigned to more than one shift per day. We found it better to identify such problems as inconsistent. Manual planners are free to accept or ignore the suggested modifications. The more flexibility the algorithm allows for doing the preliminary assignments, the more accurate the diagnosis will be. In the algorithms implemented for solving ANROM, alternative skill categories are also taken into account. Assignments for alternative qualifications can often help to satisfy Constraint 22 and 26 if the problem would be

$\forall q, (1 \leq q \leq Q):$
$\forall p, (1 \leq p \leq P), \forall t, (1 \leq t \leq T):$

$$w = w_p$$
$$pa = pat_w$$
$$IF \quad (pa \neq 0)$$
$$\begin{cases} d = t/S + 1 \\ st = start\_pattern_p \\ l = pattern\_length_{pa} \\ pa\_day_d = pattern\_day_{pa,(1+l-st)/l*7+d} \\ pa\_detail_d = pattern\_detail_{pa,(1+l-st)/l*7+d} \end{cases}$$

$schedule_{p,t}$ *is NOT available for skill category q IF*

Hard Constraints

$\{p, t\} \notin search\_space$
$schedule_{p,t} \neq 0$
$q_p \neq q \wedge q \notin QA_p$

Soft Constraints

$day\_off_{p,t/S+1} = 1$
$shift\_off_{p,t} = 1$
$|\{u \; \xi \; 1 \leq u \leq S \wedge u \neq s_t \wedge required\_assignment_{p,(d-1)*S+u} = 1\}| \neq 0$
$pa \neq 0 \quad \wedge$

$$\begin{cases} (pa\_day_d = PAT\text{-}2 \wedge pa\_detail_d \neq s_t) \vee \\ (pa\_day_d = PAT\text{-}3 \wedge (|shift\_duration_{s_t} - pa\_detail_d| > 15)) \vee \\ (pa\_day_d = PAT\text{-}5 \vee pa\_day_d = PAT\text{-}6) \vee \\ (pa\_day_d = PAT\text{-}7 \wedge s_t \in pa\_detail_d) \end{cases}$$

Figure 5.2: Unavailable timeslots

$$
\begin{array}{l}
\forall k, (1 \leq k \leq Q): \\
\\
\quad q = QO_k \\
\quad \forall p, (1 \leq p \leq P), \forall t, (1 \leq t \leq T): \\
\quad\quad occupied_{p,t} = 1 \;\; IF \\
\quad\quad \left\{ \begin{array}{l}
day\_off_{p,t/S+1} = 1 \\
shift\_off_{p,t} = 1 \\
|\{u \; \xi \; (S * (t/S - 1) + 1 \leq u \leq S * t/S) \wedge required\_assignment_{p,u} \neq 0\}| \\
\vee \; (pa \neq 0 \wedge (pa\_day_d = PAT\text{-}2 \vee pa\_day_d = PAT\text{-}5 \\
\vee \; (pa\_day_d = PAT\text{-}6 \wedge pa\_detail_d = s_t) \\
\vee \; pa\_day_d = PAT\text{-}7))
\end{array} \right. \\
\quad\quad occupied_{p,d} = 0 \;\; ELSE \\
\\
\quad \forall t, ((start - 1) * S + 1 \leq t \leq end * S): \\
\quad\quad assigned_{q,t} = |\{p \; \xi \; 1 \leq p \leq P \wedge automatic_p \neq 1 \\
\quad\quad \wedge \; (schedule_{p,t} = q \vee schedule_{p,t} = pref + q)\}| \\
\quad\quad + |\{p \; \xi \; 1 \leq p \leq P \wedge \{p, t\} \in search\_space_q \\
\quad\quad \wedge \; ((required\_assignment_{p,t} \neq 0) \vee (pa \neq 0 \; \wedge \\
\quad\quad pa\_day_d = PAT\text{-}2 \wedge s_t = pa\_detail_d))\}| \\
\\
\quad with \\
\quad \left\{ \begin{array}{l}
d = t/S + 1 \\
w = w_p \\
pa = pat_w \\
st = start\_pattern_p \\
l = pattern\_length_{pa} \\
pa\_day_d = pattern\_day_{pa,(1+l-st)/l*7+d} \\
pa\_detail_d = pattern\_detail_{pa,(1+l-st)/l*7+d}
\end{array} \right. \\
\quad \ldots
\end{array}
$$

Figure 5.3: Schematic procedure for the consistency check on the available personnel in the case of shift type personnel requirements (1)

$\ldots$

$IF \quad (assigned_{q,t} > requirements_{q,t})$

$\quad inconsistent_{q,t} = assigned_{q,t} - requirements_{q,t}$

$ELSE$

$\quad p = 1$

$$WHILE(assigned_{q,t} < requirements_{q,t} \wedge p \leq P)$$

$$\begin{cases} IF \quad \{p,t\} \in search\_space_q \wedge occupied_{p,t} = 0 : \\ \quad IF \quad (pa \wedge pa\_day_d = PAT\text{-}3 \wedge \\ \qquad\qquad |pa\_detail_d - shift\_duration_{s_t}| \leq 15) \\ \quad \forall u, (S * (t/S) + 1 \leq u \leq S * (t/S + 1)) : \begin{cases} occupied_{q,t} = 1 \\ assigned_{q,t} + 1 \end{cases} \\ p + 1 \end{cases}$$

$\quad p = 1$

$$WHILE(assigned_{q,t} < requirements_{q,t} \wedge p \leq P)$$

$$\begin{cases} IF \quad \{p,t\} \in search\_space_q \wedge occupied_{p,t} = 0 : \\ \quad IF \quad (pa \wedge pa\_day_d = PAT\text{-}1) \\ \quad \forall u, (S * (t/S) + 1 \leq u \leq S * (t/S + 1)) : \begin{cases} occupied_{q,t} = 1 \\ assigned_{q,t} + 1 \end{cases} \\ p + 1 \end{cases}$$

$\quad p = 1$

$$WHILE(assigned_{q,t} < requirements_{q,t} \wedge p \leq P)$$

$$\begin{cases} IF \quad \{p,t\} \in search\_space_q \wedge occupied_{p,t} = 0 : \\ \quad \forall u, (S * (t/S) + 1 \leq u \leq S * (t/S + 1)) : \begin{cases} occupied_{q,t} = 1 \\ assigned_{q,t} + 1 \end{cases} \\ p + 1 \end{cases}$$

$\quad IF \quad (assigned_{q,t} < requirements_{q,t})$

$\quad\quad inconsistent_{q,t} = assigned_{q,t} - requirements_{q,t}$

Figure 5.4: Schematic procedure for the consistency check on the available personnel in the case of shift type personnel requirements (2)

inconsistent for the main skill category.
The scenarios are equivalent for all the possible personnel requirement formulations (RM, RP, RIM, and RIP).
Since the system does not handle infeasible problems, the algorithm always reduces the personnel requirements if an unavoidable violation is detected. As the user starts a planning, the algorithm informs about inconsistencies in the personnel demands. The inconsistencies for minimum and preferred shift type requirements are represented in the structures $inconsistentRM_{q,t}$ and $inconsistentRP_{q,t}$. For floating requirements, they are correspondingly denoted by $inconsistentRIM_{q,t}$ and $inconsistentRIP_{q,t}$. The values are 0 when the requirements cause no inconsistency. A value larger than 0 indicates that the requirements should be increased by that value, a negative number requires a corresponding decrease. Either the user can accept the remedy by relaxing the personnel requirements when necessary or he can deliberately choose to violate the soft constraints that were checked. Both options are demonstrated in the overview of Fig. 5.1.

5.3 Freezing parts of the Schedule

The working area of rostering algorithms can be restricted for several reasons. In the most common approach, all the personnel members of a ward have to be scheduled from the first till the last day of the planning period. We distinguish 3 different reasons to limit the search space. They are explained in the following sections.

5.3.1 People who must not be scheduled automatically

Some hospital situations require an interactive way of scheduling certain personnel members. It should be possible to generate the schedule of particular nurses by hand. In order to maintain these personal schedules, it is forbidden for the rostering algorithms to add or remove any assignment. Each personnel member whose schedule must not be planned automatically have value 0 for $automatic_p$. Frozen personal schedules are evaluated against the soft constraints in the same way as the schedules of personnel members who are part of the automatic planning process. Also, frozen personal schedules do not relax the hard constraints.
Schedules with frozen areas have a restricted search space. The calculation time is generally less than for the corresponding full schedules. On the other hand, the frozen areas constrain the solution space and make it harder to find a good quality solution. With a rigid assignment of shifts and free days to parts of the schedule that are frozen for the planning algorithms, it might be much more difficult to meet the personnel requirements in a good quality solution with the remaining personnel members and within the remaining time interval.

5.3.2 Freeze parts of personal schedules

The start and end date of personal contracts do not necessarily coincide with the start and end times of the planning period. This option provides the possibility for allocating personnel temporarily in other wards. In ANROM, the minimum period during which people are allocated to a ward is one day. On that day, shift assignments can be made. The value of $employed_{p,d}$ is 0 if person p is not employed in the ward at day d, otherwise it is 1.
Soft constraints are always evaluated over the entire planning period. For people who are not employed during the entire planning period, an adjustment of the cost function is necessary for certain constraints (e.g. Constraint 3, 8, 11, etc). For example, the values A_p, AH_p, etc (introduced in Section 2.4) take the time during which the person p is not in the service of the hospital into account. The activities of personnel members working in other wards are not calculated when evaluating the hard constraints of the current ward. However, they do contribute to the evaluation of the cost function.
In Section 4.2.5, we explained how personnel members can change work regulation and/or skill category within the planning period. A transition from one work regulation to another does not affect the search space. Neither does a transition of skill categories change the global search space, but it changes the search space for the skill categories concerned (see Section 5.5). In order not to overload the pseudo code of the algorithms in this section, we assume that the skill categories remain the same within a planning period.

5.3.3 Freeze periods in time

Schedules can be frozen partially in time. In case urgent rescheduling is required, it is recommended not to disturb already existing personal schedules drastically. A typical problem in healthcare occurs when a personal member is not able to perform the assigned shifts, for example during an illness period. Instead of re-planning the ward with the modified constraints, the personnel prefer their previously generated schedules to remain unchanged as much as possible. Generating a schedule from scratch, taking the new situation of the constraints into account would probably lead to a much better overall quality in terms of the cost function, but that is not what the personnel want. Only a few personal schedules will be affected by the replacements due to the option presented in this section. In ANROM, we modelled a possibility for freezing one or two time intervals of the schedule. Frozen parts are restricted to the start and/or the end of the planning period. We denote the first day of the planning period by *start* and the last day by *end*. Whether or not the hard constraints are satisfied within these frozen parts of the search space, the rostering algorithms will not add or remove any assignment in these periods. The scheduled shifts in these periods do contribute to the evaluation of the cost function, which always looks at the entire planning period.

Fig. 5.5 explains which parts of the schedule belong to the search space of the rostering algorithms.

$$\forall p, (1 \leq p \leq P), \forall t, (1 \leq t \leq T):$$

$$\{p,t\} \in search\_space \quad IF \quad \begin{cases} automatic_p = 1 \\ employed_{p,d} = 1 \\ S*(start-1)+1 \leq t \leq S*end \end{cases}$$

Figure 5.5: Non-frozen parts of a schedule

5.4 Initialisation

The initialisation of the scheduling algorithm consists of two phases for constructing a feasible initial solution. It suffices for a schedule to satisfy all the hard constraints to be called feasible. In the first phase, the input is loaded (after an option has been selected, Section 5.4.1). The second phase makes the schedule feasible.

5.4.1 Input for the initialisation

For practical planning problems three possible strategies are introduced.

- **Current schedule**
  The option starts from the currently available schedule, which can either be the result of a previous attempt to generate a solution, or the planning that existed before certain extra restrictions occurred. Very regularly the rostering does not happen in one go. The hospital planner will often quickly calculate a schedule to check certain constraints or personal preferences, after which (by making changes to constraints or personnel requirements if necessary), a final schedule will be calculated. The option to take the current schedule as input for the initialisation phase is especially useful when urgent changes in an existing schedule are required. In real life this may happen when a personnel member is suddenly ill and has to be replaced. Of course, this emergency is not supposed to change the schedule for other people drastically. In many practical occasions, schedulers applying this option make use of the 'freezing' tool (Section 5.3.3).

- **Schedule of the previous planning period**
  This option is useful when the schedule in the previous planning period is of very high quality and when the constraints on the current and the previous planning period are similar. It is not recommended to use

the previous schedule as an input when the number of personnel is not the same in both periods nor when the planning periods include bank holidays. We also advice not to select this option when the *pattern* constraint has a period different from the planning period for some personnel members.

- **Empty schedule**
  The simplest input option starts the initialisation from an empty schedule.

Although the two first initial schedule constructors may seem very attractive, our experiments show that it is not too difficult for the meta-heuristic algorithms to produce schedules of comparable quality starting from a random initial schedule. Indeed it is often the case that with the two latter initialisations, the algorithm is already in a local minimum and has problems escaping from it.

5.4.2 Create a feasible solution

Given the entire search space of the ANROM problem, each element of that search space, called a schedule, corresponds to a potential solution. We call the schedule feasible when it satisfies all the hard constraints. In order to satisfy the hard constraints in the initial schedule, an algorithm is used that adds and/or removes shifts until the personnel requirements (according to the planning options of Section 5.6) are met. The process takes some of the soft constraints into account but it is mainly random driven. Users of the software based on ANROM suggested to force satisfaction of the personal requests for days and shifts off (Constraint 24 and 25), and satisfaction of the patterns (Constraint 22). These are precisely the constraints that play a role in determining the consistency of the data (Section 5.2). It seems contradictory to the previously explained concept of treating the cost function (which sums violations of all the constraints, Chapter 4) as the only evaluation means. However, even though the planners can freely set cost parameters, a schedule in which less personal constraints are violated will often be preferred to better quality schedules (with respect to the cost function) containing more violations of these particular constraints. Fig. 5.7 describes the entire initialisation phase but the procedure will be explained step by step. The initialisation is always executed per skill category. The group of people belonging to the skill category is extended with the people who have that skill as an alternative qualification. In Fig. 5.6, we demonstrate how the schedule is divided into sub-schedules that correspond to each skill category. Some of the personnel members have alternative qualifications in this simplified example and 1 person even changes skill category during the planning period. The initialisation algorithm makes a first attempt, only in the case when the *Empty schedule* option holds, to satisfy the pattern constraint. When a pattern requires an obligatory shift, the algorithm assigns a randomly chosen shift to the corresponding people, provided the assignment never violates the hard constraint on the number of people required. In order to be general, we have used a simplified notation *requirements* (see also Section 5.2) for the

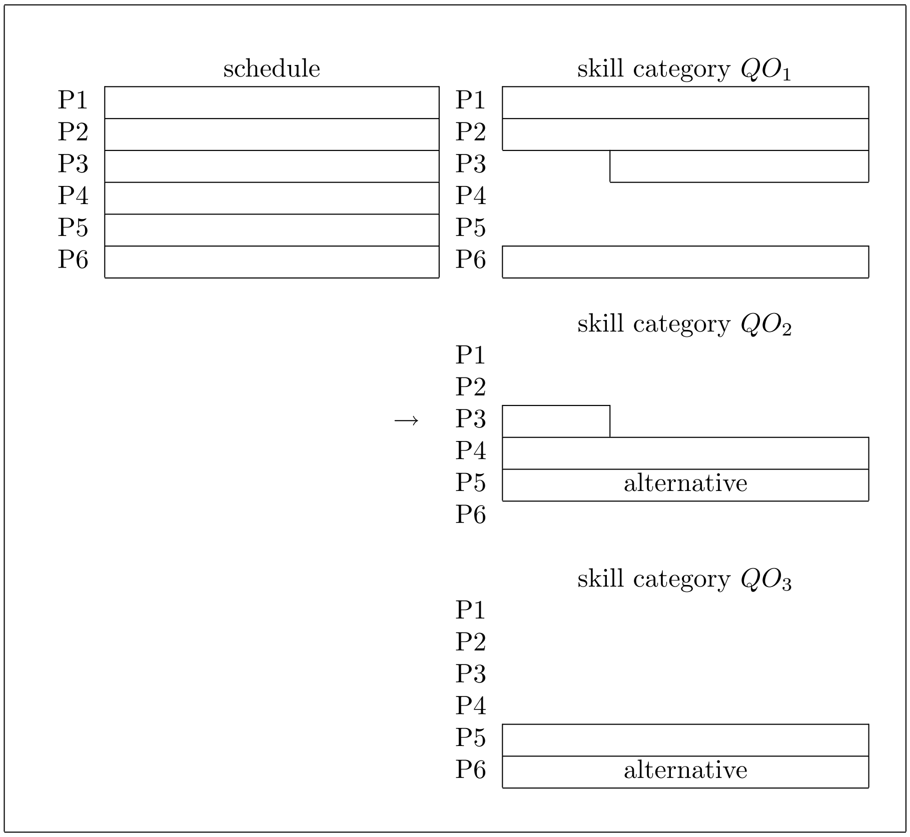

Figure 5.6: Example of the division of a schedule into partial schedules that are solved per skill category. P3 moves from skill category QO_2 to QO_1 during the planning period. P5 belongs to QO_3 but is also allowed to carry out shifts for QO_2, and P6 belongs to QO_1 with alternative qualification QO_3.

personnel requirements. The notation does not specify whether it is minimum or preferred requirements. In Section 5.6, we explain which type of personnel requirements determine the hard coverage constraints.
In case the shift type is specified (building block PAT-2 of the pattern, Constraint 22), the shift will be assigned to the person if there is a shortage in the schedule for the shift on the particular pattern day. For the 3rd building block, PAT-3, the algorithm randomly chooses a shift type of the specified duration (allowing the small deviation of the preset time, i.e. 15'), for which the personnel requirements are not yet met.
The algorithm afterwards moves to an iterative phase which stops when the personnel requirements are fulfilled for every assignment unit or when a maximum number of attempts to assign randomly is reached. The maximum number of attempts is function of the number of people who can carry out jobs for the skill

category. It is called *max_attempts* in Fig. 5.7. In case the number of scheduled personnel is too low to meet the requirements, the algorithm will assign the corresponding shift type randomly to a personal schedule, provided it is part of the search space for that qualification.

Experiments with varying real world problems led to the procedure described in this section. It turned out to be a very satisfying approach for tackling the large set of nurse rostering problems occurring in practice.

We will now explain the procedure in detail. The personnel are divided into groups with equal eagerness for the assignment. In the case where all people in a group already have an assignment for the shift (either for the currently scheduled skill category or for another), the assignment fails. The algorithm stops if a random assignment is possible in a group, that is if a person of the group has no assignment. If no such person exists, the algorithm moves to the next group. People can only belong to a group if the considered part of the schedule (assignment unit) belongs to the search space. The number of groups for adding assignments is called *add_max* in Fig. 5.7.

ADD–1 All the people having a personal request for the shift corresponding to the time to be scheduled (Constraint 26) and no assignment yet.

ADD–2 All the people working according to a predefined pattern, for which a PAT-2 type corresponds to the day and the detail to the shift to be scheduled, with an empty schedule for that assignment unit.

ADD–3 All the people working according to a predefined pattern, for which a PAT-3 type corresponds to the day and the detail to the duration of the shift to be scheduled (+/- the deviation), and an empty schedule for that assignment unit.

ADD–4 All the people who have the scheduled skill category as main skill category and do not have a personal request (with a high importance) for a day off or shift off at the time to be scheduled (see Constraint 24 and 25), and have an empty schedule for that assignment unit.

ADD–5 All the people who have the skill category as alternative and do not have a personal request (with a high importance) for a day off or shift off at the time to be scheduled, and have an empty schedule for that assignment unit.

ADD–6 All the people who belong to the skill category, have a marked assignment for another skill category that has not been scheduled in this run (a skill category which is lower in the planning order hierarchy).

ADD–7 All the people who have the skill category as alternative, have a marked assignment for another skill category that has not been scheduled in this run (a skill category which is lower in the planning order hierarchy) and which is different from the main skill category of the particular people.

ADD–8 All the people who are authorised for the skill category and have an empty schedule for the shift.

The group ADD-7 is exceptional in that a marked assignment will be removed in order to enable the assignment that will make the schedule feasible.
An analogous procedure has been developed for removing shifts when a schedule exceeds the requirements for certain shifts. We identify *rem_max* different groups (see Fig. 5.7). The hierarchy of the groups to which the removal of a shift is applied is listed below:

REM–1 All the people who have the skill category as alternative and have a marked (see Section 5.6) assignment for the skill category being scheduled.

REM–2 All the people who belong to the skill category and have a marked assignment for it.

REM–3 All the people who have the skill category as alternative and have an assignment for the skill category being scheduled.

REM–4 All the people who belong to the skill category and have an assignment for it.

In Section 5.6, we will explain how the personnel requirements for the initialisation depend on the planning options.

5.5 Planning order of skill categories

Hard constraints must never be violated (see Section 2.3), thus shifts for a skill category cannot be assigned to unqualified people. In Section 2.3.3, we explained that personnel requirements are defined per skill category. Each skill category is scheduled separately.
Dividing the rostering in sub problems reduces the search space. The number of personnel belonging to each skill category is often considerably smaller than the entire staff in the ward. The number of shifts to be assigned (whether or not translated from floating requirements) is also lower.
In case people have the permission to carry out shifts for other than their main skill category, there are a few difficulties. After the planning for a qualification has stopped, the algorithm moves on to the next qualification and temporarily freezes the already assigned shifts. The fixed shifts can hinder the planning of the following qualifications, especially when they occur in the schedule of a person who has another main skill category.
Planners can freely arrange the order in which the qualifications must be planned. We denote the planning order of skill categories by QO. The elements of this list of length Q are the qualification numbers, with QO_1 the qualification to plan first, and QO_Q the last one. In Fig. 5.8, the procedure of planning the shifts for different qualifications in a predefined order, is schematically presented.

$$\forall k, (1 \leq k \leq Q):$$

$$q = QO_k$$
$$\forall t, (1 \leq t \leq T):$$
$$scheduled_{q,t} = |\{p \ \xi \ 1 \leq p \leq P \wedge (schedule_{p,t} = q \vee schedule_{p,t} = pref + q)\}|$$

$$attempt = 0$$
$$x = 1$$

$$WHILE \ (scheduled_{q,t} < requirements_{q,s_t,t/S+1} \wedge x \leq add\_max)$$
$$\left\{\begin{array}{l} IF \ \ (random \ p \in ADD\text{-}x) \\ \left\{\begin{array}{l} schedule_{p,t} = pref + q \\ scheduled_{q,t} + 1 \\ attempts + 1 \\ IF \ \ (attempts = max\_attempts) \\ \left\{\begin{array}{l} x + 1 \\ attempts = 0 \end{array}\right. \end{array}\right. \\ ELSE \\ x + 1 \end{array}\right.$$

$$WHILE \ (scheduled_{q,t} > requirements_{q,s_t,t/S+1} \wedge x \leq rem\_max)$$
$$\left\{\begin{array}{l} IF \ \ (random \ p \in REM\text{-}x) \\ \left\{\begin{array}{l} schedule_{p,t} = 0 \\ scheduled_{q,t} - 1 \\ attempts + 1 \\ IF \ \ (attempts = max\_attempts) \\ \left\{\begin{array}{l} x + 1 \\ attempts = 0 \end{array}\right. \end{array}\right. \\ ELSE \\ x + 1 \end{array}\right.$$

Figure 5.7: Schematic representation of the construction of a feasible schedule for shift type requirements

$\forall q, (1 \leq q \leq Q):$

- Make the schedule feasible with respect to the personnel requirements for QO_q (see Section 5.4)
- Execute the planning algorithm within the group of people having QO_q as main or alternative skill, by replacing only assignments in the schedule which correspond to QO_q and $pref + QO_q$

Figure 5.8: Skill categories are scheduled sequentially

Intuitively, many schedulers choose a hierarchical order for the skill categories (i.e. starting with the head nurse qualification and ending with the cleaners). It is often beneficial, however, to first plan the most constrained categories or the qualifications for which it is unavoidable to deploy people with other main skill categories. Suppose, for example, a ward consisting of one *head nurse* and a large group of *regular nurses*. Imagine that one of the regular nurses is designated as substitute for head nurses. If there is head nurse presence required on every day shift during the weekdays, the head nurse has to be replaced whenever he/she is absent (denoted by Constraint 24 or 25). It is not very difficult, in this case, to assign the regular nurse to the required duties during the absence of the head nurse. This is not a problem when the particular regular nurse's schedule is still empty at the time the head nurse qualification is being planned. It can be a problem, however, if the regular nurse cannot replace the head nurse because there is a duty assigned already for the same assignment unit in the substitute's schedule. In Section 2.3, we stated that a person cannot be assigned to the same shift more than once. This involves that the head nurse will be assigned to the shift, although there is probably a very severe reason for being absent.

A particular diversification algorithm, which is implemented in the software system based on ANROM as part of some of the hybrid planning algorithms which are introduced in later chapters (Fig. 5.9) was especially developed to avoid such problems. This diversification algorithm only looks at a few soft constraints (Constraint 22, 24, and 25) for its moves and generally produces a worse overall solution. The algorithm investigates whether people can take over shifts assigned on absence days of other personnel members. Nurses who are absent themselves are not considered. It is the first goal to find a replacement among the personnel belonging to the skill category of the duty to be moved. If there is no such person available, people who have the skill category as alternative are considered. In a more elaborated version of this algorithm, further replacements are also considered. The details of this extended algorithm are not explained in this work but a simple example will demonstrate the aim. Suppose Nurse A has an assignment to a shift type on an absence day. Let Nurse B be skilled to carry out that shift. Suppose B has no absence on that particular day but has an assignment for the same shift type (but for another skill category) already. An extra step in the algorithm will try to move B's assignment to a

$$
\begin{array}{l}
\forall k, (QO_Q \geq k \geq 1): \\
\forall p, t, \{p,t\} \in search\_space \\
\\
\quad q = QO_k \\
\quad d = t/S + 1 \\
\quad IF \;\; ((schedule_{p,t} = q \vee schedule_{p,t} = pref + q) \wedge \\
\qquad (day\_off_{p,d} = 1 \vee shift\_off_{p,t} = 1 \vee pat\_dayoff_{p,t} = 1)) \\
\quad \begin{cases}
found = 0 \\
b = 1 \\
\;WHILE \; (found = 0 \wedge b \leq P) \\
\; \begin{cases}
IF \; (\{b,t\} \in search\_space \wedge schedule_{b,t} = 0 \wedge b_q = q \wedge b \neq p \\
\qquad \wedge \, (day\_off_{b,d} = 1 \vee shift\_off_{b,t} = 1 \vee pat\_dayoff_{b,t} = 1)) \\
\begin{cases}
schedule_{p,t} = 0 \\
schedule_{b,t} = q \\
found = 1
\end{cases} \\
b + 1
\end{cases} \\
IF \; (found = 0) \\
b = 1 \\
\;WHILE \; (found = 0 \wedge b \leq P) \\
\; \begin{cases}
IF \; (schedule_{b,t} = 0 \wedge q \in b_{QA} \\
\qquad \wedge \, (day\_off_{b,d} = 1 \vee shift\_off_{b,t} = 1 \vee pat\_dayoff_{b,t} = 1)) \\
\begin{cases}
schedule_{p,t} = 0 \\
schedule_{b,t} = q \\
found = 1
\end{cases} \\
b + 1
\end{cases}
\end{cases} \\
\\
\quad with \\
\quad \begin{cases}
pat\_dayoff_{x,t} = 1 & IF \; (pa \neq 0 \wedge pa\_day_d = PAT\text{-}5 \vee PAT\text{-}6 \\
 & \vee \, (PAT\text{-}7 \wedge s_t \in pa\_detail_d)) \\
pat\_dayoff_{x,t} = 0 & ELSE
\end{cases} \\
\quad and \\
\quad \begin{cases}
w = w_x \\
pa = pat_w \\
st = start\_pattern_p \\
l = pattern\_length_{pa} \\
pa\_day_d = pattern\_day_{pa,(1+l-st)/l*7+d} \\
pa\_detail_d = pattern\_detail_{pa,(1+l-st)/l*7+d}
\end{cases}
\end{array}
$$

Figure 5.9: Search algorithm to increase the quality of the schedule in case of an unfortunate planning order of the skill categories

third nurse's schedule, after which A's assignment can be put in B's schedule. This algorithm is not very useful unless it is followed by a proper search algorithm because it does not generally improve the overall quality.
Some real-world situations require rescheduling the personnel members of a single skill category. In that case, freezing concerns all the assignments for other skill categories.

5.6 Coverage Procedures

In practice, the number of required personnel on a certain day is not completely strict. Experienced planners know very well when it is reasonable to plan more or less personnel than required. However, because there exist no clear rules for decisions like this, planners using ANROM can optionally choose among different coverage strategies. In Section 2.3.3, we introduced the difference between minimum and preferred personnel requirements. Any solution with fewer assignments than the minimum requirements, or with more than the maximum requirements, violates the hard constraints.
The framework provides planning options to set the coverage constraints, which will be hard constraints for the rostering algorithms. It is also possible to allow a few post-planning algorithms, which can change the coverage after a schedule has been generated. We will now explain some of these options in more detail.

5.6.1 Minimum - preferred requirements

The hospital scheduler can choose to plan the minimum personnel requirements or the preferred requirements as hard constraints. During the entire planning process, the number of planned shifts (be it minimal or preferred) will not change when applying the meta-heuristics discussed later in this work.
This planning option holds for both shift type and floating requirements.

5.6.2 Plan towards preferred requirements

Instead of strictly setting the hard constraints, this option allows to work with a range in which the hard constraints are considered satisfied. The algorithm to organise this option first takes the minimum requirements as hard constraints. After a result has been calculated by the scheduling algorithms (Chapter 7 - 9), the system searches possibilities for adding shifts to the schedule wherever this does not involve an extra violation of soft constraints. For every day on which there is a difference between what is actually scheduled and the preferred requirements, the system adds a shift to that personal schedule, which improves the overall quality of the schedule most. Adding a pair of shifts on consecutive days in a personal schedule is often less harmful than adding an isolated shift. In the competition for the best candidates to assign extra shifts to, 'twin assignments' are also considered (provided none of the assigned shifts causes an excess with respect to the preferred personnel requirements of

$$
\begin{array}{l}
scheduled_{q,t} = |\{p \;\xi\; 1 \leq p \leq P \wedge (schedule_{p,t} = q \vee schedule_{p,t} = pref + q)\}| \\
\forall p, (1 \leq p \leq P) : penalty_p = penalty(schedule_p) \\
\forall t, (start * (S-1) + 1 \leq t \leq end * S) : extra_t = RP_{q,t} - scheduled_{q,t} \\
\\
extra = \sum_t extra_t \\
stop = 0 \\
\\
WHILE \quad (extra \wedge stop \neq 1) \\
\begin{cases}
best = \infty \\
IF \quad (best \leq 0) \begin{cases} schedule_{best_p, best_t} = pref + q \\ extra_{best_t} - 1 \\ IF \quad (best_{t2} \neq 0) \begin{cases} schedule_{best_p, best_{t2}} = pref + q \\ extra_{best_{t2}} - 1 \end{cases} \end{cases} \\
ELSE \quad stop = 1
\end{cases}
\end{array}
$$

Figure 5.10: Post-planning algorithm to satisfy the preferred shift type personnel requirements better for skill category q

the schedule). Since the complexity of finding optimal ‘twins’ to add to the schedule is exponential, we reduced the search to the selection of the best set of equal shift types on two consecutive days for a personal schedule. The entire procedure is illustrated in Fig. 5.10.

For floating personnel requirements, a slightly different approach is needed. Instead of searching the best candidate to assign an extra shift or an extra pair of shifts to, we have to add switches from a shift to a longer shift, from a pair of shifts to a single shift with a longer entire duration or from a single shift to a pair of shifts that last longer (see also Chapter 6).

The system allows even for a more flexible approach by providing a threshold value for the individual cost function value: $threshold_{cost}$. In this case, the algorithm will add extra assignments, whenever the personal cost function value does not exceed that threshold.
Every shift type that is added to the schedule after the planning algorithm has stopped, will be marked. The location of such marked shifts in the schedule is the result of a post-processing algorithm, while the other shift types have been assigned by search algorithms which look at the entire search space. It is recommended during some planning activities to remember which shift removals will harm the schedule less. We explained the importance of the marking in Section 5.4.2.

5.6.3 Adding hours

The option to add shifts to people with undertime does not necessarily respect the (hard) preferred personnel coverage constraints which hold during the course of the planning. It is a pure post-planning option. Once a schedule has been calculated, an algorithm searches for every personal schedule the best point in time to assign extra shifts. The constraint holds that a shift cannot be added unless such shift occurs already in the personnel requirements for that day and skill category.

By default, nothing will happen if it would increase the value of the cost function. As explained in the previous option, there is also a possibility for setting a threshold: $threshold_{hours}$. In this case, the threshold determines the maximum number of excess hours (overtime). When a personal schedule has reached this number, the algorithm does not add extra shifts. Just like in the previous section, where shifts are added towards the preferred requirements, the extra shifts are marked. The procedure is explained in Fig. 5.11. Since it can be better in terms of the cost function to add a pair of shift types on consecutive days, the possibility is also evaluated when searching for the best time to assign extra shifts.

5.7 Stop Criterion

When applying the meta-heuristics introduced in the Chapters 7-9, a stop criterion is required. The tabu search algorithms of Chapter 8 have been in use in hospitals for quite some years now and the stop criteria for these have been fine tuned all the time. This experience forms the basis for the stop criteria of the algorithms introduced in the other sections.

The search space per skill category depends largely on the number of people in that group (of which the schedule is not frozen), the length of the non frozen part of the planning period and the total number of available shift types to be scheduled. Depending on how good a schedule is supposed to be, the planners can choose among a number of options, thus combining different hybrid algorithms. For every single algorithm used in the hybrid approaches, a separate stop criterion will be applied. A steepest descent algorithm will stop as soon as there exists no more improvement in the neighbourhood. Some diversification and greedy algorithms (introduced in Section 8.3.6) stop after the first iteration without improvement.

The stop criteria for the specific meta-heuristics will be discussed when introducing the algorithms in Chapter 7-9.

$$
\begin{array}{l}
\forall p, (1 \leq p \leq P): \\
\quad penalty_p = penalty(schedule_p) \\
\quad stop = 0 \\
\quad w = w_p \\
\quad best = \infty \\
\quad hours \;=\; previous\_hours_w \\
\qquad\qquad\quad + |\{t \;\xi\; 1 \leq t \leq T \wedge schedule_{p,t} \neq 0\}| * shift\_duration\_s_t \\
\quad WHILE \; (hours < max\_hours_w - AH_p \wedge stop \neq 0) \\
\quad \left\{ \begin{array}{l}
best = \infty \\
\forall t, (start * (S-1) + 1 \leq t \leq end * S): \\
\left\{ \begin{array}{l}
IF \quad (RP_{q,t} > 0 \wedge \{p,t\} \in search\_space_q \wedge schedule_{p,t} = 0 \\
\qquad \wedge\, penalty_{p,C8} < threshold_{hours} + shift\_duration_{s_t}) \\
\left\{ \begin{array}{l}
schedule_{p,t} = pref + q \\
new\_penalty + p = penalty(schedule_p) \\
schedule_{p,t} = 0 \\
IF \quad (penalty_p - new\_penalty_p \leq best) \\
\left\{ \begin{array}{l}
best_p = p \\
best_t = t \\
best_t 2 = 0 \\
best = penalty_p - new\_penalty_p
\end{array} \right.
\end{array} \right. \\
IF \quad (RP_{q,t} > 0 \wedge RP_{q,t+S} \wedge \{p,t\} \in search\_space_q \wedge schedule_{p,t} = 0 \\
\qquad \wedge\, \{p, t+S\} \in search\_space_q \wedge schedule_{p,t+S} = 0 \\
\qquad \wedge\, (penalty_{p,C8} < threshold_{hours} + 2 * shift\_duration_{s_t})) \\
\left\{ \begin{array}{l}
schedule_{p,t} = pref + q \\
schedule_{p,t+S} = pref + q \\
new\_penalty + p = penalty(schedule_p) \\
schedule_{p,t} = 0 \\
schedule_{p,t+S} = 0 \\
IF \quad (penalty_p - new\_penalty_p \leq best) \\
\left\{ \begin{array}{l}
best_t = t \\
best_{t2} = t + S \\
best = penalty_p - new\_penalty_p
\end{array} \right.
\end{array} \right.
\end{array} \right. \\
IF \quad (best \leq 0) \left\{ \begin{array}{l}
schedule_{p,best_t} = pref + q \\
hours + shift\_duration_{s_t} \\
IF \quad (best_{t2} \neq 0) \left\{ \begin{array}{l}
schedule_{p,best_{t2}} = pref + q \\
hours + shift\_duration_{s_t}
\end{array} \right.
\end{array} \right. \\
ELSE \quad stop = 1
\end{array} \right.
\end{array}
$$

Figure 5.11: Post-planning algorithm to decrease undertime for people with skill category q

Chapter 6

Floating Personnel Requirements

6.1 Introduction

The motivation for the alternative way of modelling personnel requirements has been provided by hospital administrators/schedulers who apply the nurse rostering software based on ANROM (see Section 2.1) in practice. This approach is a direct response to the requirements and demands of difficult real world scheduling problems. The presented model aspires to accommodate the customs and practices employed by the personnel planners in hospitals and allows for a high flexibility in constructing the timetables.
This new formulation was not developed until some of the algorithms introduced in PART III of this thesis were applied in practice. However, we decided to describe these floating personnel requirements model in PART II because it is now available as an alternative path in the solution framework. All the meta-heuristics of PART III are applicable both on the traditional shift type requirements and on the new floating requirements model.

Hospitals define a high number of shift types that match the typical activities of the institution and allow for several kinds of part-time employment. The personnel requirements are nearly always expressed as a number of people required per shift type or even per day. We tackle a much more flexible approach in this chapter. Not only is the number of possible shift types higher than in most problems encountered, but also the approach to compose a schedule with different combinations of shift types is (as far as the author is aware) unique.

The meta-heuristics presented in Chapter 7-9 depend very much on categorising staff into certain shift types. However, the planners in practice often find it hard to translate the real world situation generated by their daily staff complement into rigorous staff duty categories. The approach described in this chapter concentrates on an advanced representation of the daily personnel requirements of healthcare institutions. Instead of formulating the requirements as a number of personnel needed per skill category and per shift type for each day of the planning period, we now introduce ‘floating’ personnel requirements (Section 2.3.3). Floating personnel requirements allow for the representation of the personnel requirements per day in terms of time intervals, i.e. start and end times of personnel attendance. This formulation enables the provision of a greater choice of shift work and part time work and reduces the amount of unproductive time because it enables the shifts to be split and combined. We present an algorithmic approach to handle this new formulation. We also set up a series of experiments indicating that, not only does this approach take into account the requests and requirements of hospital schedulers, but it also generates higher quality schedules when compared with shift type approaches. The obtained results are better in the sense that various specific real world soft constraints (e.g. Constraint 11, 12, 13, 17, etc) can be satisfied by scheduling appropriate shift type combinations whereas in the shift type approach fixed shift types restrict the solution space.

In Section 6.2, a few examples of similar coverage problems in literature are presented. Section 6.3 elaborates on the problem definition, starting from the real world hospital practices that induced the development of the floating personnel requirements presented in this chapter. In Section 6.4, we introduce a two-step approach to construct a shift type schedule from time based personnel requirements. The method preserves the desirable features of the metaheuristics which are explained in PART III of this thesis. Examples illustrating the working of the algorithms are given in Section 6.5. In Section 6.6, we discuss the impact of the floating requirements method on the resulting timetable. A comparison with the shift type personnel requirements model is presented in Section 6.7.

6.2 Coverage Constraints

No matter how the coverage constraints are formulated (shift type requirements or floating requirements, Section 2.3.3), the search algorithms do not violate these constraints during the iterations. Coverage remains a hard constraint.
In Table 3.8 of the literature review, a brief overview of possible shift types for the coverage constraints is presented. Many automatic scheduling systems still work with three strictly defined shifts. More recent approaches, of which some are applied in practice, provide user definable shift definitions (e.g. Chiarandini et al. [58], Kawanaka et al. [126], Meisels et al. [138], Meisels and Lusternik [140], Meyer auf'm Hofe [142, 144], Schaerf and Meisels [182], etc). Apart from ANROM, there are very few researchers who allow a time interval formulation of coverage constraints.
Bailey and Field [9] use 12-hour scheduling periods which can start and end at any time of the day. By applying a linear program, they identify shift configurations which minimise costs. Their findings are that the 12-hour ‘flexshift’

concept reduces idle time considerably compared to other approaches in which 8-hour shifts are used. For Isken and Hancock [121], coverage is a soft constraint. Under- and overstaffing are penalised. The size of tackled problems is very small compared to what is carried out in ANROM (Chapter 2). Brusco and Jacobs [30] allow flexible start and end times for the work but they are more concerned with staffing than short-term rostering. Cyclical shift-days off schedules are generated over a limited planning horizon. Tanomaru [201] allows flexible start and end times for the shifts. Coverage is not a hard constraint in [201], where even the number of personnel is not fixed. The dimensions of the tackled problems are very low compared to the problem size of ANROM. Tanomaru schedules periods of one week, and provides personal schedules in which 7 pairs of (start and end) times are presented.
The problem definition in this chapter is the result of feedback from the users of the nurse rostering package in several Belgian hospitals.

6.3 Personnel Requirements in Time Intervals: Floating Requirements

Personnel requirements express the number of personnel of each skill category needed to staff the ward. They are set by management and are usually expressed in terms of the minimum number of personnel needed and the preferred number of personnel available. The minimum number of personnel strictly meets the personnel needed to carry out all the work while the preferred number of personnel will ameliorate the atmosphere by reducing the workload of staff members (see also Section 5.6). The requirements can be formulated either in terms of shift types (which is the traditional approach used in the literature) or in terms of begin and end times. The personnel requirements can depend on the time of the day, the day of the week, etc. For a more formal description of personnel requirements, we refer to Section 2.3.3. Table 2.2 presents an example of personnel requirements on a certain day of a planning period (expressed as a number of required shift types per skill category). If we deduce the personnel

Shift types	From	Till
Short Early	7:00	13:00
Early	7:00	15:00
Day	8:00	17:00
Late	13:00	21:00
Short Late	15:00	21:00
Night	21:00	7:00

Table 6.1: Set of shift types; Dataset 1

requirements from the shift types it is possible to allocate several kinds of part-time employment over the shift period. We call this new representation 'floating

personnel requirements'.
Floating requirements are defined on a time interval basis. They are expressed as a varying number of personnel needed for each skill category during the day (see Section 2.3.3). The main goal of the approach described in this thesis is to construct a timetable covering all the personnel requirements, only using the shift types applied in the hospital. In practice, the time intervals will not always correspond exactly to the start and end time of actual shift types.
Compared to shift type requirements, the floating requirements method changes the size and structure of the problem. On one hand, the problem definition becomes more intricate and the complexity of constructing feasible solutions increases. We must find a satisfactory combination of the shift types used in the hospital to fulfil the floating requirements. On the other hand, the search space is considerably enlarged. This formulation creates an extra degree of freedom to construct a high quality timetable, because the floating requirements can usually be met with different combinations of shift types.
Considering the shift types given in Table 6.1, the period from 7:00 till 21:00 can be covered with a {Short Early - Late} combination but also with an {Early - Short Late} set. In the traditional shift type approach, the planner has to determine the shift type combination as part of the input data, thus restricting the number of possible solutions. By switching between satisfying shift type combinations we can try to improve the personal timetables (see Section 6.4.2).
For the representation of floating personnel requirements, we need:

- The shift types with their start and end times (e.g. Table 6.1).
- A depiction of which shift types can form legal sequences: *joining together tightly*. Shift types with this 'joining' relationship are those which are considered consecutive in terms of time. The implication for the floating personnel requirements approach is that 'joining' shift types can be replaced by another shift type covering the time intervals which were covered by both individual shift types. Two shift types with a common start and end time are considered to have that relationship. In practical applications, however, it is often the case that time gaps or overlaps are not considered to be restrictive. This is explained later on in Section 6.5.
- The number of personnel needed for each skill category q in terms of time intervals ti and days d (called $RIM_{q,ti,d}$ for minimum requirements and $RIP_{q,ti,d}$ for preferred requirements).

An elementary real world example (Dataset 1) is used to explain the formulation of floating personnel requirements. The problem consists of 6 different shift types, presented in Table 6.1.
In practice, legal sequences of shift types are not always equally obvious as in this example. Sometimes a gap or an overlap between consecutive shifts does not cause problems at all. In many real world situations, an overlap in time is required to consider shift types as joining together tightly. It is often necessary for 2 colleagues to have a discussion between shift changes. This can lead to situations in which a 7:00-15:00 cannot be replaced by a 7:00-11:00 – 11:00-15:00 couple. There are other examples in hospitals where it is no great matter to have

Join tightly	Short Early	Early	Day	Late	Short Late	Night
Short Early				x		
Early					x	
Day						
Late						x
Short Late						x
Night	x	x				

Table 6.2: Shift types that join together tightly in Dataset 1

an intermission between shifts. Cleaner's and Nurse Aid's tasks, for example, are not necessarily uninterrupted. In order to construct good timetables, it is important to know which shift types can precede or follow others without affecting the hospital activities.
In the Dataset 1 example, the shift types that join together tightly are those that are considered to be consecutive in terms of time (see Table 6.2) it is the simplest case which hardly ever occurs in practice. To reduce the complexity of the example, we assume that the situation described in Table 6.2 holds for all the skill categories. Table 6.3 presents the personnel requirements per day of the week. Both the minimum and the preferred number of required personnel are given, in the columns 'RIM' and 'RIP' respectively.

6.4 A Two-Step Approach

The method discussed in this chapter is a two-step approach towards a high quality timetable. The goal of the first step is to find a roster that satisfies the personnel requirements, without taking into account the soft constraints on the personal schedules. In the next step, an efficient hybrid tabu search algorithm is applied, in which the required shift types are set. The details of this algorithm are explained later on in Section 8.3.6. The meta-heuristic algorithms will never violate the hard constraints in their process of finding a schedule matching as many soft constraints as possible.

6.4.1 Initialisation

The initialisation phase employed in the more traditional shift type approach is maintained in the current algorithms (see Section 5.4). One extra initialisation step has to be performed to translate the personnel requirements from time intervals to shift types. The consistency check algorithm, which was introduced in the previous Chapter (Section 5.2), gives guidelines to repair infeasibilities but cannot translate the subjective requirements in practical situations. The result of this step is the input for the regular initialisation step (see Fig. 6.3 in Section 6.4.3). The procedure is explained in the latter part of this section.

	Head Nurse				Regular Nurse				Nurse Aid			
	From	Till	RIM	RIP	From	Till	RIM	RIP	From	Till	RIM	RIP
Monday	8:00	17:00	1	1	0:00	7:00	1	1	8:00	17:00	1	1
					7:00	13:00	2	3				
					13:00	21:00	2	2				
					21:00	24:00	1	1				
Tuesday	8:00	17:00	1	1	0:00	7:00	1	1	8:00	17:00	1	1
					7:00	13:00	2	3				
					13:00	21:00	2	2				
					21:00	24:00	1	1				
Wednesday	8:00	17:00	1	1	0:00	7:00	1	1	8:00	17:00	1	1
					7:00	13:00	2	3				
					13:00	21:00	2	2				
					21:00	24:00	1	1				
Thursday	8:00	17:00	1	1	0:00	7:00	1	1	8:00	17:00	1	1
					7:00	13:00	2	3				
					13:00	21:00	2	2				
					21:00	24:00	1	1				
Friday	8:00	17:00	1	1	0:00	7:00	1	1	8:00	17:00	1	1
					7:00	13:00	2	3				
					13:00	21:00	2	2				
					21:00	24:00	1	1				
Saturday					0:00	13:00	1	1	8:00	17:00	1	1
					13:00	15:00	2	2				
					15:00	24:00	1	1				
Sunday					0:00	13:00	1	1	8:00	17:00	1	1
					13:00	15:00	2	2				
					15:00	24:00	1	1				

Table 6.3: Minimum (RIM) and Preferred (RIP) personnel requirements of Dataset 1 for a period of one week and for three different skill categories

Initialisation in the shift type approach:

The only aim of the initialisation step is to construct a feasible solution. The quality of the solution is not taken into account because the scheduling algorithms (described in Section 6.4.2) can cope with any input, as long as it does not violate any hard constraints. ANROM provides three options for constructing the initial schedule (as presented in Section 5.4).

Initialisation in the floating personnel requirements approach:

The main goal is to find, for every day of the planning period, shift type combinations that fulfil the personnel requirements. To tackle this particular phase of the timetabling problem, we enumerate the solutions of the linear program described in Fig. 6.1. So far we have not encountered problems of the translation from floating requirements into shift type combinations that cannot be solved with an exact method. The algorithm provides all the possible shift type combinations fulfilling the personnel requirements. In the linear problem, we try to find the sets of shift types for which, at each point in time, the personnel requirements are satisfied without a surplus of personnel. If the set contains shift types that 'join together tightly' without an exactly matching start and end time, corrections are taken into account. In practice, the initialisation algorithm randomly chooses shift type combinations among the enumerated possibilities. The method used to translate the floating personnel requirements into shift type combinations, described in Fig. 6.1, will only work if there exists at least one shift type combination that matches the floating requirements. A problem arises when there is no such shift type combination because in that case the linear programming approach will not produce any solution at all. We could consider developing a more flexible heuristic for the initialisation phase in order not to have a dead end. This more flexible heuristic could provide a graceful degradation of the system, by producing a shift type combination that comes as close as possible to the floating requirements. The planners in practice find it better, however, to get a warning message when the requirements are infeasible. They expect guidelines on how to make their personnel requirements realistic (and thus translatable to shift types). This advisory warning helps the planners to increase or diminish the personnel requirements in a certain time interval, in order to find at least one shift type set satisfying the requirements. In addition to the consistency check which was introduced in Section 5.2, ANROM provides an extension which can deal with floating personnel requirements. This particular algorithm gives feedback to the users about any inconsistent hard constraints in terms of floating personnel requirements.

6.4.2 Improving the quality of the schedule

In this part of the nurse rostering algorithm, meta-heuristics are applied to the preliminary schedule in order to reduce violations of the soft constraints. For this step we can apply the same algorithms as for the shift type personnel requirements (see PART III).

SOLVE THE LINEAR PROBLEM:
ENUMERATE all possibilities for SOLUTION
SUBJECT TO
DIFFERENCE[y] $\leq$ *0* $\forall y$
in which

- *SOLUTION: list of length S giving the number of appearances for each corresponding shift type in the solution.*
- *DIFFERENCE: list of length TI, for each element x depicting the difference between RIM[x] and AVAILABLE[x]*
- *RIM: ordered list of length TI containing the personnel requirements for the corresponding time in PIT.*
- *PIT: ordered list of length TI containing all the start and end times of the time intervals in the floating requirements in addition to all the start and end times of the shift types, duplicates are removed.*
- *AVAILABLE: list of length TI, for each element x giving the number of personnel scheduled at time PIT[x] according to SOLUTION and taking RELAXATION into account*

$$\mathit{AVAILABLE}[x] = \sum_{s \in SOLUTION} (\mathit{SOLUTION}[s] + \mathit{RELAXATION}[x]) * ((shift\_start_s \leq \mathit{PIT}[x]) \text{ AND } (\mathit{PIT}[x] < shift\_end_s))$$

- *RELAXATION: list of length TI, the elements of the list give the relaxation of the personnel requirements according to the JOIN MATRIX.*

$\forall\ a,\ (1 \leq a \leq S),\ \forall b,\ (1 \leq b \leq S),\ and\ \forall\ x \in PIT:$
IF JOIN[a][b]=1
IF($shift\_end_a < x$ *AND* $x \leq shift\_start_b$*)*
THEN RELAXATION[x]=min$\{$*SOLUTION[a],SOLUTION[b]*$\}$
ELSE
IF($shift\_start_b \leq x$*) AND (*$x < shift\_end_a$*)*
THEN RELAXATION[x]=-min$\{$*SOLUTION[a],SOLUTION[b]*$\}$
ELSE RELAXATION[x]=0

- *JOIN: 1/0 matrix with dimension S*S depicting the shift types which join together tightly.*

Figure 6.1: Linear problem for the initialisation phase in case of 'floating' personnel requirements, we only present the minimum requirements: RIM

Meta-heuristics in the shift type environment:

The shift type requirements formulation is less complex than the floating requirements approach in ANROM. After the initialisation phase which satisfies all the hard constraints of the problem, we only apply algorithms maintaining the solution of the hard constraints, in a shift type schedule.
The aim of the timetabling algorithm is to reorganise the assigned shifts in order to diminish the value of the cost function. We will apply the meta-heuristics that are developed for the shift type requirements, especially the hybrid tabu search algorithms (see Section 8.3.6) implemented in the software package based on ANROM. The tabu search algorithm uses an environment where shifts can be moved from one person to another on the same day (see 'single shift-day' neighbourhood in Section 7.3.1). This step will be referred to as a 'move'. The only restriction on the moves is to conserve the satisfaction of hard constraints. A shift for a certain skill category can thus not be moved to a person who is not qualified to do it. The move of a shift to a person who is already assigned to this shift on the day considered is also forbidden. We explain the details of the moves in detail in Section 7.3.1.

Meta-heuristics in the floating requirements environment:

In this compound algorithm, shift types in the personal schedules will be moved from one person to another ('moves' as briefly mentioned above are explained in detail in Section 7.3.1), while the shift type combinations satisfying the personnel requirements will be varied ('swaps', see Fig. 6.3 in Section 6.4.3). In this alternating system, the possibility of satisfying the personnel requirements with different shift type combinations enlarges the solution space considerably, affecting the calculation time to a very high extent. In order to keep the computing time down we have tuned the alternation of 'moves' and 'swaps' experimentally by adjusting the stop criteria for each of them (see Fig. 6.2). The planning algorithm starts with the tabu search 'moves' (for a detailed description, see Section 8.3.1) until the stop criterion for the moves is reached (a number of iterations without improvement). Instead of switching to the hybridisations, in the floating personnel requirements approach, we allow for a diversification by applying these swaps in the schedule. For every day of the planning period, we search all possible alternatives for the shift type combinations. The best one of these swaps will be performed in any case (even if the quality of the timetable deteriorates). The cost function (Chapter 4) is applicable to the floating requirements without any modification because the schedules are set up with shift types. It allows for a quick calculation of the best people to assign the new set of shift types to. Suppose, for example, that person A works during the period 8:00-17:00 and that we want to swap that shift type to 8:00-12:00 - 13:00-17:00 (provided they are defined as joining together tightly). Our algorithms will find the best, in terms of the cost function, personnel pair B and C to carry out the 8:00-12:00 and the 13:00-17:00 shift type. Until a swap worsens

```
INITIALISE schedule X (result of Fig. 6.1)
BEST_SCHEDULE=X; number_steps=0

  WHILE (number_steps < maximum_number_steps)
    number_moves=0; number_swaps=0; weekend_step=0; worst_personal_schedule=0
    WHILE (number_moves < maximum_number_moves)
      X'=move(X)
      IF (f(X') < f(BEST_SCHEDULE))
        number_moves=0
        number_steps=0
        BEST_SCHEDULE=X'
      ELSE
        number_moves=number_moves+1
        number_steps=number_steps+1
      X=X'
      END
    WHILE (number_swaps = 0)
      X'=swap(X)
      IF (f(X') < f(BEST_SCHEDULE))
        number_steps=0
        BEST_SCHEDULE=X'
      ELSE
        number_swaps=number_swaps+1
      X=X'
      END
    IF (number_weekend_steps < maximum_number_weekend_steps)
      X'=WEEKEND_STEP(X)
      IF (f(X') < f(BEST_SCHEDULE))
        number_steps=0
        BEST_SCHEDULE=X'
      ELSE
        number_weekend_steps=number_weekend_steps+1
      X=X'
      END
    ELSE
      WHILE (worst_personal_schedule=0)
      X'=WORST_PERSONAL_SCHEDULE(X)
      IF (f(X') < f(BEST_SCHEDULE))
        number_steps=0
        number_weekend_steps=0
        BEST_SCHEDULE=X'
      ELSE
        worst_personal_schedule=worst_personal_schedule+1
      X=X'
      END
    END
  BEST_SCHEDULE=GREEDY_SHUFFLING(BEST_SCHEDULE)
```

maximum_number_steps, maximum_number_moves, maximum_number_weekend_steps are calculated before the algorithm starts, as function of the dimensions of the search space, f denotes the cost function

Figure 6.2: Heuristics for the scheduling phase when using 'floating' personnel requirements

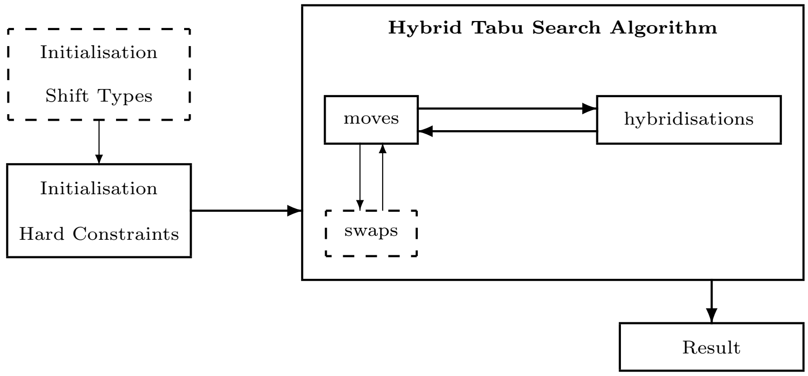

Figure 6.3: Diagram of the heuristics for the nurse rostering problem with 'floating' personnel requirements

the schedule, the swap step will be repeated.
After swapping, it is very likely that some tabu search moves will enable an improvement of the schedule again. This combined process of moves and swaps is repeated until another stop criterion, calculating the iterations without improvement, is reached. Depending on the problem characteristics and on the wishes of the planner, the next step is one of the hybridisations described above. The dimensions of the problem, (the number of personnel to be scheduled, the number of different shift types, the duration of the planning period, etc) influence the size of the solution space. The dimensions will depict the overall stop criterion as well as the stop criteria on the moves and hybridisations. Fig. 6.2 demonstrates the most advanced option, in which all the hybridisation steps (explained in Section 8.3.6) are executed.

6.4.3 Diagram of the modules

In this section we demonstrate where the newly developed parts of the algorithm are situated. The initialisation and hybridisations, which are summarised in Section 5.4 and 6.4.2 (and fully elaborated in Chapter 8) are represented by a single frame in Fig. 6.3.
The diagram only shows that part of the scheme which is affected by the floating personnel requirements. In the case where the personnel requirements are expressed as floating requirements, the pieces in dashed frames are employed. The ANROM model can still be used in the original way (with shift type personnel requirements) and it then simply skips the parts of the algorithm represented by dashed boxes in the diagram.

6.5 Examples

Considering the example given in Tables 6.1, 6.2, and 6.3, there are 3 possibilities to combine shifts for weekdays, for the minimum as well as for the preferred requirements. If we have a schedule that does not violate either of them, six different combinations of shift types can satisfy the (Regular Nurse's) personnel requirements on the weekdays Monday to Friday (see Table 6.4). Both the

	RIM			RIP		
	RIM C1	RIM C2	RIM C3	RIP C1	RIP C2	RIP C3
Short Early	2	1		3	2	1
Early		1	2		1	2
Day						
Late	2	1		2	1	
Short Late		1	2		1	2
Night	1	1	1	1	1	1

Table 6.4: Possible solutions for the Regular Nurses on a weekday (Dataset 1) RIM Cx: Minimum personnel requirements shift type Combination; RIP Cx: Preferred personnel requirements shift type Combination; the index x denotes the number of the combination

	RIM	RIP
Short Early		
Early	1	1
Day		
Late	1	1
Short Late		
Night	1	1

Table 6.5: Possible solutions for the Regular Nurses on a weekend day (Dataset 1)

possible shift type combinations for the minimum personnel requirements (denoted by MC) and for the preferred personnel requirements (PC) are given. The results for the weekend requirements are presented in Table 6.5. In this table, we can see that only one combination satisfies the personnel requirements. To study the mechanism of the swaps, we have counted (in a period of one month) the number of appearances of each shift type combination (initially and after the algorithm). The results for the weekdays only are displayed in Table 6.6. The swaps in the algorithm have given preference to the second solution (in Table 6.6). This is most probably due to the character of the soft constraints on personal schedules. In the Dataset 1 example, there was a restriction on the

	RIM C1	RIM C2	RIM C3
Initially	6	8	6
Result after the algorithm	2	13	5

Table 6.6: Appearance of the shift type combinations in the initial solution and in the final result (Dataset 1); RIM Cx: Minimum personnel requirements shift type Combination with index x

Join tightly	EE	SE	E	D	SD	L	SL	LL	N
EE						x	x		
SE				x					
E								x	
D								x	
SD									
L									x
SL								x	
LL									x
N	x	x							

Table 6.7: The shift types that join together tightly in Dataset 2

maximum number of each shift type a person could work during the planning period. The solutions MC1 and MC2 both combine a lower number of shift types, which could in some circumstances lead to violations of the particular constraint mentioned.
The next example (all information is in Tables 2.4, 6.7, 6.8 and 6.9) is also taken from the real world, the shift types are represented by an abbreviation from the names given in the particular hospital. The example illustrates the extra difficulty of the *joining tightly* constraint of shift types. It is easy to understand that the shift types SE and D join together tightly. SE lasts till 10:00 and D starts

From	Till	Requirements
00:00	06:00	1
06:00	08:00	2
08:00	17:00	4
17:00	18:00	3
18:00	22:00	2
22:00	24:00	1

Table 6.8: Personnel requirements for a single qualification on one day (Dataset 2)

Combinations	C1	C2	C3	C4	C5	C6	C7	C8
EE		2	1			1	2	2
SE	2		1			1		
E	2	2	2	2	2	2	2	2
D	2		1			1		
SD								
L		2	1		1			1
SL				2	1	1	2	1
LL	2		1	2	1	2	2	1
N	1	1	1	1	1	1	1	1

Table 6.9: Possible shift type combinations satisfying the daily personnel demand of Dataset 2

$$\begin{array}{rcl} SL + LL & \leftrightarrow & L \\ EE + L & \leftrightarrow & SE + D + LL \\ \\ & \Downarrow & \\ EE + SL & \leftrightarrow & SE + D \end{array}$$

Figure 6.4: Possible swaps between shift types of Dataset 2

at 10:00. The impact of this characteristic is that replacing one person working before and after 10:00 by two people who switch shifts at 10:00 does not affect the hard constraints of the work schedule. In the example, the only exceptions to the expected shift-type-joins are D-LL and SL-LL, as can be derived from Table 6.7. The joins are exceptional because both D and SL last until 18:00 and LL starts at 17:00. Fig. 6.4 illustrates possible swaps between shift type sets. A nurse working the L shift (from 14:00 till 22:00) can be replaced by two colleagues working the SL shift (from 14:00 till 18:00) and the LL shift (from 17:00 till 22:00) respectively, because of the allowed sequences. The opposite swap is more significant, because, if we replace two nurses (both in the SL and the LL shift) by one doing the L shift; the number of personnel in the ward between 17:00 and 18:00 is reduced by one.
Fig. 6.4 also demonstrates how 2 people (doing the EE and L shift) can be replaced by 3 people in the SE, D and LL shift, and vice versa. From these allowed 'swaps' in shift types, others can be derived, as shown in the last line of Fig. 6.4.

6.6 Test Results

Solutions that are not within reach (not satisfying the hard constraints) in the classic approach defining shift type personnel requirements, can be constructed by the swap steps in the floating requirements approach. Although there is no direct and simple way of comparing this approach with those of Chapter 8, we have specifically developed a method to compare both approaches. First, we run tests on shift type and floating requirements datasets, each with the proper algorithms for the problem type. Afterwards, tests have been carried out on the two kinds of datasets, traditional algorithms (eliminating the swaps) for the floating requirements dataset and floating algorithms (enabling swaps) for the shift type dataset.

Schematically, both kinds of experiments, set up for comparison reasons, are:

- Allowing 'swaps' in a ward with traditional shift type personnel requirements.
- Omitting the 'swaps' in a ward with 'floating' personnel requirements.

We now compare the following algorithms:

- Tabu Search Algorithm constructed for the traditional shift type personnel requirements (Section 8.3.6).
- Algorithm for the 'floating' personnel requirements (subject of this chapter).

For the purposes of comparison, we selected an algorithm used in practice (because of the trade off between time taken and quality of solution) and not the most complex algorithm described in Chapter 8. The scheduling algorithm chosen is the TS1 algorithm (described in detail in Section 8.3.6). TS1 consists of the tabu search moves combined with the 'complete weekends' and 'worst personal schedule' hybridisations. They will be discussed in detail later on in Section 6.4.2. It is the aim of the model explained in this chapter to demonstrate the possibility of formulating a time interval problem and producing a shift type solution of high quality. The results of the experiments with data obtained from schedulers applying ANROM are presented in Tables 6.10 and 6.11. The column Value represents the value of the evaluation function. The duration of the calculations on an IBM Power PC RS6000 is given in the column Time.
Test sets can be downloaded from http://extern.kahosl.be/greet.vandenberghe/. For reasons of confidentiality, the available data sets do not correspond to any of the hospitals that apply the software based on ANROM.
It was necessary to reformat the input data slightly for these tests. We will explain here how the user data constructed for the shift type formulation is made fit for the floating approach and vice versa. In the traditional problem 1 (TP1: 12 people, 6 shift types), we defined the shift types which join together tightly in the most careful way. We forbid gaps and overlaps between shift types and only allow shift type swaps if they cover exactly the same time period. For the

Traditional Problem		RM		RM - RP	
		Value	Time	Value	Time
TP1	Traditional Algorithm	643	8’27”	618	8’31”
	Floating Algorithm	511	1h15’36”	500	1h18’07”
TP2	Traditional Algorithm	1862	21’44”	1715	22’13”
	Floating Algorithm	1009	3h57’12”	932	5h24’30”

Table 6.10: Test results of the Traditional and Floating Algorithm on a problem with a traditional personnel demand formulation

second problem (TP2: 20 people, 8 shift types), we constructed more possibilities for the swaps. There is a case in which we allow a swap with a 30’ overlap and legalise another swap with a 60’ gap.
The test data formulated as ‘floating’ requirement problems, already introduced on previous pages as Dataset 1 and Dataset 2, required reformatting in order to match the shift type formulation. Translating the floating requirements into shift type personnel requirements, which is necessary to create an input for the traditional algorithm, was performed in two different ways. FP1 and FP2 are slight variations on Dataset 1; FP3 is the Dataset 2 problem. In the first approach (FP1), the requirements are equal to the combination of shift types resulting from the branch and bound algorithm (the first solution). Several days of the planning period have exactly the same ‘floating’ requirements and therefore the result in terms of shift type requirements will be the same on these days. The FP2 data set was constructed from the same original set as FP1, but it differs in the translation to shift type requirements. Instead of taking the first solution, we construct a random solution for every day of the planning period. Like FP2, the daily shift type combinations of FP3 are chosen randomly from the possibilities in Dataset 2.
Experiments matching the minimum personnel requirements exactly (‘Minimum Requirements’ in the Tables 6.10 and 6.11) and with a feasible domain between the minimum and the preferred personnel requirements (‘Min-Pref’, as explained in Section 5.6.1) obviously lead to different results. As will be stated in Chapter 9, tests prove that minimising the calculation time and maximising the solution quality are not compatible.
The calculation time for the Floating Algorithm is much higher than for the Traditional Algorithm. The conclusion holds for both experiments (Table 6.10 and 6.11). This is according to our expectations because the number of possible solutions is increased considerably by not restricting the schedule to a given shift type combination. In all the test examples, the quality of the result is better for the Floating Algorithm. Splitting long shift types into two or more shorter shifts and assigning them to different people, or vice versa, can overcome soft constraint obstacles in the personal schedules.
Although the FP1 and FP2 data set of Table 6.11 are basically the same, it is not surprising that the Traditional Algorithm produces better results for the

'Floating' Problem		RIM		RIM - RIP	
		Value	Time	Value	Time
FP1	Traditional Algorithm	1555	4'20"	1456	4'34"
	Floating Algorithm	1197	1h47'08"	1014	3h59'46"
FP2	Traditional Algorithm	1799	5'54"	1714	6'04"
	Floating Algorithm	1197	1h47'08"	1014	3h59'46"
FP3	Traditional Algorithm	2641	17'27"	2598	18'04"
	Floating Algorithm	1622	5h08'48"	1014	12h57'27"

Table 6.11: Test results of the Traditional and Floating Algorithm on a problem with a 'floating' personnel demand formulation

FP1 variant. Since the shift type combinations are chosen randomly in FP2, the initialisation results in a wide variety of daily shift type combinations. The Traditional Algorithm has no swap steps thus the scattered shift type combinations will be maintained during further calculations. Many soft constraints (e.g. the minimum number of consecutive shifts of the same type) are much easier to solve when the shift types on consecutive days are equal.
The considerable increase of the solution space, due to the high number of shift type combinations satisfying the requirements, reveals itself in the calculation time of FP3. No further comparison with other approaches is possible because problems similar to this one have not been solved before.

6.7 Conclusion

The formulation of 'floating' personnel requirements simplifies the practical use of the nurse rostering model, in that it corresponds very well to the real world situation. The developed approach, which deals with the expanded nurse rostering problem, produces even better results than approaches developed for the less complex problem.
Floating personnel requirements have been identified to reflect the particular difficulties that hospital planners face when automating their personnel rostering process. Users of planning software were asked to carefully define their personnel requirements in order to avoid infeasibilities in the translation to shift types. The floating requirements mechanism simplifies this task. The input data for the personnel rostering software with floating personnel requirements matches the real world practice in hospitals better. Moreover, work can be structured more around patients' needs and thus unproductive time will often be reduced.
Enabling a considerably higher number of shift type combinations to staff hospital tasks provides more possibilities for individual personnel members to satisfy their private needs and wishes. In spite of being more time consuming than rostering personnel problems on a shift type base, the 'floating' personnel

approach induces much better quality schedules, with respect to the personal constraints of the staff (the soft constraints). The search space with feasible solutions is considerably larger than it is when shift type personnel requirements are defined. The higher degree of freedom allows for tackling particular soft constraint problems by making the most of different shift type combinations in the construction of the timetable. It allows for many kinds of part time employment without requiring restrictive decisions on shift type combinations from the personnel manager or planner.
Experiments have shown that both personnel and hospitals benefit from this new approach. Everything which is possible with the meta-heuristics developed for shift type requirements (Chapter 7-10) remains possible with the current methods. In fact, the search space in the shift type approach is a subset of the search space for floating requirements. 'Floating' personnel requirements are an important improvement of the shift type system. This approach provides a higher level of personnel satisfaction, creates plenty of possibilities for part time employment, and leads to efficient and flexible organisations.

The benefits of allowing floating personnel requirements have been demonstrated in this chapter. Both the hospital and the patients are served better, but especially for the personnel, this new model offers plenty of possibilities for part time contracts and for combining personal objectives with the organisational requirements. All the meta-heuristics in PART III of this thesis can be carried out both with shift type and floating requirements and are thus applicable in a very wide range of personnel rostering environments.

Part III

Meta-Heuristics and Hybrids

Chapter 7

Variable Neighbourhood Search

7.1 Introduction

The nurse rostering meta-heuristics discussed in PART III of this work are not aimed at specific individual hospitals. On the contrary, the intention is that such algorithms should be applicable across the whole healthcare sector. Applying nurse rostering algorithms to real-world problems often involves very generic heuristics to deal with widely varying hospital customs and requirements. Escaping from local optima can be very hard for the search algorithms because of the broad variety of constraints. Some constraints refer to particular duties of the nurses while other constraints restrict consecutive shifts, days, weekends, etc (see Chapter 2).
The research presented in this chapter attempts to make more use of problem specific characteristics to dynamically change the search heuristics and their neighbourhoods in order to overcome some typical drawbacks of meta-heuristics for complex combinatorial problems. Hidden parts of the solution space become accessible by applying appropriate problem specific neighbourhoods. The method allows for a better exploration of the search space, by combining short sighted neighbourhoods, and very greedy ones. Experiments demonstrate how heuristics and neighbourhoods can be assembled for finding good quality schedules in a relatively short calculation time.
Other meta-heuristics (see Chapter 8 and 9) developed for solving the nurse rostering problem of ANROM apply the neighbourhoods which are introduced in this chapter in their local search algorithms. While studying many different real-world implementations of the rostering problem, cases appeared in which previously developed heuristics did not manage to overcome some difficulties that originate from very particular constraint combinations. The overall quality of the resulting schedule in terms of the cost function is not necessarily bad but it is hard for hospital planners to accept results, which they can make more

acceptable by some very small manipulations to the schedule by hand. The characteristics of the constraints are so different good quality solutions are very hard to find in the search space.
The main ideas of this variable neighbourhood approach were published as *E.K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe: Variable Neighbourhood Search for Nurse Rostering Problems, Proceedings of 4th Metaheuristics International Conference, MIC2001, Porto, 2001, 755-760* [38].

The chapter is organised as follows. We introduce variable neighbourhood search in Section 7.2. Different neighbourhoods for the search heuristics are defined in Section 7.2. In Section 7.4, we explain how the heuristics can be combined in order to reach results that might remain behind big barriers when using single neighbourhood strategies. A few ideas to combine, repair and restart different heuristics applying several neighbourhoods are explored. The results of the developed variable neighbourhood algorithms are discussed in Section 7.5, and we conclude in Section 7.6.

7.2 Variable Neighbourhood Search

The idea of changing neighbourhoods while performing a meta-heuristic search was already introduced by Glover [103], (1986), as a means for increasing the performance of the algorithms. He suggests diversification strategies to explore the search space of combinatorial problems effectively. It is often the only possible way to reach regions behind barriers in the landscape of solutions. Variable neighbourhood search [148] combines local search heuristics, which stop in local optima, and neighbourhood changes to escape from these local optima. The approach is applicable in combination with meta-heuristic algorithms as a diversification strategy for the local search.
Variable neighbourhood search has been applied to several NP hard problems by Hansen and Mladenovic. Examples are the travelling salesman problem, the location-allocation problem, a clustering problem, the bilinear programming problem with bilinear constraints [113, 148]. Other applications are: the linear ordering problem (Gonzáles and Pérez-Brito [108]), scheduling problems (Davidovic et. al. [72], den Besten and Stützle [79]), vehicle routing (Crispim and Brandão [67]), the p-median (Crainic et. al. [65], Hansen and Mladenović [113]), the max-cut [93], and the k-cardinality tree problem (Mladenović and Urošević [150]), etc.
Different approaches exist for selecting neighbourhoods and for going from one neighbourhood to another. It is often recommended to *shake* the solution, i.e. to randomly swap to a solution in the neighbourhood of the current one.

7.3 Variable Neighbourhood Search for the Nurse Rostering Problem

For the approach presented in this chapter, a set of neighbourhood structures, which use specific information about the problem, are defined. When a search heuristic fails to improve the solution (within a certain amount of time or a number of iterations), the algorithm dynamically chooses a different neighbourhood.

In ANROM, the personnel requirements are hard constraints and we call all the solutions satisfying them 'feasible' solutions. Any solution must provide a sufficient number of qualified personnel at any time during the entire planning period. In all the meta-heuristics developed for this nurse rostering problem, we remain in the feasible part of the search space during the iterations. Consider the matrix representation of Fig. 4.1. Feasibility with respect to the coverage constraints (Section 2.3.3) can only persist if no other than vertical displacements of assignments are allowed. In order to satisfy the hard constraints on required skills (Section 2.3.2), a procedure in the algorithms prevents assignments from being shifted to unqualified personnel (this was already introduced in the procedures of Chapter 5). To guarantee the satisfaction of the hard constraints, shifts will thus only be moved to another person's schedule on the same day. The moves are not allowed unless the person is qualified to perform the duty and provided this person is not yet assigned to the same shift.

During the search process, the algorithms aim at minimising the number of violations of the soft constraints taking cost parameters into account (see Section 4.2.4). The cost function is the motor of the search heuristics, but since it does not interpret the problem characteristics, the algorithms are quite blind to certain improvements. While improving the schedule with respect to one constraint, it might make the solution much worse with respect to others.

Some of the constraints are of particular importance for the research presented in this chapter. We have constructed neighbourhoods in order to especially satisfy a number of these constraints (see Section 7.3.1).

7.3.1 Neighbourhoods for the nurse rostering problem

We introduce a number of different neighbourhoods, which enable the heuristics to search for good solutions with respect to the evaluation function of Chapter 4. We expand this group with sets of new neighbourhoods related to soft constraints, and greedier neighbourhoods, which are inspired by manual scheduling processes.

Single shift-day (D)

The simplest neighbourhood of a schedule includes all the feasible solutions that differ in the position of one scheduled shift. It is the basic neighbourhood for all the meta-heuristic approaches (Chapter 7 - 10) executed on the problem described in Chapter 2. Note that the position refers to the personnel member

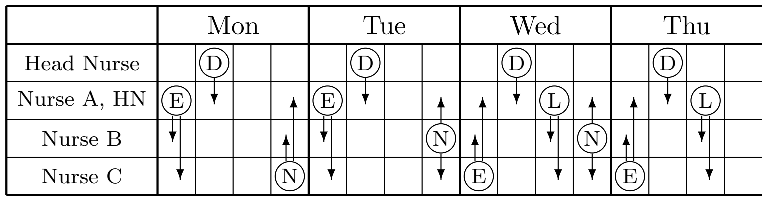

Figure 7.1: Possible moves in a *single shift-day* neighbourhood; Shifts are Early (E), Day (D), Late (L), and Night (N)

whose schedule the assigned shift belongs to. The single shift-day neighbourhood considers the solutions in the nearest environment of the current solution. In order to create the neighbourhood, it suffices to consider all the allowed displacements of a scheduled shift from the personal schedule, which contains the shift type, to another person's schedule which is empty for that shift type on the same day. We will further refer to these displacements as 'moves'. The solution corresponding to that move does not violate the hard constraints, provided the 2nd person is qualified to work that shift type. Fig. 7.1 presents the allowed moves in the single shift-day neighbourhood. A very small ward, consisting of one head nurse and three regular nurses, is presented. One of the regular nurses has the head nurse skill as an alternative. This will be the person to replace the head nurse during absence. A very small part of a realistic planning period is shown and there are four shift types: Early (E), Day (D), Late (L), and Night (N). Arrows demonstrate the possible moves in the neighbourhood. Note that the head nurse's Day shifts cannot be moved into the schedules of Nurse B and C because that would violate the hard constraint on skills. Neither can shifts for the regular nurses (Nurse A, B, and C) be moved to the head nurse's schedule. Shifts cannot be moved horizontally in the schedule either because that would disturb the equilibrium between required personnel members and scheduled ones.

Soft constraint related neighbourhoods

The neighbourhoods introduced in this section are not comparable to the others because they perceive the landscape of the solution space in a different way. While searching schedules, which better satisfy one particular soft constraint, the algorithms are blind to the overall quality of the result. This is one of the main reasons why such neighbourhoods are not applied in the final phase of a search algorithm.
The inspiration for developing these neighbourhoods comes from real-world suggestions from hospital planners (they wanted the violations on weekend constraints to be reduced in the solution for example). Looking at an automatically generated schedule, they can point out shortcomings with respect to some sensitive constraints. Solving these problems does not necessarily lead

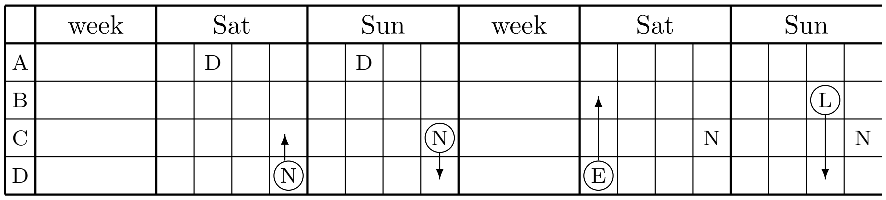

Figure 7.2: Possible moves in a *weekend* neighbourhood

the search to a very interesting part of the solution space but it generally does not hinder the search either.
Although it is against the philosophy of working with abstractions of the individual constraints in the search for better solutions (see also [36]), we propose the use of these neighbourhoods even if they only act as a means of diversification.
Examples of such soft constraint related neighbourhoods are:

Weekend neighbourhood (W)
This neighbourhood consists of all the solutions differing from the current solution by the assignment of one shift on a day of the weekend. This weekend neighbourhood is of importance only in the case where the constraint of 'complete weekends' (Constraint 15) is applied to at least one of the personnel members. The weekend neighbourhood is empty if the constraint on complete weekends is fully satisfied. If it is not completely satisfied, all the personal schedules which are subject to the complete weekends constraint have one violation of this constraint less than the current solution. Fig. 7.2 illustrates the possible moves in the weekend neighbourhood for a very simple example.

Overtime - Undertime neighbourhood (OU)
This neighbourhood only considers moving shifts from people with overtime (violation of Constraint 8) to people with undertime (violation of Constraint 9 or people for which additional assignments do not generate a penalty on overtime). An extension of this neighbourhood includes all the moves that do not increase the sum of the overtime and undertime violations in the schedule.

Alternative qualifications neighbourhood (AQ)
Experienced people have the authority to carry out work for other skill categories in order to replace absent personnel members. However, it is better for the quality of a schedule when the number of replacements is low (Constraint 2). This neighbourhood consists of all schedules which have one assignment, that involves a skill category replacement, less.

Personal requests neighbourhood (PR)
The soft constraint on personal requests (Constraint 24, 25, and 26) has a

modifiable cost parameter, like all other soft constraints. In many circumstances, the result of the scheduling algorithm will violate a few of these constraints. The nature of the cost function (which sums the violations of soft constraints) guarantees a solution, which is not biased towards solving a particular constraint.
Nurses can be very sensitive about their personal request for a certain shift or day off. This particular neighbourhood has been developed to search for solutions which satisfy the personal requests. By moving from one solution to another in this personal requests neighbourhood, the size of the neighbourhood should decrease. Ideal schedules without penalties for personal requests have this neighbourhood empty.

The most violated constraint neighbourhood (MV)
The modular nature of the cost function allows for isolating constraints. This neighbourhood pays more attention to moves affecting one particular constraint, namely the constraint that is violated to the highest extent. We consider the number of violations per constraint in every personal schedule in order to determine the most violated constraint. The MV neighbourhood itself contains all the solutions of the simplest neighbourhood (single shift-day) but the evaluation function temporarily takes a higher value for the cost parameter of the most violated constraint. By doing this, the search will be guided towards different parts of the search space. After having applied the MV neighbourhood, the cost parameters are set back to their original values.

Swapping large sections of personal schedules

The system based on ANROM allows hospital schedulers to change the schedule manually. Their manipulations often have the aim of creating schedules, which are visually more satisfying. This inspired us to design this category neighbourhood, in which we try to imitate very common real-world manipulations of schedules.
Unlike the previous group of neighbourhoods, in which neighbouring solutions only differ in the position of one single shift type, this set of neighbourhoods looks at schedules which differ considerably from the original solution. Reallocating larger groups of shifts at once is often less harmful for the quality of a schedule than moving single shifts around. The drawback of applying this category of neighbourhoods is that the number of neighbouring solutions is very large, and thus so is the calculation time. Examples are:

Shuffle neighbourhood (SH)
The 'shuffle' environment considers switches from a part of the worst, in terms of the evaluation function, personal schedule with any other schedule. Instead of moving duties (as in the simple single shift-day neighbourhood), all the assignments in a period from one day to a number of days equal to half the planning period, are switched between the person with the worst schedule and another person in the ward. All possible feasible shuffles during the planning period are

considered (see Fig. 7.3 for a part of the shuffle neighbourhood).

Greedy shuffling neighbourhood (GS)
The greedy shuffling environment is comparable to the shuffle environment, but it is much bigger. It consists of all possible shuffles between any set of two people in the schedule. We call this shuffling greedy because the neighbourhood is very large and very time-consuming to evaluate, and also because the steps involve large sections of the schedule.

Core shuffle neighbourhood (CS)
Compared to the shuffle neighbourhood, we apply an extra shuffle, moving an internal part of the shuffle section back (see Fig. 7.4). The core shuffle neighbourhood considers two consecutive swaps between a pair of personal schedules at a time. In the first phase, a move from the greedy shuffling neighbourhood is performed. Within the swapped time interval of that move, a new time interval, also consisting of full days, is swapped back in the second phase. The second interval must start at least one day after the beginning of the first time interval and end at least one day before the other ends.

7.3.2 Shaking the solution

Shaking allows the algorithm to explore the solution space in a random manner. It is defined as the move to a random element of the neighbourhood. Some moves within the 'soft constraint neighbourhoods' already act as shakes. They do not generally improve the overall quality of the schedule but they provide a different viewpoint in the search space. Examples of the other shakes are:

Shake a shift
Making a random move in the single shift-day neighbourhood is seldom an interesting shake. Neither are the chances high that the move will improve the solution or take the schedule to an unexplored area because most of the environment remains unchanged after moving a single shift.

Shake weekends (swap a weekend between two personnel members)
The nature of some weekend constraints often prohibits the single shift-day neighbourhood to remove weekend shifts from a personal schedule. Removing (or adding) a single shift on Saturday or Sunday can create violations of the 'complete weekends' (Constraint 15) and very often also on some consecutiveness constraints. Removing or adding simultaneously a Saturday-Sunday shift pair in a person's schedule can overcome barriers in the cost function. Even if a weekend shake does not improve the quality of the schedule, it has taken the solution into a considerably different area of the search space.

Shake 2 people (swap two personal schedules)
Swapping personal schedules for people with different work regulations will normally make both schedules worse. Even if two people have the same work

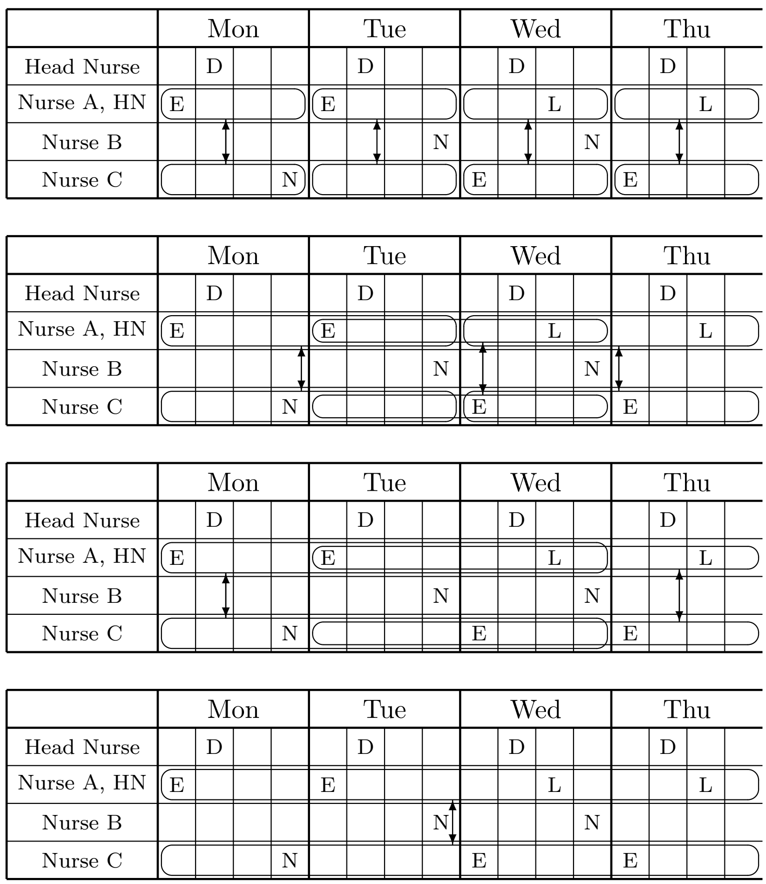

Figure 7.3: Possible moves in a *shuffle* neighbourhood between the personal schedules of Nurse A and Nurse C; for clarity, the moves are presented on 4 instances of the schedule

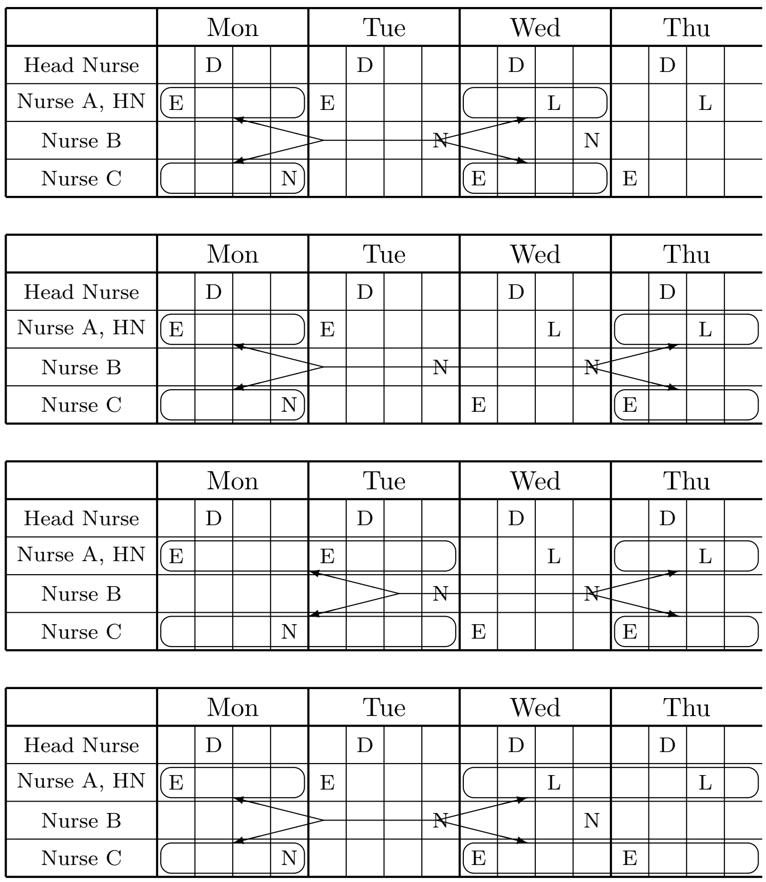

Figure 7.4: Examples of moves between Nurse A and Nurse C in a core shuffle neighbourhood

Neighbourhood		Steepest Descent	Tabu Search
D	single shift-day		x
PR	personal requests	x	
W	weekend		x
MV	worst constraint		x
SH	shuffle	x	
GS	greedy shuffle	x	
CS	core shuffle	x	

Table 7.1: Selection of pairs of neighbourhoods and heuristics for the test results of Table 7.2

regulations, their personal schedules can be very different in terms of the value of the cost function. For the purpose of diversification, this is certainly a good shake. The value of the cost function, however, will rarely drop after this shake.

7.4 Variable Neighbourhood Approaches

We refer to Section 5.4 for the initialisation phase. In the variable neighbourhood experiments presented in this chapter, we applied the last initialisation option, in which all the required shifts are assigned at random. The heuristics start from feasible schedules (see procedure of Fig. 5.7 in Section 5.4.2).
The nurse rostering problem is solved by scheduling each skill category separately as explained in Section 5.5. We opted for applying two different search algorithms to this problem, namely steepest descent and tabu search. The details of the tabu search algorithm will be explained thoroughly in Chapter 8. For both steepest descent and tabu search, the decision for a move is made at random out of the set of equally good solutions. Several layouts have been implemented for swapping between these algorithms, combining different neighbourhoods from the set defined in Section 7.3.1. The experiments which have been carried out in Section 7.5 to test the variable neighbourhood approach have fixed algorithm-neighbourhood pairs. Table 7.1 presents all the combinations. Previous experiments indicated that it is better to combine the single shift-day neighbourhood with tabu search than with steepest descent. In problems of realistic size, there are plenty of non-tabu moves which can guide the search out of the environment of a local optimum, whereas steepest descent ends in the first local optimum. As long as there are improving moves, the search in shuffle neighbourhoods continues. We deliberately combine these neighbourhoods with steepest descent because they generally reach good quality local optima (provided they are explored after a search in a smaller size neighbourhood). Applying tabu search in the shuffle neighbourhoods would increase the calculation time enormously. Experiments have also been carried out with 'shaking' the neighbourhood (Section 7.3.2). Shaking did not generally contribute to finding good quality schedules in preliminary experiments and we

decided not to keep it in the algorithm. Rather, most of the neighbourhoods of the soft constraint class (2nd category in Section 7.3.1), have the effect of shakes since they search improvements for a partial set of soft constraints only. Searching in these neighbourhoods can be seen as a process which reduces the contribution of a particular soft constraint to the overall cost function and is thus much more relevant than random shakes.

The steepest descent algorithm (obviously) stops when the neighbourhood contains no better solution than the current one. The stop criterion for the tabu search algorithm is defined as a number of iterations without improvement (see Section 8.3.4). The number depends on the problem dimensions (number of people, number of shift types, length of the planning period, etc). When the stop criterion for a heuristic and neighbourhood combination is reached, the heuristic switches to another neighbourhood, or to the other search algorithm.

Starting from an initial solution, local search is applied in the first neighbourhood. If the local optimum thus found is better than the current best solution, the algorithm moves there and continues the search in the first neighbourhood; otherwise, it employs the next neighbourhood and applies the corresponding search method. The algorithm stops when the search in the last neighbourhood does not lead to an improvement. Fig. 7.5 presents a schematic overview of the procedure.

```
initialise:
    select a set of algorithm-neighbourhoods pairs
    (neighbourhoods N_k, k = 1 ... k_max)
    set success rate success_k
    define a local stop criterion
    construct an initial feasible solution x
search:
    set k = 1
    WHILE k ≤ k_max
        IF success_k ≥ 1
            explore the neighbourhood until local stop criterion
            is met
            IF the best solution x' is better than x
                x = x'
                k = 1
            ELSE
                success_k = success_k − 1
                k = k + 1
```

Figure 7.5: Pseudo code for the variable neighbourhood approach

Neighbourhoods can be applied in many different orders, which affect the result of the search considerably. It is shown to be the best approach to develop algorithms exploring neighbourhoods with increasing size. Whenever a neighbourhood generates a better solution, the algorithm starts over from the first (finest) neighbourhood, which is generally also the least time consuming. As will be explained in detail in in Chapter 8 and 9, it is always beneficial for use in practice when, after the most greedy step, no finer neighbourhood is explored. The same holds for the core shuffle neighbourhood as it brings the solution into a 'finalised shape'. The nature of the GS and CS environments leads to solutions that are judged positively by the schedulers. Additional moves which do not worsen the quality, might bring the solution into a new area that stimulates hospital planners to explore it manually. The greedy shuffling, and even more so the core shuffling neighbourhood, are not recommended for very large problems when the calculation time is limited. Exploring the entire neighbourhood is (for both approaches) an extremely intensive task.
The soft constraint related neighbourhoods are not equally interesting for every type of problem. We therefore developed a method to avoid those neighbourhoods that never contribute to better solutions. The probability of selecting a particular soft constraint neighbourhood will change during the course of the calculations, depending on the results produced by that neighbourhood. We introduce a parameter $success_k$ which is decreased by 1 each time neighbourhood k does not lead to better solutions (see also Fig. 7.5) When $success_k$ is less than 1, the neighbourhood k will not be applied in later iterations. The single shift-day neighbourhood initially has a very high value for $success_k$ whereas soft constraint related neighbourhoods preferably start with smaller values (1 or 2).
When changing the neighbourhood, it is possible to start from the most recent solution reached in the previous algorithm-neighbourhood combination; or from the overall best solution found. It appeared from experiments that the best solution is always a recommendable start position.
Since the variable neighbourhood search has been applied to real-world problems, we cannot ignore the calculation time. The test data sets are complex and large, and hospital schedulers expect a schedule to be generated within a reasonable calculation time.

7.5 Test Results

Experiments have been carried out on real-world data with different combinations of the neighbourhoods defined in Section 7.3.1. Test data can be obtained from the website http://extern.kahosl.be/greet.vandenberghe/. Depending on the nature of the test data (whether certain soft constraints are applied or not, whether the corresponding cost parameter is high, etc) the effect of the neighbourhoods corresponding to soft constraints is completely different.
In Table 7.2, the test results on a rather simple real-world problem are presented

Algorithm	r	rsD	rs	Result	Time
D CS	v		v	572	23'19"
D W CS	v		v	572	23'13"
D SH CS				527	21'16"
D W GS	v			572	8'09"
D W MV GS (see Fig. 7.6)	v	v		572	9'14"
D PR W SH GS	v			572	8'12"
D PR W SH GS	v	v		527	7'34"
D PR W SH GS	v		v	527	7'13"
D PR W SH CS	v		v	572	21'05"
D W SH GS CS	v		v	527	11'16"
D W SH GS				602	3'30"
D W SH GS	v			573	4'09"
D W SH GS	v	v		573	4'09"
D W SH GS	v		v	587	3'41"
D PR W SH GS CS				527	9'18"
D PR W SH GS CS	v		v	527	13'24"

Table 7.2: Test results for algorithms combining different neighbourhoods in the search

for a variety of algorithms. The scheduled ward consists of 20 personnel members, 6 shift types and very stringent soft constraints for which simultaneous satisfaction can never produce a feasible schedule. The combination of applied neighbourhoods is denoted by the abbreviations in the column 'algorithm'. All the abbreviations stand for the neighbourhoods, which can be found in Section 7.3.1. To summarise, the neighbourhoods used in the test example are: single shift-day (D), weekend (W), most violated constraint (MV), shuffle (SH), greedy shuffle (GS), and core shuffle (CS). Columns 2-4 present restart options:

r repeat 'large section' neighbourhoods with the best solution found after the global stop criterion is reached for the first time

rsD restart from the first neighbourhood with the best solution found after the global stop criterion is reached for the first time

rs identical to rsD but skip the single shift-day neighbourhood

Fig. 7.6 schematically presents the scenario for one of the test algorithms. The algorithm applies 4 different neighbourhoods: the single shift-day neighbourhood D, two soft constraint related neighbourhoods W and MV, and one large section neighbourhood GS. The smallest box shows the nearest neighbourhood whereas bigger peripheries of the neighbourhoods are represented by bigger boxes. Note that D and W are explored with tabu search while in the other 2 neighbourhoods steepest descent is applied. The numbers 1-8 explain the order in which the neighbourhoods are passed through. After the first exploration of

GS was finished, the numbers 4-6 demonstrate how the option rsD is applied. It makes the search restart from the smallest neighbourhood. Number 7 indicates option r, for which GS is applied at the end of the search, starting from the best solution found.

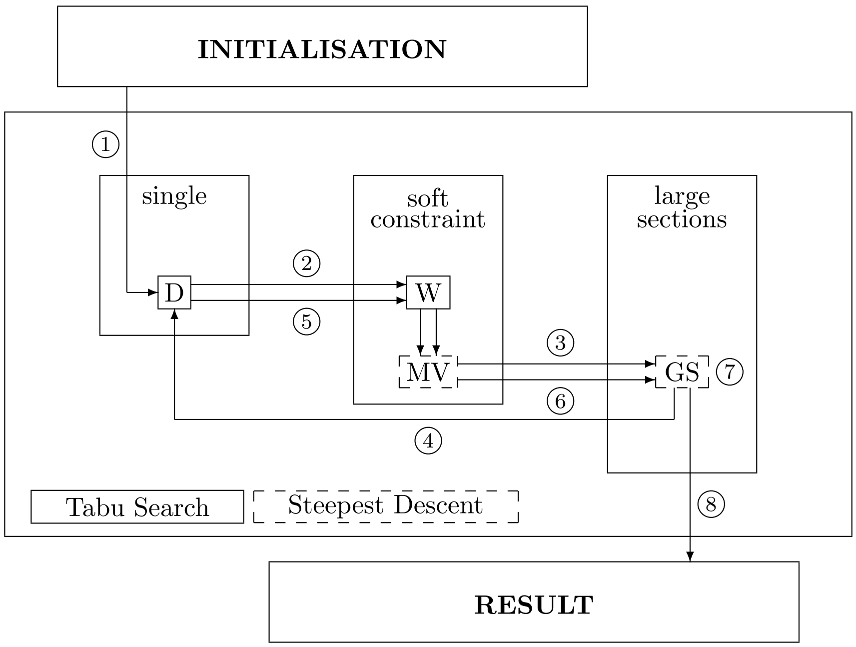

Figure 7.6: Diagram of the scenario for the algorithm D W MV GS with options r and rsD, the initialisation starts from an empty schedule and there are no post planning options selected (Fig. 5.1)

The Result is the value of the cost function, i.e. the weighted sum of the violations of soft constraints, summed over all the personnel members of the ward (see Chapter 4). The calculation time was recorded on an IBM RS6000 PowerPC. It is presented in the column Time. Fig. 7.7 schematically shows the effect of applying the neighbourhoods of Fig. 7.6 on the solution quality. The progress of the quality corresponds very well to our intuitive findings after testing many different combinations. When applying the single shift-day neighbourhood, the number of violations drops drastically. The value of the cost function increases while exploring the soft constraint related neighbourhoods, because they do not take the overall quality into account when searching improvements for one particular constraint. In the greedy neighbourhood, more time is required for improving the solution than in the single neighbourhood. Although the improvement is small, the overall satisfaction of planners is much higher after applying this neighbourhood. This cannot be stated with the figure, however, because there is not much difference in terms of the cost

Figure 7.7: Effect of applying the neighbourhoods of Fig. 7.6 on the solution quality and the computation time

function value.

Many algorithms reached a solution with value 527, which is the best cost function value found. The solutions are all different, however. Some constraints cannot be satisfied and their violation appears in all the solutions, all be it in the schedule of different people. It is remarkable that the category of algorithms, which reached the value 527, all make use of the greedy and/or core shuffle neighbourhood. Larger scale swaps are very useful at the end of the search. This finding will be confirmed for other meta-heuristics in Section 8.4 and 9.5.

The order in which neighbourhoods are explored is very important. It would be a waste of effort to use a greedy neighbourhood to improve the randomly obtained initial solution. Greedy algorithms require a lot of time to explore the entire search space and they would improve the schedule in a very slow manner. The single shift-day neighbourhood is rather small and quickly brings the initial solution into an area with acceptable quality.

It is also remarkable that a combination of the single shift-day (D) and the core shuffle (CS) neighbourhood alone is not interesting at all. The solution quality is not impressive and the calculation time is very high. With the D neighbourhood only, the search stops in a solution which is the result of single shift swaps. The CS neighbourhood finds many changes which make the solution better and therefore requires a lot of calculation time. When applying the CS neighbourhood after a series of smaller scale neighbourhoods (but larger than single shift-day) the possible improvements are smaller, and so is the calculation time.

Some medium scale (soft constraint related) neighbourhoods have been used in the algorithms: weekend, personal requests and most violated constraint. Searches in these neighbourhoods are rather considered as swaps than as real improvement steps. They do not contribute by generating overall better solutions but they act as a diversification in the search (see Fig. 7.7). If the soft constraint neighbourhood does not contribute it is eliminated by the *success* variable. The soft constraint related neighbourhoods are not necessarily developed to solve particular soft constraints but rather to explore parts of the solution space in which these constraints are satisfied. Later iterations might take the solution back to a schedule with a reduced number of violations for that constraint. Since the problem specific neighbourhoods, which the soft constraint neighbourhoods are, cannot consume much calculation time, the chances of finding good solutions in a reasonable amount of time increase.

The discussion in this chapter only presents the final findings after plenty of experiments on real-world (but confidential) data. We figured out that it is not worth spending effort on constructing a good quality initial solution for the problem defined by ANROM (in contrast with what is often claimed in literature on meta-heuristics), but that the local search heuristics on the single shift-day neighbourhood are able to quickly improve the quality of a random initial solution towards a reasonable value of the evaluation function. In comparison with most other solution methods for the nurse rostering problem (see conclusions of the literature review in Section 3.5), the approach presented in this thesis was tested in practice in widely varying healthcare environments. The observation of how experienced planners manually modify the resulting schedules inspired us to implement neighbourhoods such as greedy shuffling and core shuffling, that attempt to finalise the search in a way the manual schedulers appreciate best. In fact, it turned out to be often more satisfying to spend quite a lot more calculation time on calculating minor improvements with respect to the evaluation function value, but important improvements with respect to the overall impression of users in practice.

7.6 Conclusion

Changing neighbourhoods, when applying meta-heuristics to the nurse rostering problem, enables the system to find schedules which are hidden for single neighbourhood heuristics.
The nature of the problem tackled is such that it has a very complex search space compared to other problems reported upon in literature. Meta-heuristics are not always effective enough to explore the search space thoroughly. In the novel approach presented in this chapter, we demonstrate how adding problem specific neighbourhoods to straightforward ones increases the applicability of general heuristics while keeping the calculation time down.
Experiments revealed that it is often beneficial to perform intensive local search in the immediate surroundings of a an obtained schedule. After reaching a local optimum, we recommend the exploration of wider environments.

Several algorithms reach results of equally good quality. The fastest among these is D PR W SH GS, with the single shift-day, two soft constraint related neighbourhoods, and the shuffle and greedy shuffle as large section neighbourhoods. When the stop criterion is reached for the first time, the algorithm passes through all the stages again, except the smallest neighbourhood one (rs option). The algorithm ends with exploring the GS neighbourhood proceeding from the best solution found. The second runner up is nearly the same algorithm but it restarts with the rsD instead of the rs option. It does not find a better solution although it explores an extra neighbourhood after restarting (which explains the longer calculation time). The worst algorithm in terms of quality is D W SH GS, but it is the fastest. Algorithms with a long calculation time do not necessarily produce good results (see D CS and D W CS) because they perform intensive local search in the environment of early found local optima.
We developed a method to organise the changes of neighbourhoods, and to choose particular soft constraint-related neighbourhoods, which are beneficial for the search, resulting in schedules with a low value of the overall cost function. It is often more beneficial to apply simple heuristics such as steepest descent, with a variety of neighbourhoods than to use sophisticated heuristics which are blind to large parts of the search space.

The variable neighbourhood approach is not the most robust meta-heuristic for the nurse rostering problem. In the following chapters, we will introduce more hybrid approaches, which all apply neighbourhoods introduced in Section 7.3.1.

Chapter 8

Hybrid Tabu Search

8.1 Introduction

The complexity of the problem described in Chapter 2 requires other than pure mathematical approaches in real-world applications. For the implementation of ANROM for practical use (Section 2.1), we developed hybrid tabu search algorithms. These heuristics deserve a full chapter in this work because they still form the basis of the software application used in practice. A slightly modified version of this chapter was published as *E.K. Burke, P. De Causmaecker, and G. Vanden Berghe: A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem, X. Yao et al. (Eds.): Simulated Evolution and Learning 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, 1999, 187-194, Springer* (Burke et al. [39]).
In Section 8.2, we briefly introduce tabu search and some applications of the meta-heuristic. Section 8.3 describes the details of the tabu search implementation for this nurse rostering problem. The hybridisations in Section 8.3.6 have been developed to make the algorithms more generally applicable and to increase their overall performance. Some test results for real world problems are presented in Section 8.4. We conclude in Section 8.5.

8.2 Tabu Search

The term tabu search was first introduced by Glover in 1986 [103], at the same time as the term meta-heuristics. Both the operations research (OR) and artificial intelligence (AI) domains have contributed to the foundations of tabu search. While OR was focussing on optimisation and mathematical results, AI was more into qualitative analysis.
Tabu search is a meta-heuristic which guides a local search procedure to explore the solution space beyond local optimality. The local procedure is a search heuristic that makes use of a move to reach the neighbourhood of any given solution (Glover and Laguna [104]). It is an improvement on a descent search

with a neighbourhood structure and a cost function [117].

Tabu search iteratively explores all the possible solutions in the neighbourhood of a current solution and moves to the best one. In order to prevent the heuristic from cycling, a memory structure, called the tabu list, is incorporated in the method. Tabu search applies this list to force the search away from solutions selected for recent iterations. The neighbourhood of a solution thus depends on the iteration number. The tabu list is referred to as the short term memory of the heuristic.

The tabu list, which is based on certain attributes of the most recent moves, can function in accordance with the first in, first out principle. If a move satisfies certain tabu list conditions, it should be rejected. Aspiration criteria are required in order not to ignore moves to solutions that are better than any previous solution. Attribute based tabu lists avoid cycling by preventing the search from going back to an already visited local optimum. It also excludes many solutions that have not been visited before, by forcing the search into unexplored regions.

There is no rule to set the tabu list sizes for different problems. The determination of the best list size can only acquired by experiments on the particular type of problems for which the algorithms are developed. Intuitively, large scale problems should benefit from long tabu lists to avoid cycling. Longer tabu lists are expected to be more effective against falling back in the same local optimum. If cycling must be completely avoided, the calculations and memory use are very expensive [106].

In the tabu search algorithm for nurse rostering (Section 8.3), we have implemented *hashing functions*. They were first introduced by Hansen and Jaumard [112]. Woodruff and Zemel [223] have explored the ideas in detail to keep the memory use for tabu lists down. The hashing functions introduced in Section 8.3.1 are applied in the algorithms for ANROM. *Vectors* resulting from moves are mapped to integers, which can be stored for a large number of recently visited solutions. Hash functions are the mappings from vectors to integers and the hash list is the list of hash functions for recently visited solutions.

Tabu search enables moves to strictly selected parts of the search space. It is very probable that the list prevents the search from visiting attractive solutions. Aspiration conditions therefore overrule the tabu status of certain moves at some occasions. If a move leads to a solution of higher quality than any solution found before, the move should be accepted.

Memory in tabu search can also be used for learning in a more long-term meaning. Intensification and diversification are two important components that help tabu search to behave intelligently. It is interesting to investigate if good solutions have common properties. Intensification can either restrict the neighbourhood or change the current solution to satisfy beneficial properties which occurred in previously visited good solutions. It afterwards discourages the properties from being violated during the search.

Pure intensification is insufficient to guarantee good results for different kinds of optimisation problems. It is necessary to apply diversification to allow the most effective search of the solution space. Diversification guides the search to

contrasting regions.
In the literature overview of Chapter 3, several examples of tabu search implementations are discussed: nurse rostering (Berrada et al. [21] and Dowsland [84], Section 3.3.4) and other personnel scheduling problems (e.g. Chiarandini et al. [58] in Section 3.4).
Many other researchers have made important contributions to tabu search and developed a large number of very successful applications. There are also examples of tabu search in manufacturing (Srivastava and Chen [200]), planning and scheduling (Barnes and Laguna [12], Brandimarte [27], Brucker and Schumacher [29]), transportation and routing (Gendreau et al. [102], Semet and Taillard [190]), layout problems (Blacewicz et al. [23], Bennell and Dowsland [19]), graph colouring (Costa [63]), graph timetabling (Hertz [116], White and Xie [219], Di Gaspero and Schaerf [81]), assignment problems (Ferland [92]), etc.

8.3 Tabu Search Algorithm for the Nurse Rostering Problem

8.3.1 Original tabu search algorithm

The original tabu search algorithm developed for this nurse rostering problem applies the single shift-day neighbourhood (D). It is a move of a duty from one person to another on the same day. Essentially we move one assignment within a column of the solution representation given in Fig. 7.1. The move is not allowed if the goal person is not of the right skill category of or is already assigned to that duty. Hence the hard constraints will still be respected. For a detailed description of this neighbourhood, we refer to Section 7.3.1.
For each skill category (for each iteration) possible moves will be calculated and the move leading to the highest benefit will be performed. If the highest benefit is negative, the move will be performed anyway, unless this move is forbidden by the tabu list. When a move is accepted, a rectangular area in the roster around the roster point where the duty comes from and where it is moved to may not be changed.
For comparison purposes only, we introduced a steepest descent algorithm in which the neighbourhood of the moves is exactly the same as in the tabu search algorithm. After evaluating all the possible moves in the neighbourhood, the best one will be performed, unless this best move does not improve the schedule, in which case the algorithm stops. The algorithm chooses at random among moves leading to equally good solutions. These algorithms turned out not to be powerful enough to produce good solutions for complex problems as is shown in the 'steepest descent' and 'tabu search' experiments in Table 8.1 and 8.2 (Section 8.4). The tabu search algorithm performs better than the steepest descent algorithm and is therefore used as a local search heuristic in the hybrid algorithms described in Section 8.3.6.

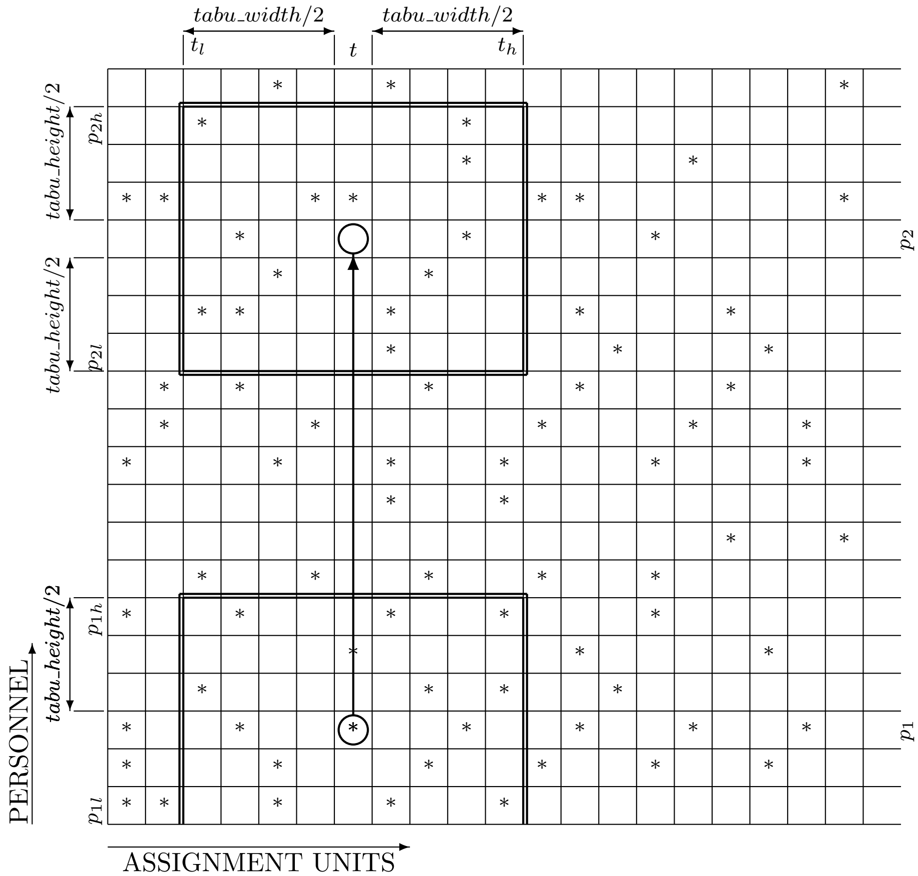

Figure 8.1: Illustration of the rectangles around the positions of a swap, which are used in the hash function

8.3.2 Tabu list

The tabu list for this application is implemented with a hashing function. We store in each hash element the serial numbers of the people whose schedules are involved in the swap, the assignment unit of the swap and a character table in which the roster positions, in the surrounding areas of the swap origin and goal of the move, are saved. The surrounding areas of both origin and goal, are rectangles with dimensions *tabu_height* and *tabu_width*. For both origin and goal of the swap, we consider *tabu_height*/2 people with a lower serial number than the person whose schedule is being manipulated, and *tabu_height*/2 people with a higher serial number. It is demonstrated in Fig. 8.1, which makes use of the representation in Fig. 4.2 for presenting the schedule. Each assignment unit of the planning period has a separate column. Analogously, the schedule of *tabu_width*/2 assignment units before and an equal number after the assignment unit considered are kept in memory for all these personal schedules. For some swap positions, which are near the borders, no rectangle of the required size can be drawn within the schedule. The hash function simply works with rectangles of smaller width and height in that case, as is indicated in Fig. 8.1. As is explained in Section 5.5, the only personal schedules considered are those of the people who are skilled for the category being scheduled. In the program, we temporarily map the serial numbers of the personnel members to a new list with length P_q.

A hash element h has three fields for integer values, 2 for saving the serial numbers of the people whose schedule is involved in the swap and 1 for the time of the swap. It has an extra field for a table representing the assignments in the rectangles around the swap positions. The $address_{h1}$ of the hash element corresponding to the origin node of the move is calculated by the following hash function:

$$\begin{aligned} address_{h1} = \quad & [\ |\{u, p \ \xi \ t_l \leq u \leq t_h \wedge p_{1l} \leq p \leq p_{1h} \wedge schedule_{p,u} \neq 0\}| \\ & +(t + (P_q + p_1) * (P_q + p_2))\ ]\ \%HASH\_SIZE \end{aligned}$$

Analogously, the hash element which corresponds to the goal position of the move is calculated as follows:

$$\begin{aligned} address_{h2} = \quad & [\ |\{u, p \ \xi \ t_l \leq u \leq t_h \wedge p_{2l} \leq p \leq p_{2h} \wedge schedule_{p,u} \neq 0\}| \\ & +(t + (P_q + p_1) * (P_q + p_2))\ ]\ \%HASH\_SIZE \end{aligned}$$

The value of HASH_SIZE in the formulas equals $T * P_q * 0.3$ (P_q being the number of people who can work for skill category q). The value was determined after some experiments.

In our application, we keep the memory use down by not allocating memory for hash elements unless we need it. We check for collision by comparing the people and the time of the swap when an address in the hash list appears to be occupied. If necessary, the assignments for the schedule positions in the rectangles can be compared. The chance for collision does not need to be 0. Thanks to the aspiration criterion, any better solution than the best one found will be accepted anyway.

8.3.3 Aspiration

The simplest tabu criterion would be to prevent the reversal of a move for a certain number of subsequent iterations. Due to the characteristics of the tabu elements, it is not unthinkable that a tabu move leads to an overall better solution.

As an example, we consider the part of an imaginary schedule as it is presented in Fig. 8.1. If subsequent moves change the schedule in areas which do not overlap with the rectangles corresponding to the presented move, the opposite move does not necessarily lead to an already visited solution because the rest of the schedule has changed. The algorithm therefore evaluates the quality of a new solution before it checks the tabu list.

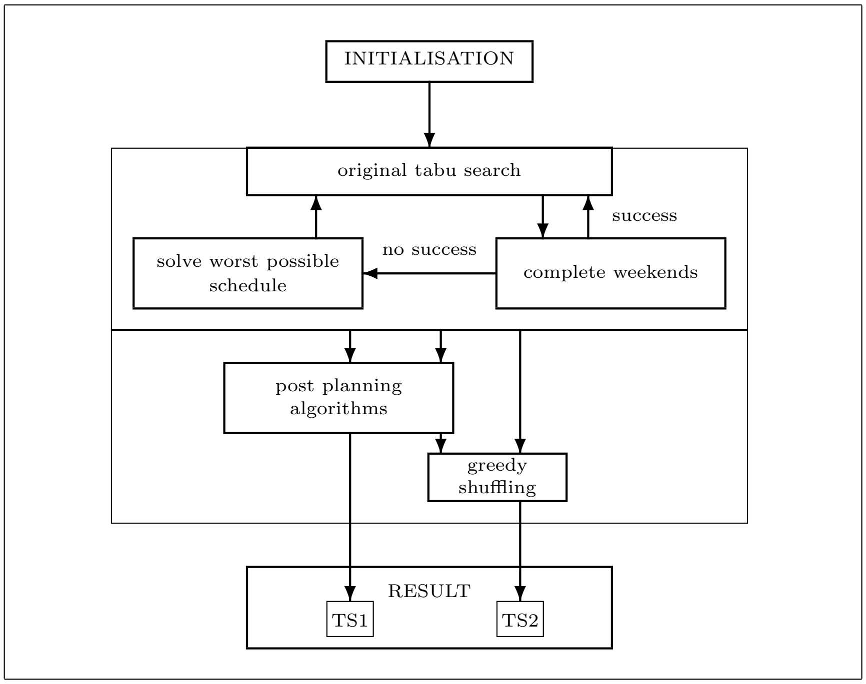

Figure 8.2: Diagram of the hybrid tabu search algorithms for the nurse rostering problem, plug-in for Fig. 5.1

8.3.4 Stop criterion

The original tabu search algorithm stops after a number of iterations without improvement. We have empirically set the value of that number equal to $P_q * T$. This number does not take the coverage into account (schedules with a high coverage have a smaller search space than schedules with a low coverage). Hybridisations such as problem specific diversification moves turned out to be more beneficial for the quality of the algorithms than increasing the number of iterations in the simplest tabu search environment (see Section 8.3.6). After diversification steps, the iterations counter restarts from 0, whether or not an improvement was found. A simplified representation of the flow diagram of the hybrid tabu search algorithms described in this section can be seen in Fig. 8.2. We refer to Chapter 5 for a thorough description of the possible procedures but the general pre- and post-planning procedures are not copied in this figure. After the initialisation phase, in which a feasible schedule is generated, the algorithm enters the tabu search part of the process. Depending on the chosen hybridisation, the results for **TS1** or **TS2** will be generated.

8.3.5 Some diversification heuristics for the problem

Here we describe some heuristics that can be employed (in conjunction with the original tabu search algorithm) to improve the solution.

Diversification 1: Complete weekend

For this diversification, the weekend neighbourhood (W) from Section 7.3.1 is applied. Although the users of the program based on ANROM can assign a cost parameter to this constraint, it is very hard to find satisfactory solutions. The problem is that there are so many constraints and the degree of freedom of ANROM is so high that it is likely to find solutions satisfying many other constraints but not this one. In the graphical user interface, incomplete weekends really catch the eye, while other constraints such as overtime or too many morning shifts on Mondays,... are not immediately visible. Since it is almost impossible to guarantee good solutions with a certain setting of the parameters, we decided to provide a manner of solving this problem the hard way, by not caring about possible problems for other constraints. The stop criterion for the Diversification 1 step is very simple. It stops when no further improvement with respect to the weekend constraint (Constraint 15) is possible. The search space for this step is rather small, the number of weekends in a schedule equals $W * P_q$, for the skill category q. The maximum number of moves required to go from the worst case (highest possible number of violations of this constraint) to the best (lowest possible number of violations) is approximately $P_q/2 * W$. In the worst case, none of the weekends for none of the personnel members is a complete weekend, which means that there are $P_q * W$ violations. The best case has no violations. By moving a weekend shift from one weekend to another, the first one becomes empty (no violations), unless there were several shifts planned on the same day for one person, and the other weekend becomes a complete weekend.

The weekend steps are diversifications because they reach regions which cannot easily be explored by the original tabu search moves. Moreover, good schedules cannot afford high violations of the weekend constraints. They have certain features in common, which reflect good assignments during weekends. This means that the Diversification 1 step can also be looked at from an intensification point of view. It aims at obtaining certain good properties in the solution.

Diversification 2: Consider the worst personal schedule

The Diversification 2 algorithm makes use of the shuffle neighbourhood (SH) introduced in Section 7.3.1. If the complete weekend function (above) has not changed the schedule, it can be beneficial to look at the people with the worst schedule (according to the evaluation function). For every person (within the category being scheduled) it is possible to calculate the value of the evaluation function after exchanging a part of the schedule of the people involved. The parts of the schedule always contain full days and the maximum length is half the planning period. After all possibilities have been calculated, which is quite time consuming, the best exchange (chosen at random from equal values) is performed. The result of this process often provides a better solution. However, it is possible that the best shuffle does not improve the schedule. Therefore, a separate tabu list is maintained to prevent successive shuffles from cycling.

Greedy shuffling: Model human scheduling techniques

The neighbourhood used for this search is the greedy shuffling (GS) neighbourhood introduced in Section 7.3.1. In practical applications, there was a problem with the results of the tabu search algorithm because sometimes a human could improve the visual result by making a small change. This process calculates all possible Diversification 2 (above) moves for every pair of people. After listing the gain in the cost function for every possible exchange, the shuffle leading to the best improvement will be performed. Afterwards, the next best improvement in the list is performed, provided none of the considered personal schedules were already involved in an earlier shuffle. As long as there are improving exchanges in the list, they are carried out. The whole procedure starts over again until none of the possible exchanges improves the quality of the schedule. Note that greedy shuffling only considers improvements. Unlike Diversification 2, greedy shuffling will never move to an equally good or worse solution and thus withstands cycling.
The improvements on the schedule, which can be obtained by employing this procedure together with tabu search (described as **TS2** below), are considerable but the biggest advantage of this step is that it creates schedules for which it is almost impossible for a human to improve the schedule.

8.3.6 Hybrid tabu search algorithms

After extensive testing of hybrid versions of the tabu search algorithm and the above heuristics, two algorithms were developed. The first one produces schedules when a very short computation time is required (as it often is when planners must react to unforeseen events such as staff absenteeism). The second algorithm needs more computation time but generates schedules of a considerably higher quality. Both algorithms are briefly described below.

Tabu search + diversification: TS1

The aim of this algorithm is to provide reliable solutions in a very short time. In practice this algorithm has proved to be very useful to check whether the constraints are realistic, whether during the holiday periods it will be possible to plan good schedules if every person gets their desired holiday period, etc.
The algorithm is constructed quite simply from the original tabu search algorithm. If after a number of iterations no improvement is found, the weekend step is performed. In case the weekend step does not result in a different schedule the second diversification step is performed. After this diversification step,

Problem 1	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
steepest descent	2594	1'26"	2395	1'37"	2657	1'36"
TS	2435	2'05"	2214	2'06"	1928	1'59"
TS stop crit. x50	1915	40'58"	1675	41'21"	1534	23'58"
TS1	1341	6'00"	1089	5'59"	929	5'27"
TS2	1264	20'15"	1011	24'39"	809	28'08"

Table 8.1: Value of the evaluation function and results of the steepest descent and variants of the hybrid tabu search algorithm for Problem 1, planning order of the qualifications as chosen by the customer

the original tabu search algorithm is used again and so on. The calculations stop after a global number of iterations without improvement.

Tabu search + greedy shuffling: TS2

This algorithm combination requires more time but the results are considerably better from the human point of view. Anecdotal evidence suggests that the level of satisfaction with schedules produced by this algorithm is actually higher than the cost function indicates. The main reason for this is that after the shuffling step the users cannot easily improve the results.
It is important for the greedy shuffling step at the end of the calculations because its real aim is to perform the exchanges a human user would perform. It is because of the exhaustive search character of the shuffling that this step takes a lot of time. It is very important to calculate this step until there are no further improvements because otherwise the goal of excluding manual improvements to the schedule might be lost (Greedy shuffling in Section 8.3.5).

8.4 Test Results

The tests in this chapter are restricted to planning the minimum requirements RM, planning between the minimum and the preferred requirements RMP (= $RM + \textit{coverage option}$ Fig. 5.10), and planning according to the calculated demands RMC (= $RM + \textit{inconsistent}_{RM}$) as explained in Section 5.2. All the experiments started from an empty initial schedule, which is made feasible according to the algorithm described in Fig. 5.7 (Section 5.4.2). If we look at Fig. 8.2, it means that we only present experimental results for shift type requirements in this chapter. For the latter we decided to perform the step 'add shifts towards preferred personnel requirements' (see Section 5.6.1) as a post planning option, whenever this does not cause a violation of the soft constraints (Section 5.6). In Table 8.1 and 8.2, the results of the variants of the tabu search algorithm are compared to the steepest descent algorithm. The test examples Problem 1 and Problem 2 are rather hard to solve real world problems and in

Problem 2	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
steepest descent	1338	44”	1338	45”	1134	47”
TS	1189	57”	1189	58”	933	1’03”
TS1	843	3’18”	843	3’18”	867	2’14”
TS2	809	6’25”	809	6’25”	588	10’19”

Table 8.2: Value of the evaluation function and results of the steepest descent and variants of the hybrid tabu search algorithm for Problem 2, planning order of the qualifications as chosen by the customer

both cases the personal demands make a good schedule almost impossible.
The column 'Value' shows the value of the evaluation function for the entire schedule (see Chapter 4: the cost parameter per constraint times the extent the constraint is violated). The column 'Time' contains the computation time on an IBM Power PC RS6000. Test examples are available at http://extern.kahosl.be/greet.vandenberghe/.
The third set of results, where the demands are adapted to the constraints as described in Section 5.2 (RMC: calculating more realistic demands), are better than the results in the first column. This is according to our expectations because the hard constraints were changed in order to prevent some unavoidable violations. The steepest descent algorithm in the first table performs worse. In Problem 2, there was no difference between the minimum and the preferred requirements.
For all the considered examples, the tabu search algorithm performs better than the steepest descent algorithm. We decided to organise the stop criterion for the tabu search algorithm such that the computation time is of the same order of magnitude as the time required to do steepest descent. Only Table 8.1 contains the results of the original tabu search algorithm for a longer computation time. It is obvious from the experimental results that it is much better to implement problem specific diversification steps and to hybridise it than to increase the computation time for the simplest algorithm.
The behaviour of the hybrid algorithms is better than the behaviour of the normal tabu search algorithm (with a short computation time). Even considering the computation time, for application in practice it is worth using the hybrid tabu search algorithm because the degree of confidence the users have in the program is much higher.

8.5 Conclusion

The tabu search algorithm developed for the nurse rostering problem of Chapter 2 respects the hard constraints during the search. Some diversification steps, which are larger moves in the solution space, have proven more beneficial

for the quality of the results than an increase of the calculation time for the simplest version of the tabu search algorithm. Different users will choose different algorithms, depending on their opinions and their requirements. The runtime/quality trade-off depends very much on the individual planner. Some users are really interested in the lowest possible value of the evaluation function, no matter how long the calculations take, particularly in smaller hospitals where a single planning officer generates the roster for the whole hospital and will not mind if roster generation takes an overnight run. Others, for instance in very big hospitals with many wards to be scheduled by individual head nurses, needed quick calculations and a slightly lower quality of the schedule is good enough, since each head nurse may have a very tight window in which to generate a schedule.
Combining the original tabu search algorithm with some specific problem solving heuristics not only guarantees better quality rosters but also satisfies the hospital planners to a very high extent because it is almost impossible for experienced planners to improve the results (considering the soft constraints) manually. For many practical scheduling problems the higher quality of the solutions produced by the hybrid algorithm compared to the simple tabu search algorithm compensates for the increase in calculation time.

For application in practice, the hybrid tabu search approach combines fast scheduling and good quality results. However, the approach has some shortcomings when applied to extremely difficult real world examples, in which the application can benefit from extra information given by the users. In the next chapter, we attempt to overcome some of these problems, by defining heuristics which explore a much larger part of the solution space.

Chapter 9

Memetic Algorithms

9.1 Introduction

In this chapter, we introduce a range of new memetic approaches for the nurse rostering problem. The algorithms apply steepest descent improvement heuristics within a genetic algorithm framework. The main ideas of the memetic algorithms described in this chapter were published as *E.K. Burke, P. Cowling, P. De Causmaecker, and G. Vanden Berghe: A Memetic Approach to the Nurse Rostering Problem, Applied Intelligence special issue on Simulated Evolution and Learning, Vol. 15, Number 3, 2001, 199-214* [34].
Tabu search heuristics can be made effective, particularly for obtaining reasonably good solutions quickly for smaller rostering problems, as mentioned in Chapter 8. The tabu search algorithms do sometimes display considerable shortcomings for practical applications, particularly that they are not sufficiently robust to handle difficult problems well. This provided the motivation to investigate population based approaches for the same problem. We discuss the robustness problems, which arise in practice for tabu search heuristics. We provide empirical evidence to demonstrate the best features of a memetic algorithm for the rostering problem, particularly the nature of an effective recombination operator, and show that these memetic approaches can handle initialisation parameters and a range of instances more robustly than tabu search algorithms, at the expense of longer solution times. Having presented tabu search (Chapter 8) and memetic approaches (both with benefits and drawbacks) we finally present an algorithm that is a hybrid of our memetic approaches and tabu search approaches. This technique produces better solutions than either of the earlier approaches and it is relatively unaffected by initialisation and parameter changes, with a running time comparable to that of the memetic approaches, combining some of the best features of each approach to create a hybrid which is greater than the sum of its component algorithms. Any successful solution method must be robust enough to cope with widely varying cost functions and problem instances. In Section 9.2, we introduce evolutionary algorithms, and

make a distinction between genetic and memetic algorithms. In the Sections 9.3 and 9.4 we present algorithms for ANROM. Section 9.3 presents several different genetic and memetic algorithms and Section 9.4 brings together Sections 8.3.6 and 9.3 to consider hybridisations between the two different approaches. In Section 9.5 we compare and contrast the performance of the algorithms on specific real-world problems. We present conclusions in Section 9.6.

9.2 Evolutionary Algorithms

9.2.1 Genetic algorithms

Darwin's work, The Origin of Species [71], forms the basis of the evolutionary computation which has applications in artificial intelligence, optimisation, game theory, etc. In the 1960's, evolutionary algorithms have been introduced to model natural evolution processes (Bremermann [28] and Fraser [97]). In 1975, Holland [118] presented the first genetic algorithm for discrete domains. Schwefel [187] introduced evolution strategies for continuous optimisation and Goldberg [107] explains how genetic algorithms can be applied to search, optimisation, and learning. Reeves [176] demonstrates that genetic algorithms have become very popular for solving hard combinatorial problems. Fogel [95] presents an extensive overview of early genetic approaches.

Genetic algorithms are inspired by biological processes. Natural species undergo a process of slow evolution by mutation and crossover in genes. Gradually, good qualities will emerge and uninteresting characteristics will disappear while the species adapt themselves to their environment. Mutation perturbs single individuals and thus avoids local minima while crossover combines characteristics of two (or more, in some approaches) parents. Solutions that are maintained in the next generation are selected on the basis of their fitness.

A genetic algorithm maintains a population of individuals or solutions, which are all built up by a set of genes. The quality of such a solution is given by its fitness which is the value of the evaluation function. During the genetic process, new generations will replace old ones. The members of a new generation are constructed by applying the genetic operators on the individuals of the current generation.

In Section 3.3.4, some examples of genetic algorithm approaches for nurse scheduling are presented (Aickelin and Dowsland [5], Easton and Mansour [85], and Tanomaru [201]). Other timetabling applications of population based approaches are school timetabling (Burke and Newall [41], Burke et al. [43], Carrasco and Pato [48], Paechter et al. [167], Ueda et al. [211]) or examination timetabling (Ross et al. [180]), general personnel scheduling (Corne and Ogden [62]), bin packing (Pimpawat and Chaiyaratana [172]), etc. Evolutionary approaches have also been successfully implemented in engineering (Garcia et al. [100]), robotics (Han and Oh [111]), control (Ge et al. [101]), optimisation (Cercueil and François [53], Myung and Kim [149]), games (Darwen and Yao [70], Reiser and Riddle [177]), neural networks (Kaise and Fujimoto [124]), forecasting (Lam [129], Liu and Yao [132]), pattern recognition (Scott et al. [188], Tsjujimura and Gen [210]), programming (Liang et al. [130]), data mining (Sierra [192]), etc.

9.2.2 Memetic algorithms

Hybrid evolutionary algorithms, which incorporate local search methods, are named memetic algorithms. The term was first introduced by Moscato and Norman [152] in 1992, while Radcliffe and Surry [174] further formalised the idea.
Apart from genes, which are known to evolve in biology, also ideas can evolve. Dawkins [73] describes a meme as a unit of cultural transmission, which can be an idea or behaviour. Ideas can be combined to form new ideas (crossover), and they can change by accident (mutation). Good ideas will survive longer than bad ones. The difference with genes, however, is that memes can be improved by individuals before they are passed on. A local improvement algorithm is performed on each meme.
There are examples of memetic algorithms for a range of optimisation problems: vehicle routing (Moscato and Norman [152]), timetabling (Burke et al. [42, 43], Peachter et al. [168, 169]), job shop scheduling (Cheng and Gen [57]), maintenance scheduling (Burke and Smith [44]), etc.

9.3 Evolutionary Algorithms for Nurse Rostering

A basic genetic algorithm with just mutation and crossover operators can be employed but to reach convergence it is important to have crossover operators that combine parts of good solutions to produce good new solutions. A difficulty with rostering problems is often that the quality of a solution is not necessarily a sum or a combination of the qualities of the partial solutions. We have carried out a number of experiments with several crossover operators, either conserving the 'building blocks' as much as possible or repairing the roster if it is destroyed by the crossover. This section will describe a memetic algorithm which incorporates the tabu search and hybrid tabu search algorithms of Section 8.3.6 into a genetic algorithm. The components of the algorithms introduced in this section and in Section 9.4, especially developed for solving ANROM, can be seen in Fig. 9.1. We will describe how the algorithms can be constructed with the components displayed. In the memetic algorithm used for the nurse rostering problem, an initial population consists of N feasible schedules, generated randomly using the initialisation techniques discussed in Section 5.4. There are several possibilities for recombination to create new offspring. It is very important to organise the recombination so that the children inherit the good characteristics of a parent generation. Since the quality of a schedule is the sum of the schedule quality for each person, it is important to get these personal schedules right. The characteristics of the constraints are such that

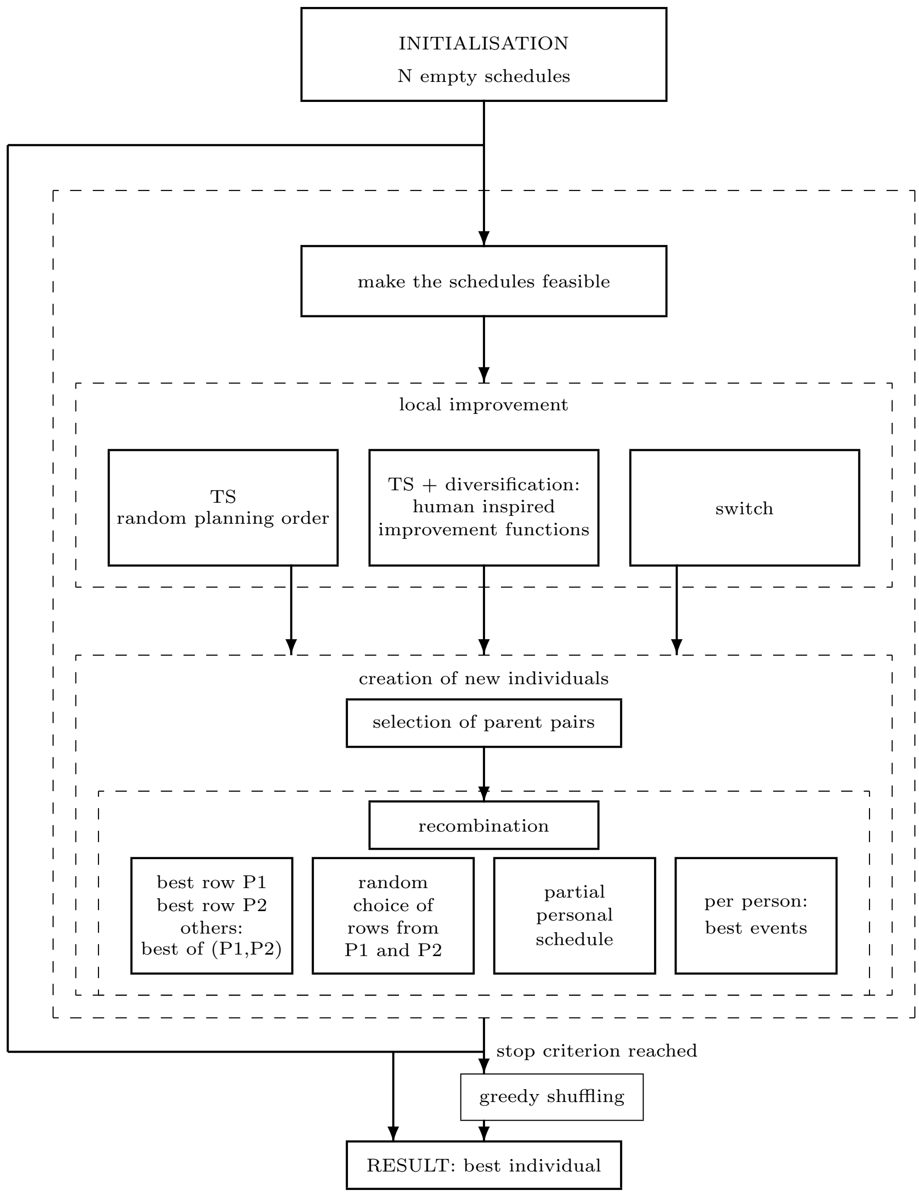

Figure 9.1: Diagram of the components of the genetic and memetic algorithms for the nurse rostering problem, plug-in for Fig. 5.1

mixing up the scheduled events for a person usually leads to very bad schedules. In the following algorithms, we have used many variants of the recombination operators. Some preserve personal schedules to a high extent, others do not but preserve the position of well-placed events. Each generation requires significant calculation time, so we decided not to plan a high number of generations. Each of the memetic algorithms stops when no improvement arises during two consecutive generations. The variants of the memetic algorithm (described below) contain different recombination mechanisms.

Original memetic algorithm: M

In the simplest memetic algorithm, a steepest descent is performed for each individual. The steepest descent algorithm applies the single shift-day neighbourhood (D in Section 7.3.1) for the moves as the simplest tabu algorithm (Section 8.3.1) where the planning order of skill categories (see Section 5.5) is as given by the user. After evaluating all the possible moves in the neighbourhood, the best one is performed, unless this best move does not improve the schedule, in which case steepest descent stops. After this step, there is a simple tournament selection of the best individuals for creating offspring. For each pair of parents, two new individuals are created. The first child contains the best personal schedule (referred to as 'row' in the schedule) from the first parent + the best personal schedule from the second parent (different from the first one selected). The other personal schedules are chosen in a pair wise tournament between the rows of the parents. This normally does not result in a feasible schedule, so to make the child schedules feasible, shifts are added or taken away at random where necessary (see algorithm Fig. 5.7 in Section 5.4.2). This leads to diverse schedules of poor quality, prior to the application of the steepest descent heuristic. The algorithm is schematically presented in Fig. 9.2.

Diverse memetic algorithm: DM

With the shortcomings of the tabu search algorithms in mind, we decided not to plan the skill categories according to the planning order chosen by the customers (Section 5.5) as in algorithm **M** above. In the **DM** algorithm, each time the steepest descent algorithm is performed, the planning order of the skill categories is randomly chosen for each schedule, causing additional diversity in the population. All other features of the algorithm are the same as for algorithm **M**.

Diverse memetic algorithm with random selection: DMR

Here we use the steepest descent approach and other features of the **DM** algorithm, but rather than choose the rows of the child rosters by tournament selection, each personal schedule is chosen randomly from one of the parents.

- *create N different schedules using random initialisation*

  while stop criterion is not reached (stop criterion: no improvement during two generations), repeat:
- *make all the schedules feasible by randomly adding and deleting appropriate shifts (algorithm of Fig. 5.7 in Section 5.4.2)*
- *perform steepest descent on each of the individuals*
- *select parents from the individuals by tournament selection*
- *recombine the parents,*

  per pair of parents, generate two children:

  child 1:

  - *best personal schedule (row) from parent 1*
  - *best personal schedule (row) from parent 2*
  - *(or the second best one if the same row is best for both parents)*
  - *for the other child rows use tournament selection from parent rows*

  child 2:

  - *best personal schedule (row) from parent 2*
  - *best personal schedule (row) from parent 1*
  - *(or the second best one if the same row is best for both parents)*
  - *for the other child rows use tournament selection from parent rows*

Figure 9.2: Schematic representation of the original memetic algorithm

Memetic algorithm with string recombination: MSR

In this algorithm, a different technique is used to generate offspring. It no longer copies an entire schedule from one of the parents. For each personal schedule (row) in the current solution, a time unit (day and shift, corresponding to a column in the current period of Fig. 4.1) is randomly chosen. The part of the schedule between the start of the planning period and this time unit is copied from the first parent and the remainder from the second parent. The procedure is repeated for the second child, except that the first part of each row of this child is taken from the second parent and the remainder from the first parent. Apart from this new recombination operator, this algorithm is the same as the **DM** algorithm.

Memetic algorithm copying the x best events: MEx

The difference between the **MEx** algorithm and the **DM** algorithm considered previously, is the way in which child schedules are generated from their parents.

Good results were found when copying the 'best placed' events for every person, from the parents to the children. The 'best placed' events are those events that would lead to the worst increase in the cost function when removed. Experiments have been carried out that copy x = 2, 3, 4, and 5 events per parent to the child. If the best personal events are the same in both parents, this will lead to only x events in the offspring. Again, the schedules are made feasible by randomly adding the other events (according to Fig. 5.7), which of course leads to more diversity.

9.4 Combining the Qualities of the Hybrid Tabu Search and the Evolutionary Algorithm

Tabu search algorithms using different initial solutions and randomising the planning order of the skill categories: TSPOP

The tabu search algorithm (without hybridisations) and the memetic algorithms do not always lead to excellent solutions for the complex problems hospital planners have to deal with. Unfortunately, the quality of the solution depends strongly on the initial schedule. Not that the schedule has to be good, on the contrary, very good initial schedules are sometimes hard to improve by the methods considered. The problem is actually that the small move made during the tabu search (Section 8.3.1) cannot lead the solution away from some of the local minima. In the commercial version based on this research, only the hybrid versions of the tabu search algorithm are used. The users' informal feedback about the hybrid algorithms points to insensitivity to the random seed (initial solution). The strength of the memetic algorithm approach is that many different starting points are taken and a diversity of different schedules is maintained throughout. An even bigger advantage is the possibility of planning the skill categories in different orders. It might be argued that starting our tabu search from multiple different starting points might produce solutions of comparable quality to the memetic algorithms and hybrids. In order to compare directly the performance of our tabu search approaches and our memetic approaches and hybrids, given similar time, the **TSPOP** algorithm first produces a population of initial solutions that are one by one improved by the **TS1** algorithm (Section 8.3.6) except that the ordering of skill categories is random. Greedy shuffling (explained in Section 7.3.1) is applied to the best individual solution of the population (see also Fig. 9.1).

Memetic algorithm with human inspired improvement functions: MEH

If we combine the extra functions of the tabu search algorithm with the memetic algorithm, either by using them as local improvement functions or by performing them on the best individual of the memetic algorithm, the results get much better. The solutions thus found are of the same quality as, or better than the

1. *create N random individual solutions while the stop criterion is not reached*
   (stop criterion: no improvement during two generations), repeat steps 2 - 5:
2. *make all the schedules feasible by adding and removing appropriate shifts randomly (Fig. 5.7 in Section 5.4.2)*
3. *perform the TS1 on each of the individuals*
   (choose the planning order of the skill categories at random)
4. *select parents from the individuals by tournament*
5. *recombine the parents as explained in the ME4 algorithm*
6. *perform the greedy shuffling step on the best individual*

Figure 9.3: Schematic representation of the memetic algorithm with human improvement functions

solutions found with the hybrid tabu search algorithm. The reason why better solutions are found is the diversification of the algorithm. By starting the calculations from different starting situations, and by changing the planning order of the skill categories, the probability of finding better solutions is increased. For the tests, we used the **ME4** algorithm from Section 9.3. On the best individual obtained with this **ME4** algorithm, the greedy shuffling step (Section 7.3.1) is applied. The algorithm is schematically presented in Fig. 9.3.

Switch: SWT

In all the previous algorithms, the number of staff of each skill category and each shift remained constant once the initial feasible solution was found. Here we add an additional move where, every now and then, the schedule is randomly changed, for a random person and at a random day and shift. In case nothing was scheduled for this person at this time, we introduce a new 'event' in the schedule (i.e. if $schedule_{p,t} = 0$, we insert an assignment: $schedule_{p,t} = q_p$). In case something was scheduled already for the skill category being planned, we remove the scheduled event (i.e. if $schedule_{p,t} \in \{q_p, pref + q_p\}$, we remove an assignment: $schedule_{p,t} = 0$). The random changes may not violate the hard constraints. As long as the number of scheduled people is not lower than the minimum number and not higher than the preferred number, and as long as the person in whose schedule there is a switch is of the right skill category, there is no problem. Let us for example assume that the number of late shifts for caretakers is minimum 2 and preferably 3 on a certain day in the planning period. Let us also assume that in the current solution 3 late shifts are scheduled on that day. Assume that the 'switch' move, described in this section, randomly picks this day and the late shift for the move. In case the randomly chosen person was not assigned to the late duty that day, the algorithm cannot change this because adding a late duty for that person would violate the hard constraints (4 late duties planned instead of the preferred number of 3). The other possibility is that the algorithm picks at random a person who was already scheduled for the late duty. In that case, the late duty will be removed. The removal causes no violation of the hard constraints since the minimum required number of scheduled late duties is still scheduled. With the 'switched' tabu search algorithm, we allow more flexibility for staff to be scheduled at lower or higher levels than usual, where this is permitted, according to the hard constraints on coverage discussed in Section 2.3.3. We decided to alternate the ordinary tabu search and the switch function within the **SWT** algorithm, since after a switch the resulting solution is often of poor quality. The **SWT** algorithm is the **ME4** algorithm with this additional change.

9.5 Results

We tested our algorithms on four difficult real-world rostering problems (two are identical to the test problems discussed in Chapter 8) arising in Belgian hospitals. Due to complex confidentiality and operating requirements, gathering each set of problem data required significant amounts of time and effort. Each of the four rostering problems has different characteristics and consistent performance across these four different problems provides strong empirical evidence of performance overall. It is not appropriate to test our algorithms using random data that would not have the same problems associated with solution as are encountered in practice. The results of applying our heuristics to the problems are given in Tables 9.1 - 9.4. We have given the value of the evaluation function and the time taken for the planning of the minimum staff requirements (RM), planning towards the preferred numbers of staff at each skill category (RMP, see Section 5.6), and planning the recalculated requirements which will aid satisfaction of the soft constraints (RMC) as explained in Section 5.2. In all the tables, the column 'Value' shows the value of the evaluation function. The column 'Time' contains the calculation times on an IBM Power PC RS6000. None of the four problems is feasible. At http://extern.kahosl.be/greet.vandenberghe/ some test examples are available.
The consistency check procedure suggests to modify the hard constraints in all the cases.

Problem 1 is large with many conflicting soft constraints, being by far the most difficult problem of this group. Problems 2 and 4 are smaller problems, they also have fewer difficult soft constraints and are thus much easier than Problem 1. In Problem 2, a feasible solution requires many alternative skill category assignments. Some of the skill categories are understaffed, and many people go on long leave during the planning period. Problem 3 is of intermediate

Problem 1	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
M (10 generations)	1445	52'33"	1296	52'42"	1103	50'18"
M (100 generations)	1435	4h46'34"	1293	4h59'04"	1100	4h22'41"
DM	1334	52'40"	1298	52'58"	1091	50'28"
DMR	1993	1h00'01"	1829	1h00'20"	1716	59'01"
MSR	1991	31'14"	1806	31'49"	1695	34'51"
ME2	1397	1h24'25"	1266	1h24'36"	1203	1h16'57"
ME3	1300	2h10'18"	1209	2h10'40"	1120	2h13'45"
ME4	1208	2h08'30"	1087	2h08'41"	1003	2h00'43"
ME5	1322	2h02'43"	1199	2h02'52"	1127	1h58'22"
TS	2435	2'05"	2214	2'06"	1928	1'59"
TS1	1341	6'00"	1089	5'59"	929	5'27"
TS2	1264	20'15"	1011	24'39"	809	28'08"
TSR	2893	2'20"	2714	2'21"	2983	2'22"
TS1R	1911	18'33"	1692	35'14"	1573	35'03"
TS2R	1911	34'16"	1691	53'55"	1573	40'09"
TSPOP (12 individuals)	1352	1h38'33"	1089	1h42'55"	736	1h41'47"
TSPOP (24 individuals)	1352	3h40'16"	1083	3h07'30"	746	2h52'02"
MEH	1192	2h22'04"	904	2h28'28"	769	2h31'51"
SWT	1090	1h45'17"	1094	2h20'26"	807	30'14"

Table 9.1: Comparison between the algorithms for Problem 1

complexity. We deliberately tested the approach on four test problems of a very different nature, in order to demonstrate the sensitivity of the newly developed algorithms. Some of the hybrid tabu search results of chapter 8 are repeated in this section for Problem 1 and 2, for comparison only.

The original tabu search algorithm **TS** is very good at producing a reasonable starting solution from a random initialisation in a short time. It is however a very slow method if it is used to generate acceptable solutions. Though the results of the hybrid tabu search algorithms **TS1** and **TS2** are considerably better, there is no indication how much more they could be improved. Unacceptable solutions usually arise when the constraints on the problem are contradictory. It is then very hard to find the very narrow valleys in the solution space, which contain good schedules. Giving a very high value to the cost parameter corresponding to a particular constraint does not necessarily guarantee that the solution will be free from violations of this constraint.

Since the test problems consist of wards with different skill categories, we also tested what the influence of changing the planning order was (see Section 5.5). There are particular difficulties when the requirements for people with a certain skill category are higher than the number of people available. They are only significant in the tabu search algorithm where every ward is planned skill category by skill category. The customers can freely decide upon the order in which they want the skill categories to be planned. When planning such a skill category, the algorithm is free to place the required shifts on every day that the people within the category under consideration (or with this category

Problem 2	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
M (10 generations)	1245	4'09"	1245	4'09"	1060	4'12"
M (100 generations)	1245	10'53"	1245	10'53"	1060	10'22"
DM	800	4'10"	800	4'10"	823	4'40"
DMR	1037	2'56"	1037	2'56"	992	2'56"
MSR	1104	2'51"	1104	2'51"	1080	2'59"
ME2	1123	22'02"	1123	22'02"	1130	22'02"
ME3	752	22'16"	752	22'16"	748	21'42"
ME4	698	22'01"	698	22'01"	707	22'10"
ME5	782	21'51"	782	21'51"	769	22'05"
TS	1189	57"	1189	58"	933	1'03"
TS1	843	3'18"	843	3'18"	867	2'14"
TS2	809	6'25"	809	6'25"	588	10'19"
TSR	2614	48"	2614	50"	1584	49"
TS1R	1875	1'04"	1875	1'06"	554	8'21"
TS2R	790	15'10"	790	15'11"	554	9'37"
TSPOP (12 individuals)	885	29'36"	885	29'37"	464	32'34"
TSPOP (24 individuals)	892	50'19"	892	50'21"	457	53'48"
MEH	980	23'54"	980	23'54"	535	24'42"
SWT	992	19'09"	992	19'09"	578	19'26"

Table 9.2: Comparison between the algorithms for Problem 2

Problem 3	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
M (10 generations)	567	23'46"	560	23'58"	547	21'20"
M (100 generations)	552	1h37'14"	541	1h37'25"	547	1h15'14"
DM	403	23'57"	402	24'10"	396	24'10"
DMR	636	28'45"	629	28'58"	620	27'51"
MSR	612	27'12"	610	27'30"	604	27'28"
ME2	526	1h17'17"	521	1h17'30"	518	1h11'42"
ME3	472	57'03"	466	57'16"	459	56'54"
ME4	398	1h16'43"	392	1h16'41"	391	1h00'23"
ME5	397	1h09'55"	393	1h10'05"	390	1h04'42"
TS	422	2'05"	418	2'08"	415	2'06"
TS1	398	7'38"	389	7'42"	390	7'38"
TS2	391	13'52"	380	13'56"	377	14'11"
TSPOP (12 individuals)	624	1h35'50"	620	1h48'06"	583	1h32'27"
TSPOP (24 individuals)	608	2h56'11"	608	3h16'12"	583	2h55'00"
MEH	378	1h20'44"	379	1h22'45"	369	1h28'03"
SWT	381	1h17'09"	375	1h30'10"	364	1h26'32"

Table 9.3: Comparison between the algorithms for Problem 3

Problem 4	RM		RMP		RMC	
	Value	Time	Value	Time	Value	Time
M (10 generations)	226	8'47"	226	8'47"	224	8'09"
M (100 generations)	225	33'27"	225	33'27"	224	32'54"
DM	241	9'21"	241	9'21"	237	9'42"
DMR	266	9'33"	266	9'33"	260	10'39"
MSR	273	8'40"	273	8'40"	265	9'15"
ME2	200	42'11"	200	42'11"	205	39'23"
ME3	184	46'31"	184	46'31"	184	46'22"
ME4	186	45'40"	186	45'40"	187	47'19"
ME5	191	42'16"	191	42'16"	190	44'57"
TS	231	52"	231	53"	227	1'03"
TS1	190	2'21"	190	2'23"	189	2'14"
TS2	189	4'25"	189	4'26"	186	4'38"
TSPOP (12 individuals)	269	22'16"	269	23'09"	266	24'28"
TSPOP (24 individuals)	264	39'03"	264	40'21"	263	43'10"
MEH	182	23'54"	182	24'35"	175	24'42"
SWT	179	19'09"	179	19'59"	176	19'26"

Table 9.4: Comparison between the algorithms for Problem 4

as an alternative possibility) are available. When this step of the algorithm stops, the shifts planned for this skill category are frozen. This sometimes causes difficulties in planning the shifts for other skill categories, because of the overlap between categories, demonstrated in Fig. 5.6. The other algorithms use a random ordering of the skill categories. Practical experiments have shown that it is the best strategy to plan those skill categories which are understaffed first, but the decision as to the planning ordering is a difficult one requiring the human planner's expertise. We only present here an extract of these results, namely the results of planning the skill categories in an order, which is the reverse of the order initially chosen by the customer. We suppose this to be the worst case and our results certainly support that this is a very bad choice. The results for Problem 1 and Problem 2 are in the **TSR**, **TS1R** and **TS2R** rows of Tables 9.1 and 9.2 respectively. For Problem 2, we actually find that the reverse planning order for the skill categories generates a better solution than the selected order (namely in the case of TS2-TS2R). Many hospital planners have the habit of setting the planning order according to the hierarchical importance of the skill categories. This example indicates that it is not always a good option for generating good quality schedules. It is actually for solving this frequently occurring problem that we set up the algorithm of Fig. 5.9. It can overcome unfortunate settings of the planning order and it is included as a diversification option in the software based on ANROM. This very time consuming algorithm does not improve the quality, however, when planners have enough insight in their specific problem to set the planning order right.

We see that in general, using a poor ordering of skill categories produces much poorer results. This is particularly true for the difficult Problem 1. Surprisingly, the **TS2** algorithm produces slightly (2%) better results for the relatively easy Problem 2 when the planning order is reversed in **TS2R**. This behaviour is due to the greedy shuffling step performed at the very end of the **TS2** algorithm's calculation. This step goes again through all the skill categories following the planning order. It is thus possible to make changes in the schedule of previously planned skill categories again. The original tabu search algorithms (**TS**) and **TS1** go through the schedule skill category by skill category, visiting each category only once, with no chance to rectify poor choices later. They are thus more strongly affected by the bad planning order of skill categories.

Unless indicated otherwise in the tables with test results, all the memetic algorithms stop after two generations without improvement, this is typically less than 20 generations. All the algorithms used to produce the test results of the Tables 9.1 - 9.4 have a population size of 12.

We can see that for each of the problems, the extra benefit from allowing our original memetic algorithm **M** to run ten times longer produces little improvement in the final solution, since this approach does not generate sufficient diversity. However, we see that for Problems 1, 2, and 3 the **DM** heuristic produces better rosters than the **M** heuristic, since the random ordering of skill categories gives greater population diversity. However, we can also see that the **DMR** heuristic, which introduces still more diversity through choosing random rows from the parent schedules instead of the best rows in each case has arguably introduced too much diversity since its results are worse than those for the **DM** heuristic. The same could be said for heuristic **MSR** that takes an appropriate segment of each row from each parent, which demonstrates a similar performance. Each of these memetic algorithms has a population size of at least 12, which explains the slower running time of the memetic algorithms. We will consider below the comparison between the memetic approaches and a multi start tabu search approach **TSPOP**.

Copying large partial schedules of high quality from the parent schedules to the children often turns out not to be a good idea. The steepest descent technique used by the memetic algorithms is not powerful enough to generate better schedules. We have obtained better results by copying small parts (with good qualities) from the parent schedules, so that the degree of freedom after making feasible solutions is high enough to provide diversity. Hence we see that the memetic algorithms **ME2**, **ME3**, **ME4**, and **ME5** which are more selective about which parental traits are passed on, generate significantly better schedules than the algorithms **M**, **DM**, **DMR**, and **MSR**. The algorithms copying the smallest parts of the parent schedules are the **MEx** algorithms. We found that the number **x** of events which are copied per personal schedule (row) has a great influence on final solution quality. The results of copying 2, 3, and 4 events per row are increasingly better but from 5 events on the results get worse again. We believe that these 'best placed' events strongly influence the position of all other events, so that the freedom of the solution to evolve is restricted to good areas of the search space and the steepest descent heuristic is particularly effective in improving the diverse schedules generated. **ME4** represents the best compromise between a diverse population of solutions and the ability to focus on interesting areas of the search space.

When we compare the best of our memetic algorithms, **ME4**, with **TS2**, the best of our tabu search algorithms, we see that significantly better solutions are produced by the **ME4** algorithm for Problems 1 and 2 and comparable results for Problems 3 and 4, at the expense of longer running times. A very important advantage of the memetic algorithms with respect to the tabu search algorithms, however, is the fact that the results of the memetic algorithms are not dependent on the planning order of the skill categories chosen by the user. The chance of becoming trapped in a local minimum, which is very far from the optimal, is reduced. Problem 1 has a high number of very strict soft constraints, with high cost parameters attached to them. Problem 2, on the other hand, has fewer personnel and duty types and has few soft constraints. Problem 1's search space (only depending on soft constraints) will thus be much hillier and full of traps for algorithms based upon neighbourhood search, so the problem will differentiate more clearly between the algorithms. This explains why the improvements of the hybridisations **TS1** and **TS2** are considerably higher for Problem 1 than those of the original tabu search algorithms **TS**, and the additional improvements yielded by the memetic approaches. We can see in the **TSPOP** rows of Tables 9.1 - 9.4 that applying the **TS1** algorithm using random ordering of skill categories starting from a number of different initial solutions (without any recombination or switch) is not effective - showing clearly the dependence upon the human planner's knowledge of the sequencing which must be applied to the ordering of skill categories. It seems to be a good idea not to do the time consuming greedy shuffling step on every individual but only on the best one. The algorithms make use of a random generator (to create an initial solution and to choose among equally good steps). This explains why some of the experiments with more individuals lead to worse solutions than with fewer individuals. However, the memetic hybrids **SWT** and **MEH** and the memetic algorithm **ME4** demonstrate that the recombination operator consistently improves performance over the **TSPOP** algorithm which has no recombination given similar time to solve the problems, and moreover, they do not require the user to specify the order in which skill categories should be scheduled.

The memetic/tabu hybrid **MEH** shows excellent performance over the more difficult Problems 1 and 4 bettering all other solution methods except the hybrid **SWT** algorithm. This demonstrates the better solutions obtainable and the increased robustness offered by a hybrid approach.

Originally, when planning according to option RMP, we start planning the minimum personnel requirements and at the end add duties to the schedule whenever this does not introduce new violations of soft constraints (see Section 5.6.1). We are inclined to think that better results can be obtained by adding duties while the planning algorithm is still active. The **SWT** algorithm, which was developed to test this, indeed leads to good results for all the examples tested. Note that this algorithm has no greedy shuffling step in the end, in contrast to the **TSPOP** and the **MEH** algorithms.

9.6 Conclusion

By automating the nurse rostering problem for Belgian hospitals, the scheduling effort and calculation time are reduced considerably from the manual approach that was previously used. The time for automatic schedule generation can be tailored to suit the time available. Fast tabu search algorithms can quickly find reasonably good schedules in response to events such as staff absenteeism. The memetic algorithms are robust enough to produce excellent solutions to hard problems when more time is available. The quality of the automatically produced schedules is much higher than the quality of the manual schedules.
We have described several memetic approaches and compared them to previously obtained tabu search results (Chapter 8). The hybrid tabu search algorithm runs quickly and does produce good solutions but it is highly dependent on the initialisation parameters, requiring the expertise of human planners to judge the correct order of skill categories and displays a lack of robustness to generate good schedules for all problems. The memetic approaches take much longer to run than the tabu search approaches. Those memetic approaches which copy only a carefully selected part of each parent schedule to the child schedules use this extra time to good effect to produce better solutions and the dependence on the initialisation and parameter changes is very much reduced. The hybrid memetic algorithms, which combined the basic approach with the hybrid tabu search provide good solutions in a similar time to the other memetic algorithms. The solutions are significantly better than the best tabu search solution and they are relatively unaffected by initialisation and parameter changes. We believe that these approaches are particularly robust to handle the variety of instances that occur in the real-world.

Part IV

A Different Framework

Chapter 10

Multi Criteria Approach

10.1 Introduction

In this chapter, we present a multi criteria approach to nurse scheduling which overcomes some of the practical difficulties that personnel schedulers in hospitals often face. Users of hospital personnel planning software often cope with the complex task of translating their needs into several constraints of a very different nature and with differing cost parameters (see Section 4.2.4). The approach presented in this chapter is an attempt to address this issue.
Compared to the previously developed cost function guided methods (Chapter 7 - 9), this method allows for a much more flexible formulation of the problem specific requirements. It is no longer the user's responsibility to compare different quality measures, whereas in the cost function approach, violations of completely different constraints are added up for the evaluation (Chapter 4). In this approach, schedules are evaluated by measuring their position in a preference space. We especially construct that preference space in order to evaluate all the criteria in dimensionless units.
In the multi criteria approach, users can express the importance of criteria according to their preferences. This corresponds more to the hospital customs than aiming at the overall best schedule in terms of a particular cost function. The new evaluation framework presented in this chapter is accepted for publication as *E.K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe: A Multi Criteria Meta-heuristic Approach to Nurse Rostering, Proceedings of Congress on Evolutionary Computation, CEC2002, Honolulu, IEEE Press, 2002, 1197-1202* [35].
Some problems for the user of the personnel planning software are outlined in Section 10.3. Section 10.4 elaborates on the developed multi criteria method in order to tackle these particular drawbacks of the personnel scheduling problem. We carried out a set of tests and the results are explained in Section 10.5. In Section 10.6, some conclusions on the new multi criteria approach for the personnel planning problem are drawn.

10.2 Multi Criteria Decision Making

The assignment is subject to a set of constraints which vary from hospital regulations to specific personal requests such as holidays and days off. In the multi criteria approach, we still maintain the rule that the coverage constraint (assign the requested number of skilled personnel at any time) can never be violated during the course of the search. All of the meta-heuristics which were discussed in Chapter 7 - 9 could be applied to the multi criteria approach discussed in this chapter.

The nurse rostering problem is a complex combinatorial problem, which is characterised by multiple goals. Some of the constraints are easier to satisfy than others and that should be taken into consideration within the search algorithm. Very few researchers have worked on multi criteria approaches to nurse timetabling problems. In most approaches, personnel coverage is treated as a goal, unlike in this thesis, where it is a hard constraint.

Some publications on multi criteria or goal programming for nurse scheduling are mentioned in the literature overview. Arthur and Ravindran [8] combine goal programming with a heuristic assignment of shifts. Musa and Saxena [153] developed an interactive approach for small size problems. The scheduling problem tackled by Franz et al. [96] differs considerably from the problem defined in Chapter 2 in that personnel works at scattered locations. Ozkarahan [162] and Ozkarahan and Bailey [166] consider coverage and time related objectives, in addition to maintaining the usage of full time staff. Chen and Yeung [56] apply goal programming in an expert system. Berrada et al. [21] have developed a tabu search approach with multi objective mathematical programming for small size problems and Jaszkiewicz [122] applies simulated annealing in a multi criteria approach.

There are also applications of multi criteria approaches for other timetabling problems. Lotfi and Cerveny [133] developed an exam scheduling package at a large university. They give priorities to three different quality measures for a schedule. Similar approaches to the method described in this chapter have been developed for timetabling problems by Burke et al. [33] and Paechter et al. [170]. Although the main concept for the distance measure is the same, the high number of constraints in the nurse rostering approach makes it inconvenient to rely upon the planner's practical experience for setting the targets (as presented in [170]). Burke et al. [33] developed their multi criteria approach for examination timetabling as a two-phase algorithm. The first phase applies a graph colouring heuristic to generate a set of timetables with a high quality with respect to the criteria separately. Afterwards, an iterative search in the neighbourhood of these timetables tries to improve the quality in terms of the other criteria. The approach is similar to the multi criteria method developed for ANROM in that weights of criteria reflect the importance of constraints and they are set by the scheduler.

10.3 Drawbacks of the Cost Function Approach

The development of ANROM started in 1993 but the model has not stopped evolving as new hospital users appeared with different demands and planning habits. The program currently in use is a very complex system based on ANROM, which can be fine-tuned by the hospital planners in order to meet their requirements.

ANROM has been adapted to an increasing number of user defined constraints, in order to meet the high requirements of all the different hospital wards. The system provides modifiable functionalities to all customers. However, the growing number of constraints renders the task of assigning cost parameters to constraints increasingly difficult. Also, it is rather artificial to unify constraints with completely different characteristics, such as overtime, weekend work, replacing people with a different qualification class, etc.

The cost function per personal schedule is defined as a linear combination of the violations of the constraints (see Chapter 4). For evaluating the entire schedule, the cost function values per personnel member are summed.

The problem tackled in this thesis copes with extremely tough constraints. It aims at producing a satisfiable schedule even when violations of the soft constraints are unavoidable.

Compared to similar problems described in the literature (Section 3.2.6), ANROM offers plenty of possibilities for the schedulers to modify the evaluation function by defining different flexible work regulations and by setting the cost parameters related to the soft constraints. The latter, however, sometimes requires more insight into the search heuristics than the hospital planners can be reasonably expected to have.

An often encountered difficulty when applying ANROM for scheduling in practice is the fact that planners have no knowledge about the profile of the search space. By increasing the cost parameter of one particular soft constraint, they do not necessarily obtain better quality solutions in general, nor even with respect to that constraint in some particular cases. The experience we built up while testing a large set of rostering problems from practice learned that increasing one cost parameter can create very high barriers, which cannot easily be crossed, in the search space.

10.4 Multi Criteria Approach for the Nurse Rostering Problem

10.4.1 Soft constraints

In this section, we set up the modelling of nurse scheduling problems as multi criteria problems. We opted for the approach in which each criterion measures the number of violations of one soft constraint. Violations of the soft constraints are measured in different units: hours, shifts, days, weekends, and their combinations. A multi criteria approach enables constraints of different nature, expressed in different units, to be treated simultaneously. The soft constraints enumerated in this section can take different values for different personnel members in the ward. We will briefly describe the criteria groups based on the units in which they are expressed.

Hours: The violation of this particular set of constraints can be measured in terms of hours. Examples of such constraints are:

- Overtime (Constraint 8)
- Undertime (Constraint 9)
- Some of the counter constraints (Constraint 23)

Shifts: In order to calculate the violation of constraints belonging to this group, we count assignments of shift types to the personnel members. Depending on the constraint values, some sequences or occurrences will lead to violations. This group contains constraints concerning a minimum or maximum total number or sequence of shifts. Also belonging to this group are constraints which require people to work together, constraints on replacing people with a different qualification, etc. Some of these constraints are:

- Allow work for an alternative skill category (Constraint 2)
- Maximum number of assignments (Constraint 3)
- Maximum number of assignments per day of the week (Constraint 10)
- Maximum number of assignments for each shift type (Constraint 11)
- Maximum number of a shift type per week (Constraint 12)
- Number of consecutive shift types (Constraint 13)
- Maximum number of assignments on bank holidays (Constraint 20)
- Restriction on the succession of shift types (Constraint 21)
- Some of the counter constraints (Constraint 23)
- Shifts off (Constraint 25)
- Requested assignments (Constraint 26)
- People who should work together (Constraint 27)
- People who should not work together (Constraint 28)

Days: These constraints are independent from the actual shift types or number of hours scheduled. This constraint category needs information whether or not a person works on a certain day. Violations of the minimum or maximum number of days and consecutive days are expressed (as might be expected) as a number of days. Examples are:

- Maximum number of consecutive days (Constraint 4)
- Minimum number of consecutive days (Constraint 5)
- Assign 2 free days after night shifts (Constraint 14)
- Some of the counter constraints (Constraint 23)
- Days off (Constraint 24)

Weekends: The weekend constraint group is treated separately because weekends seem to attract more attention in real-world examples than other constraints. Violations are expressed as a number of weekends in the following constraints:

- Assign complete weekends (Constraint 15)

- Assign identical shift types during the weekend (Constraint 17)
- Maximum number of consecutive working weekends (Constraint 18)
- Maximum number of weekends in a period of 4 weeks (Constraint 19)
- Some of the counter constraints (Constraint 23)

Miscellaneous: This set of constraints actually covers several of the constraint groups described above. They represent very complex requirements to personnel schedules and require a calculation of hours, shifts, days, etc for the evaluation. Examples of such constraints are:

- Patterns (Constraint 22)
- Counters (Constraint 23)

10.4.2 Search space

The search space for the proposed multi criteria approach is defined in this Section. We use compromise programming, which is based on the concept of the distance from an ideal solution [226]. Each solution for the problem (a schedule) is represented as a point in a criteria space whose dimension is equal to the number of criteria. Two points in the criteria space have a special meaning: the ideal point and the anti-ideal point. In the ideal point, all criteria take their best value. Very often no solution corresponding to this ideal point exists. We also define an anti-ideal point, represented by the worst value for all the criteria.
In order to tackle all the criteria in dimensionless units, the criteria space is mapped to the preference space [171]. The quality of a solution is expressed in terms of the number of violations of a constraint. Larger co-ordinates indicate worse solutions. For each criterion, the best value is thus mapped to 0 and the worst value is mapped to w_c, where w_c denotes the relative importance (weight) of the criterion c. Obviously, the ideal point is mapped to a point in the preference space, whose co-ordinates are all equal to 0, while the anti-ideal point is mapped to a point whose co-ordinates are all equal to the weights of the criteria. A simplified example of a preference space is explained in detail in Section 10.4.3 (see Fig. 10.1). In this approach the quality of a schedule will be measured by its distance from the ideal point in the preference space. A smaller distance indicates a higher quality.
In order to define a mapping from the criteria space of the nurse rostering problem onto a preference space, a best and worst value are calculated for each criterion (here soft constraint). We call all the schedules which do not violate a particular constraint *ideal schedules* with respect to that constraint. These ideal schedules are mapped to points with co-ordinate 0 for the criterion corresponding to the constraint. An example of a schedule which is ideal in terms of the maximum number of hours worked (Constraint 8) is a solution in which the constraint on overtime is not violated in any of the personal schedules. For the constraint concerning the minimum number of consecutive days (Constraint 5), it suffices to have no shorter sequence of working days in any of the personal schedules, to consider the solution ideal. It is clear from experience that the best value lies in an infeasible part of the solution space for nearly all real-world

problems.
The calculation of the worst values of the criteria is more complex. We will illustrate the idea by using a number of examples. Consider the constraints with hourly measures. An estimation of the worst case in terms of overtime is a schedule in which people have to work day and night, without any break during the entire planning period. In this situation, both overtime and the minimum time required between shifts will be violated to the highest extent. In order to violate the undertime constraint (Constraint 9) to the highest extent, we consider for all the personnel members, a schedule without any assignment.
For the constraints from the shift category which limit the number of assignments (Constraint 11, 12, etc) the procedure is analogous to that for the hourly constraints. The consecutiveness constraints, however, require a more careful approach. In order to violate the constraint on the number of consecutive shift types (Constraint 13) to the highest extent, we imagine a schedule with as many forbidden sequences as possible. The highest number of forbidden sequences is obtained by alternating the smallest forbidden sequence of assigned shifts and a free day. The constraint on complete weekends (Constraint 15) can be violated most by a schedule in which all Saturdays are free and all Sundays are assigned (or the other way round), for all the personnel members.
Some constraints take the previous planning period into account. If the previous planning period ends with a scheduled day, we will start the worst schedule for 'Minimum number of consecutive free days' with a free day, and vice versa.
For some criteria in which more than one person is involved, we combined full and empty schedules to simulate the worst case. Table 10.1 gives an overview of the schedule types which represent the anti-ideal points for each of the criteria. 'Full' and 'empty' schedules are those in which all the possible shifts are assigned and those in which no assignment is made during the planning period.

10.4.3 An algorithm for multi criteria search

Search Algorithms

Our aim is to use the previously developed meta-heuristics for ANROM (Chapter 7 - 9) in a new multi criteria environment. Instead of a cost function summing the violations of all the constraints, the driving force is now a minimisation of the distance from a solution to the ideal point in the preference space.
The tests performed for this chapter make use of the hybrid tabu search algorithm described in Section 8.3.6. After an initialisation phase, which basically consists of assigning all the requested shifts to qualified people randomly, the tabu search algorithm is applied. The algorithm never violates any hard constraints during the calculations. Also in the multi criteria approach, the algorithm used to demonstrate this multi criteria approach makes use of two hybridisations: sorting out full weekend work and improving the schedule of the person with the worst result (called TS1 in Section 8.3.6).
In the cost function approach, the program enables users to set a cost parameter

Constraint		Schedule Characteristics
1	Minimum time between two assignments	Full
3	Maximum number of assignments	Full
4	Maximum number of consecutive days	Full
8	Maximum number of hours worked	Full
10	Maximum number of assignments per day of the week	Full
11	Maximum number of assignments for each shift type	Full
12	Maximum number of a shift type per week	Full
18	Maximum number of consecutive weekends	Full
19	Maximum number of working weekends in 4 weeks	Full
20	Maximum number of assignments on bank holidays	Full
24	Days off	Full
25	Shifts off	Full
23	Counters	Full for this one, empty schedule for all the other people
27	People who should work together	1 Full and 1 Empty schedule
28	People who should not work together	Full schedule for both people
6	Maximum number of consecutive free days	Empty
9	Minimum number of hours worked	Empty
13	Number of consecutive shift types	Alternating the smallest forbidden sequence and free days
21	Restriction on the succession of shift types	Repeated forbidden sequences
26	Requested assignments	Empty
5	Minimum number of consecutive days	Alternating scheduled and free days
7	Minimum number of consecutive free days	Alternating scheduled and free days
14	Assign two free days after night shifts	Alternating Night Shifts and empty days
15	Assign complete weekends	Every Saturday planned, every Sunday empty
17	Assign identical shift types during the weekend	A shift type every Saturday a different shift type every Sunday
22	Patterns	Opposite of the pattern

Table 10.1: Schedules representing the worst cases of the corresponding constraints

for every constraint (see Section 4.2.4). The value of the cost function is the weighted sum of the violations of soft constraints. In an attempt to find a good schedule a set of practical cost parameters, set by the customers of the software were copied into the weights of the multi criteria approach. Section 10.5 also presents results with other weight values.

Quality of the Schedule

Each personal schedule is considered separately by measuring its distance from the ideal point in the preference space. The goal for the search algorithm is to minimise the sum of the distances for P people in the schedule. We use as a distance measure a family of L_x metrics, which gives a wide range of geometric measures for different values of x. A distance from a personal schedule $schedule_p$ for person p to the ideal point is denoted by $L_x(p)$ (Definition 5) where $schedule_p$ has co-ordinates $w_c s_{p,c}$ (c = 1 ... C: number of criteria) in the preference space.

Definition 5 $L_x(p) = (\sum_c [w_c s_{p,c}]^x)^{1/x}$*;* $1 \leq x \leq \infty$

In the definition, $s_{p,c}$ is $\frac{f_{p,c}}{f_{p,c,worst}}$, where $f_{p,c}$ is the value of criterion c in the schedule of person n and $f_{p,c,worst}$ is the value of the criterion c for the worst possible schedule for p. Smaller values of x allow for compensation among criteria values, i.e. high satisfaction for one constraint can counterbalance low satisfaction for another one. If the distance measure is ∞, the distance $L_x(p)$ defined in Definition 5 is the value of the largest co-ordinate in the preference space. Only the most violated constraint will contribute to the value.

Definition 6 $L_\infty(p) = max_c[w_c s_{p,c}]$

The distance of the entire schedule S is the sum, for all the P nurses in the ward, of the personal schedules (as denoted by Definition 7).

Definition 7 $L_x(S) = \sum_{p=1}^{P} L_x(p)$

The mapping of the criteria space into the preference space, with 2 criteria, is depicted in Fig. 10.1 and Fig. 10.2 presents the pseudo code for a multi criteria search algorithm.

10.5 Experiments

10.5.1 Test problems

In order to carry out preliminary experiments, it was our deliberate choice to tackle simplified examples. In comparison with ANROM, the test problems in this chapter have a limited number of constraints, smaller problem dimensions and thus a smaller search space. These data enabled us to carry out preliminary experiments with several approaches in a limited amount of calculation time. After promising early results, this multi criteria approach will be tested on more complex real-world data.

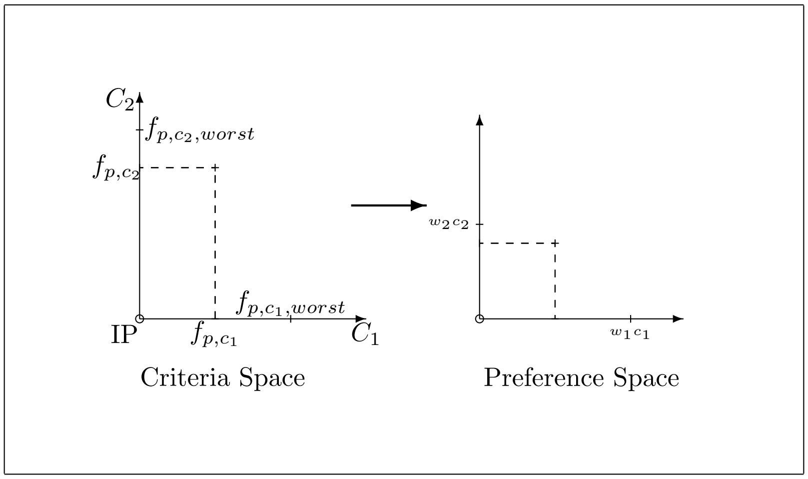

Figure 10.1: Mapping from the criteria space into the preference space for person p's schedule

In Example 1, the personnel requirements consist of 4 different shift types, and a planning period of 4 weeks is considered. It is much simpler than any of the problems tackled in PART III of this thesis because this completely new approach is still in a very early stage. The ward employs 9 people belonging to 2 different skill categories; one head nurse and 8 regular nurses. The head nurse has a special work regulation (called Head Nurse) which allows day shifts only and no weekend work. Among the regular nurses, five of them have a Full Time work regulation, one has a personalised full time regulation (forbidding late and night shifts on Tuesday, by Constraint 25) and the other two are Half Time nurses. This particular problem has some extra difficulties, affecting the worst case values of some personal constraints. One full time nurse is hired by another

- initialise a feasible schedule S
- $BEST \leftarrow S$
- WHILE stop criterion is not reached
    - perform best non-tabu move to S' in the neighbourhood of S: $S \leftarrow S'$
      IF $L_p(S) \leq L_p(BEST)$
      THEN $BEST \leftarrow S'$
- return $BEST$

Figure 10.2: Pseudo code for multi criteria search

ward during the first two weeks of the planning period. Another full time nurse leaves the ward after two weeks. Furthermore, one half time and one full time nurse swap work regulations after the first week of the planning period. The

Multi Criteria			Violations	
Weights	x	$L_x(S)$	Constraints	#
all weights 1	2	14	patterns	8
patterns 2 other weights 1		56	patterns	8
patterns 3 other weights 1		117	patterns	7
patterns 4 other weights 1		336	patterns	7
patterns 5 other weights 1		525	patterns	7
patterns 10 other weights 1		1300	patterns	7
patterns 50 other weights 1		22707	# weekends in 4 weeks min. consecutive days patterns	1 1 5
cost parameters		225	patterns	5
Cost Function			Violations	
Cost Parameters		Value	Constraints	#
		250	patterns	5

Table 10.2: Results for Example 1

data in Example 2 represent the same ward as in Example 1 but the planning period is different. It also involves different qualifications and work regulations for some personnel members, another position in the cyclic patterns (because of the value $start\_pattern_p$), different personal requests for days off, different bank holidays, and a slightly different personnel demand on some days.
The tests have been carried out with distance measure x=1,2, and ∞ in the preference space.

10.5.2 Discussion

Table 10.2 presents the test results for Example 1 and in Table 10.3 some test results are displayed for Example 2. Experiments have been carried out with different values for the weights. In the last two columns, the violated constraints are presented together with the number of times the constraint is violated. The last column sums the number of violations when they occur in different personal schedules. An ideal schedule does not violate any of the soft constraints. Both test examples demonstrate that increasing the relative importance of a particular constraint often results in a schedule with less violations of that constraint.

Multi Criteria			Violations	
Weights	x	$L_x(S)$	Constraints	#
all weights 1	1	19	min. consecutive days	1
			patterns	9
patterns 10		38	hours	3
other weights 1			min. consecutive days	2
			patterns	2
patterns 50		8	hours	1
other weights 1			min. consecutive days	7
patterns 50, min. cons. days 10		312	hours	6
other weights 1			patterns	12
all weights 1	2	72	max. consecutive days	3
			patterns	7
patterns 2		114	hours	2
other weights 1			patterns	8
patterns 5			hours	2
other weights 1			patterns	6
patterns 10			min. consecutive days	2
other weights 1			identical weekends	2
			patterns	2
cost parameters		0		
all weights 1	∞	125	hours	13
			max. assignments	1
			# weekends in 4 weeks	3
			day off	3
			min. consecutive days	5
			complete weekends	1
			identical weekends	1
			patterns	9
patterns 10		777	hours	74
other weights 1			max. assignments	6
			# weekends in 4 weeks	3
			day off	4
			min. consecutive days	13
			complete weekends	2
			identical weekends	6
			patterns	14
cost parameters		3642	hours	192
			max. assignments	18
			max. consecutive days	6
			# weekends in 4 weeks	1
			min. time between	78
			day off	5
			min. consecutive days	6
			complete weekends	1
			identical weekends	7
			patterns	14
Cost Function			Violations	
Cost Parameters		Value	Constraints	#
		7	min. consecutive days	7

Table 10.3: Results for Example 2

This does not generally hold because of the local optima. Although we realise that it is not possible to compare results for different objectives, the general tendency of applying a multi criteria method will be explained.

The best solution for Example 1 violates the 'pattern' constraint 5 times. We know, after investigating the simplified problem data thoroughly, that there is no solution which corresponds to the ideal point. When assigning a moderate weight to the pattern constraint, it is violated more often. Since the data of the cost function approach is available anyway, we experimented with copying the cost parameters into the weight factors of the corresponding constraints. It is not irrelevant to assume a correspondence between the cost parameters set by the users and the importance of the corresponding constraints. This option for the weights led to the best results for the multi criteria approach. However, it is not clear that these preliminary results will also hold for real-world problems.

The results from the Example 2 dataset clearly show that a lower value of the distance measure x enables compensations between constraints. With higher values of x, the algorithm tends to generate schedules which reduce the penalty of the most violated constraint because its contribution to the value of the cost function is very high. The results with distance measure ∞ are not promising at all. The first experiment was carried out with all weights equal to 1. It is only the pattern constraint which determines the value of the distance, although many other constraints are also violated. Increasing the weight for the patterns seems the most obvious thing to do, but as is presented on the next row in Table 10.3, the results are even worse. For some people, the violation of that particular constraint has decreased at the expense of other constraints. In the schedule of some personnel members, when other constraints are responsible for the distance, the violation of patterns has even increased. Copying the cost parameters into the weights gives the worst results of all, whereas this was the best option for the x=2 distance measures.

Increasing the weights for the most violated constraint almost always leads to better quality solutions in which less constraints are violated. Also for this test example, copying the cost parameters into the weights is the best option. It is not possible to compare the results obtained using the cost function and the multi criteria approach because they are based on different evaluation functions. In fact, the comparison of these two approaches based on numbers of all the constraint violations in the obtained schedules is a multi criteria problem in itself. However, in one particular case, we can see that the multi criteria approach outperforms the single cost function approach. It happened in the Example 2 dataset when the multi criteria approach gave 0 violations of all of the constraints while the cost function approach resulted in 7 violations on minimum consecutive days (Constraint 5). Although there is no proof of the quality of the newly developed multi criteria approach for realistic problems in which simultaneously satisfying all the soft constraints is not possible, the first obtained results are very promising.

10.6 Conclusion

The new multi criteria approach presented in this chapter overcomes some practical difficulties for automated nurse rostering, which is a multi criteria problem by nature. It enables handling dissimilar constraints in a better way than a single cost function approach does, by taking the ranges of possible values of criteria into consideration. The developed multi criteria approach, incorporated in a meta-heuristic, allows the scheduler to control the compensation of constraints. Some of the constraints are easier to satisfy than others and that should be taken into account within the search algorithm. The multi criteria approach described in this chapter allows the users to set weights that reflect the relative importance of the constraints. The search for a solution corresponds more to reality in which the scheduler controls the compensation of constraints. Instead of aiming at the lowest number of violations of soft constraints (taking cost parameters into account), the current approach aims at an ideal point in a preference space. Tests on real-world data have shown that previously developed search algorithms perform very well using the new multi criteria method. Although comparing the results of different objectives is not possible, simple tests provide strong indications that this new multi criteria approach responds better to the planner's changed objectives. Assigning a larger weight to a criterion will make the corresponding constraint more important during the search, without increasing the risk for cutting off parts of the search space.
In general, the multi criteria approach presented in this chapter has potential to accommodate every day practice of hospital schedulers better than the single cost function approach.

Chapter 11

General Discussion

11.1 Problem Definition and Model Development

Nurse rostering is a very complex combinatorial problem, for which hardly any software system exists. During this study it became clear that, especially in Belgian hospitals, any assistance for the head nurses or ward managers to automatically generate their monthly rosters could save a lot of time and effort. Several levels of decision making can be distinguished in nurse scheduling but the problem dealt with in this thesis is situated at the short-term timetabling level. Its main objective is to understand and automatically generate comfortable shift schedules for personnel members in order to meet the staff coverage. The first major achievement of this research is **the determination of the key parameters in this issue and the development of a highly flexible model for nurse rostering**.

Hospital personnel scheduling deals with particular difficulties, originating from real-world practical issues such as unpredictable work load, round the clock work, a large number of different skill categories, flexible work contracts such as part time work, night nurses, etc. One of the major contributions of this thesis is that it **captured the extensive set of realistic constraints, and integrated them, together with explicit and implicit objectives, in a general, flexible model** called ANROM (Advanced Nurse ROstering Model). The requirements of hospital planners motivated the choice for considering personnel coverage as a hard constraint, which can be relaxed in a controlled manner. In our model, the entire set of time related constraints on personal schedules are all treated as soft constraints.
Although strategic decisions, such as hiring trained people for specific skill categories or assigning personnel to specific wards, are not considered part of rostering, we did include the results as input to our model. Furthermore, within the limits of the available information, we did an effort to describe the problem

as accurately as possible, including modifiable constraints and parameters, so that it is applicable to a wide range of personnel scheduling problems.

The development of the solution framework, with modifiable evaluation tools and a large set of heuristics targeting specific objectives, constitutes a third contribution of this thesis. It enables to set and modify constraints and selects a suitable combination of procedures for particular requirements, without expecting a feasible problem formulation from the planners. Thanks to the **modular and abstract evaluation method**, the calculation of the solution quality can easily be fine-tuned and incorporates violations of all the possible soft constraints. The evaluation procedure provides feedback for the planners in practice, and assists them in making their problem formulation more coherent.
We made an attempt to guide the planners by performing a consistency check on their settings. Moreover, we provide procedures that assist in relaxing certain hard constraints, or in avoiding understaffing by better aiming at the preferred personnel coverage, and even in diminishing 'undertime' when some personnel members are in danger of not fulfilling their contracts, etc. Floating personnel requirements have been introduced in this thesis as a completely new time interval based formulation of personnel requirements. In many cases, floating requirements reflect much better than any other approach, what the hospital planners want. Although this option substantially enlarges the search space, it also enables the calculation of good quality solutions, by allowing different shift type combinations for satisfying the coverage constraints.
This set of modules in the solution framework increases the applicability of the rostering algorithms in different real-world settings. All the modules are compatible with the search heuristics that form the third part of this thesis.

A set of meta-heuristics and hybrids are included in that solution framework, as the central search force for solving ANROM problems. As the research evolved, we gradually obtained more implicit knowledge about the problem's search space and the behaviour of meta-heuristics. The development of many heuristics and hybridisations often originates in the understanding of shortcomings of preceding approaches. We therefore consider **the design of new meta-heuristics and hybrids for nurse rostering** as another key contribution of this thesis.

While maintaining the hard constraints, a large set of search environments can be considered. They were introduced as part of the variable neighbourhood search heuristic. The approach enables the algorithms to escape from local optima, especially by applying neighbourhoods which aim at solving particular constraints and providing very sensitive optional characteristics in the solution.

The nurse rostering package based upon ANROM has become commercially available and is now used in many wards in Belgian hospitals. It includes the hybrid tabu search approach that was introduced in

this thesis and provides different algorithm combinations, suitable for several requirements. Fast tabu search algorithms can quickly find reasonably good schedules in response to events such as staff absenteeism. Some hybridisations have demonstrated that algorithms can be made much more effective through the implementation of diversification and a greedy steepest descent search in an environment inspired by watching interactions of users. The users of the model often put an emphasis on the higher quality of the solution because the algorithms provide an objective schedule treating all nurses equally and in which the number of violated constraints is very low. Planners are pleased by the fact that the system gives the ability to generate consistently better solutions than manual planning procedures while providing a high level of robustness. They acknowledge also that experienced planners cannot easily improve the schedules by hand.
The new memetic approach is effective enough to produce excellent solutions to hard problems when more time is available. Memetic algorithms outperform the heuristics in which single solutions evolve, mainly because they provide a search from different starting points but also due to our effort to find good characteristics of solutions.

Another achievement of this thesis is that we **gained insight in the behaviour of applying heuristics and in making use of different problem specific neighbourhoods**.

No commercial package, nor models presented in literature were suitable for tackling the widely varying nurse rostering problems encountered in practice. They are either set up to solve a problem in one particular hospital, or they have been tested on theoretical problems only. Consequently, benchmarking is not possible and therefore we made use of test data provided by users of the system based on ANROM. They represent a relevant group of hospital schedulers, since the system is applied in over 40 Belgian hospitals, some of which consist of 100 wards and more.

None of the developed meta-heuristics violate hard constraints during the iterative search. They all aim at reducing the value of the cost function. Only the multi criteria approach makes use of a different quality measure, namely the distance from the solution to an ideal point in the preference space. **The multi criteria approach of this thesis opens perspectives for releasing the planners from setting the cost parameters.** It is more realistic and increases the flexibility in setting the weights and thus modifying the relative priority of the constraints. The multi criteria approach enables combining objectives of different nature and importance in the preference space. It is no longer necessary to compare costs for incommensurable quantities, such as overtime, consecutive free days, etc. Some soft constraints are easy to satisfy, others cannot be satisfied while maintaining feasibility. The definition of an ideal and an anti-ideal point in the preference space simplifies the assessment of good quality solutions.

By automating the nurse rostering problem for Belgian hospitals, the scheduling effort and time are reduced considerably as compared to the manual approach that was previously used. The time for automatic schedule generation can be tailored to suit the time available by selecting appropriate search heuristics.
The proposed solution method saves lots of time for the planners in practice and provides an unbiased way of generating the schedules for all the personnel members. It enables simple verification of the constraints, helps redefining unrealistic hard constraints, and thus leads to an overall higher satisfaction among the personnel, as is manifest in many applications.

11.2 Directions for Future Research

For inexperienced users of the system based on ANROM, the large set of soft constraints is sometimes rather inconvenient. The model does not foresee many options to warn the planners in case of conflicting requirements. Also, application in practice revealed that hospital planners themselves are not always very strict in satisfying what they define as hard constraints. The developed nurse rostering model often requires more insight in the characteristics of specific data than planners in practice can be expected to have. Some of the planning procedures, which can be combined with the meta-heuristic search algorithms, already assist in setting feasible hard constraints or relaxing them when necessary. It will be beneficial for the model to take the idea of relaxing the rather strict distinction between hard and soft constraints further.
The consistency check procedure already deals with some precedence soft constraints, and guides the hospital planners towards a more consistent definition of their problem. However, the remaining set of soft constraints that are not pre-checked is very large. The consistency check procedure will merit from an extension with pre-evaluation of a larger set of soft constraints. This will provide interesting guidelines for planners to formulate their problem so that feasible solutions exist.
Most personnel members attach a high importance to the personal constraints such as days off, shifts off, requested assignments, and to patterns in which these constraints appear. It is likely that these constraints are of less importance for other people, who might, for example, be more concerned about overtime. Since the problem is too complex to compute whether a solution exists that satisfies all the constraints simultaneously, a more user dependent consistency check would be very helpful.

The meta-heuristic approaches have depicted several possibilities for improving their performance in widely varying real world applications. We have built a set of neighbourhoods for local search, which are rather generally applied in the algorithms. Future refinements of the algorithms will make more use of problem-specific information to apply the best suitable neighbourhoods in their search.

Future research will certainly build upon the promising early findings of testing our multi criteria approach on nurse rostering. With this novel method, we have initiated an interesting new direction in nurse rostering research since the problem is clearly a multi criteria problem by nature. We will pay attention to appropriately calculating both the best and the worst values for all the soft constraints related to every personnel member's schedule. Coverage constraints are objectives rather than hard constraints in real-world practice. Other criteria that deserve some attention are: evenly dividing violations of certain constraints among people, combining soft constraints (e.g. all the constraints related to overtime), compensating unwanted schedules from the past, etc.
Our future work will also include testing the behaviour of search algorithms that modify the weights of criteria in order to escape from local optima. It will consider the fact that some constraints are more easily satisfied, and adapt the weights specified for the corresponding criteria accordingly.

Although the nurse rostering model was developed explicitly to address hospital personnel rostering, the techniques and methods developed as a result of this research are certainly applicable to other personnel scheduling problems. Of course the presented algorithms deal with the extensive set of soft constraints, of which many are only valid in healthcare. Moreover, other sectors require the evaluation of constraints on locations, equipment, etc irrelevant in nurse rostering. Nevertheless, future development of widely applicable real-world scheduling models can definitely profit from this thesis and build upon its different modules, such as 1) the development of the main building blocks of the model, with user definable constraints, modifiable variables and parameters, 2) the inclusion of an explanatory evaluation function, 3) the use of pre- and post-planning algorithms to assist in formulating the requirements and to relax certain constraints, and lastly the application of abstract meta-heuristics that generate good quality solutions.

Personnel rosters in healthcare are not only schedules for arranging patient care but they also affect diverse aspects of the hospital organisation and directly influence the lives of the personnel. The nursing profession is known to involve very hard and stressful work, irregular hours, and lots of night and weekend work. Shortage of staff is very common in hospitals and therefore this work could (and does already) contribute to scheduling available staff more equitably and more efficiently.

Bibliography

[1] E. Aarts, J.K. Lenstra: Local Search in Combinatorial Optimization, Wiley, ISBN 0-471-94822-5, 1997

[2] W.J. Abernathy, N. Baloff, J.C. Hershey, S. Wandel: A Three-Stage Manpower Planning and Scheduling Model - A Service-Sector Example. Operations Research 22, 1973, 693-711

[3] J. Ahmad, M. Yamamoto, A. Ohuchi: Evolutionary Algorithms for Nurse Scheduling Problem, Proceedings of the 2000 Congress on Evolutionary Computation, CEC00, San Diego, ISBN 0-7803-6375-2, 2000, 196-203

[4] U. Aickelin, Genetic Algorithms for Multiple-Choice Optimisation Algorithms, PhD thesis, European Business Management School University of Swansea, 1999

[5] U. Aickelin, K. Dowsland: Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem, Journal of Scheduling, Volume 3 Issue 3, 2000, 139-153

[6] H.K. Alfares: Days-off Employee Scheduling Over a Three-Week Work Cycle, E.K. Burke, W. Erben (Eds.): Proceedings of the 3rd international conference on the Practice and Theory of Automated Timetabling, ISBN 3-00-003866-3, 2000, 251-258

[7] M. Anzai, Y. Miura: Computer Programme for Quick Work Scheduling of Nursing Staff, Medical Informatics 12, 1987, 43-52

[8] J. Arthur, A. Ravindran: A Multiple Objective Nurse Scheduling Model, AIIE Transactions, Vol. 13, No. 1, 1981, 55-60

[9] J.E. Bailey: Integrated days off and shift personnel scheduling, Computers and Industrial Engineering 9, 1985, 395-404

[10] J. Bailey, J. Field: Personnel Scheduling with Flexshift Models, Journal of Operations Management, Vol. 5, 1985, 327-338

[11] V.A. Bardadym: Computer-Aided School and University Timetabling: The New Wave, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, 1995, 22-45

[12] J.W. Barnes, M. Laguna: A tabu search experience in production scheduling, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1993, 141-156

[13] J.E. Beasley, B. Cao: A Dynamic Programming based Algorithm for the Crew Scheduling Problem, Computers and Operations Research, Vol. 25, No. 7, 1998, 567-582

[14] S.E. Bechtold, M.J. Brusco, M.J. Showalter: A Comparative Evaluation of Labor Tour Scheduling Methods, Decision Sciences 22, 1991, 683-699

[15] S.E. Bechtold, L.W. Jacobs: Implicit modeling of flexible break assignments in optimal shift scheduling, Management Science 36, 1990, 1339-1351

[16] S.E. Bechtold, M.J. Showalter: A Methodology for Labor Scheduling in a Service Operation System, Decision Sciences 18, 1987, 89-107

[17] G. R. Beddoe, S. Petrovic, G. Vanden Berghe: Case-based reasoning in employee rostering: learning repair strategies from domain experts, submitted to International Transactions in Operational Research

[18] G. R. Beddoe, S. Petrovic, G. Vanden Berghe: Storing and adapting repair experiences in personnel rostering, Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, Gent, 2002, abstract accepted for publication

[19] J. Bennell, K.A. Dowsland: A Tabu Tresholding Implementation for the Irregular Stock Cutting Problem, International Journal of Production Research, Vol. 37, No. 8, 1999, 4259-4275

[20] I. Berrada, Planification d'horaires du personnel infirmier dans un établissement hospitalier, PhD Dissertation, Département d'informatique et de recherche opérationelle, Université de Montréal, Montréal, 1993

[21] I. Berrada, J. Ferland, P. Michelon: A Multi-Objective Approach to Nurse Scheduling with both Hard and Soft Constraints, Socio-Economic Planning Science 30, 1996, 183-193

[22] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, M. Spadoni, A heuristic procedure for the crew rostering problem, European Journal of Operational Research, Vol. 58, 1992, 272-283

[23] J. Blacewicz, P. Hawryluk, R. Walkowiak: Using a Tabu Search Approach for Solving the Two-dimensional Irregular Cutting Problem, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1995, 313-325

[24] R. Blau: Multishift Personnel Scheduling with a Microcomputer, Personnel Administrator, Vol. 20, No. 7, 1985, 43-58

[25] R. Blau, A. Sear: Nurse Scheduling with a Microcomputer, Journal of Ambulance Care Management, 1983, 1-13

[26] D. Bradley, J. Martin: Continuous Personnel Scheduling Algorithms: A Literature Review, Journal of the Society of Health Systems 2, 1990, 8-23

[27] P. Brandimarte: Routing and scheduling in a flexible job shop by tabu search, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1993, 157-183

[28] H.J. Bremermann: The Evolution of Intelligence. The Nervous System as a Model of its Environment. Technical Report No.1, Contract No. 477 (17), Dept. of Mathematics, University of Washington, Seattle, 1958

[29] P. Brucker, D. Schumacher: A New Tabu Search Procedure for an Audit-scheduling Problem, Journal of Scheduling, Vol. 2, No. 4, 1999, 157-173

[30] M.J. Brusco, L.W. Jacobs: Cost analysis of alternative formulations for personnel scheduling in continuously operating organisations, European Journal of Operational Research 86, 1995, 249-261

[31] M.J. Brusco, L.W. Jacobs, R.J. Bongiorno, D.V. Lyons, B. Tang: Improving personnel scheduling at airline stations, Operations Research, Vol. 43, No. 5, 1995, 741-751

[32] H.-J. Bürckert, K. Fischer, G. Vierke: Holonic Transport Scheduling with TELETRUCK, Journal of Applied Artificial Intelligence 14, 2000, 697-725

[33] E.K. Burke, Y. Bykov, S. Petrovic: A Multicriteria Approach to Examination Timetabling, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 118-131

[34] E.K. Burke, P. Cowling, P. De Causmaecker, G. Vanden Berghe: A Memetic Approach to the Nurse Rostering Problem, Applied Intelligence special issue on Simulated Evolution and Learning, Vol. 15, Number 3, Springer, 2001, 199-214

[35] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: A Multi Criteria Meta-heuristic Approach to Nurse Rostering, Proceedings of Congress on Evolutionary Computation, CEC2002, Honolulu, IEEE Press, 2002, 1197-1202

[36] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: Fitness Evaluation for Nurse Scheduling Problems, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 1139-1146

[37] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: Floating Personnel Requirements in a Shift Based Timetable, working paper KaHo Sint-Lieven, 2001

[38] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe: Variable Neighbourhood Search for Nurse Rostering Problems, Proceedings of 4th Metaheuristics International Conference, MIC2001, Porto, 2001, 755-760

[39] E.K. Burke, P. De Causmaecker, G. Vanden Berghe: A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 187-194

[40] E.K. Burke, D.G. Elliman, P.H. Ford, R.F. Weare: Examination Timetabling in British Universities - A Survey, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference Edinburgh, Springer, 1995, 76-90

[41] E.K. Burke, J.P. Newall: A multi-stage evolutionary algorithm for the timetabling problem, IEEE Transactions on Evolutionary Computation, Vol.3, No.1, 1999, 64-73

[42] E.K. Burke, J.P. Newall, R.F. Weare: A Memetic Algorithm for University Timetabling, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference Edinburgh, Springer, 1995, 241-250

[43] E.K. Burke, J.P. Newall, R.F. Weare: Initialisation strategies and diversity in evolutionary timetabling, Evolutionary Computation Journal (Special Issue on Scheduling), Vol. 6, No. 1, 1998, 81-103

[44] E.K. Burke, A.J. Smith: Hybrid evolutionary techniques for the maintenance scheduling problem, IEEE Transactions on Power Systems, Vol. 15, No.1, 2000, 122-128

[45] R. Burns, M. Carter: Work force size and single shift schedules with variable demands, Management Science, Vol. 31, 1985, 599-607

[46] J.L. Byrne, R.B. Potts: Scheduling of toll collectors, Transportation Science 7, 1973, 224-245

[47] A. Caprara, F. Focacci, E. Lamma, P. Mello: Integrating constraint logic programming and operations research techniques in the crew rostering problem, Software Practice & Experience, Vol. 28, 1998, 49-76

[48] M.P. Carrasco, M.V. Pato: A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 3-17

[49] M.W. Carter: A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 64-82

[50] M.W. Carter, G. Laporte: Recent Developments in Practical Examination Timetabling, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, 1995, 3-21

[51] M.W. Carter, G. Laporte: Recent Developments in Practical Course Timetabling, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling, Second International Conference, Toronto, Springer, 1997, 3-19

[52] L.J. Cavaiola, J.P. Young: An integrated system for patient assessment and classification and nurse staff allocation for long term care facilities, Health Services Research 15, 1980, 281-306

[53] A. Cercueil, O., François: Monte Carlo Simulation and Population-Based Optimization, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 191-198

[54] V. Černý: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, Journal of Optimization Theory and Applications 45, 1985, 41-51

[55] P. Chan, G. Weil: Cyclical Staff Scheduling Using Constraint Logic Programming, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 159-175

[56] J.-G. Chen, T. Yeung: Hybrid Expert System Approach to Nurse Scheduling, Computers in Nursing, 1993, 183-192

[57] R.W. Cheng, M. Gen: Parallel machine scheduling problems using Memetic Algorithms, Computers & Industrial Engineering, 33: No. 3-4, 1997, 761-764

[58] M. Chiarandini, A. Schaerf, F. Tiozzo: 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, ISBN 3-00-003866-3, 2000, 298-302

[59] J. Collins, E. Sisley: Automated Assignment and Scheduling of Service Personnel, IEEE Expert, 5, 1994, 33-38

[60] A. Colorni, M. Dorigo, V. Maniezzo: Genetic Algorithms: A New Approach to the Time-Table Problem, NATO ASI Series, Vol. F 82, Springer, 1990, 235-239

[61] P. Coppens: Geautomatiseerde personeelsplanning bij het A.Z. Sint-Erasmus. GET info 9, 1995, 1-3

[62] D. Corne, J. Ogden, Evolutionary Optimisation of Methodist Preaching Timetables, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling, Second International Conference, Toronto, Springer, 1997, 142-155

[63] D. Costa: On the use of some known methods for T-colorings of graphs, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1993, 343-358

[64] P. Cowling, G. Kendall, E. Soubeiga: A Hyperheuristic Approach to Scheduling a Sales Summit,E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 176-190

[65] T.G. Crainic, M. Gendreau, P. Hansen, N. Hoeb, N. Mladenović: Parallel Variable Neighbourhood Search for the p-Median, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 595-599

[66] H.A.J. Crauwels, C.N. Potts, L.N. Van Wassenhove: A branch and bound algorithm for single machine scheduling with batching to minimize the number of late jobs, European Journal of Operational Research, Vol. 90, 1996, 200-213

[67] J. Crispim, J. Brandão: Reactive Tabu Search and Variable Neighbourhood Descent Applied to the Vehicle Routing Problem with Backhauls, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 631-636

[68] G.B. Dantzig: A Comment on Edie's Traffic Delays at Toll Booths, Operations Research, Vol.2, No.3, 339-341, 1954

[69] S.J. Darmoni, A. Fajner, N. Mahe, A. Leforestier, O. Stelian, M. Vondracek, M. Baldenweck: Horoplan: computer-assisted nurse scheduling using constraint based programming, Journal of the Society for Health Systems, Vol.5, 41-54, 1995

[70] P.J. Darwen, X. Yao: Why More Choices Cause Less Co-operation in Iterated Prisoner's Dilemma, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 987-994

[71] C. Darwin: The Origin of Species, 1859

[72] T. Davidović, P. Hansen, N. Mladenović: Variable Neighborhood Search for Multiprocessor Scheduling Problem with Communication Delays, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 737-741

[73] R. Dawkins: The Selfish Gene, Oxford University Press, 1976

[74] P. De Causmaecker, P. Demeester, P. De Pauw-Waterschoot, G. Vanden Berghe: Agents in a Route Planning Application, JORBEL Special Issue Orbel-14: Emerging Challenges in Operations Research, Vol. 40 (1-2), 2000, 105-116

[75] P. De Causmaecker, P. Demeester, Y. Lu, G. Vanden Berghe: Agent Technology for Timetabling, working paper at KaHo Sint-Lieven, 2002

[76] P. De Causmaecker, P. Demeester, Y. Lu, G. Vanden Berghe: Using Web Standards for Timetabling, accepted for publication in Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, Gent, 2002

[77] P. De Causmaecker, G. Vanden Berghe: Artificiële intelligentie voor personeelsplanning in de gezondheidszorg, NVKVV 4de Colloquium automatisering en zorgverlening, 1998

[78] P. De Causmaecker, G. Vanden Berghe: Relaxation of Coverage Constraints in Hospital Personnel Rostering, accepted for publication in Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, Gent, 2002

[79] M. den Besten, T. Stützle: Neighborhoods Revisited: An Experimental Investigation into the Effectiveness of Variable Neighborhood Descent for Scheduling, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 545-549

[80] G. de Vries: Nursing workload measurement as management information, European Journal of Operational Research 29, 1987, 199-208

[81] L. Di Gaspero, A. Schaerf: Tabu Search Techniques for Examination Timetabling, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 104-117

[82] D. Dowling, M. Krishnamoorthy, H. Mackenzie, D. Sier: Staff rostering at a large international airport, Annals of Operations Research, Vol. 72, 1997, 125-147

[83] K. Dowsland: Genetic Algorithms - a Tool for OR? Operations Research 47, 1996, 550-561

[84] K. Dowsland: Nurse scheduling with Tabu Search and Strategic Oscillation. European Journal of Operations Research 106, 1998, 393-407

[85] F. Easton, N. Mansour: A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems, Conference on Genetic Algorithms, San Mateo, 1993, 360-367

[86] F. Easton, D. Rossin, W. Borders: Analysis of Alternative Scheduling Policies for Hospital Nurses, Production and Operations Management, Vol.1, No. 2, 1992, 159-174

[87] F. Easton, D. Rossin: Sufficient working subsets for the tour scheduling problem, Management Science 37, 1991, 1441-1451

[88] A.E. Eiben, B. Jansen, Z. Michalewicz, B. and Paechter: Solving CSPs with evolutionary algorithms using self-adaptive constraint weights, Genetic and Evolutionary Computation Conference, GECCO, 2000

[89] M.A.S. Elmohamed, P. Coddington, G. Fox: A Comparison of Annealing Techniques for Academic Course Scheduling, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling, Second International Conference, Toronto, Springer, 1997, 92-112

[90] H. Emmons, D. Fuh: Sizing and scheduling a full-time and part-time workforce with off-day and off-weekend constraints, Annals of Operations Research, Vol. 70, 1997, 473-492

[91] W. Erben: A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 132-156

[92] J.A. Ferland: Generalised Assignment-Type Problems: A Powerful Modeling Scheme, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling, Second International Conference, Toronto, Springer, 1997, 53-77

[93] P. Festa, P.M. Pardalos, M.G.C. Resende, C.C. Ribeiro: GRASP and VNS for Max-Cut, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 371-376

[94] K. Fischer, J.P. Müller, M. Pischel: Cooperative Transportation Scheduling: An Application Domain for DAI, Journal of Applied Artificial Intelligence, Special Issue on Intelligent Agents, Vol. 10, No. 1, 1996, 1-33

[95] D. Fogel: Evolutionary Computation: The Fossil Record, IEEE Press, 1998

[96] L.S. Franz, H.M. Baker, G.K. Leong, T.R. Rakes: Mathematical model for scheduling and staffing multiclinic health regions, European Journal of Operational Research, Vol. 41, No. 3, 1989, 277-289

[97] A.S. Fraser: Simulation of genetic systems by automatic digital computers. II. Effects of linkage on rates under selection, Australian Journal of Biol. Sci. Vol. 10, 1957, 492-499

[98] B. Fries: Bibliography of Operations Research in Health-Care Systems, Operations Research 24, 1976, 801-804

[99] E. Fuller: Tackling Scheduling Problems Using Integer Programming, Master Thesis, University of Wales Swansea, 1998

[100] J. Garcia, H.J. Besada Portas, J.M. Molina Lopez, G. de Miguel, J. Portillo Garcia: Design of Interacting Multiple Model Filters based on Evolution Strategies, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 449-456

[101] S.S. Ge, T.H. Lee, F. Hong: Robust Controller Design with Genetic Algorithm for Flexible Spacecraft Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 1033-1039

[102] M. Gendreau, A. Hertz, G. Laporte: A Tabu Search Heuristic for the Vehicle Routing Problem, Management Science, Vol. 40, No. 10, 1994, 1276-1290

[103] F. Glover: Future Paths for Integer Programming and Links to Artificial Intelligence, Computers and Operations Research, Vol. 13, 1986, 533-549

[104] F. Glover, M. Laguna: Tabu Search, Kluwer Academic Publishers, ISBN 0-7923-8187-4, 1997

[105] F. Glover, C. McMillan: The General Employee Scheduling Problem: An Integration of Management Science and Artificial Intelligence, Computers and Operations Research, Vol. 13, No. 5, 1986, 563-593

[106] F. Glover, E. Taillard, D. De Werra: A user's guide to tabu search, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1993, 3-28

[107] D.E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989

[108] C.G. Gonzáles, D. Pérez-Brito: A Variable Neighbourhood Search for Solving the Linear Ordering Problem, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 181-185

[109] R. Gopalan, T. Talluri: Mathematical Models in Airline Schedule Planning: A Survey, Annals of Operations Research, Vol. 76, 1998, 155-185

[110] L.G. Graff, M.J. Radford, Formula for Emergency Physician Staffing, American Journal of emergency medicine, Vol. 8, No. 3, 194-199

[111] S.-J. Han, S.-Y. Oh: Evolutionary Algorithm Based Neural Network Controller Optimization for Autonomous Mobile Robot Navigation, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 121-127

[112] P. Hansen, B. Jaumard: Algorithms for the maximum satisfyability problem, Computing 44, 1990, 279-303

[113] P. Hansen, N. Mladenović: An introduction to variable neighborhood search, S. Voss et al. (Eds.): Advances and Trends in Local Search Paradighms for Optimization, Kluwer Academic Publishers, Boston, MA, 1999, 433-358

[114] R. Heckmann, T. Lengauer: A Simulated Annealing Approach to the Nesting Problem in the Textile Manufacturing Industry, Annals of Operations Research, Vol. 57, 1995, 103-133

[115] W.B. Henderson, W.L. Berry: Heuristic Methods for telephone operator shift scheduling: An experimental analysis, Management Science 22, 1976, 1372-1380

[116] A. Hertz: Tabu Search for Large Scale Timetabling Problems, European Journal of Operational Research, 54, 1991, 39-47

[117] A. Hertz, E. Taillard, D. de Werra: Tabu search, E. Aarts, J.K. Lenstra (Eds.): Local Search in Combinatorial Optimization, Wiley, ISBN 0-471-94822-5, 1997, 121-136

[118] J.H. Holland: Adaptation in Natural and Artifical Systems, University of Michigan Press, Ann Arbor, 1975

[119] R. Hung: A Cyclical Schedule of 10-Hour, Four-Day Workweeks, Nursing Management, Vol 22, No 5, 1991, 30-33

[120] R. Hung: Hospital Nurse Scheduling, Journal of Nursing Administration, Volume 25, number 7/8, 1995, 21-23

[121] M. Isken, W. Hancock: A Heuristic Approach to Nurse Scheduling in Hospital Units with Non-Stationary, Urgent Demand, and a Fixed Staff Size, Journal of the Society for Health Systems, Vol. 2, No. 2, 1990, 24-41

[122] A. Jaszkiewicz: A metaheuristic approach to multiple objective nurse scheduling, Foundations of Computing and Decision Sciences, Vol. 22, No. 3, 1997, 169-184

[123] B. Jaumard, F. Semet, T. Vovor: A generalized linear programming model for nurse scheduling, European Journal of Operational Research, Vol. 107, 1998, 1-18

[124] N. Kaise, Y. Fujimoto: Applying the Evolutionary Neural Networks with Genetic Algorithms to Control a Rolling Inverted Pendulum, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 223-230

[125] T. Kampe, Simulated Annealing: Use of a New Tool in Bin Packing, Annals of Operations Research, Vol. 16, 1998, 327-332

[126] H. Kawanaka, K. Yamamoto, T. Yoshikawa, T. Shinogi, S. Tsuruoka: Genetic Algorithm with the Constraints for Nurse Scheduling Problem, Proceedings of Congress on Evolutionary Computation, Seoul, IEEE Press, 2001, 1123-1130

[127] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi: Optimization by Simulated Annealing, Science 220, 1983, 671-680

[128] M. Kostreva, K. Jennings: Nurse Scheduling on a Microcomputer, Computers and Operations Research 18, 1991, 731-739

[129] S.S. Lam: A Genetic Fuzzy Expert System for Stock Market Timing, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 410-417

[130] K.-H. Liang, X. Yao, C.S. Newton: Dynamic Control of Adaptive Parameters in Evolutionary Programming, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 42-49

[131] J.S. Liebman, J.P. Young, M. Bellmore: Allocation of Nursing Personnel in an extended care facility, Health Services Research 7, 1972, 209-220

[132] Y. Liu, X. Yao: Evolving Neural Networks for Hand Seng Stock Index Forecast, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 156-260

[133] V. Lotfi, R. Cerveny: A Final Exam-Scheduling Package, Journal of Operations Research, Vol. 42, 1991, 205-216

[134] J.P. Lyons, J.P. Young: A staff allocation model for mental health facilities, Health Services Research 11, 1976, 53-68

[135] C. Maier-Rothe, H.B. Wolfe: Cyclical Scheduling and Allocation of Nursing Staff, Socio-Economic Planning Science, Vol. 7, 1973, 471-487

[136] R. Marett, M. Wright: A Comparison of neighbourhood search strategies for multi-objective combinatorial problems, Computers and Operations Research, Vol. 23, No.5, 1996, 465-483

[137] J.D. Megeath: Successful hospital personnel scheduling, Interfaces, Vol. 8, No. 2, 1978, 55-60

[138] A. Meisels, E. Gudes, G. Solotorevski: Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference Edinburgh, Springer, 1995, 93-105

[139] A. Meisels, E. Gudes, G. Solotorevski: Combining rules and constraints for employee timetabling, Journal of Intelligent Systems, Vol. 12, 1997, 419-439

[140] A. Meisels, N. Lusternik: Experiments in Networks of Employee Timetabling Problems, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling II, Second International Conference Toronto, Springer, 1997, 130-141

[141] A. Meisels, A. Schaerf: Modelling and Solving Employee Timetabling Problems, Applied Intelligence, accepted for publication, (2001)

[142] H. Meyer auf'm Hofe: 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, 1997, 257-271

[143] H. Meyer auf'm Hofe: Finding regions for local repair in partial constraint satisfaction, Advances of Artificial Intelligence, No 1504 in LNAI, 1998, Springer-Verlag, 57-68

[144] H. Meyer auf'm Hofe: Solving Rostering Tasks as Constraint Optimization, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 191-212

[145] H.H. Millar, M. Kiragu: Cyclic and non-cylcic scheduling of 12h shift nurses by network programming, European Journal of Operational Research, Vol. 104, 1998, 582-592

[146] H.E. Miller: Implementing Self Scheduling, The Journal of Nursing Administration, Vol. 14, March 1984, 33-36

[147] H.E. Miller, W. Pierskalla, G. Rath: Nurse Scheduling Using Mathematical Programming. Operations Research 24, 1976, 857-870

[148] N. Mladenović, P. Hansen: Variable Neighborhood Search, Computers & Operations Research, Vol. 24, 1997, 1097-1100

[149] H. Myung, J.-H. Kim: Multiple Lagrange Multiplier Method for Constrained Evolutionary Optimization, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 2-9

[150] N. Mladenović, D. Urošević:

Variable Neighbourhood Search for the k-Cardinality Tree, MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 743-747

[151] E. Morgado, J. Martins: An AI based approach to crew scheduling, Proceedings of CAIA-93, 1993, 71-77

[152] P. Moscato, M.G. Norman: A 'Memetic' Approach for the Traveling Salesman Problem - Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems, M. Valero, E. Onate, M. Jane, J.L. Larriba, and B. Suarez (Eds.): Parallel Computing and Transputer Applications, IOS Press, Amsterdam, 1992, 187-194

[153] A. Musa, U. Saxena: Scheduling Nurses Using Goal-Programming Techniques, IEEE, 1984, 216-221

[154] N. Muslija, J. Gaertner, W. Slany: Efficient Generation of Rotating Workforce Schedules, E.K. Burke, W. Erben (Eds.): Proceedings of the 3rd international conference on the Practice and Theory of Automated Timetabling, ISBN 3-00-003866-3, 2000, 314-332

[155] N. Musliu, A. Schaerf, W. Slany: Local Search for Shift Design, MIC'2001, Proceedings of the 4th Metaheuristics International Conference, Porto, 2001, 465-469

[156] R. Nanda, J. Browner: Introduction to Employee Scheduling, Van Nostrand Reinhold, New York, 1992

[157] R. Narasimhan: An algorithm for single shift scheduling of hierarchical workforce, European Journal of Operational Research, Vol. 96, 1996, 113-121

[158] M. Okada: An approach to the Generalised Nurse Scheduling Problem - Generation of a Declarative Program to represent Institution-Specific Knowledge. Computers and Biomedical Research 25, 1992, 417-434

[159] M. Okada, M. Okada: Prolog-Based System for Nursing Staff Scheduling Implemented on a Personal Computer. Computers and Biomedical Research 21, 1988, 53-63

[160] J.H. Oldenkamp: Investigating Reasoning in Maternity Care Scheduling, Knowledge and Policy, Vol. 5, No. 3, 1992, 67-76

[161] I. Ozkarahan: A Flexible Nurse Scheduling Support System, Arizona State University, Tempe, Arizona, (unpublished doctoral dissertation), 1987

[162] I. Ozkarahan: A Disaggregation Model of a Flexible Nurse Scheduling Support System, Socio-Economical Planning Science, Vol. 25, No. 1, 1991, 9-26

[163] I. Ozkarahan: An Integrated Nurse Scheduling Model, Journal of the Society for Health Systems, Vol. 3, No. 2, 1991, 79-101

[164] I. Ozkarahan: Allocation of surgical procedures to operating rooms, Journal of Medical Systems, Vol. 19, No. 4, 1995, 333-352

[165] I. Ozkarahan: Allocation of surgeries to operating rooms by goal programming, Journal of Medical Systems, Vol. 24, No. 6, 2000, 339-378

[166] I. Ozkarahan, J.E. Bailey: Goal Programming Model Subsystem of a Flexible Nurse Scheduling Support System, IEEE Transactions, Vol. 16, 1988, 306-316

[167] B. Paechter, A. Cumming, H. Luchian: The use of local seach suggestion lists for improving the solution of timetabling problems with evolutionary algorithms, T. Fogerty (Ed.): Evolutionary Computation, Springer Lecture Notes in Computer Science, Vol. 993, 1995, 86-93

[168] B. Paechter, A. Cumming, M.G. Norman, H. Luchian: Extensions to a Memetic Timetabling System, E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, 1995,

251-265

[169] B. Paechter, R.C. Rankin, A. Cumming: Improving a Lecture Timetabling System for University-Wide Use, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling II, Second International Conference, Toronto, Springer, 1997, 156-165

[170] B. Paechter, R.C. Rankin, A. Cumming, T.C. Fogarty: Timetabling the Classes of an Entire University with an Evolutionary Algorithm, Parallel Problem Solving from Nature (PPSN) V, Springer, LNCS 1498, 1998

[171] S. Petrovic, R. Petrovic: Eco-EcoDispatch: DSS for multicriteria loading of thermal power generators, Journal of Decision Systems 4 (4), 1995, 279-295

[172] C. Pimpawat, N. Chaiyaratana: Using a Co-Operative Co-Evolutionary Genetic Algorithm to Solve a Three-Dimensional Container Loading Problem, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 1197-1204

[173] A.P. Punnen, Y.P. Aneja: A tabu search Algorithm for the Resource-Constrained Assignment Problem. Journal of the Operational Research Society 44, 1995, 214-220

[174] N.J. Radcliffe, P.D. Surry: Formal Memetic Algorithms, T.C. Fogarty (Ed.): Evolutionary Computing, Lecture Notes in Computer Science 865, Springer, 1994, 250-263

[175] S. Randhawa, D. Sitompul: A Heuristic Based Computerised Nurse Scheduling System, Computers and Operational Research 20, 1993, 837-844

[176] C. Reeves: Genetic Algorithms for the Operations Researcher, INFORMS, Vol.9, No. 3, 1997, 231-250

[177] P.G.K., Reiser, P.J. Riddle: Evolving Logic Programs to Classify Chess-Endgame Positions, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 138-145

[178] K.K. Ringl, L. Dotson: Self-Scheduling for Professional Nurses, Nursing Management, Vol. 20, 1989, 42-44

[179] E.S. Rosenbloom, N.F. Goertzen: Cyclic Nurse Scheduling, European Journal of Operational Research 31, 1989, 19-23

[180] P. Ross, E. Hart, D. Corne: Some Observations about GA-based Exam Timetabling, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling, Second International Conference, Toronto, Springer, 1997, 115-129

[181] A. Schaerf: A Survey of Automated Timetabling, Artificial Intelligence Review, 13(2), 1999, 87-127

[182] A. Schaerf, A. Meisels: Solving employee timetabling problems by generalised local search, Proceedings Italian AI, 1999, 493-502

[183] D. Schneider, K. Kilpatrick: An Optimum Manpower Utilization Model for Health Maintenace Organisations, Operations Research, Vol. 23, No 5, 1975, 869-889

[184] J.A.M. Schreuder: Assigning magistrates to sessions of the Amsterdam Criminal Court (abstract), E.K. Burke, W. Erben (Eds.): Proceedings of the 3rd international conference on the Practice and Theory of Automated Timetabling, ISBN 3-00-003866-3, 2000, 333-334

[185] J.A.M. Schreuder: Personnel Timetabling based on OR Models: two recent case studies (abstract), E.K. Burke, W. Erben (Eds.): Proceedings of the 3rd international conference on the Practice and Theory of Automated Timetabling, ISBN 3-00-003866-3, 2000, 259-260

[186] J.A.M. Schreuder, J.A. van de Velde: Timetables in Dutch High Schools, J.P. Brans (ed): Operational Research '84, Elsevier Science Publishers, 1984, 601-612

[187] H.-P. Schwefel: Numerical Optimization of Computer Models, Wiley, 1981

[188] G.P. Scott, D.I. Clark, T. Pham: A Genetic Clustering Algorithm Guided by a Descent Algorithm, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 734-740

[189] M. Segal: The operator scheduling problem: A network-flow approach, Operations Research 22, 1974, 808-823

[190] F. Semet, E. Taillard: Solving real-life vehicle routing problems efficiently using tabu search, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research 41, 1993, 469-488

[191] M.J. Showalter, V.A. Mabert: An evoluation of a full- /part-time tour scheduling methodology, International Journal of Operations and Production Management 8, 1998, 54-71

[192] A. Sierra: Evolution of Fisher's discriminants, Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 2001, 747-752

[193] S.P. Siferd, W.C. Benton: Workforce staffing and scheduling: Hospital nursing specific models, European Journal of Operational Research 60, 1992, 233-246

[194] D. Sitompul, S. Randhawa: Nurse Scheduling Models: A State-of-the-Art Review, Journal of the Society of Health Systems 2, 1990, 62-72

[195] L.D. Smith: The application of an Interactive Algorithm to Develop Cyclical Rotational Schedules for Nursing Personnel, INFOR, Vol. 14, February, 1976, 53-70

[196] L.D. Smith, D. Bird, A. Wiggins: A Computerised System to Schedule Nurses that Recognises Staff Preferences, Hospital & Health Service Administration, 1979, 19-35

[197] L.D. Smith, A. Wiggins: A Computer-Based Nurse Scheduling System, Computers and Operations Research, Vol. 4, No. 3, 1977, 195-212

[198] V. Smith-Daniels, S. Schweikhart, D. Smith-Daniels: Capacity Management in Health Care Services: Review and Future Research Directions, Decision Sciences, Vol.19, 1988, 891-919

[199] D. Sosnowska, J. Rolim: Fleet Scheduling Optimization: A Simulated Annealing Approach, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 227-241

[200] B. Srivastava, W.-H. Chen: PArt type selection problem in flexible manufacturing systems: Tabu search algorithms, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 1993, 279-297

[201] J. Tanomaru: Staff Scheduling by a Genetic Algorithm with Heuristic Operators, Proceedings of the IEEE Conference on Evolutionary Computation, 1995, 456-461

[202] P.E. Taylor, S.J. Huxley: A break from tradition for the San Fransisco Police: Patrol officer scheduling using an optimization-based decision support system Interfaces 19, 1989, 4-24

[203] H.M.M. ten Eikelder, R.J. Willemen: Some Complexity Aspects of Secondary School Timetabling Problems, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 18-27

[204] J. Tien, A. Kamiyama: On Manpower Scheduling Algorithms, Society for Industrial and Applied Mathematics, 1982, 275-287

[205] G.M. Thompson: A Simulated-Annealing heuristic for shift-scheduling using non-continuously available employees, Computers and Operations Research 13, 1996, 275-288

[206] J. Thompson, K.A. Dowsland: Multi-Objective University Examination Scheduling, EBMS/1993/12, European Business Management School, Swansea University, UK

[207] J.M. Thompson, K.A. Dowsland: Variants of Simulated-Annealing for the examination timetabling problem, Annals of Operations Research 63, 1996, 105-128

[208] V.M. Trivedi, M. Warner: A branch and bound algorithm for optimum allocation of float nurses, Management Science vol. 22, No 9, 1976, 972-981

[209] E. Tsang, C. Voudouris: Fast local search and guided local search and their application to British Telecom's workforce scheduling problem, Operations Research Letters, Vol. 20, No. 3, 1997, 119-127

[210] Y. Tsjujimura, M. Gen: Evolutionary Recognition of Features from CAD Data, B. McKay et al. (Eds.): Simulated Evolution and Learning, 1998, Lecture Notes in Artificial Intelligence, Vol. 1585, Springer, 1999, 276-284

[211] H. Ueda, D. Ouchi, K. Takahashi, T. Miyahara: A Co-evolving Timeslot/Room Assignment Genetic Algorithm Technique for University Timetabling, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 48-63

[212] S. Vanderhaeghe: Dienstroosters in de gezondheidszorg: ongezond? Informatiedossier, Stichting Technologie Vlaanderen, 2de oplage, mei 1998

[213] A. van Weert, J.A.M. Schreuder: Construction of Basic Math Schedules (BMS) for Sports Competition by Using Graph Theory, E.K. Burke, M. Carter (Eds.): Practice and Theory of Automated Timetabling II, Second International Conference Toronto, Springer, 1997, 201-210

[214] M. Warner: Nurse Staffing, Scheduling, and Reallocation in the Hospital, Hospital & Health Services Administration, 1976, 77-90

[215] M. Warner: Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach. Operations Research 24, 1976, 842-856

[216] M. Warner, J. Prawda: A Mathematical Programming Model for Scheduling Nursing Personnel in a Hospital, Management Science 19, 1972, 411-422

[217] D. Wedelin: An algorithm for a large scale 0-1 integer programming with application to airline crew scheduling, Annals of Operations Research, Vol. 57, 1995, 283-301

[218] G. Weil, K. Heus, P. Francois et al.: Constraint Programming for Nurse Scheduling, IEEE Engineering in Medicine and Biology, 1995, 417-422

[219] G.M. White, B.S. Xie: Examination Timetables and Tabu Search with Longer-Term Memory, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 85-103

[220] R.J. Willis, S.B. Huxford: Staffing rosters with breaks - A case study, Journal of the Operational Research Society 42, 1991, 727-731

[221] H. Wolfe, J.P. Young: Staffing the Nursing Unit: Part I, Nursing Res. 14 (3), 1965, 236-234

[222] H. Wolfe, J.P. Young, J.P.: Staffing the Nursing Unit: Part II, Nursing Res. 14 (4), 1965, 299-303

[223] D.L. Woodruff, E. Zemel: Hashing vectors for tabu search, F. Glover, M. Laguna, E. Taillard, D. De Werra (Eds): Annals of Operations Research, Vol. 41, 123-137

[224] A. Wren: Scheduling, Timetabling, and Rostering - A Special Relationship? E.K. Burke, P. Ross (Eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, Springer, 1995, 46-75

[225] A. Wren, J.-M. Rousseau: Bus Driver Scheduling - An Overview, J.R. Daduna, I. Branco, I., J.M.P. Paixao (Eds.): Computer Aided Transit Scheduling, Springer Verlag, 1995, 173-187

[226] M. Zeleny: A Concept of Compromise Solutions and the Method of the Displaced Ideal, Computers and Operations Research, Vol. 1, 1974, 479-496

[227] K. Zervoudakis, P. Stamatopoulos: A Generic Object-Orienteed Constraint-Based Model for University Course Timetabling, E.K. Burke, W. Erben (Eds.): Practice and Theory of Automated Timetabling, Third International Conference, Konstanz, Springer, 2000, 28-47