Examensarbete

Rotating Workforce Scheduling

Caroline Granfeldt

LiTH - MAT - EX - - 2015 / 08 - - SE

Rotating Workforce Scheduling

Optimeringslära, Linköpings Universitet

Caroline Granfeldt

LiTH - MAT - EX - - 2015 / 08 - - SE

Examensarbete: **30 hp**

Level: **A**

Supervisor: **Torbjörn Larsson**,
Optimeringslära, Linköpings Universitet

Supervisor: **Ann Bertilsson**,
Schemagi AB

Examiner: **Elina Rönnberg**,
Optimeringslära, Linköpings Universitet

Linköping: **November 2015**

Abstract

Several industries use what is called rotating workforce scheduling. This often means that employees are needed around the clock seven days a week, and that they have a schedule which repeats itself after some weeks. This thesis gives an introduction to this kind of scheduling and presents a review of previous work done in the field. Two different optimization models for rotating workforce scheduling are formulated and compared, and some examples are created to demonstrate how the addition of soft constraints to the models affects the scheduling outcome. Two large realistic cases, with constraints commonly used in many industries, are then presented. The schedules are in these cases analyzed in depth and evaluated. One of the models excelled as it provides good results within a short time limit and it appears to be a worthy candidate for rotating workforce scheduling.

Keywords: Optimization, Rotating schedules, Cyclical scheduling

URL for electronic version:

http://liu.diva-portal.org/smash/record.jsf?pid=diva2:867722

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Torbjörn Larsson, at Linköping University for all the support and pleasant conversations throughout the work on this thesis. Your mentorship is the foremost reason I am inspired to continue working in the field of optimization. I would also like to thank Elina Rönnberg and Nils-Hassan Quttineh at Linköping University for all your encouragement.

Furthermore, I would like to dedicate appreciation to my supervisor, Ann Bertilsson, at Schemagi AB for all your thoughtful and valuable input.

I thank my opponents, David Larsson and Rebecka Håkansson, for your support and constructive thoughts which greatly helped improve my work.

Lastly, I thank my wonderful fiancé Oscar, whose love and support I would not have made it this far without, and my friends and family for always encouraging me. You have all helped bring out the best in me, through good and bad times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis explores different approaches to a specific type of staff scheduling called *rotating workforce scheduling*. The definition of this type of schedules is discussed in Section 1.1, along with other terms used in this thesis.

1.1 Background

In many industries, it is required that work should be carried out 24 hours a day, 7 days a week. Typical work schedules in such contexts consist of a cycle repeating itself after a few weeks. The work is usually divided into different shifts, typically day (D), evening (E) and night (N) shifts. Figure 1.2 below shows an example of how a *rotating workforce schedule* can look like, where the empty spaces indicate the rest periods. The schedule repeats itself every six weeks. It should be mentioned right away that in every model and example in this thesis, the occurrence of more than one shift per day in each weekly schedule is prohibited.

Person	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
A			D	D	D	D	D	D	D					E	E	N	N	N	N			E	E	E	E	E			D	D	D	D	D			D	D	D	D	D	D	D
B	D	D				E	E	N	N	N	N				E	E	E	E	E			D	D	D	D	D			D	D	D	D	D	D	D			D	D	D	D	D
C	N	N	N	N				E	E	E	E	E			D	D	D	D	D			D	D	D	D	D	D	D			D	D	D	D	D	D	D			E	E
D	E	E	E	E	E			D	D	D	D	D			D	D	D	D	D	D	D			D	D	D	D	D	D	D			E	E	N	N	N	N		
E	D	D	D	D	D			D	D	D	D	D	D	D			D	D	D	D	D	D	D			E	E	N	N	N	N			E	E	E	E	E		
F	D	D	D	D	D	D	D			D	D	D	D	D	D	D				E	E	N	N	N	N			E	E	E	E	E			D	D	D	D	D		

Figure 1.1: *Example of a schedule with day, evening and night shifts.*

The observant reader notices that all persons in the above schedule are on the same six-week schedule, with the only difference being an offset in the starting week. Hence, the schedule can be represented compactly by only showing one person's schedule. In the schedule shown in Figure 1.2, person A begins the cycle on week 1, while person B begins the cycle on week 2, person C begins the schedule on week 3, and so on. At the end of each week, each person moves down to the following week and the last person moves up to the first week. Hence, Figures 1.1 and 1.2 illustrate the exact same schedule.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1			D	D	D	D	D
2	D	D				E	E
3	N	N	N	N			
4	E	E	E	E	E		
5	D	D	D	D	D		
6	D	D	D	D	D	D	D

Figure 1.2: *Example of a schedule with day, evening and night shifts.*

To be able to construct rotating workforce schedules, a workload matrix is required. Simply put, the matrix shows the staffing demand on each specific shift every week. Table 1.1 shows an example of a workload matrix, which matches the schedule given in Figure 1.2.

Table 1.1: *Example of a workload matrix.*

Shift	Mo	Tu	We	Th	Fr	Sa	Su
N	1	1	1	1			
D	3	3	3	3	3	2	2
E	1	1	1	1	1	1	1

The process of constructing the above described type of schedules is, not surprisingly, called *rotating workforce scheduling*. In general, staff scheduling is a very complex problem. A difficulty with rotating workforce scheduling is that various constraints make it difficult to find an optimal solution; health and safety regulations need to be taken into consideration. For example, it is not allowed to work a day shift directly after a night shift, while working too many night shifts in a row is tiresome and thus causes a proneness to more accidents. Additionally, most work places and their employees have specific requests, which are unique compared to other employers. This could for example be that every employee should at some time during the cycle work with every other employee, or that a small group of employees should work more frequently together.

An advantage of these types of schedules is that they are fair. The scheduling process can be problematic since it is difficult to design schedules that contains the same properties, but are still different and adapted to individual preferences. With rotating workforce schedules, this issue is non-existent since everyone is on the same schedule.

Another property is that it is the staffing demand together with the design of shifts that decides the number of scheduling weeks. For example, the workload matrix in Table 1.1 contains 30 shifts, and each shift is in this case 8 hours long. A person works on average 40 hours per week, and thus 5 shifts. Hence, $30/5 = 6$ persons are needed to cover the staffing demand, which leads to 6 scheduling weeks. In another example, every shift might be 10 hours long and the workload matrix requires 28 shifts. On average, a person works 4 shifts every week. In total, this requires $28/4 = 7$ persons and thus 7 scheduling weeks.

1.2 Aims and Goals

In short, this thesis explores the possibilities of how to perform rotating workforce scheduling, taking into consideration different demands from the customer as well as constraints following from laws and regulations. More specifically, the main objectives of this thesis can be divided into three different parts:

1. Research what has previously been done in this field.
2. Construct basic optimization models: an integer model and a network model.
3. Extend the models and see how they perform on two larger cases.

1.3 Method

To begin with, a good literature base is researched and studied. The literature below was used as starting point:

- G. Laporte. The Art and Science of Designing Rotating Schedules. *The Journal of the Operational Research Society*, 50:1011–1017, 1999
- N. Musliu. Heuristic Methods for Automatic Rotating Workforce Scheduling. *International Journal of Computational Intelligence Research*, 2:309–326, 2006
- M. Rocha, J.F. Oliveira and M.A. Carravilla. Cyclical Staff Scheduling: Optimization Models for Some Real-Life Problems. *Journal of Scheduling*, 16:231–242, 2013

Chapter 2 compiles all the information found in the literature. Two basic mathematical models were developed, both found in Chapter 3. These models are written as integer linear programs and implemented in the mathematical programming language AMPL, which allows the optimization models to be expressed in a simple way. To solve the optimization problems written in AMPL, the solver CPLEX is used. The interested reader can find more information about AMPL and CPLEX in [4]. A small test case was constructed by hand, which makes it possible to see if the models perform adequately. Unfortunately, only one of the models is useful in practice with the current software, something which is further explained at the end of Section 3.2. Two larger cases, based on constraints that are fairly common in real-life, were later produced, which helped to develop and push the boundaries of one of the models further. These cases, along with the resulting schedules given by the model, are discussed in Chapters 5 and 6, respectively.

1.4 Topics Covered

This thesis consists of seven chapters, including this introduction. A summary of previous work done in this field is found in Chapter 2. Chapter 3 contains the two basic models explored in this thesis. The following chapter, Chapter 4, gives a few examples of how different constraints - when added to the basic model - affect the outcome. Case 1 is covered in Chapter 5. More specifically, the constraints used in that case and the different resulting schedules are showed. Chapter 6 is like the previous chapter, but instead looks at Case 2. Last but

not least, Chapter 7 contains a discussion of the results and also suggests ideas for future work in this field.

Chapter 2

Previous Work

The scheduling process in general is an issue that has been studied in several contexts. If the reader wishes to elaborate further on the subject, an extensive bibliography has been written by Ernst et al. [5] in 2004.

Work on rotating workforce scheduling has been done before, with many different approaches. This chapter discusses a few of them, by beginning to give the most common constraints appearing throughout the literature. It should be mentioned that early solvers, especially for integer problems which are common in scheduling applications, were not capable of handling large scale problems. Most likely, this is the reason why a lot of articles contain a special-purpose algorithms, instead of simply using a standard solver like CPLEX.

2.1 Models and Common Constraints

A couple of constraints seem to appear frequently in rotating workforce scheduling, regardless of context. Some of the most common ones, in no particular order, are given in Table 2.1. Several of these constraints depend on the setting and could thus of course be modified according to the situation at hand.

2.1.1 Integer Model

The most common way to model the rotating workforce scheduling problem, and probably the first idea to come to mind, is by an integer model. Although this is often used today, it was not the case in the earlier days. The digital revolution has greatly helped develop the scheduling field during the last decades. This section gives two examples of integer models.

Laporte et al. [7] suggested an algorithm in 1980 which uses an integer linear programming model. The algorithm begins by creating segments consisting of shifts followed by some days off which varies in length and start on different days of the week. With the help of an integer model, it is then possible to select a set of work segments that cover all shifts. The model can be used (recall that the paper is from 1980) because of the small lengths of segments, thus avoiding large scale optimization. The algorithm continues by merging the segments together using an enumeration process, resulting in a set of feasible schedules. The best schedule with regard to desirable properties not included in the integer model,

Table 2.1: *Typical constraints used in rotating workforce scheduling. The last column shows examples of articles which include the constraint.*

Nr	Constraint	Articles
1	Shift change, like going from a day shift to an evening shift, can only occur after at least one day off.	[1],[6],[7], [8]
2	The number of consecutive work days must not exceed a preset bound, typically 6 or 7, and must not be less than 2 or 3.	[1],[2],[6], [7],[8],[9], [10],[11], [12],[13]
3	The number of consecutive rest days must not exceed a preset bound, typically 6 or 7, and must not be less than 2 or 3.	[1],[6], [7], [8],[10]
4	Long work periods are normally followed by long rest periods and short work periods should be followed by short rest periods.	[1],[7]
5	Schedules should contain as many full weekends off as possible.	[1],[6],[7], [8]
6	Weekends off should be well spaced out in the cycle.	[1], [7], [8]
7	In some contexts, teams of employees must remain intact, that is, they cannot be divided and recombined differently.	[1]
8	Forward rotations (day, evening, night) should be used instead of backward rotations (day, night, evening).	[1],[14]
9	Staffing demand should always be satisfied. Alternatively, a maximum number of shortage or excess workers might be acceptable.	[1],[3],[6], [13],[15], [16]
10	Exactly one shift should be used each day.	[1],[15],[16]
11	Every week should contain 2 consecutive days off.	[15],[16], [17]

for example how weekends off are spread out, is then selected by the user. The algorithm was successful at its time and was used to create schedules for at least two police stations and one fire station.

In 2013, Rocha et al. [3] proposed a mixed integer problem to solve the rotating workforce scheduling problem. In a mixed integer problem, some of the variables are constrained to be integers while the others are allowed to be fractional. The paper begins by giving a general mathematical model, which the authors adapt to two different real-life case studies: a glass industry and a continuous care unit. In the glass industry the workforce is homogeneous concerning skills, with everyone working full-time. The continuous care unit is heterogeneous, with different demands for different shifts and both full-time as well as part-time workers. Results are presented for both adjusted models, which culminates in the conclusion that the models are able to produce optimal solution in an adequate amount of time.

2.1.2 Days-Off Scheduling

A common type of problems in earlier days of rotating workforce scheduling is the so called days-off scheduling problem, where we have a 5-day work week but a 7-day operating week. It is often also required that every employee is entitled to 2 consecutive days off each week. The problem consists of determining the off-work days for each worker, thus the name of the problem, while maintaining the minimum required workforce size.

During the middle of the seventies, Baker [15, 16] suggested methods for solving the cyclic days-off scheduling problem. In 1974, Baker [15] developed a simple algorithm for the objective to find a minimum staff size capable of meeting the requirements. The approach is heuristic and consists of two different steps. In the same year, Baker [16] tackles the problem of scheduling with both full-time and part-time staff. Just like in [15], he develops an algorithm though with the objective this time being to minimize the number of part-time employees. Unlike the procedure in this thesis, the approaches used by Baker in both papers allow finding the solutions by hand calculations. In 1977, Baker and Magazine [18] continued to work on the problem. In this paper, they examine a number of methods to solve different day-off policies. For each case, a formula for optimal workforce size together with an algorithm for constructing a feasible schedule is included. Other contributions in the area are given by Bennett and Potts [17] in 1968, Bechtold [19] in 1981 and by Alfares [20] in 1998.

The days-off scheduling problem is a little different from what is tried to achieve in this thesis. Its type of constraints is usually included in the rotating workforce scheduling problem. Hence, the days-off scheduling constraints can be viewied upon as a subset of the constraints found in the later problem.

2.1.3 Network Model

Another possible approach to the rotating workforce scheduling problem is to use a network flow model. It is usually difficult to incorporate all the constraints in the network, which makes it necessary to have side constraints. Hence, the resulting models are usually only network flow-based, as is the case in the paper by Balakrishnan and Wong [21] from 1990. Their model deals with issues like rest-period identification and work/rest period sequencing. All the constraints are included in the network itself, with the exception of the constraints describing the staffing demand which are thus treated as side constraints. An optimal solution corresponds to a path in the network. The uniqueness with their approach is that, unlike previous suggested algorithms during that time, it handled all requirements simultaneously. The common problems of infeasibility and non-optimality, which can arise in a sequential approach, are therefore avoided. For example, the work periods might be constructed in the first step while sequenced in the next, ergo a sequential approach.

Balakrishnan and Wong use nodes that correspond to the days within the planning horizon, where the source node represents the first day and the sink node represents the last day. Each arc in the network corresponds to a work period allocated to a specific shift, or alternatively to a rest period. Each path from the source node to the sink node will consequently represent a sequence of work and rest periods, thus a rotating workforce schedule. The shortest path, where the length of the path is modeled as the cost of that sequence, gives the

best resulting schedule with regard to the cost. The number of paths in the network will of course be very large for large scale problems, which made a complete enumeration non-viable at that time. The authors therefore decided to use a solution technique based on Lagrangian duality theory followed by a K-shortest path enumeration process.

In this thesis, although different from the model suggested by Balakrishnan and Wong, a network flow-based model is also constructed.

2.2 Solution Approaches

Before discussing different solution methods for the rotating workforce scheduling problem, a manual approach suggested by Laporte [1] in 1999 is looked upon. In his paper, Laporte argues that finding an optimal schedule can be more of an art than a science. He states that design by hand, where it is possible to relax certain constraints, can sometimes yield very reasonable schedules. Designing schedules with the constraints mentioned in Table 2.1 is often relatively easy and can be done by hand without difficulty. Difficulties arise however when, for example, the number of working hours per week must fall within an acceptable range. Still, by bending the rules a little, workable solutions can be found. The following are a few suggestions, given by Laporte in his paper, of how this can be done.

If we look at Constraint 1, it states that a shift change can only occur after at least one day off, while Constraint 2 states that breaks should be at least two days long. One day breaks are often avoided since they are usually not favoured by the staff. However, breaking the first rule might be acceptable if some other unappealing property is eliminated, for example a very long work stretch.

Another aspect is that of extending shifts to satisfy working hours per week. If the number of work hours per week is too low, it can be adjusted by extending some shifts so that they overlap each other. The problem with this adjustment might nonetheless be that it causes redundancies with too many workers on duty when the shifts overlap, but also adding extra strain on the workers these extended shifts.

If the number of workers is large, such that they work in teams, it is possible to introduce an occasional relief team. The relief team, consisting of fewer workers than an ordinary team, works the same schedule every week. This schedule coincide with the schedule of one of the weeks from the ordinary cyclical schedule, say the first week for simplicity. The regular team then take turns between themselves to take the first week off, while being replaced by the relief team. Although this breaks Constraint 7, it introduces the possibility of having an occasional week off.

Aside from the suggestions above, the article by Laporte also includes an example of manpower scheduling at Quebec Ministry of Transportation. In the presented case, a couple of other constraints besides those in Section 2.1 are added. Solving this scheduling case fairly easy, Laporte continued by arguing that scheduling by hand combined with the techniques suggested by him and described above gave a fast and satisfactory result, and that this is also the case in many other examples.

In this thesis, we do not schedule by hand but rather introduce models to solve the problem for us. However, the models include soft constraints, which

are constraints that are relaxed and penalized in the objective function (see the beginning of Chapter 3 for a more detailed definition). Thus, even though we do not schedule by hand, we can still "bend the rules" to obtain schedules with desired properties simply be using different values of penalty parameters.

Exhaustive Enumeration

Early on, simple exhaustive enumeration procedures were commonly used. An example of this is the paper by Smith [22] from 1976, which includes an algorithm that attempts to minimize the staffing demand subject to constraints regarding rest periods. A form of feedback loop is often used between steps to correct any infeasible or less desirable schedules, like for example adjusting the staffing demand and consequently increasing or decreasing the workforce size. This decomposition approach might seem good in theory, but the algorithms are rarely able to manage large scale problems even with today's computer technology.

Commercial Software

The rotating workforce scheduling problem is common in several industries. Thus, it is only natural that a need for commercial software to simplify the process has arisen. An example of this is the *First Class Scheduler*, developed by Gärtner et al. [23] in 2000. The purpose of this software is to find high-quality schedules that are at least as good as solutions found by professional planners, and to generate these schedules in a reasonably small amount of time. Furthermore, hard constraints should be easy to control by the user and also individual preferences should be taken into account. In 2002, Musliu et al. [9] extended the software by adding a four-step algorithm which focuses on splitting the scheduling process into smaller problems. In each of these problems, a human decision-maker is involved in order to incorporate preferences regarding soft constraints. Combined with problem-oriented intelligent backtracking algorithms, the method delivers good solutions for real-life problems.

Another software for rotating software scheduling is *CP-Rota*, developed by Triska and Musliu [12] in 2011. The software uses constraint programming, further explained in Section 2.2.1, to solve real-life problems from the industry. The purpose of the software is to complement previously mentioned *First Class Scheduler*.

2.2.1 Constraint programming

Constraint programming can be counted as both a model and a method. The technique originates from artificial intelligence, where certain problems can not be solved within polynomial time. Just like the name implies, the programming paradigm is founded on relations between variables as constraints.

The use of constraint programming for rotating workforce scheduling was first introduced by Chan and Weil [24] in 2000. Laporte and Pesant [14] further applied the concept in their paper from 2004, when they developed an algorithm which is able to handle a large variety of constraints and has been applied to several real-life instances and produced good-quality solutions in each case.

2.2.2 Heuristic Approaches

Heuristics is a class of methods often used to solve difficult optimization problems such as integer problems (see Lundgren et al. [25]). In short, a heuristic is a method which generates a good solution within a limited solution time but without a guarantee of the solution's quality. The solution found is however in most cases near-optimal, but it is often troublesome to decide how close to the optimum it is. These methods are usually developed for a particular class of optimization problems and adapted to take advantage of that problem's specific structure. In this thesis, heuristics are not used to find feasible schedules but this subsection gives some examples of how heuristics could be utilized.

In 1987, Burns and Koop [6] introduced a heuristic approach for finding feasible schedules. These schedules use no more than the minimum number of workers necessary while still satisfying the constraints given in Table 2.1. This heuristic algorithm is among the first to include multi-shifts (different types of shifts), as previous algorithms only focused on single-shift (one type of shift) manpower scheduling.

Montoya and Mejía [13] used in 2006 an iterative procedure with local search. Local search is a method which iteratively improves a feasible solution until a termination criterion is met. As the name suggests, the resulting objective value is a local optimum. In their article, the authors avoid local optima by guaranteeing that the algorithm never returns to a previously visited solution. The algorithm takes several constraints into account, among them minimum and maximum number of working hours and rest periods. The results appear promising as optimal solutions are found on all test instances, and the procedure also shows a good behaviour in terms of execution times.

In 2005-6, Musliu [11, 26, 2] introduced the concept of combining tabu-search with min-conflict heuristics to solve the scheduling problem. Tabu-search is a metaheuristic which combines local search with the ability to move towards solutions with an inferior objective value. This property gives a global dimension to the local-search since it enables more local optima to be found. Min-conflict heuristics are often used to solve constraint satisfaction problems. The algorithm selects at each iteration a random conflicted variable, which thus violates one or more constraints. A new value is then assigned for the selected variable such that it minimizes the number of conflicts. This process is iterated until a solution is found, or a pre-selected maximum number of iterations is reached. Returning to the paper, the suggested algorithms appear to perform well and produce good results. However, a feasible solution is not always guaranteed. Nevertheless, the methods work on all the test instances (given by literature and real-life problems from a broad range of organizations) included in the paper.

Musliu and Mörz [10] presented in 2004 a genetic algorithm. A genetic algorithm is a search heuristics which mimics the process of natural selection, where solutions are generated by techniques inspired by evolution principles such as inheritance, mutation, selection and crossover. The results were promising and the algorithm succeeded in generating feasible solutions, but was nevertheless outperformed by earlier methods (especially the one used in [9]). The authors do however leave suggestions for further improvement.

2.2.3 Polynomial Method

A more algebraic computational approach is a Boolean polynomial method, which was recently suggested by Falcón et al. [8]. The main idea is to count and enumerate all feasible rotating schedules. The polynomial structure is improved by the use of Gröbner bases, which reduces the computational time significantly. The method produces good results on all instances tested. Another advantage with this method is that problems which are infeasible are easily detected, making it possible to analyze constraints and how they affect the feasibility of the problem.

Chapter 3

Basic Models

In this chapter, two basic models are introduced. Section 3.1 presents an integer model, while Section 3.2 presents a network model.

In general, two different types of constraints are used when modeling:

- *Hard* constraints are always satisfied and written as explicit constraints in the model. An example of this is that exactly one schedule should be chosen each week. All constraints in this chapter are hard unless otherwise stated.
- *Soft* constraints are constraints which we would like to be satisfied, although it is not necessary. For instance, we may wish that our schedule contains five consecutive days off somewhere in the cycle. The way this is accomplished is by adding a penalty parameter and an auxiliary variable in the objective function, where the variable becomes active if the constraint is not satisfied. The penalty parameter, usually a very large number when minimizing, will then cause an increase in the objective function value. The advantage of soft constraints is that it is possible to prioritize different constraints to different needs with the help of the penalty parameters.

Chapter 4 gives a few examples of hard constraints as well as soft constraints, and also how they affect the outcome.

It should be mentioned that possible weekly schedules are generated by enumeration. Infeasible schedules, with regard to some existing hard constraints, are then filtered out. An example of such a constraint is the maximum number of consecutive work days that schedule should contain. Thus, several hard constraints are hidden in the generation of possible weekly schedules. This method can unfortunately not cover all the hard constraints. Instead, explicit hard constraints are added to the models.

3.1 The Integer Model

Our first mathematical description of the problem to construct rotating workforce schedules is henceforth denoted as *the integer model*. The scheduling is done over a cycle of W weeks, with N possible weekly schedules to choose from every week. The main variable is defined as

$$x_{vi} = \begin{cases} 1 & \text{if weekly schedule } i \text{ is chosen for week } v \\ 0 & \text{otherwise.} \end{cases}$$

Exactly one weekly schedule should be used each week, thus $\sum_i x_{vi} = 1, \forall\, v$.

Satisfying the staffing demand is something of high priority. To model this, parameters

$$H_{pd} = \text{the staffing demand for shift type } p \text{ on day } d,$$

and

$$b_{pid} = \begin{cases} 1 & \text{if weekly schedule } i \text{ contains shift type } p \text{ on day } d \\ 0 & \text{otherwise} \end{cases}$$

and auxiliary variables

$$y^u_{pd} = \text{the number of personnel lacking for shift type } p \text{ on day } d, \quad y^u_{pd} \geq 0,$$

$$y^o_{pd} = \text{the number of extra personnel for shift type } p \text{ on day } d, \quad y^o_{pd} \geq 0,$$

are introduced. The staffing demand is modeled as two soft constraints, one for y^u_{pd} and one for y^o_{pd}, and the penalty parameter M in the objective function is assigned a large value. This guarantees that the staffing demand is always satisfied, if this is at all possible. The constraints are written as

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \leq y^u_{pd}, \qquad \forall\, p, d,$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d.$$

Another useful constraint is to fix a shift to an arbitrary week and day. Suggestively, this shift should coincide with the staffing demand. For instance, if the first week's Monday should contain a day shift the constraint below is obtained:

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

There are two reasons for this constraint:

1. It is easier to read and compare schedules if they all begin alike.
2. Every schedule has W equivalent solutions since the full schedule is cyclical with W weeks, with the only difference being which week begins the schedule. By adding this constraint, symmetry is avoided as it removes $W - 1$ of the equivalent solutions. Observe that this does not remove any unique solutions since this shift has to be included in some week. The symmetry constraint only decides which week that is.

The difficulty with rotating workforce scheduling lies in merging the possible weekly schedules together in the best possible way without violating any constraints. To help with this, the following parameter is introduced:

$$a_{ij} = \begin{cases} 1 & \text{if weekly schedule } i \text{ week } v \text{ can be followed by weekly schedule } j \text{ week } v+1 \\ 0 & \text{otherwise} \end{cases}$$

where a_{ij} can capture different desired constraints. The a_{ij}-parameters are created when enumerating possible weekly schedules.

Looking at the problem of merging week v and $v+1$ with each other, the parameter a_{ij} can be expressed in a matrix structure. Figure 3.1 gives a graphical description of this.

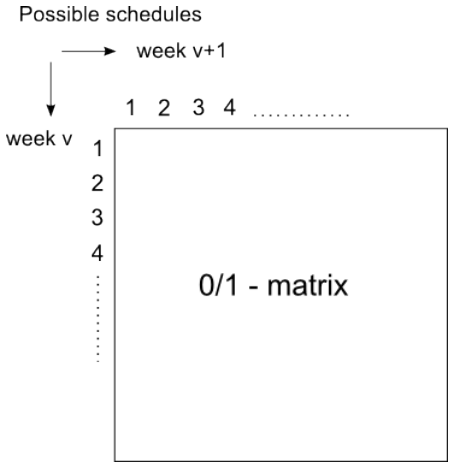


Figure 3.1: *The matrix containing the elements a_{ij}.*

Two constraints are needed for the merging:

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \geq x_{vi}, \qquad \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \geq x_{(v \bmod W)+1,j}, \qquad \forall\, v, j$$

where W is the number of weeks the full schedule consists of. The modulus operation is used because the full schedule is cyclical, meaning that $(W+1) \bmod W = 1$. If $x_{vi} = 1$ in the top constraint, then at least one of the elements in the sum must be used. In other words, if a possible weekly schedule i is chosen a certain week, then it must be able to be followed by a possible weekly schedule j. The second constraint work the other way around: if a weekly schedule j is chosen a specific week, then it has to be preceded by a weekly schedule i.

To summarize, the introduced constraints say that exactly one weekly schedule should be used every week and that the staffing demand should be satisfied. Additionally, the constraints also considers which weekly schedules that are allowed to follow a week v. Hence, with the information above, a very simple optimization model is now formed:

$$\min z = M \sum_p \sum_d (y^u_{pd} + y^o_{pd})$$

subject to

$$\sum_i x_{vi} = 1, \qquad \forall\, v$$

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \leq y^u_{pd}, \qquad \forall\, p, d$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d$$

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \geq x_{vi}, \qquad \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \geq x_{(v \bmod W)+1,j}, \qquad \forall\, v, j$$

$$x_{vi} = 0/1, \qquad \forall\, v, i$$

M is redundant here since the objective only penalizes unfulfilled staffing demand and not violation the of any other soft constraints.

3.2 The Network Model

Our second mathematical description of the problem is a network model with additional side constraints, where each node represents a weekly schedule. Hence, it is called *the network model*. In Figure 3.2 below, if the arc is used, then schedule i is used week v while schedule j is used week $v + 1$. The arcs work the same way as the parameter a_{ij} does in the integer model, thus the arcs only exist if the constraints allow schedule i to be followed by schedule j.

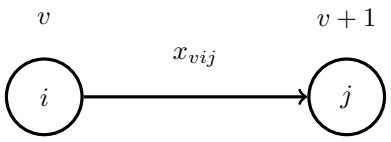


Figure 3.2: *Network definition*

An example of a network can be seen Figure 3.3. The rows consists of all possible weekly schedules, while the columns corresponds to weeks. The last week, the arc connects back to the weekly schedule chosen the first week.

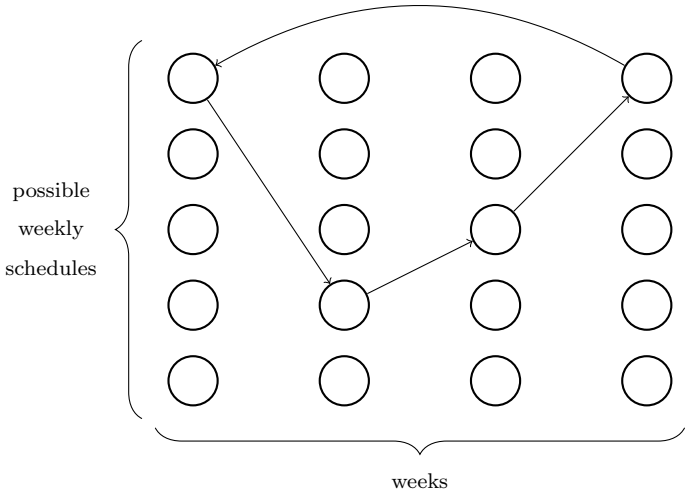


Figure 3.3: *Example of a network.*

Like with the first model, a few variables and parameters need to be introduced. The main variable is defined as follows:

$$x_{vij} = \begin{cases} 1 & \text{if arc between node } i \text{ and node } j \text{ is used from week } v \text{ to week } v+1 \\ 0 & \text{otherwise.} \end{cases}$$

The network model uses the same parameters as the integer model, with the only difference being the parameters a_{ij}. Instead the set A is used, which contains all arcs from week v to week $v+1$. Observe that the set A is the same for all weeks since every week has the same possible schedules. The arcs however represent a_{ij} (they are actually created from that matrix) which basically means that a_{ij} is still used, but represented by a set of all feasible arcs. The relation between the previous variables x_{vi} and the new x_{vij} is $\sum_i x_{vi} \sim \sum_{(i,j)\in A} x_{vij}$. In some constraints, the first sum can simply be replaced by the latter.

The first constraint introduced is the following flow conservation constraint:

$$\sum_{(i,k)\in A} x_{vik} - \sum_{(k,j)\in A} x_{(v \bmod W)+1,k,j} = 0, \qquad \forall\, v, k$$

This constraint says that everything that flows into a node must also flow out of it.

Just like before, exactly one weekly schedule should be used each week. This is enforced by

$$\sum_{(i,j)\in A} x_{1,ij} = 1,$$

together with the flow conservation constraint.

To avoid symmetry, the constraint below is added:

$$\sum_{(i,j)\in A^r} x_{1,ij} = 1,$$

where A^r is a subset of A restricted such that A^r contains all the possible weekly schedules that have a day shift on the Monday.

The staffing demand is, just like in the previous model, given by the following two soft constraints:

$$H_{pd} - \sum_{v} \sum_{(i,j)\in A} b_{pid} x_{vij} \leq y^u_{pd}, \qquad \forall\, p, d,$$

$$\sum_{v} \sum_{(i,j)\in A} b_{pid} x_{vij} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d.$$

The only difference is that $\sum_i x_{vi}$ is replaced with $\sum_{(i,j)\in A} x_{vij}$. The entire network model is therefore the following:

$$\min z = M \sum_{p} \sum_{d} (y^u_{pd} + y^o_{pd})$$

subject to

$$\sum_{(i,j)\in A} x_{1,ij} = 1$$

$$\sum_{(i,j)\in A^r} x_{1,ij} = 1$$

$$H_{pd} - \sum_{v} \sum_{(i,j)\in A} b_{pid} x_{vij} \leq y^u_{pd}, \qquad \forall\, p, d$$

$$\sum_{v} \sum_{(i,j)\in A} b_{pid} x_{vij} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d$$

$$\sum_{(i,k)\in A} x_{vik} - \sum_{(k,j)\in A} x_{(v \bmod W)+1,k,j} = 0, \qquad \forall\, v, k$$

$$x_{vij} = 0/1, \qquad \forall\, v, i, j$$

Unfortunately, this model resulted in very long solution times, which made it unusable in practice. The most probable reason for this behaviour is that although the problem has a network structure with additional side constraints, the variables used are integers. The current CPLEX-solver is of limited applicability since it does not have features and options that enables solving side-constrained network problems with integer variables by exploiting the network structure. The consequence of this is increased solution times, as seen when running the network model. Hence, the results in this report will only cover the integer model, which from now on will be shortened to *the model*.

Chapter 4

Illustrative Examples

In this chapter, a few examples of hard and soft constraints and the resulting schedules that the integer model produces are given. The different shifts used can be seen in Table 4.1, while the staffing demand is shown in Table 4.2. The tables make it clear that the scheduling is done with a demand for night, day and evening shifts. The staffing demand is for 30 shifts, and, with the given shift lengths, a person works on average 5 shifts a week. Thus, the staffing demand requires 6 persons, which leads to 6 scheduling weeks. In every example, information is also given about the size of the problem and the solution times.

Table 4.1: *The shifts used in this chapter.*

Shift	Time	Break
N	$22^{00} - 06^{00}$	0 min
D	$06^{00} - 14^{00}$	0 min
E	$14^{00} - 22^{00}$	0 min

Table 4.2: *The workload matrix used in this chapter.*

Shift	Mo	Tu	We	Th	Fr	Sa	Su
N	1	1	1	1			
D	3	3	3	3	3	3	3
E	1	1	1	1	1		

The following hard constraints are taken into consideration in the enumeration process:

- It is prohibited to work more than 6 consecutive days.
- After the last night shift in a working period, there should be at least a period of 50 hours off until the next shift starts.

The possible weekly schedules are enumerated, and then the infeasible schedules with regard to the above constraints are filtered out. The a_{ij}-parameters captures these restrictions and assist in merging of weekly schedules.

Recall the basic model from Chapter 3.1:

$$\min z = M \sum_p \sum_d (y^u_{pd} + y^o_{pd})$$

subject to

$$\sum_i x_{vi} = 1, \qquad \forall\, v$$

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \leq y^u_{pd}, \qquad \forall\, p, d$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d$$

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \geq x_{vi}, \qquad \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \geq x_{(v \bmod W)+1,j}, \qquad \forall\, v, j$$

$$x_{vi} = 0/1, \qquad \forall\, v, i$$

If this model is solved, using the two restrictions mentioned above, the schedule seen in Figure 4.1 is produced.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D	D		D
2	D	E	E	E	E		D
3	N	N	N	N			
4	E		D	D	D	D	D
5		D				D	
6	D	D	D	D	D	D	

Figure 4.1: *Resulting schedule when solving the basic model.*
Solution time: 7 s
 # possible weekly schedules: 563
 # variables: 3420
 # constraints: 6805

This schedule might not look very pleasant to use, but it satisfies all of the requirements. The hard constraints are clearly met since only one schedule each week is used, and the first week's Monday contains a day shift. In addition the constraints described by the a_{ij}-parameters are also fulfilled since the resulting schedule has a maximum of 6 consecutive work days, and after the last night shift there is 84 hours off. The staffing demand is obviously also met since the

numbers from the workload matrix, Table 4.2, match the shifts in the resulting schedule.

In the following examples, and addition of some constraints to the basic model is done to see what happens. Thus, the resulting schedules are always compared to the schedule in Figure 4.1.

Example 4.1. In this example, the model is extended by adding a constraint saying that if you work anything during the weekend, you must also work the entire weekend. This implementation is incorporated in the enumeration of the possible weekly schedules and in the values of the a_{ij}-parameters. Solving the basic model now instead gives the schedule in Figure 4.2.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D	D		
2	E	E	E	E	E		
3	D	D		D	D	D	D
4	D		D	D	D	D	D
5		D	D			D	D
6	N	N	N	N			

Figure 4.2: *Resulting schedule in Example 4.1, where the a-parameters have an added constraint.*
Solution time: 1 s
 # possible weekly schedules: 298
 # variables: 1830
 # constraints: 3625

Looking at the resulting schedule, it is apparent that the only significant difference between this schedule and the schedule in Figure 4.1 is how the weekends appear. The constraints from the basic model are still satisfied, but with the added condition of having the weekends clustered together. If you work Saturday the condition forces you to also work Sunday and vice versa.

□

Example 4.2. In this example, single days off are unfavourable and thus to be penalized. Hence, soft constraints which have the penalty parameter Q in the objective function are added. To help with this, the following parameters are added:

$$e_i = \begin{cases} 1 & \text{if weekly schedule } i \text{ has a single day off in the middle of the week} \\ 0 & \text{otherwise,} \end{cases}$$

$$e_{in}^s = \begin{cases} 1 & \text{if weekly schedule } i \text{ ends with exactly } n \text{ days off, } n = 0, 1 \\ 0 & \text{otherwise,} \end{cases}$$

$$e_{in}^b = \begin{cases} 1 & \text{if weekly schedule } i \text{ begins with exactly } n \text{ days off, } n = 0, 1 \\ 0 & \text{otherwise.} \end{cases}$$

The auxiliary variable

$$s_v = \begin{cases} 1 & \text{if a single day off occurs in the merging of week } v \text{ and } v+1 \\ 0 & \text{otherwise} \end{cases}$$

is used in the objective function. A single day off occurs if a week ends with a single day off and the next week begins with work, or if a week ends with work and the next week begins with a single day off. The idea is to capture this phenomenon with the help of e^s_{in},e^b_{in} and s_v. If the described situation arises, s_v will be activated and thus giving an increase to the objective function value. Using the parameters and auxiliary variable above, the basic model can be expanded:

$$\min z = M \sum_p \sum_d (y^u_{pd} + y^o_{pd}) + Q \sum_v (\sum_i e_i x_{vi} + s_v)$$

subject to

$$\sum_i x_{vi} = 1, \forall v$$

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \leq y^u_{pd}, \qquad \forall\, p, d$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \leq y^o_{pd}, \qquad \forall\, p, d$$

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \geq x_{vi}, \qquad \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \geq x_{(v \bmod W)+1,j}, \qquad \forall\, v, j$$

$$\sum_i e^s_{i,1} x_{vi} + \sum_j e^b_{j,0} x_{(v \bmod W)+1,j} \leq s_v + 1, \qquad \forall\, v$$

$$\sum_i e^s_{i,0} x_{vi} + \sum_j e^b_{j,1} x_{(v \bmod W)+1,j} \leq s_v + 1, \qquad \forall\, v$$

$$x_{vi} = 0/1, \qquad \forall\, v, i$$

Solving the above model yields the schedule seen in Figure 4.3.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	E	E			D
2	N	N	N	N			
3	E	E			D	D	D
4	D		D	D	D	D	D
5		D	D	D	E		
6	D	D	D	D	D	D	

Figure 4.3: *Resulting schedule in Example 4.2.*
Solution time: 6 s
 # possible weekly schedules: 563
 # variables: 3426
 # constraints: 6817

This schedule has 3 single days off, while the schedule in Figure 4.1 has 6 single days off. Hence, the added constraint cuts the single days off in half without loosing any other quality (with regard to the constraints).

□

Example 4.3. This example examines the possibility of penalizing a certain number of consecutive work days. To help us with this, we introduce the penalty parameter Q_n, where n is the number of consecutive work days. The following parameters are used:

$$f_{in} = \begin{cases} 1 & \text{if weekly schedule } i \text{ contains } n \text{ consecutive work days in the} \\ & \text{middle of the week} \\ 0 & \text{otherwise,} \end{cases}$$

$$f^s_{in} = \begin{cases} 1 & \text{if weekly schedule } i \text{ ends with exactly } n \text{ consecutive work days} \\ 0 & \text{otherwise,} \end{cases}$$

$$f^b_{in} = \begin{cases} 1 & \text{if weekly schedule } i \text{ begins with exactly } n \\ & \text{consecutive work days} \\ 0 & \text{otherwise.} \end{cases}$$

An auxiliary variable is also needed:

$$y_{nv} = \begin{cases} 1 & \text{if the joint between week } v \text{ and week } v+1 \text{ accumulates } n \\ & \text{consecutive work days} \\ 0 & \text{otherwise.} \end{cases}$$

The constraint used in this example is similar to the constraint for single days off in Example 4.2. If a weekly schedule has n consecutive work days in the middle of the week, it is penalized in the objective function by the parameter Q_n. If some work period length is feasible, the penalty parameter is simply zero for that length n. Consecutive work days can also appear when weekly schedules are merged together. The parameters f^s_{in} and f^b_{in} keep track of how many work days a weekly schedule ends with and begins with respectively. If

an unwanted work period length occurs in the merging, the auxiliary variable y_{nv} is activated. For example, if a week v ends with 2 work days and schedule $v+1$ begins with 3 work days, then the merging accumulates a work period of 5 days. The auxiliary variable $y_{5,v}$ is then activated, which leads to a penalty in the objective function value according to Q_5. The resulting model is given below.

$$\min z = M \sum_p \sum_d (y^u_{pd} + y^o_{pd}) + \sum_n Q_n \sum_v (y_{nv} + \sum_i f_{in} x_{vi})$$

subject to

$$\sum_i x_{vi} = 1, \qquad \forall\, v$$

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \le y^u_{pd}, \qquad \forall\, p, d$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \le y^o_{pd}, \qquad \forall\, p, d$$

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \ge x_{vi}, \qquad \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \ge x_{(v \bmod W)+1,j}, \qquad \forall\, v, j$$

$$\sum_i f^s_{id} x_{vi} + \sum_j f^b_{j,c-d} x_{(v \bmod W)+1,j} \le y_{cv} + 1, \qquad \forall\, v,\ c = 1,..,6,\ d = 0,..,c$$

$$x_{vi} = 0/1, \qquad \forall\, v, i$$

The resulting schedule becomes as in Figure 4.4, where single work days have been penalized.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	E	E	E		D
2	N	N	N	N			
3		D	D			D	D
4	E	E		D	D	D	D
5	D		D	D	D		
6	D	D	D	D	D	D	

Figure 4.4: *Resulting schedule in Example 4.3.*
Solution time: 7 s
 # possible weekly schedules: 563
 # variables: 6841
 # constraints: 7057

The schedule above clearly has no single work days, while the schedule in Figure 4.1 has 3 of them.

□

Example 4.4. A restriction is added to the basic model about how many weekends off there should be in the full schedule. Specifically, the schedule should have at least $L = 3$ weekends off. A consequence of this constraint together with the staffing demand used in this chapter is that if a work day occurs during the weekend, both Saturday and Sunday should be worked. Thus, the resulting schedule should be somewhat similar to the result in Example 4.1. We introduce the parameter

$$l_i = \begin{cases} 1 & \text{if weekly schedule } i \text{ has the weekend off} \\ 0 & \text{otherwise,} \end{cases}$$

and the auxiliary variable

$$t = \text{the number of weekends missing to satisfy } L = 4 \text{ weekends off}, \quad t \geq 0$$

to model this soft constraint. Like in Example 4.2, Q is used as the penalty parameter. If weekends off are missing to satisfy the requirement, the auxiliary variable t will be activated and a penalty in objective function value is received. The resulting mathematical model can be seen below.

$$\min z = M \sum_p \sum_d (y^u_{pd} + y^o_{pd}) + Qt$$

subject to

$$\sum_i x_{vi} = 1, \forall\, v$$

$$H_{pd} - \sum_v \sum_i b_{pid} x_{vi} \leq y^u_{pd}, \forall\, p, d$$

$$\sum_v \sum_i b_{pid} x_{vi} - H_{pd} \leq y^o_{pd}, \forall\, p, d$$

$$\sum_i b_{2,i,1} x_{1,i} = 1$$

$$\sum_j a_{ij} x_{(v \bmod W)+1,j} \geq x_{vi}, \forall\, v, i$$

$$\sum_i a_{ij} x_{vi} \geq x_{(v \bmod W)+1,j}, \forall\, v, j$$

$$L - \sum_v \sum_i l_i x_{vi} \leq t$$

$$x_{vi} = 0/1, \qquad \forall\, v, i$$

If this model is solved, the schedule in Figure 4.5 is received.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D		D	D
2	E		E		D	D	D
3	D	E		D	D	D	D
4		D	D	E	E		
5		N	N	N	N		
6	D	D	D	D	D		

Figure 4.5: *Resulting schedule in Example 4.4.*
Solution time: 7 s
 # possible weekly schedules: 563
 # variables: 3421
 # constraints: 6806

The schedule above has 3 full weekends off, as requested, without breaking any of the other constraints. If compared to Figure 4.2 in Example 4.1, it is seen that they are very similar. Besides the obvious result of both having 3 work weekends, they both also have entire work periods of night shifts and evening shifts. The structure of the day shifts are a little dissimilar, but the difference is negligible since we could easily move the shifts in either schedule to receive the other without any notable negative effect.

□

Example 4.5. In this final example, the model from Example 4.4 is used but instead with $L = 4$. In other words, 4 weekends off is required. However, this is impossible without failing to satisfy the staffing demand. Id the staffing demand is penalized harder than the weekend constraints, the schedule seen in Figure 4.6 is received.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D		D	D
2	D	D	D		D	D	D
3		D		D	D	D	D
4	D		D	D	D		
5		N	N	N	N		
6	E	E	E	E	E		

Figure 4.6: *Resulting schedule when solving the initial setup in Example 4.5.*
Solution time: 6 s
 # possible weekly schedules: 563
 # variables: 3421
 # constraints: 6806

The resulting schedule, as expected, does not give 4 weekends off. Instead, just as it should be, it has full demand coverage. However, it does have 3 full weekends off since this accommodates to the constraint in the best possible way. As the reader might recall, we penalize harder the more number of weekends

that are missing to satisfy the constraint. If the staffing demand is penalized softer than the weekends off, the schedule in Figure 4.7 is received.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D		D	D
2	D	D	D		D	D	D
3				D	D		
4	D	D	D	D	D		
5		N	N	N	N		
6	E	E	E	E	E		

Figure 4.7: *Resulting schedule when solving the modified setup in Example 4.5.*
Solution time: 7 s
 # possible weekly schedules: 563
 # variables: 3421
 # constraints: 6806

In this schedule, we see that there are 4 weekends off, and thus personnel is missing one weekend.

□

We have now gone through the basics of the mathematical modelling and how soft constraints work. Thus, we are ready to move on to the case studies. The model used in the case studies are not shown in detail since the principle already has been shown in this chapter. Hence, the constraints used are mentioned but not how they are mathematically modeled.

Chapter 5

Case 1

Case 1 follows the standard of a six week schedule for six persons, just as in previous examples. However the types of shifts are a little different from what has been seen before in this thesis, with two night shifts and two day shifts. At the end of this chapter, the expansion of scheduling for twelve weeks is explored to see if this results in different solutions. Every schedule contains information about the size of the problem and the solution times, just as in Chapter 4.

5.1 Shifts and Staffing Demand

In this case, we use four different shifts as seen in Table 5.1. Table 5.2 shows the staffing demand.

Table 5.1: *The different shifts occurring in Case 1.*

Shift	Time	Break
NL	$19^{00} - 7^{00}$	0 min
NS	$23^{00} - 7^{00}$	0 min
DL	$7^{00} - 19^{00}$	0 min
DS	$7^{00} - 15^{00}$	0 min

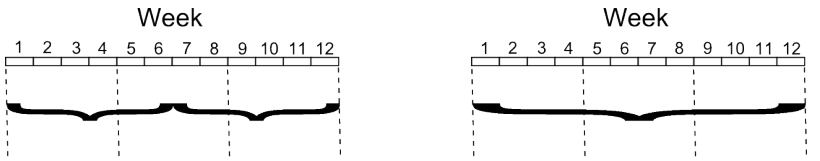
Table 5.2: *The workload matrix for Case 1.*

Shift	Mo	Tu	We	Th	Fr	Sa	Su
NL	1	1	1	1			
NS	1	1	1	1			
DL	1	1	1	1	2	2	2
DS			1	1			

5.2 Constraints

Case 1 has several hard and soft constraints besides the most basic ones given in Chapter 3:

- Every day should include at least 11 consecutive hours off. The beginning of the day cycle can be set at a chosen time and does not have to start at midnight.
- Every week should contain at least 36 consecutive hours off.
- For every four week interval, from a chosen starting point, the average working week should not exceed 40 hours. If the starting point is set to the beginning of the first week, three different intervals will appear until the pattern repeats itself. Remembering that the full schedule is cyclical, Figure 5.1a shows these three intervals, with the braces representing the six week schedule. If we instead schedule for 12 weeks, this hard constraint will cover the intervals seen in Figure 5.1b.



(a) *The four-week intervals when scheduling for 6 weeks.*

(b) *The four-week intervals when scheduling for 12 weeks.*

Figure 5.1: *The two different four-week constraints.*

- To avoid symmetry, the first week's Monday must contain shift DL.
- After the last night shift in a work period, there should be a period of at least 50 hours off until the next shift starts.
- A schedule should contain a maximum of 5 consecutive work days.
- We wish to find a schedule where everyone works with everyone else as much as possible. In other words, person A should never encounter person B only once every cycle, but everyone else several more times per cycle. However, how these encounters are distributed during the cycle does not matter, as we are only looking on how many times a person works with another person. It should be clarified that shifts NL and NS count as a meet, as well as DL and DS.
- If someone works a shift during the weekend, then both Saturday and Sunday should be workdays. With the staffing demand given for this case, this means that every cycle contains two work weekends and four weekends off.
- It is prohibited to work two weekends in a row. This means that in a schedule with a six week cycle, there are two different possibilities of weekend layouts. There is either one weekend off between the work weekends, or two weekends off between the work weekends. Figure 5.2 clarifies this. We are interested in both possibilities.

Week	Sa	Su
1	x	x
2		
3	x	x
4		
5		
6		

(a) *One weekend off between work weekends.*

Week	Sa	Su
1	x	x
2		
3		
4	x	x
5		
6		

(b) *Two weekends off between work weekends.*

Figure 5.2: *The two possible layouts of weekends.*

The case is divided into three different subcases. The first two subcases, Case 1a and Case 1b, are very alike. The difference between them is the desired number of consecutive days off and consecutive working days. In Case 1a, we look at longer work periods as well as longer periods of days off. The objective is that the schedule should consist of periods of 4-5 consecutive days of work and that at least 6 consecutive days off should appear at least once. Case 1b considers shorter periods of work and days off. Here, 4 consecutive work days is allowed once, otherwise there should be a maximum of 3 consecutive work days. The consecutive days off should be 2 or 3. Case 1c however schedules for 12 weeks instead of 6 weeks, to see if it is possible to improve any schedules received in Case 1a and Case 1b.

5.3 Results

In all simulation results shown below, all the constraints presented above - both from the basic model and from this case - are used with the exception of two soft constraints:

- that everyone should work with everyone
- distribution of weekends

As mentioned above the distribution of weekends have two different possibilities, and the constraint that everyone should work with everyone will either be active or inactive. Thus, this gives four different combinations of constraints in each subcase.

5.3.1 Case 1a - Longer Work Periods

To begin with, the schedules with one weekend off between work weekends are considered. The resulting two schedules can be seen in Figure 5.3, where subfigure 5.3b include the additional soft constraint that everyone should work with everyone as much as possible.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL					
2	NL		NS	NS	NL		
3	NS		NL	NL	NS		
4			DS	DS	DL	DL	DL
5			DL	DL	DL		
6						DL	DL

(a) *No constraint that everyone should work with everyone.*
Solution time: 16 s
 # possible weekly schedules: 790
 # variables: 4882
 # constraints: 10 060

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL	DL	DL			
2			DS	DS	DL	DL	DL
3			NL		NL		
4					DL	DL	DL
5	NL		NL				
6	NS		NS	NS	NS		

(b) *Everyone should work with everyone.*

Solution time: 25 s
 # possible weekly schedules: 790
 # variables: 5392
 # constraints: 11 330

Figure 5.3: *Resulting schedules for longer work periods with one weekend off between work weekends.*

When studying the two schedules, it becomes apparent that they do not differ very much. Schedule 5.3a has a work period of 3 shifts, while schedule 5.3b lacks this period and instead has a work period of 2 shifts. Schedule 5.3a has an entire week off, something which does not occur in schedule 5.3b. Still, it contains 6 consecutive days off which is also good. In addition, schedule 5.3b has a rest period of 5 consecutive days, which schedule 5.3a does not have. In conclusion, when it comes to days off, the schedules are of equal quality. Nonetheless, if we summarize the differences mentioned, schedule 5.3a can be considered to be the better choice since we generally prefer longer work periods.

However, it is also interesting to see how well the "everyone should work with everyone" constraint actually performed. To see this we sum up how many times person A works with persons B, C, D, E and F, respectively. It is enough to look at just one person. The reason for this is that since the schedule is cyclical, the collaboration distribution will look alike for everybody. The only difference will be which person they work with, but the number of times they work with the others will be exactly the same. We define the persons such that person A begins the schedule on week 1, person B on week 2 and so on, with person F on week 6 being the last. Then, for example, if person A works 5 times with person B, a direct consequence of the cyclical feature will be that person B works 5 times with person C. The reader might do well in convincing himself of this property before continuing. The resulting collaboration distribution can be seen in Table 5.5.

Table 5.5: *The collaboration distribution found in the schedules in Figure 5.3.*

Person	#times
B	7
C	2
D	0
E	2
F	7

(a) *The number of times, in schedule 5.3a, that person A works with the other staff.*

Person	#times
B	4
C	3
D	4
E	3
F	4

(b) *The number of times, in schedule 5.3b, that person A works with the other staff.*

As is evident from Table 5.5, the added soft constraint gives a significant effect

on the collaboration distribution. In schedule 5.3a, person A never works with person D, but works pretty often with persons B and F. When we add the constraint and obtain a new collaboration distribution, we clearly see that person A works evenly with everyone.

The differences mentioned above are the only dissimilarities between the schedules with regard to the objective function and constraints. The question whether schedule 5.3a or 5.3b is the best is thus a matter of preferences, as it often is with workforce scheduling.

Moving on to the other type of weekend distribution, Figure 5.4 shows the resulting schedules when there should be two weekends off between work weekends.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL					
2	NS		NL	NL	NL		
3			DL	DL	DL	DL	DL
4			DS	DS	DL		
5	NL		NS	NS	NS		
6						DL	DL

(a) *No constraint for everyone should work with everyone.*
Solution time: 15 s
 # possible weekly schedules: 790
 # variables: 4882
 # constraints: 10 060

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL	DS	DL	DL		
2						DL	DL
3	NS	NS					
4	NL	NL	NS	NS			
5			DL	DS	DL	DL	DL
6			NL	NL			

(b) *Everyone should work with everyone.*

Solution time: 40 s
 # possible weekly schedules: 790
 # variables: 5392
 # constraints: 11 330

Figure 5.4: *Resulting schedules for longer work periods with two weekends off between work weekends.*

Just like in Figure 5.3, the largest difference between these schedules is the lengths of work periods. In schedule 5.4a, we have a period of 3 shifts which is replaced by a period of 2 shifts in Figure 5.4b. In both schedules we get the desired week off. Schedule 5.4a contains 2 consecutive days off twice, implying that these occurrences are inevitable. However, the same thing happens in schedule 5.4b, which means that the added soft constraint does not worsen this behaviour.

Like before, we are of course interested in how well the added constraint actually performed. Table 5.7 shows the collaboration distribution in the respective schedules given in Figure 5.4.

Table 5.7: *The collaboration distribution found in the schedules in Figure 5.4.*

Person	#times
B	3
C	0
D	12
E	0
F	3

(a) *The number of times, in schedule 5.4a, that person A works with the other staff.*

Person	#times
B	2
C	5
D	4
E	5
F	2

(b) *The number of times, in schedule 5.4b, that person A works with the other staff.*

As clearly shown by Table 5.7a, person A and person D work a lot with each

other. Person A also works a few times with persons B and F, but never meets persons C and E. If we add the soft constraint that everyone should work with everyone, we see a notable difference. Person A works considerably less with person D and instead has a fairly even distribution amongst all its colleagues. Interestingly enough, person A works the most with persons C and E when the soft constraint is added, while these are the only ones person A does not at all work with without the constraint.

5.3.2 Case 1b - Shorter Work Periods

Just like in Case 1a, we commence this section by looking at the soft constraint that require one weekend off between work weekends. Figure 5.5 shows the resulting schedules while Table 5.9 shows the collaboration distribution.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL				DL	DL
2			NS	NS			
3	NL	NL				DL	DL
4			DL	DS	DL		
5	NS	NS	NL	NL			
6			DS	DL	DL		

(a) *No constraint for everyone should work with everyone.*
Solution time: 23 s
 # possible weekly schedules: 790
 # variables: 4882
 # constraints: 10 061

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL				DL	DL
2			NS	NS			
3	NL	NL				DL	DL
4			DL	DL	DL		
5			DS	DS	DL		
6	NS	NS	NL	NL			

(b) *Everyone should work with everyone.*

Solution time: 34 s
 # possible weekly schedules: 790
 # variables: 5392
 # constraints: 11 331

Figure 5.5: *Resulting schedules for longer work periods with two weekends off between work weekends.*

Table 5.9: *The collaboration distribution found in the schedules in Figure 5.5.*

Person	#times
B	0
C	7
D	4
E	7
F	0

(a) *The number of times, in schedule 5.5a, that person A works with the other staff.*

Person	#times
B	3
C	4
D	4
E	4
F	3

(b) *The number of times, in schedule 5.5b, that person A works with the other staff.*

When looking at schedules 5.5a and 5.5b, it becomes apparent that they almost do not differ at all. This is a very interesting result, because if we look at the collaboration distribution in Table 5.9, we see that they obviously differ a lot with regard to this property. This means that we can add the collaboration distribution soft constraint without giving up on anything else. Hence, if an even collaboration distribution is wished for, schedule 5.5b is without doubt the best candidate.

We continue by looking at the scenario where we have 2 weekends off between weekends of work. Figure 5.6 shows the resulting schedules and Table 5.11 gives the collaboration distribution for each schedule respectively.

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL	NS	NS			
2	NS		NL			DL	DL
3			DS	DL	DL		
4			NL	NL			
5	NL		NS			DL	DL
6			DL	DS	DL		

(a) *No constraint for everyone should work with everyone.*
Solution time: 15 s
 # possible weekly schedules: 790
 # variables: 4882
 # constraints: 10 061

Week	Mo	Tu	We	Th	Fr	Sa	Su
1	DL			DL	DL		
2			DS	DS	DL		
3	NS		NL			DL	DL
4			NS		NL		
5	NL	NS		NL	NS		
6		DL	DL			DL	DL

(b) *Everyone should work with everyone.*

Solution time: 25 s
 # possible weekly schedules: 790
 # variables: 5392
 # constraints: 11 331

Figure 5.6: *Resulting schedules for longer work periods with one weekend off between work weekends.*

Table 5.11: *The collaboration distribution found in the schedules in Figure 5.6.*

Person	#times
B	0
C	0
D	18
E	0
F	0

(a) *The number of times, in schedule 5.6a, that person A works with the other staff.*

Person	#times
B	4
C	3
D	4
E	3
F	4

(b) *The number of times, in schedule 5.6b, that person A works with the other staff.*

At a first glance, the schedules seem to be a little more dissimilar this time. However, with regard to the hard and soft constraints, they actually do not differ that much. They both have a work period of 4 shifts, which is allowed once, and then only work periods of 2 or 3 shifts.

The collaboration distribution is however radically different in the schedules. In schedule 5.6a, person A *only* works with person D and never with anyone else. With the added soft constraint, everyone in the staff work with each other as evenly as it is possible. Consequently, schedule 5.6b is probably preferred in most situations.

5.3.3 Case 1c - Schedule with 12 Weeks

In this section, the scheduling is extended to 12 weeks instead of 6 weeks like before. It is interesting to see how this change affects the results, since it gives the process of scheduling more degrees of freedom. The constraints remain the same, except the layout of the weekends which is a little different. This is due to the availability of more possible combinations.

Looking only at longer work periods like in Case 1a, the resulting schedule without any constraint on the collaboration distribution is seen in Figure 5.7. Figure 5.8 shows the result when the collaboration distribution constraint is added.

Week	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL	DS			DL	DL	DL	DL	DS				
3				DL	DL	DL	DL							
5	NL	NL	NL	NL							DL	DL	DL	DL
7										DL	DS	DL	DL	DL
9			DL	DS	DL			NL	NL	NL	NS			
11	NS	NS	NS	NS				NS	NS	NS	NL			

Figure 5.7: *Resulting schedule when scheduling for 12 weeks in Case 1, with no constraint on the collaboration distribution.*
Solution time: ∼2 min
 # possible weekly schedules: 790
 # variables: 9 704
 # constraints: 20 052

Week	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
1	DL	DL	DS	DL	DL			NL	NS	NS	NS			
3						DL	DL	DL	DL					
5	NL	NS	NS	NL						DS	DL	DL	DL	DL
7			NL	NS									DL	DL
9	NS	NL						NS	NL	NL	NL			
11			DL	DS	DL	DL	DL			DL	DS	DL		

Figure 5.8: *Resulting schedule when scheduling for 12 weeks in Case 1, with added constraints for the collaboration distribution.*
Solution time: ∼25 min
 # possible weekly schedules: 790
 # variables: 11 732
 # constraints: 22 582

In comparison with the schedules found in Figures 5.3 and 5.4, these results are clearly improved. Looking at the schedule in Figure 5.7 to begin with, it has two work periods with 3 shifts. Recall that one work period of 3 shifts occurs in both schedules 5.3a and 5.4a. This is the exact same result since the above schedules contains twice as many weeks. In addition, compared to previous schedules which both have one week off, the new schedule has three full weeks off. The schedule in Figure 5.8 is also an upgrade compared to the schedules in Figures 5.3b and 5.4b. Instead of two work periods with 2 shifts, the result with a 12-week schedule is that one of the work periods becomes a work period of 3 shifts. Since longer work periods are desired, this is clearly an improvement. The number of weeks off also makes an upswing since the new schedule contains two weeks off, while schedule 5.3b contains one week off and schedule 5.4b has no full week off. The only disadvantage of using 12 weeks instead of 6 is the increased solution times, where seconds becomes minutes.

The conclusion is that there is no loss and only gain, besides the increased solution times, when scheduling for 12 weeks compared to 6 weeks. However, it

is also interesting to see how well the scheduling performs with respect to the collaboration distribution. Table 5.12 shows the collaboration distribution in both schedules.

Table 5.12: *The collaboration distribution when scheduling for 12 weeks.*

Person	#times
B	10
C	1
D	14
E	1
F	10

(a) *The number of times, in schedule 5.7, that person A works with the other staff.*

Person	#times
B	7
C	8
D	6
E	8
F	7

(b) *The number of times, in schedule 5.8, that person A works with the other staff.*

Undoubtedly, the collaboration distribution is affected by adding a soft constraint for it. If the schedule in Figure 5.7 is used, person A will only meet persons C and E once every 12 weeks. The schedule in Figure 5.8 however allows person A to work approximately the same number of shifts with everyone in the remaining staff.

Chapter 6

Case 2

In Case 2, the staffing demand is different every other weekend. Thus, the scheduling basis is extended to two weeks instead of one week like before. The staffing demand requires three persons to satisfy, which gives six scheduling weeks (since the basis is two weeks).

6.1 Shifts and Staffing Demand

We here use five different shifts. Table 6.1 shows the shifts, while Table 6.2 shows the staffing demand. Notice that unlike previous workload matrices, this one covers two weeks which have somewhat different demands.

Table 6.1: *The different shifts occurring in Case 2.*

Shift	Time	Break
D	$7^{00} - 15^{00}$	30 min
E	$14^{30} - 22^{00}$	30 min
Dw	$9^{00} - 16^{00}$	30 min
Ew	$15^{30} - 22^{00}$	30 min
L	$9^{00} - 22^{00}$	1 h

Table 6.2: *The workload matrix for Case 2.*

Shift	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
D	1	1	1	1	1			1	1	1	1	1		
E	1	1	1	1	1			1	1	1	1	1		
Dw						1	1							
Ew						1	1							
L													1	1

6.2 Constraints

Case 2 has several constraints in common with Case 1, and these will not be thoroughly explained since they have already been described in Section 5.2.

- Every day should include at least 11 consecutive hours off. Just like before, the beginning of the day cycle can be set at a chosen time and does not have to start at midnight. In practice, this means that shift E can not be followed by shift D and shift L can not be followed by shift D.
- Every week should contain at least 36 consecutive hours off.
- Every four weeks, the average working time should not surpass 40 hours.
- To avoid symmetry, the first week's Monday must contain a day shift.
- The shifts and days off should be as evenly distributed in time as possible.
- It is not feasible to work more than three consecutive days if some of these days include a weekend shift.
- If you work Friday evening, you must also work at least one shift during the weekend.
- It is not feasible to work more than two weekends in a row.

6.3 Results

The resulting schedule can be seen in Figure 6.1.

Week	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
1	D	E	E	E		Ew	Dw	D				E	L	
3		D	D	D	D			E	E	E	E			L
5	E				E	Dw	Ew		D	D	D	D		

Figure 6.1: *Resulting schedule in Case 2.*
Solution time: 10 s
 # possible weekly schedules: 1606
 # variables: 9768
 # constraints: 19 846

In general, the work periods in this schedule are pretty short. The constraints allow a maximum of 5 consecutive shifts (since it is prohibited to work more than 3 consecutive days in connection with the weekends). However, we get no work periods with 5 shifts, but several work periods of 4 shifts in the resulting schedule. The 2 single days off that occur are not desired, but appears to be inevitable as they are penalized very hard in the model.

It was also possible to avoid many work weekends in a row. Whenever someone do work the weekend, there is at most 3 consecutive shifts as desired. Also, every time someone works Friday evening, they also work the Saturday and/or Sunday. Hence, the schedule produced by the model satisfies all the requirements and has a fairly even distribution of shifts and days off.

6.4 Results When Altering a Constraint

We are curious as to see how well the model performs when we push it a little harder, which we do by strengthening a constraint. If we modify the next to last constraint to include the Friday's day shift as well, we should hopefully observe a change in the schedule. The result can be seen in Figure 6.2.

Week	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
1	D	D	D	D		Ew	Dw	E				E	L	L
3			E	E				D	E	E	E			
5	E	E			E	Dw	Ew		D	D	D			

Figure 6.2: *Resulting schedule when altering a constraint in Case 2.*
Solution time: ∼2 h
 # possible weekly schedules: 1572
 # variables: 9564
 # constraints: 19438

A very interesting result is that of the hugely increased solution time, which went from 10 seconds to approximately 2 hours. The reason for this is probably connected to the lack of day shifts on both Fridays, which the observant reader might have noticed. There are 2 shifts that should be covered every Friday, which implies that we also have 2 workers that are obligated to work during the weekend. At the same time, we have a constraint saying that only 3 consecutive work days are allowed if you work during a weekend. These properties, together with the prohibition of working more than two weekends in a row, makes it impossible to satisfy the staffing demand. Hence, we get the resulting schedule in Figure 6.2 where we lack 2 shifts, and it took the solver a long time to establish that there is no schedule that satisfies all staffing demands.

Chapter 7

Discussion

The purpose of this thesis was to construct two models for rotating workforce scheduling. The integer model performed very well. Besides giving reasonable schedules, the short solution times were a surprise, since the anticipated times were of the order of hours, not seconds or minutes. The network model, however, required too long solution times, even for the initial attempts, and was therefore not further studied.

A chapter with illustrative examples demonstrated how soft constraints can be used to include additional requirements in the model. The cases were based on constraints commonly used in cyclical schedules, thus giving a good estimate of how the model would perform in real-life scenarios. They also had an increased amount of constraints compared to the basic model, which further tested the integer model's capability.

Case 1, which was the larger of the two cases and included three subcases, provided interesting results. Because of the prevalence of different requirements on the schedule, both longer and shorter work periods were worthwhile to examine. The results in Case 1a, which looked at longer periods, were without doubt adequate. By comparing the resulting schedules with each other, it was possible to see the consequences when adding certain constraints. In Case 1b, shorter periods were instead of interest. The received schedules were undeniably satisfactory, thus proving that the integer model works very well. Case 1c was of a different nature and changed the scheduling process by considering 12 weeks instead of 6 like in previous cases. The purpose of this extension was to see if the increased degrees of freedom would yield schedules with properties that are even better. It was only of interest to look at longer work periods, since the schedules with shorter work periods had no relevant downside. The results were an improvement compared to the schedules received in Case 1a, but the solution times unfortunately increased. However, since solution times in the order of minutes are still considered good, there were no significant drawbacks to extending the schedule length.

Case 2 explored the option of using a basis of two weeks, instead of one, since the staffing demand varied every other week. The produced schedule appeared to be as good as possible with regard to the used constraints. This case also included a experiment to examine the integer model's behaviour when constraints are impossible to satisfy, which led to severely increased solution times.

To conclude, the integer model is a good and valid option for rotating workforce scheduling and appears to have no large disadvantage compared to other models or methods previously developed. A favourable asset is also the ability to easily add or remove constraints to fit the desired scheduling conditions.

7.1 Suggestions for Future Development

There are several opportunities for how the integer model could be developed. First off, it might be worth investigating how the number of possible weekly schedules affects the solution times. Most likely, the growth is non-linear. However, with only the current data at hand, it is impossible to predict anything. The growth could be exponential or logarithmic, or it might be that it is neither of these. Actually, it is possible that the growth is irregular and thus not following any pattern. One the same theme, it could be interesting to look more thoroughly at why the running time for Case 2 changed so dramatically when we altered one constraint.

When solving optimization problems, it is not always obvious if alternative solutions exist or not. We only receive one solution, and thus one schedule. The easiest way to find alternative solutions is probably to forbid the optimal solution found, and then resolve the problem again. This would provide us with a new solution, which we also forbid before resolving. If this process is repeated until an inferior objective value is received, all the alternative optimal solutions have been obtained, thus giving more options for the user when choosing schedule.

A couple of the constraints might gain more desirable properties if they are rewritten, with new parameter definitions. For example, the constraint used in Example 4.2 in Chapter 4 for single days off is not ideal. Recall the parameter e_i, which equals 1 if a weekly schedule contains a single day off. A weekly schedule could however have single days off appearing twice, see Figure 7.1. The constraint will however not take this into consideration, and penalize this property just as much as if the schedule has a single day off once. Hence, there is risk that we receive schedules that are inferior to alternative, but occasionally better, solutions. Fortunately, this problem has not appeared in practice, yet, but the possibility still exists. Thus, refining some of the constraints in the future might be a good idea. It could also be interesting to try different values, as well as order of magnitudes, on the penalty parameters to see if this has any effect on the resulting schedule and solution times.

Mo	Tu	We	Th	Fr	Sa	Su
D		D	D	D		D

Figure 7.1: *Example of a weekly schedule where single days off appear twice.*

Another task would be to develop the network model further. As mentioned in Section 3.2, the solution times were very long with this model. Since there is currently no available software to handle this model in a good way, an idea might be to explore how the model could be modified to provide better solution times. If it possible to rewrite and thus change a few parameter and variable

definitions, the CPLEX-solver could hopefully handle the network structure better. Another possibility is to develop a special-purpose solution method for the network model, by exploiting its inherent properties. An algorithm based on column generation, where a "column" is a "cycle in the network" might be an approachable method.

Bibliography

[1] G. Laporte. The Art and Science of Designing Rotating Schedules. *The Journal of the Operational Research Society*, 50:1011–1017, 1999.

[2] N. Musliu. Heuristic Methods for Automatic Rotating Workforce Scheduling. *International Journal of Computational Intelligence Research*, 2:309–326, 2006.

[3] M. Rocha, J.F. Oliveira and M.A. Carravilla. Cyclical Staff Scheduling: Optimization Models for Some Real-Life Problems. *Journal of Scheduling*, 16:231–242, 2013.

[4] ILOG. ILOG AMPL CPLEX System, User's Guide, 2006.

[5] A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens and D. Sier. An Annotated Bibliography of Personnel Scheduling and Rostering. *Annals of Operations Research*, 127:21–144, 2004.

[6] R.N. Burns and G.J. Koop. A Modular Approach to Optimal Multiple-Shift Manpower Scheduling. *Operations Research*, 35:100–110, 1987.

[7] G. Laporte, Y. Nobert and J. Biron. Rotating schedules. *European Journal of Operational Research*, 4:24–30, 1980.

[8] R. Falcón, E. Barrena, D. Canca and G. Laporte. Counting and Enumerating Feasible Rotating Schedules by means of Gröbner Bases. Mathematics and Computers in Simulation, 2014.

[9] N. Musliu, J. Gärtner and W. Slany. Efficient Generation of Rotating Workforce Scheduling. *Discrete Applied Mathematics*, 118:85–98, 2002.

[10] N. Musliu and M. Mörz. Genetic Algorithm for Rotating Workforce Scheduling Problem. In *Second IEEE International Conference on Computational Cybernetics*, 2004.

[11] N. Musliu. Applying Tabu Search to the Rotating Workforce Scheduling Problem. In *The 5th Metaheuristics International Conference*, 2005.

[12] M. Triska and N. Musliu. A Constraint Programming Application for Rotating Workforce Scheduling. In *Developing Concepts in Applied Intelligence*, 2011.

[13] C. Montoya and G. Mejía. Heuristic Algorithm for Workforce Scheduling Problems. *Brazilian Journal of Operations and Production Management*, 3:35–48, 2006.

[14] G. Laporte and G. Pesant. A General Multi-Shift Scheduling System. *The Journal of the Operational Research Society*, 55:1208–1217, 2004.

[15] K.R. Baker. Scheduling a Full-Time Workforce to Meet Cyclic Staffing Requirements. *Management Science*, 20:1561–1568, 1974.

[16] K.R. Baker. Scheduling Full-Time and Part-Time Staff to Meet Cyclic Staffing Requirements. *Operational Research Quarterly*, 25:65–76, 1974.

[17] B.T. Bennett and R.B. Potts. Rotating Roster for a Transit System. *Transportation Science*, 2:14–34, 1968.

[18] K.R. Baker and M.J. Magazine. Workforce Scheduling with Cyclic Demands and Day-Off Constraints. *Management Science*, 24:161–167, 1977.

[19] S.E. Bechtold. Work Force Scheduling for Arbitrary Cyclic Demands. *Journal of Operations Management*, 1:205–214, 1981.

[20] H.K. Alfares. An Efficient Two-Phase Algorithm for Cyclic Days-Off Scheduling. *Computers and Operations Research*, 25:913–923, 1998.

[21] N. Balakrishnan and R.T. Wong. A Network Model for the Rotating Workforce Scheduling Problem. *Networks*, 20:25–42, 1990.

[22] L.D. Smith. The Application of an Interactive Algorithm to Develop Cyclical Rotational Schedules for Nursing Personnel. *Infor*, 14, 1976.

[23] J. Gärtner, N. Muslija and W. Slany. First Class Scheduler: a System to Generate Rotating Workforce Schedules. Technical Report, Vienna University of Technology, 2000.

[24] P. Chan and G. Weil. Cyclical Staff Scheduling Using Constraint Logic Programming. In *Lecture Notes in Computer Science 2079*, 2000.

[25] J. Lundgren, P. Värbrand and M. Rönnqvist. *Optimeringslära*. Studentlitteratur, Lund, 2011.

[26] N. Musliu. Min Conflict Based Heuristics for Rotating Workforce Scheduling Problem. In *The Sixth Metaheuristics International Conference*, 2005.

LINKÖPINGS UNIVERSITET

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances. The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© 2015, Caroline Granfeldt