NEAR-OPTIMAL STAFF SCHEDULING USING A MIXED INTEGER MODEL

May 2011

Guðríður Lilla Sigurðardóttir
Master of Science in Decision Engineering

NEAR-OPTIMAL STAFF SCHEDULING USING A MIXED INTEGER MODEL

Guðríður Lilla Sigurðardóttir
Master of Science
Decision Engineering
May 2011
School of Science and Engineering
Reykjavík University

M.Sc. RESEARCH THESIS

ISSN 1670-8539

Near-optimal staff scheduling using a mixed integer model

by

Guðríður Lilla Sigurðardóttir

Research thesis submitted to the School of Science and Engineering
at Reykjavík University in partial fulfillment of
the requirements for the degree of
Master of Science in **Decision Engineering**

May 2011

Research Thesis Committee:

Eyjólfur Ingi Ásgeirsson, Supervisor
Assistant professor , School of Science and Engineering, Reykjavík Univeristy

Páll Jensson
Professor, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland

The undersigned hereby certify that they recommend to the School of Science and Engineering at Reykjavík University for acceptance this research thesis entitled **Near-optimal staff scheduling using a mixed integer model** submitted by **Guðríður Lilla Sigurðardóttir** in partial fulfillment of the requirements for the degree of **Master of Science** in **Decision Engineering**.

_____

Date

_____

Eyjólfur Ingi Ásgeirsson, Supervisor
Assistant professor , School of Science and Engineering,
Reykjavík Univeristy

_____

Páll Jensson
Professor, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland

_____

Date

_____

Guðríður Lilla Sigurðardóttir
Master of Science

Near-optimal staff scheduling using a mixed integer model

Guðríður Lilla Sigurðardóttir

May 2011

Abstract

Making a feasible staff schedule is both a time consuming and a difficult task for companies that have employees working on irregular schedules. We introduce a mathematical model, using mixed integer programming, which makes a feasible schedule that fulfills as many of employees requests as possible while satisfying all hard constraints and as many soft constraints as possible. Our goal with this model is to develop a mathematical model that creates feasible staff schedules of high quality. The results from the model are in worst case 10% worse than the optimal solution.

We present the mixed integer programming model and show results from four Icelandic companies and institutions as well as comparing it to a local-search based algorithm. The results show that it is possible to use mathematical programming techniques to make staff schedules, even though it is a problem with multiple and often changing objectives and goals.

Blandað heiltölubestunarlíkan sem skilar lausn sem nálgast bestu lausn fyrir vaktatöflu vandamál

Guðríður Lilla Sigurðardóttir

Maí 2011

Útdráttur

Það er bæði tímafrekt og erfitt verkefni að útbúa vaktaplan fyrir stór fyrirtæki, með starfsfólk sem vinnur á breytilegum vöktum. Í þessari grein kynnum við blandað heiltölulíkan sem býr til vaktaplan sem uppfyllir eins margar óskir starfsmanna um vaktir eins og mögulegt er án þess þó að brjóta skorðurnar sem verða til vegna kjarasamninga og samninga við starfsmenn og með því að brjóta eins fáar aðrar skorður og mögulegt er. Markmiðið með líkaninu er að búa til stærðfræðimódel sem býr til gæða vaktaplan. Í versta falli er lausnin á líkaninu 10% frá bestu lausn.

Heiltölulíkanið sem og helstu niðurstöðurnar frá fjórum íslenskum fyrirtækjum eru kynntar. Niðurstöður gefa til kynna að hægt er að nota stærðfræðilegar aðferðir til að útbúa vaktatöflur, jafnvel þó vaktatöfluvandamálið hafi oft óljóst markfall.

Acknowledgements

Thanks to Vaktaskipan ehf and Eyjólfur Ingi Ásgeirsson for the accessibility to data and results.

Contents

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

2 Literature review **5**

3 Constraints **7**
3.1 Hard constraints 7
3.2 Soft constraints 8

4 Model description **11**
4.1 Basic model 11
4.1.1 Indexes 12
4.1.2 Data 13
4.1.3 Weight factors 14
4.1.4 Penalty variables 15
4.1.5 Binary variables 15
4.1.6 Model 16
4.1.7 Model limitations 18
4.2 Model verification 19
4.2.1 Test case 1 19
4.2.2 Test case 2 20

5 Experimental results **23**
5.1 The nursing home 24
5.2 Call center 1 25
5.3 Call center 2 27
5.4 The airport ground service 28

6 Comparison **31**
6.1 The nursing home 31
6.2 Call center 1 33
6.3 Call center 2 35
6.4 The airport ground service 35
6.5 Summary of the comparisons 36

7 Conclusion **39**

8 Appendix **41**

Bibliography **45**

List of Figures

4.1 Test case 2: Schedule 21

6.1 Comparison of schedules: The nursing home 32
6.2 The nursing home, comparison 32
6.3 Comparison of schedules: Call center 1 33
6.4 Call center 1, comparison 34
6.5 Comparison of schedules: Call center 2 34
6.6 The call center 2, comparison 35
6.7 Comparison of schedules: Airport ground service 36
6.8 The airport ground service, comparison 37

List of Tables

4.1 Test case 1: Data created 20
4.2 Test case 1: Solution 21

5.1 The weight factors for the nursing home 24
5.2 Sum of the penalties for the nursing home 25
5.3 Results for the nursing home 25
5.4 The weight factors for call center 1 26
5.5 Sum of the penalties for call center 1 26
5.6 Results for call center 1 27
5.7 The weight factors for call center 2 27
5.8 Sum of the penalties for call center 2 27
5.9 Results for call center 2 28
5.10 The weight factors for the airport ground service 29
5.11 Sum of the penalties for the airport ground service 29
5.12 Results for the airport ground service 30

Chapter 1

Introduction

The staff scheduling problem is a problem that is known to all companies that are concerned with scheduling workforce to meet demand for manpower that vary within a day and/or within a week. When dealing with a staff scheduling problem, shifts need to be covered by available employees. The main problem in staff scheduling is to determine which employees should cover which shifts so that the demand for manpower is met at every time period and without breaking any regulations or contracts. Making a staff schedule can be both a time consuming and a difficult task. However an inefficient staff scheduling can be very expensive in financial terms, since manpower represents a large component of total operating cost in companies. The main goal in staff scheduling is therefore to efficiently utilize time and effort, to evenly distribute the workload among staff and to attempt to satisfy as many personnel preferences as possible. A development of a robust and powerful staff scheduling system that could handle a wide range of requirements and constraints would provide significant benefits for company's administrators and staff [7]. One of the best known problem in scheduling is concerned with scheduling the working hours for hospital personnel, these problems are known as nurse rostering or nurse scheduling problems.

The nurse rostering problem has been studied by personnel managers, operations researchers and computer scientists for more than 45 years, making it therefore a well known problem. The wide fluctuation in demand that can occur throughout the day and from one day to the next is what makes hospital personnel scheduling so challenging and difficult. Because the nurse rostering problem is well known and since it can include many types of constraints and cover a large set of staff scheduling problems, most of the previous work is focused on nurse rostering even though our problem is not specific for hospital

personnel. As a matter of fact the terms nurse rostering and nurse scheduling have been used over the years to cover several types of personnel scheduling problems [8].

In nurse rostering there are three key approaches: cyclical scheduling, preference scheduling and self scheduling [5]. A cyclical scheduling problem is a scheduling problem in which several sets of schedules are generated that cover a certain period of time i.e. a month or three months. Then the hospital personnel are assigned to a schedule that best fits their preferences so that all demands for manpower are met. The schedules are then repeated for each period making the cyclical scheduling inflexible and therefore not able to adjust rapidly to changes in the environment [14]. The main advantage of the cyclical scheduling is that the personnel knows their schedule a long time in advance.

In preference scheduling hospital personnel list their preferences for the personnel manager who then creates schedules, trying to fulfill as many preferences as possible but also makes sure that all demands for manpower and all work restrictions are met. Thus the personnel manager has a great deal of responsibility for the quality of the schedules. The preference scheduling has many advantages, the major ones being its flexibility and its individual tailoring. However the preference scheduling has a big flaw, it is a very time consuming approach for the personnel manager.

In self scheduling the hospital personnel makes the schedules themselves instead of the personnel manager and are therefore responsible for the schedule. These schedules are created by each employee signing up for their preferred shifts knowing the minimum and maximum number of staff needed for each shift with the requirement that the resulting schedule must be a feasible one. The self scheduling reduces the time involved in creating a schedule. The biggest advantages of the self scheduling beside the time saving are the potentially greater staff satisfaction, more commitment and reduced staff turnover, since the employees are empowered by making the schedules themselves. However until recently self scheduling has not been a good approach since it was too difficult to execute this method fairly [3]. Before the Internet was invented the greatest downside in pure self scheduling was how hard it was to implement it fairly, the order in which the personnel sign up did matter, there was a possibility that the system might get manipulated by some personnel, new employers were unfamiliar with the system and might therefore be disadvantaged and some employees might not sign up for any shifts at all. However with the advent of the Internet it has become easier to implement self scheduling fairly, putting self scheduling back on the map as a possible good approach.

A good way to execute self scheduling is by mixing preference scheduling and pure self scheduling. Here the hospital personnel sign up for shifts, making a draft that the personnel manager then turns into feasible schedule. The personnel manager makes sure that the

demand for manpower is met at every time and that no work regulations are broken. In this approach the personnel is responsible for creating the schedule but the final responsibility of creating the schedule lies with the personnel manager, making this approach better and more fair than either preference scheduling or pure self scheduling. This is the approach used in this paper.

It is often believed that mathematical programming techniques are too rigid to deal with the multiple and often changing, objectives and goals of staff scheduling [8, 16]. However one of our goals is to challenge the belief by using mixed linear programming formulation to determine the optimum staff schedule. The model was developed in MPL and solved using Gurobi 2.0.3 and 4.0.1 Optimization. Two test cases were made to verify the model and then it was tested on four different Icelandic companies and institutions.

The paper is structured as follows: in Chapter 2 there is a literature review of the most relevant work, Chapter 3 covers the problems hard and soft constraints, in Chapter 4 the model is developed and described and two test cases created to show how the model works, in Chapter 5 the result of testing the model with real data is discussed and analyzed, Chapter 6 compares our result with the result from [16] and in Chapter 7 main conclusions, possible areas of extension as well as further research are noted.

Chapter 2

Literature review

The origin of staff rostering and scheduling can be traced back to 1954, to Edie's work on traffic delays at toll booths [11]. Since then staff rostering and scheduling methods have been applied to many fields [4, 6, 15, 17]. Vast amount of literature on personnel scheduling has been made over the years [10, 12, 13].

Ever since staff scheduling was first applied many different methods have been used to solve the staff scheduling problem. These are methods like Mathematical programming, Goal programming, Artificial intelligence methods, Heuristics and Metaheuristic scheduling like Tabu search and Genetic algorithms [8]. Even though many different methods have been and are used to solve staff scheduling problems, only three different key approaches are usually used when solving the staff scheduling problem, these are cyclical scheduling, self scheduling and preference scheduling [5]. The self scheduling approach was first documented in 1963 by Jenkinson [3], however the possibilities of using this approach have increased with the emergence of the computers since it is possible to implement this approach more fairly using computers.

In this paper we use mathematical programming which has not been used much over the years to solve staff scheduling problems since mathematical programming is believed to be too rigid to deal with the multiple and often changing, objectives and goals of staff scheduling. Another possible reason for not using mathematical programming to solve staff scheduling problems is hardware limitations. However throughout time there have been several articles published that have used mathematical programming, this review will be limited to the most relevant work.

Abernathy et al. [1] presented a staff planning and scheduling model that had specific application in the nurse-staffing process in acute hospitals in 1973. That model was solved using mathematical (stochastic) programming techniques. In 1972 Warner and

Prawda [19] developed a model to solve the nursing personnel scheduling problem. They presented a mixed-integer quadratic programming formulation to calculate the number of specifically skilled nurses to undertake a number of shifts per day. Warner improved the previous formulation by adding weights or fairness levels and introduced it in 1976 [18].

Burns and Carter [9] presented a paper in 1985 where they developed lower bounds on the workforce size and then introduced them as an additional constraint in a linear programming model to ensure integer solution. That same year Bailey and Field [2] introduced a general mathematical model for the nurse scheduling problem. This model is still today one of the few published models that allow shifts to start at any time during the day. But since the nurse scheduling is becoming more and more complicated and complex this approach cannot address the current needs of today's hospitals.

Brunner et al. [6] formulated a mixed-integer program for physicians shift scheduling and solved it with the CPLEX optimization software package. Even though physician scheduling and nurse scheduling are similar problems, physician scheduling is more complex than nurse scheduling. In 2010 Ásgeirsson introduced the same problem as this paper addresses using a local-search based algorithm to find a solution [16].

Chapter 3

Constraints

Staff scheduling problems have a large number of constraints that need to be satisfied. Those constraints can be divided into two groups, hard constraints and soft constraints. Hard constraints must always be satisfied in order to have a feasible schedule. Hard constraints are often a result of physical resource restrictions and legislations. Soft constraints are requirements that are desirable but not obligatory and therefore allowed to be violated if necessary but it will result in a penalty in the model. Soft constraints are often used to evaluate the quality of feasible schedules.

3.1 Hard constraints

The hard constraints are mainly based on contracts with the employees and union contracts and must therefore be satisfied at all times. Not all the constraints are the same for all employees although usually the main constraints are the same. The constraints that are usually not the same for all employees are the work limit constraints. In the model the following hard constraints are considered:

- **Restrictions on working hours and rest periods from union regulations and employee contracts.** The union regulations about rest periods, maximum lengths of continuous work within a day, maximum number of continuous days worked, minimum length of continuous rest between shifts and other limits have to be met. Employee contracts can include restrictions on when employee can work, for example an employee that will never work nights or weekends.
- **Vacation request.** This needs to be a hard constraint so employees will never be assigned to a shift while they are on vacation.

- **Requests for time off.** Each employee has a right to some time off, how many hours depending on the company and the employee contract. In our settings this needs to be a hard constraint so these requests won't be violated.
- **Working weekends.** There can be limits on how many weekends employees are allowed to work in each scheduling period. Each employee has to receive at least A out of every B weekends off, where A $\leq$ B. These are limits like 2 or 3 weekends off out of every 4 consecutive weekends.
- **Special shifts, training sessions or meetings.** Employees often have work related duties that are not flexible and are often not included in the number of employees on duty. Since training sessions and meetings are not flexible these constraints must be satisfied.
- **Other limits on shifts or working hours, for example split shifts.** Split shifts are defined as two separate shifts within the same day, where the time between the shifts is less than the minimum resting period between shifts. It can differ between companies whether splits shifts are allowed or not. Splits shifts are only hard constraints when split shifts are not allowed.

Each company is different when it comes to number of employees, contracts, habits and regulations, therefore the constraints differs from one company to another. Each company wants to be able to quickly generate a high quality schedule that satisfies all hard constraints and as many of the soft constraints as possible.

3.2 Soft constraints

Each time a soft constraint is violated the schedule receives a penalty that appears in the objective function. How high the total penalty is depends on which constraints are violated and how often they are violated. The penalties have different weight factors, depending on how serious a violation of the relevant constraint would be. In the model the following soft constraints are considered:

- **Minimum and maximum staff level.** An estimate of the demand for manpower at every time slot over the whole period the schedule is supposed to cover is necessary. This estimate can vary greatly between companies depending on how good their forecast for the demand of manpower is. Some companies use minimum and maximum staff level for every time slot while others give exact number of employees needed for every time slot. We want the on-duty employees in the schedule

to be between the minimum and maximum staff level or as close as possible to the exact number of required on-duty employees, otherwise the schedule will be penalized.

- **Minimum and maximum number of on-duty hours for each employee.** In every employee contract a number of required on-duty hours are given. However since the employees are often working irregular hours, there must be some flexibility in required on-duty hours for each scheduling period. Therefore the required on-duty hours are interpreted as minimum and maximum number of on-duty hours for each employee. Minimum and maximum numbers of on-duty hours for each employee are calculated based on monthly working hours given in the contracts and accumulated deviations from the required on-duty hours from the previous period.
- **Employee requests for shifts.** The first step in making a schedule is to make each employee signs up for their preferred shifts knowing the minimum and maximum number of staff needed for each shift. This encourages employees to create their own work schedule and makes the schedule more acceptable for the employees. It is therefore important to meet as many requests as possible.
- **Employees assigned to shifts on weekends contiguous to their vacations.** If an employee is finishing his vacation on Friday or beginning his vacation on Monday it is unlikely he wants to work the adjacent weekend. So unless otherwise requested we will try to have the adjacent weekend free.

Chapter 4

Model description

4.1 Basic model

When a mathematical model is used to approximate a real-life problem like in this paper it is essential that a manager goes through the result from the model and makes changes to the schedule if necessary. An example why this is essential is for example if an incident occurs where too few employees are available. In this case the personnel manager needs to be able to deal with some of the unavailable employees to be available and on-duty so there will be enough manpower. Here the personnel manager needs to change the schedule by hand because the model can't deal with the employees, it can only identify whether they are available or not. The personnel manager also has to be able to connect past, present and future staffing schedules.

The objective of the mixed integer model is to minimize the penalties that occur if soft constraints are broken while satisfying all the hard constraints. To formulate the problem, five sets of binary decision variables are used. The first set of variables tells whether an employee is working a shift or not. Let

$$x_{ijk} = \begin{cases} 1, if\ employee\ i\ works\ shift\ j\ starting\ on\ day\ k; \forall\ i \in I, j \in J, k \in K \\ 0, otherwise \end{cases} \tag{4.1}$$

where I is the set of employees, J is the set of available shifts and K the set of days.

In the model the time horizon is divided into days and each day is further divided into fixed length time periods. The length of the time periods can be different from one company to the next, the length might be a quarter, a half an hour or an hour. The second set of variables indicates whether an employee is working in the period t on day k or not.

Let

$$y_{itk} = \begin{cases} 1, if\ employee\ i\ works\ period\ t\ on\ day\ k\ ; \forall\ i \in I, t \in T, k \in K \\ 0, otherwise \end{cases} \tag{4.2}$$

where T is the set of time periods.

Union regulations and employees contracts have some restrictions on rest periods between shifts, these rest periods are expressed as rest shifts. The purpose of the rest shifts is to act as a witness that the regulations on rest periods are being fulfilled. It is therefore essential to have a variable that tells us whether an employee is taking a rest shift or not, that would be the third set of variables. Let

$$z_{ilk} = \begin{cases} 1, if\ employee\ i\ has\ a\ rest\ shift\ l\ starting\ on\ day\ k; \forall\ i \in I, l \in L, k \in K \\ 0, otherwise \end{cases} \tag{4.3}$$

where L is the set of available rest shifts.

Employees are only allowed to work a certain number of consecutive days and therefore we need a variable that tells us whether an employee is working on a day or not. The fourth set of variables indicates whether an employee is working on a particular day or not. Let

$$d_{ik} = \begin{cases} 1, if\ employee\ i\ works\ on\ day\ k; \forall\ i \in I, k \in K \\ 0, otherwise \end{cases} \tag{4.4}$$

There are some regulation concerning how many weekends employees are allowed to work. It is therefore not enough only to know whether an employee is working a on day or not, we need to have the fifth set of variables that tells what weekends employees are working if any. Let

$$w_{i,\omega} = \begin{cases} 1, if\ employee\ i\ is\ working\ on\ the\ weekend\ that\ starts\ on\ Saturday\ \omega\ ; \\ \quad \forall i \in I, \omega \in W \\ 0, otherwise \end{cases} \tag{4.5}$$

where W is the set of Saturdays in the scheduling period.

4.1.1 Indexes

The model uses the following 7 set of indexes to represent employees, shifts, days, time periods, rest shifts, weekends and consecutive days.

I: set of employees

J: set of shifts, $J = \{t_1, t_2, ..., t_{|J|}\}\ where\ t_j \in T, j = 1, 2, ..., |J|$

K: set of days

T: set of time periods

L: set of rest shifts, $L = \{t_1, t_2, ..., t_{|L|}\}\ where\ t_l \in T, l = 1, 2, ..., |L|$

W: set of weekends, $K \supseteq W = \{k_6, k_{13}, ..., k_{|W|}\}\ where\ k_\omega \in K, \omega = 1, 2, ..., |W|$

C: set of consecutive days

4.1.2 Data

The following 7 data sets are used as inputs in the model.

$available_{itk}$ 1 if employee $i \in I$ is available at time period $t \in T$ on day $k \in K$, 0 otherwise

$availableonshift_{ijk}$ 1 if employee $i \in I$ is available to work shift $j \in J$ beginning on day $k \in K$, 0 otherwise

$requestshifts_{itk}$ 1 if employee $i \in I$ requests working time period $t \in T$ on day $k \in K$, 0 otherwise

$shifts_{jkt}$ connects shifts and time

$restshifts_{lkt}$ connects rest shifts and time

$weekend_{\omega k}$ connects weekends (Saturdays) and days

$kdays$ needs to be calculated as: number of days k - maximum number of consecutive days

Right hand side

In the model we use the following 7 parameters.

$demand_{tk}^{min}$ minimum number of employees required at time period t in day k; $for\ t \in T, k \in K$

$demand_{tk}^{max}$ maximum number of employees required at time period t in day k; $for\ t \in T, k \in K$

$time_i^{min}$ minimum number of time periods employee i should work in the scheduling period; $for\ i \in I$

$time_i^{max}$ maximum number of time periods employee i should work in the scheduling period; $for\ i \in I$

$timewhithinaday_{ik}^{max}$ maximum number of time periods employee i can work within day k; $for\ i \in I, k \in K$

$weekends_i^{max}$ maximum number of weekends employee i can work in the scheduling period; $for\ i \in I$

$days_i^{max}$ maximum number of consecutive days; $for\ i \in I$

4.1.3 Weight factors

The penalties each have a different weight factor depending on how severe it is to break the relevant constraint. These weight factors are marked with numbers corresponding to the relevant constraint in the model.

$c4.7$: Weight factor if there are too few employees at any time period

$c4.8$: Weight factor if there are too many employees at any time period

$c4.14$: Weight factor if there are too few on duty hours in the schedule for any employee

$c4.15$: Weight factor if employees are below the minimum on duty hours

$c4.16$: Weight factor if there are too many on duty hours in the schedule for any employee

$c4.17$: Weight factor if there are too many on duty hours within a day for any employee

$c4.18$: Weight factor if we do not fulfill a request for a shift

$c4.19$: Weight factor if there are too many shifts within a day for any employee

$c4.22$: Weight factor if there are too many on duty weekends in the schedule for any employee

The weight factors all have different weights depending on how serious a violation of the relevant constraint is. The lowest weight is on the requests because we want to fulfill as many of them as possible but it is not necessary. The highest weights are high compared to the other weights making it very important not to violate the corresponding constraints , these constraints are the constraints we classify as hard constraints. By putting a penalty

on hard constraints we are able to monitor if hard constraints are broken and which hard constraints are broken if any.

4.1.4 Penalty variables

$p4.7_{tk}$: penalty because of too few employees at time period $t \in T$ at day $k \in K$

$p4.8_{tk}$: penalty because of too many employees at time period $t \in T$ at day $k \in K$

$p4.14_i$: penalty because an employee $i \in I$ works too few hours in the scheduling period

$p4.16_i$: penalty because an employee $i \in I$ works too many hours in the scheduling period

$p4.17_{ik}$: penalty because an employee $i \in I$ works too many hours within day $k \in K$

$p4.18_{itk}$: penalty because an employee's $i \in I$ request for a shift $j \in J$ in day $k \in K$ is not fulfilled

$p4.19_{ik}$: penalty because an employee $i \in I$ is working more than one shift in day $k \in K$

$p4.22_i$: penalty because an employee $i \in I$ is working too many weekends

A reference number is connected to each constraint, making it easy to refer to both the constraints and penalties. Both the factors and the penalties are marked with numbers corresponding to the relevant reference number in the model.

4.1.5 Binary variables

In the model there are 6 binary variables five of which are covered in Section 4.1 but one is a penalty to count how many employees go under the minimum on duty hours.

x_{ijk}: 1 if employee $i \in I$ works shift $j \in J$ beginning on day $k \in K$, 0 otherwise

y_{itk}: 1 if employee $i \in I$ works time period $t \in T$ on day $k \in K$, 0 otherwise

z_{ilk}: 1 if employee $i \in I$ has a rest shift $l \in L$ beginning on day $k \in K$, 0 otherwise

d_{ik}: 1 if employee $i \in I$ starts working on day $k \in K$, 0 otherwise

$w_{i\omega}$: 1 if employee $i \in I$ is working on the weekend that starts on Saturday $\omega \in W$, 0 otherwise

$p4.15_i$: 1 if employee $i \in I$ is below the minimum duty hours in the scheduling period, 0 otherwise

4.1.6 Model

The following mixed integer model was developed.

$$\begin{aligned}
\min \quad & c4.7 \times \sum_{t\in T}\sum_{k\in K} p4.7_{tk} + c4.8 \times \sum_{t\in T}\sum_{k\in K} p4.8_{tk} + c4.14 \times \sum_{i\in I} p4.14_i \\
& + c4.15 \times \sum_{i\in I} p4.15_i + c4.16 \times \sum_{i\in I} p4.16_i + c4.17 \times \sum_{i\in I}\sum_{k\in K} p4.17_{ik} \\
& + c4.18 \times \sum_{i\in I}\sum_{t\in T}\sum_{k\in K} p4.18_{itk} + c4.19 \times \sum_{i\in I}\sum_{k\in K} p4.19_{ik} \\
& + c4.22 \times \sum_{i\in I} p4.22_i
\end{aligned} \tag{4.6}$$

$$\text{s.t.} \quad \sum_{i\in I} y_{itk} \geq demand^{min}_{tk} - p4.7_{tk} \quad \forall t \in T, k \in K \tag{4.7}$$

$$\sum_{i\in I} y_{itk} \leq demand^{max}_{tk} + p4.8_{tk} \quad \forall t \in T, k \in K \tag{4.8}$$

$$y_{itk} = \sum_{j\in shifts} x_{ijk} \quad \forall i \in I, t \in T, k \in K \tag{4.9}$$

$$y_{itk} \leq 1 - z_{ilk} \quad \forall i \in I, t \in T, l \in L, k \in K \; where \; t \in L \tag{4.10}$$

$$\sum_{l\in L} z_{ilk} = 1 \quad \forall i \in I, k \in K \tag{4.11}$$

$$y_{itk} \leq available_{itk} \quad \forall i \in I, t \in T, k \in K \tag{4.12}$$

$$x_{ijk} \leq availableonshift_{ijk} \quad \forall i \in I, j \in J, k \in K \tag{4.13}$$

$$\sum_{t\in T}\sum_{k\in K} y_{itk} \geq time^{min}_i - p4.14_i \quad \forall i \in I \tag{4.14}$$

$$\sum_{t\in T}\sum_{k\in K} y_{itk} \geq time^{min}_i * (1 - p4.15_i) \quad \forall i \in I \tag{4.15}$$

$$\sum_{t\in T}\sum_{k\in K} y_{itk} \leq time^{max}_i + p4.16_i \quad \forall i \in I \tag{4.16}$$

$$\sum_{t\in T} y_{itk} \leq timewhithinaday^{max}_{ik} + p4.17_{ik} \quad \forall i \in I, k \in K \tag{4.17}$$

$$requestshift_{itk} - y_{itk} \leq p4.18_{itk} \quad \forall i \in I, t \in T, k \in K \tag{4.18}$$

$$\sum_{j\in J} x_{ijk} = d_{ik} + p4.19_{ik} \quad \forall i \in I, k \in K \tag{4.19}$$

$$x_{ijk} \leq d_{ik} \quad \forall i \in I, j \in J, k \in K \tag{4.20}$$

$$\sum_{c \in C} d_{ik+c} \leq days_i^{max} \quad \forall i \in I, k \in K where\ k < kdays \tag{4.21}$$

$$d_{ik} + d_{i(k+1)} = 2 * w_{i\omega} \quad \forall i \in I, k \in K, \omega \in W\ where\ k \in W \tag{4.22}$$

$$\sum_{\omega \in W} w_{i\omega} \leq weekends_i^{max} + p4.22_i \quad \forall i \in I \tag{4.23}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in I, j \in J, k \in K \tag{4.24}$$

$$y_{itk} \in \{0,1\} \quad \forall i \in I, t \in T, k \in K \tag{4.25}$$

$$z_{ilk} \in \{0,1\} \quad \forall i \in I, l \in L, k \in K \tag{4.26}$$

$$d_{ik} \in \{0,1\} \quad \forall i \in I, k \in K \tag{4.27}$$

$$w_{i\omega} \in \{0,1\} \quad \forall i \in I, \omega \in W \tag{4.28}$$

$$p4.15_i \in \{0,1\} \quad \forall i \in I \tag{4.29}$$

$$p4.7_{tk}, p4.8_{tk}, p4.14_i, p4.16_i, p4.17_{ik}, p4.18_{itk}, p4.19_{ik}, p4.22_i \geq 0 \tag{4.30}$$

$$\forall i \in I, t \in T, k \in K$$

$$\tag{4.31}$$

The objective function (4.6) minimizes the sum of all the penalties. The penalties each have a different weight factor depending on how serious it is to break the relevant constraint. The first constraints, (4.7) and (4.8), make sure that the number of on-duty employees is as close as possible to the estimated demand for manpower at every time period as in minimum and maximum number of on-duty hours for each employee soft constraint. Constraints (4.9) and (4.10) tie together time periods and shifts, both normal shifts and rest shifts. A connection between variables x and y is necessary so that employees won't be assigned to multiple shifts at the same time. The same applies for rest shifts, a connection between z and y is necessary so that an employee will not be assigned to work on a shift that overlaps the rest shift. Every employee has to have one rest shift every day to make sure the employee get the mandatory rest, like in the hard constraint restrictions on working hours and rest periods from union regulations and employee contracts, constraint (4.11) defines that mandatory rest. An employee can't be assigned to a shift while on a vacation or taking a time off as in the hard constraints vacations request and requests for time off. Information about vacations and times off are documented in the data sets *available-on-shift* and *available* respectively. Constraints (4.12) and (4.13) assign vacations and times off to the schedule. Each employee contract features information about minimum and maximum time periods the employee can work in the scheduling period. In constraints (4.14) and (4.16) the soft constraints about minimum and maximum time periods for each employee is met. To minimize the number of employees that are below the minimum time periods constraint (4.15) is essential. While constraint (4.14) penalizes for every time period below the minimum on duty hour, constraint (4.15) penalizes for every employee that is

below the minimum on duty hour. Constraint (4.17) ensures that an employee does not work too much within a day.

Employees make requests about which hours they prefer to work as in the soft constraint employee requests for shifts and that information is saved in the dataset *requestshift*. We want to meet as many requests as possible and therefore assign a penalty if a request is not met, constraint (4.18). We decided that instead of employees requesting special shifts, the employees request the time periods included in the shifts the employees want to work. This method is thought to be more fair since we might be able to fulfill some of the time periods in the requested shift but not all. For example if an employee requests a 8-16 shift but we assign that employee to a 9-17 shift then we are not fulfilling the request however if an employee requests all the time periods between 8 and 16 and we assign him to a 9-17 shift we are fulfilling 7 out of 8 requests. Making requests about time periods instead of shifts is not always more fair, however for the collaborating companies and institutions in this paper, requesting time periods instead of shifts is considered appropriate. Nonetheless it would be easy to change the model so it would consider requested shifts instead of requested hours. Constraint (4.19) records whether an employee is working a particular day or not and assigns penalty if employee has more than one shift in any day. If split shifts are allowed another constraint, constraint (4.20) is needed to record whether an employee is working a particular day or not. Employees are only allowed to work a specific number of consecutive days as in the hard constraint restrictions on working hours and rest periods from union regulations and employee contracts, constraint (4.21). If an employee is working on a weekend then that employee will be assigned to a shift both on Saturday and Sunday since employees are only allowed to work a specific number of weekends in the scheduling period and those weekends are defined as working both Saturday and Sunday. Constraint (4.22) makes certain that employees either work both Saturday and Sunday or neither Saturday nor Sunday, as well as documenting whether an employee is working a weekend or not. The hard constraint working weekends has limits on how many weekends employees can work in the scheduling period, constraint (4.23). Constraints (4.24), (4.25), (4.26), (4.27), (4.28) define x, y, z, d and w as binary variables. Constraint (4.29) makes sure that (4.15) is a binary variable and constraint (4.30) that the rest of the penalties can not be negative.

4.1.7 Model limitations

The model is not without limitations. For example the staff scheduling problem is a problem that has a fuzzy objective and therefore it is difficult to use mathematical programm-

ing to solve it, since the mathematical programming does not incorporate any fuzziness. Another example about the models limitations is the fairness, every employee should be treated the same in the model, however, since there is a penalty for every employee that goes under the minimum on duty hours the model tends to fulfill as many minimum on duty hour constraints as possible so the employees not fulfilling their on duty hours might therefore be missing a lot of hours. This might create a unfairness for those employees that go below the minimum on duty hours, the staff manager needs to keep an open eye out for this. Another limitation is that the problem is NP-hard so the time that it takes to find the optimal solution is exponential in the model size. We had to take this time problem under consideration so instead of making the model find the optimal solution, the model finds a solution that is in worst case 10% worse than the optimal solution, see details in Chapter 5. Another way to deal with this time problem is to split the scheduling period up to smaller scheduling periods or make the time periods within a day longer. Another problem with the time is the boundary problem, for example are employees only allowed to work a certain number of consecutive days but when going from one scheduling period to the next these constraints might get broken. Since our goal is to show that it is possible to formulate a mathematical model that works we do not take this under considerations. However it would be possible to solve this problem but to do so we would have to add constraints and/or change parameters.

4.2 Model verification

4.2.1 Test case 1

A very small test case was created to check whether the model worked properly. The test case consists of four employees who are suppose to cover four shifts that span two adjacent days where each day has four time periods. Each employee is only allowed to work max 2 time periods per day and max 4 time periods over the whole scheduling period. However each employee must work at least 2 time periods over the whole scheduling period. Due to the simplicity of test case 1, constraints (4.19),(4.21), (4.22) and (4.23) are not used. This model has 168 constraints and it uses 140 continuous variables and 96 binary variables. The matrix density is 0.018 and the number of objective function coefficients is 60. All the weight factors in test case 1 are equal to 1. Table 4.1 contains information about the minimum and maximum expected demands for manpower for every time period, as well as information about when the employees are available and what shifts they want to work.

$Day, k:$	1				2			
$Time, t:$	1	2	3	4	1	2	3	4
$Shift, j:$	1		2		1		2	
$Restshift, l:$	1	2	3	4	1	2	3	4
$demand^{min}:$	1	1	2	2	2	1	1	1
$demand^{max}:$	2	2	3	3	2	2	2	1
$Available\ time\ periods,\ employee\ 1:$	x		x	x	x	x		x
$Available\ time\ periods,\ employee\ 2:$	x	x	x	x	x	x	x	x
$Available\ time\ periods,\ employee\ 3:$	x		x	x		x	x	x
$Available\ time\ periods,\ employee\ 4:$		x	x		x	x	x	x
$Requested\ shifts,\ employee\ 1:$			x		x			
$Requested\ shifts,\ employee\ 2:$	x				x			
$Requested\ shifts,\ employee\ 3:$			x				x	
$Requested\ shifts,\ employee\ 4:$			x				x	

Table 4.1: Test case 1: Data created

The model was solved using the Gurobi Optimizer 4.0.1 solver which is supported in the MPL modeling system. The solutions for test case 1 are stated in Table 4.2. For test case 1 it can be seen that if an employee is working a shift then the employee is not available in those time periods the shift covers. When an employee is not assigned to a shift than the employee can be assigned to a rest shift. The only constraint that is broken is the constraint concerning the requested shifts, since employee 3 and 4 both requested to work two shifts but only got one each. The value of the objective function in test case 1 is therefore two times the weight for the penalty in constraint (4.18).

4.2.2 Test case 2

Test case 1 only has two adjacent days and therefore the constraint telling how many weekends the employees can work is not relevant. To test whether the weekend constraints worked properly test case 2 was created. Test case 2 consists of six employees who are supposed to cover twenty eight shifts that span fourteen adjacent days, or two weeks, where each day has four time period and the first day of the schedule is Monday, see Figure 4.1. This second model has 1906 constraints and it uses 1570 continuous variables and 936 binary variables. The matrix density is 0.0017 and the number of objective function coefficients is 634. The only constraints violated in test case 2 are the request constraints (4.18). In test case 2 it is not effective to keep all the weight factors equal to 1, since if we violate a request for one weekend it would result in penalty of 4 (one shift or two time periods in Saturday and one shift or two time periods in Sunday) however if we assign an employee to one too many weekends it would only result in penalty of 1. But the

$Day, k:$	1				2			
$Time, t:$	1	2	3	4	1	2	3	4
$Shift, j:$	1		2		1		2	
$Restshift, l:$	1	2	3	4	1	2	3	4
$On\ duty\ hours,\ employee\ 1:$			x	x	x	x		
$On\ duty\ hours,\ employee\ 2:$	x	x			x	x		
$On\ duty\ hours,\ employee\ 3:$			x	x				
$On\ duty\ hours,\ employee\ 4:$							x	x
$On\ duty\ shifts,\ employee\ 1:$			x		x			
$On\ duty\ shifts,\ employee\ 2:$	x				x			
$On\ duty\ shifts,\ employee\ 3:$			x					
$On\ duty\ shifts,\ employee\ 4:$							x	
$Restshiftemployee\ 1:$	x						x	
$Restshiftemployee\ 2:$			x				x	
$Restshiftemployee\ 3:$	x				x			
$Restshiftemployee\ 4:$	x				x			

Table 4.2: Test case 1: Solution

request constraints have the lowest significance in the model therefore the weight factor for request constraints needs to be lower than the ones for too many weekends. In test case 2 weight factor c4.18 is 2 and all the others are 10 making the value of the objective function 56, since 28 requests are not met.

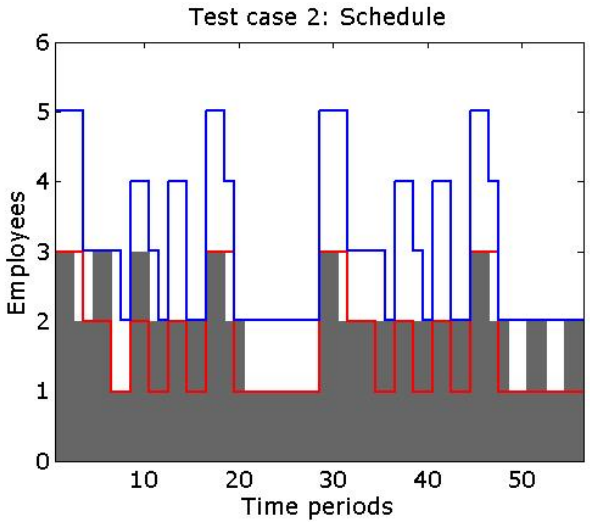

Figure 4.1: The schedule for test case 2. The blue line denotes the maximum required staff on duty at each time and the red line the minimum required staff, the gray area denotes the number of employees on duty.

Chapter 5

Experimental results

Actual data from four Icelandic companies and institutions are used to evaluate the performance and quality of the model. These companies and institutions are a nursing home, two call centers and an airport ground service company. Since we have real data from two call centers we will refer to them as call center 1 and call center 2 in this paper.

When solving an integer model some parameters can be changed to get a faster outcome at the expense of getting a worse outcome. One parameter is the relative gap. The formula for the relative gap is the absolute value of the best solution minus the best bound and then that outcome is divided by the best bound,

$$|best\ solution - best\ bound|/best\ bound \tag{5.1}$$

The MIP search will stop when the relative gap is less than or equal to the criterion value, the default value of the criterion value is 0.0. The criterion level we use for the experimental data is 0.1, that is our result will in worst case be 10% worse than the optimal solution. When solving a model most of the time often goes into going from a good solution to the optimal solution or proving that the current solution is indeed optimal, therefore we will get a faster outcome using a 10% relative gap. Another parameter is integrality but the integrality tolerance specifies the amount by which an integer variable can be different than an integer and still be considered feasible. The integrality we use for our data is 0.001.

Different schedules can be obtained by changing the values of the weight factors. The combinations we choose use a very low weight factor for not fulfilling requests since requests are only preferable but not an obligation. Minimum demand, maximum demand, minimum on duty hours and maximum on duty hours are soft constraints and all have low

weight factors. However the weight factor for the minimum demand is slightly higher than the others since it is least desirable to be understaffed and the weight factor for the maximum demand is a bit lower since we are more willing to be overstaffed than having employees working too few hours. The weight factor for number of employees below the minimum on duty hours is higher since we want to have as few employees under the minimum on duty hours as possible. The weight factors for too many hours worked in one day, more than one shift worked within a day and more weekends worked than the maximum weekends allow are very high because if these constraints are broken then the model is either not fulfilling the union regulations or an employee contract and we are obligated to fulfill all the union regulations and all the employee contracts.

In Sections 5.1, 5.2, 5.3 and 5.4 each company and institution will be covered and discussed separately.

5.1 The nursing home

The first data set is from a nursing home with 55 employees. Out of these there are 50 employees that are used as inputs in the optimization, the other 5 employees have fixed shifts and are therefore not included in the optimization. The scheduling period is 6 weeks or 42 days with 30 minute intervals or 48 time periods each day. In the schedule there are 18 possible shifts per day where the length of each shift varies from 4 hours to 8.5 hours.

The weight factors for the nursing home are recorded in Table 5.1 and the sum of the penalties are recorded in Table 5.2. With these weight factors the value of the objective function is in our solution 5602 and it took approximately 21 seconds to solve the model.

c4.7	c4.8	c4.14	c4.15	c4.16	c4.17	c4.18	c4.19	c4.22
15	2	10	100	10	10,000	1	10,000	10,000

Table 5.1: The weight factors for the nursing home where the numbers correspond to the relevant constraint. We are obligated to fulfill c4.17, c4.19 and c.4.22 and therefore these weight factors are really high. c4.18 is preferable but not an obligations and is therefore very low. And c4.7, c4.8, c4.14, c4.15 and c4.16 all have low weight factors but the difference is because of different importances in fulfilling these constraints.

For the nursing home the following hard constraints are considered. An employee cannot work on more than 6 consecutive days. The minimum length of a shift is 4 hours and the maximum length of a shift is 8.5 hours. The maximum number of working hours

p4.7	p4.8	p4.14	p4.15	p4.16	p4.17	p4.18	p.4.19	p4.22
218	0	3	1	8	0	2122	0	0

Table 5.2: Sum of the penalties for the nursing home where the numbers correspond to the relevant constraint. p4.7, p4.8, p4.14, p4.16, p4.17 and p4.18 are all based on 30 minutes intervals or 48 time periods each day.

in any 24 hour period is 9 hours and each employee must get a rest period of at least 8 consecutive hours in any 24 hour period.

The main results for the nursing home are summarized in Table 5.2. About 6% of the time the nursing home is understaffed and the nursing home is never overstaffed. There is only one employee that does not fulfill his minimum working hours, but this employee is only 1.5 hours below his minimum requirement. There are four employees that are over the maximum working hours, two employees are 0.5 hours over, one employee is 1 hour over and one employee is 2 hours over the maximum working hours in the scheduling period. Union regulations and employees contracts about restrictions on working hours and rest periods and working weekends are always met. Finally more than 98% of the requests made by the employees in the optimization are met.

Scheduled man-hours	4774
Man-hours understaffed	109
Man-hours overstaffed	0
Employees below minimum duty hours	1
Employees over the maximum duty hours	4
Employees over the maximum duty hours within a day	0
Employees working more than one shift per day	0
Employees working more then the maximum weekends	0
Requests met	108,758
Requests not met	2,122

Table 5.3: Results for the nursing home: A summation of the main results for the nursing home.

5.2 Call center 1

The second data set is from the first call center. Call center 1 has 92 employees out of which 74 are used as inputs in the optimization. The scheduling period is 6 weeks or 42 days with 30 minute intervals. In the scheduling period there are 4074 different shifts that employees in the optimization are allowed to work. The length of each shift ranges from 4 hours up to 8.5 hours.

Table 5.4 records the weight factors for call center 1 and the sum of the penalties are recorded in Table 5.5. The solution time is around 10 minutes and with these weight factors the value of the objective function is 13,358.

c4.7	c4.8	c4.14	c.4.15	c4.16	c4.17	c4.18	c4.19	c4.22
15	2	10	100	10	10,000	1	10,000	10,000

Table 5.4: The weight factors for call center 1 where the numbers correspond to the relevant constraint. We are obligated to fulfill c4.17, c4.19 and c.4.22 and therefore these weight factors are really high. c4.18 is preferable but not an obligations and is therefore very low. And c4.7, c4.8, c4.14, c4.15 and c4.16 all have low weight factors but the difference is because of different importances in fulfilling these constraints.

p4.7	p4.8	p4.14	p4.15	p4.16	p4.17	p4.18	p.4.19	p4.22
156	80	606	10	0	0	3,798	0	0

Table 5.5: Sum of the penalties for call center 1 where the numbers correspond to the relevant constraint. p4.7, p4.8, p4.14, p4.16, p4.17 and p4.18 are all based on 48 time periods each day.

Call center 1 has the following hard constraints, an employee cannot work on more than 6 consecutive days, where the maximum number of working hours in each 24 hour period is 9 hours. In any 24 hour period, each employee must get at least 11 consecutive hours of rest.

Table 5.6 summarizes the main results for call center 1. Call center 1 is understaffed about 2.2% of the scheduling period and overstaffed about 2.6% of the scheduling period. There are 10 employees below the required minimum on duty hours but out of these 10 employees there are 5 employees that are taking a vacation and are therefore not available but have some minimum on duty hours they are suppose to meet. Out of the 5 remaining employees, two employees are only one hour below the minimum on duty hours, one employee is 14.5 hours below, one is 29 hours below and the last one is 37 hours below the minimum on duty hours. The employee with 37 hours below the minimum on duty hours does not have enough available time periods to fulfill his requirements about minimum on duty hours and has therefore so many hours below the requirements. No employee is working more than the maximum on duty hours. All union regulations and employees contracts about restrictions on working hours, rest periods and working weekends are fulfilled. The percent of requests met in the result is skewed since we are not able to fulfill the requests made by employees requesting shifts while they are on a vacation. We are not able to fulfill these requests since we are only able to fulfill requests if employees are available and employees on a vacation are not available. We are able to fulfill about 79% of requests made by employees in the optimization.

Scheduled man-hours	10,577.5
Man-hours understaffed	78
Man-hours overstaffed	40
Employees below minimum duty hours	10
Employees over the maximum duty hours	0
Employees over the maximum duty hours within a day	0
Employees working more than one shift per day	0
Employees working more then the maximum weekends	0
Requests met	14,310
Requests not met	3,798

Table 5.6: Results for call center 1: A summation of the main results for call center 1.

5.3 Call center 2

The third data set is from the second call center. Call center 2 has 62 employees and out of these only 46 are used as inputs in the optimization. The scheduling period is only 31 days but with 15 minute intervals or 96 time periods every day. Call center 2 specifies the exact number of employees that should be on-duty at each time instead of minimum and maximum number of employees that should be on-duty in each interval. The minimum length of a shift is 4 hours while the maximum length of a shift is 11 hours.

Call center 2 has the weight factor given in Table 5.7 and the sum of the penalties in Table 5.8. With these weight factors it took 3 hours 47 minutes and 9 seconds to solve the model and the software was very vulnerable while solving the model, that is it would report an error and need to close if interfered with during execution. With the given weight factors the value of the objective function is 24,299.

c4.7	c4.8	c4.14	c.4.15	c4.16	c4.17	c4.18	c4.19	c4.22
15	2	10	100	10	10,000	1	10,000	10,000

Table 5.7: The weight factors for call center 2 where the numbers correspond to the relevant constraint. We are obligated to fulfill c4.17, c4.19 and c.4.22 and therefore these weight factors are really high. c4.18 is preferable but not an obligations and is therefore very low. And c4.7, c4.8, c4.14, c4.15 and c4.16 all have low weight factors but the difference is because of different importances in fulfilling these constraints.

p4.7	p4.8	p4.14	p4.15	p4.16	p4.17	p4.18	p.4.19	p4.22
149	936	1096	17	0	0	7,532	0	0

Table 5.8: Sum of the penalties for the call center 2 where the numbers correspond to the relevant constraint. p4.7, p4.8, p4.14, p4.16, p4.17 and p4.18 are all based on 15 minutes intervals or 96 time periods each day.

The hard constraints for call center 2 are that employees cannot work more than 6 consecutive days, in every 24 hour period there must be a rest period of at least 11 consecutive hours for each employee and each employee can work at most 11 hours of work in every 24 hour period.

The main results for call center 2 are summarized in Table 5.9. Around 3.7% of the time call center 2 is understaffed and around 21.8% of the time it is overstaffed. 17 employees are below their minimum on duty hours and two of those 17 employees are on a vacation but have minimum on duty hours that they are suppose to fulfill. Five of the remaining 15 employees are less than 4 hours below the minimum on duty hours, four employee are between 4 and 14 hours below, two employees are between 14 and 24 hours below and the remaining four are between 37 and 50 hours below the minimum on duty hours. All employees are below the maximum on duty hours and all the restrictions from union regulations and employee contracts about maximum working hours, minimum rest and maximum weekends worked are met. About 65% of the requests made by employees in the optimization are fulfilled. A lot of employees are requesting the same time periods making the number of employees requesting to be on duty more than the maximum on duty demand. This results in many requests being discarded so the model can rather fulfill the maximum on duty demand constraint than the requests constraints and therefore the percent of requests met is so low.

Scheduled man-hours	6,071
Man-hours understaffed	37.25
Man-hours overstaffed	234
Employees below minimum duty hours	17
Employees over the maximum duty hours	0
Employees over the maximum duty hours within a day	0
Employees working more than one shift per day	0
Employees working more then the maximum weekends	0
Requests met	13,961
Requests not met	7,532

Table 5.9: Results for call center 2: A summation of the main results for call center 2.

5.4 The airport ground service

The fourth data set is from an airport ground service with 53 employees and out of these there are 52 employees used as inputs in the optimization. The scheduling period is 6 weeks or 42 days with 30 minute intervals. The on-duty demand for manpower depends on the flight schedules at the airport. Here there are many flights that leave during the early

morning and again in the afternoon. There are almost no flights scheduled at any time apart from the mornings and afternoons and because of this the demand for manpower peaks during the mornings and again in the afternoons but drops sharply during other times. Due to this the employees often work a short morning shift and then another short afternoon shift with a few hour break in between.

The weight factors for the airport ground service are a bit different than for the other three companies and institutions because here it is allowed to work more than one shift per day so the weight factor for more than one shift per day is 0 here. The weight factors for the airport ground service are recorded in Table 5.10 and the sum of the penalties in Table 5.11. It took 28 minutes and 24 seconds to solve the model and the value of the objective function is 1523.

c4.7	c4.8	c4.14	c.4.15	c4.16	c4.17	c4.18	c4.19	c4.22
15	2	10	100	10	10,000	1	0	10,000

Table 5.10: The weight factors for the airport ground service where the numbers correspond to the relevant constraint. We are obligated to fulfill c4.17 and c.4.22 and therefore they are really high. c4.18 is preferable but not an obligations therefore very low and c4.19 is not relevant in this case. c4.7, c4.8, c4.14, c4.15 and c4.16 all have low weight factors but the difference is because of different importances in fulfilling these constraints.

p4.7	p4.8	p4.14	p4.15	p4.16	p4.17	p4.18	p.4.19	p4.22
22	100	0	0	11	0	883	552	0

Table 5.11: Sum of the penalties for the airport ground service where the numbers correspond to the relevant constraint. p4.7, p4.8, p4.14, p4.16 p4.17 and p4.18 are all based on 30 minutes interval.

The hard constraints that must be satisfied for the airport ground service are that there must be a minimum continuous rest of 11 hours in any 24 hour period, the maximum number of consecutive working days is 5 and the employees cannot work more than 12 consecutive hours.

Table 5.12 summarizes the results for the airport ground service. The airport ground service is understaffed around 1% of the scheduling period and overstaffed about 5% of the scheduling period. All employees do fulfill the minimum on duty hours requirements, but six employees are working more than the maximum on duty hour. Four employees are half an hour above the maximum on duty hours, two employees are 1 hour above and one employee is 1.5 hours above the maximum on duty hours. All employees except one work more than one shift in one day somewhere in the scheduling period, but since they do not work more than the maximum number of working hours in any 24 hour period and all employees get the mandatory rest this is not a problem. All union regulations and

employee contracts about the maximum number of weekends worked in the scheduling period are satisfied. For the airport ground service we are able to meet about 89% of the requests made by the employees in the optimization.

Scheduled man-hours	6613
Man-hours understaffed	11
Man-hours overstaffed	50
Employees below minimum duty hours	0
Employees over the maximum duty hours	6
Employees over the maximum duty hours within a day	0
Employees working more than one shift per day	52
Employees working more then the maximum weekends	0
Requests met	7173
Requests not met	816

Table 5.12: Results for the airport ground service: A summations of the main results for the airport ground service.

Chapter 6

Comparison

Mathematical programming is believed by many to be too rigid an approach to solve the complicated problem of making a feasible schedule [8, 16]. This is true up to a certain level, this approach is for example more rigid than the local-search based algorithm which Ásgeirsson used in [16]. An example would be that Ásgeirsson allows employees to break constraints if they want to but that is not possible in our model. But despite this we are going to compare our results from all four companies and institutions with those Ásgeirsson got in his article.

In the datasets there are employees that have fixed assignments either for the whole period or only for parts of it. The availability, minimum and maximum demand as well as the minimum and maximum working hours are altered to regard these fixed assignments since these fixed assignments are not part of the optimization. In the following comparisons we have added these fixed assignments as fulfilled requests to be able to compare requests met with those in [16].

6.1 The nursing home

The MIP model generates a different result than the Ásgeirsson local-search based algorithm (ÁLS-algorithm) for the nursing home. The MIP result tends to have understaffed man-hours but the ÁLS-algorithm result tends to have overstaffed man-hours see Figure 6.2, this difference can also be seen in the last three days in Figure 6.1. The MIP model result has 109 man-hours understaffed and 0 man-hour overstaffed but the ÁLS-algorithm result has 21 man-hours understaffed and 220 man-hours overstaffed. The scheduled man-hours for the MIP model are 4774 but 5198 for the ÁLS-algorithm. But while the ÁLS-

algorithm result has no employees below minimum on duty hours the result for the MIP model has 1 employee below minimum on duty hours. The MIP model fulfills more than 98% of the requests made by the employees while the ÁLS-algorithm about 97.2%.

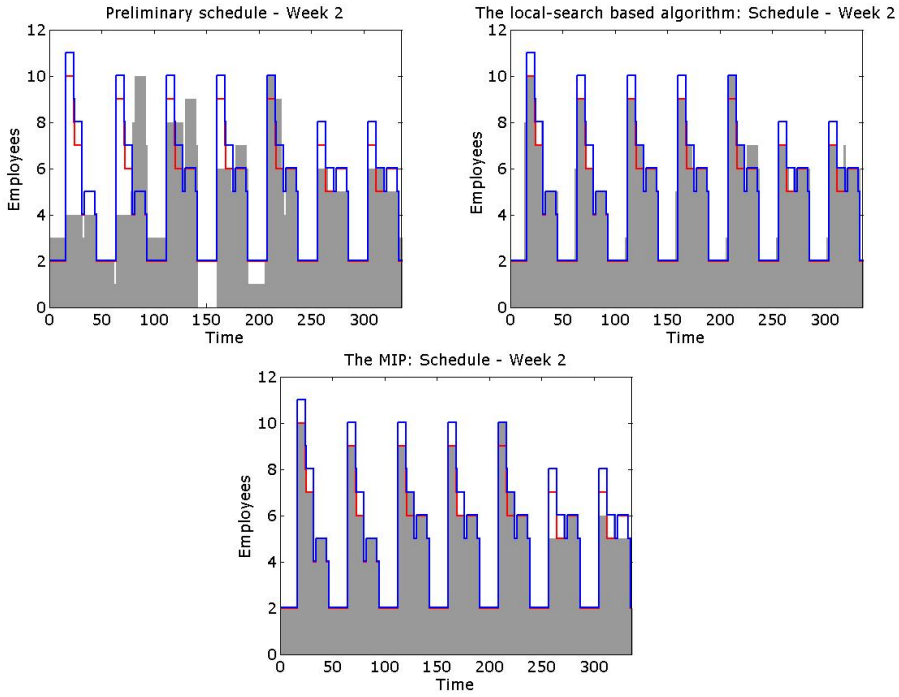

Figure 6.1: A comparison between the preliminary schedule, the schedule made from the results from the Ásgeirsson local-search based algorithm [16] and the schedule made from the results from the MIP model for the nursing home. The gray area denotes the number of employees on duty, the blue line denotes the maximum required staff on duty at each time and the red line the minimum required staff.

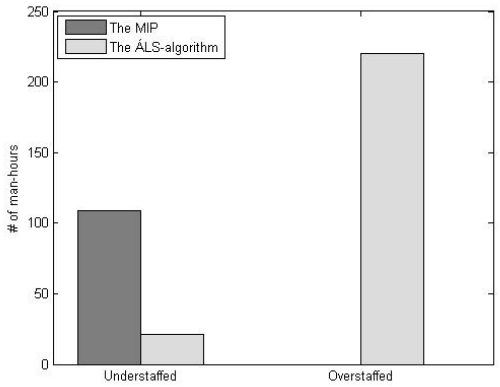

Figure 6.2: The nursing home: A comparison between the MIP model and the Ásgeirsson local-search based algorithm in [16] of the number of man-hours under- and overstaffed.

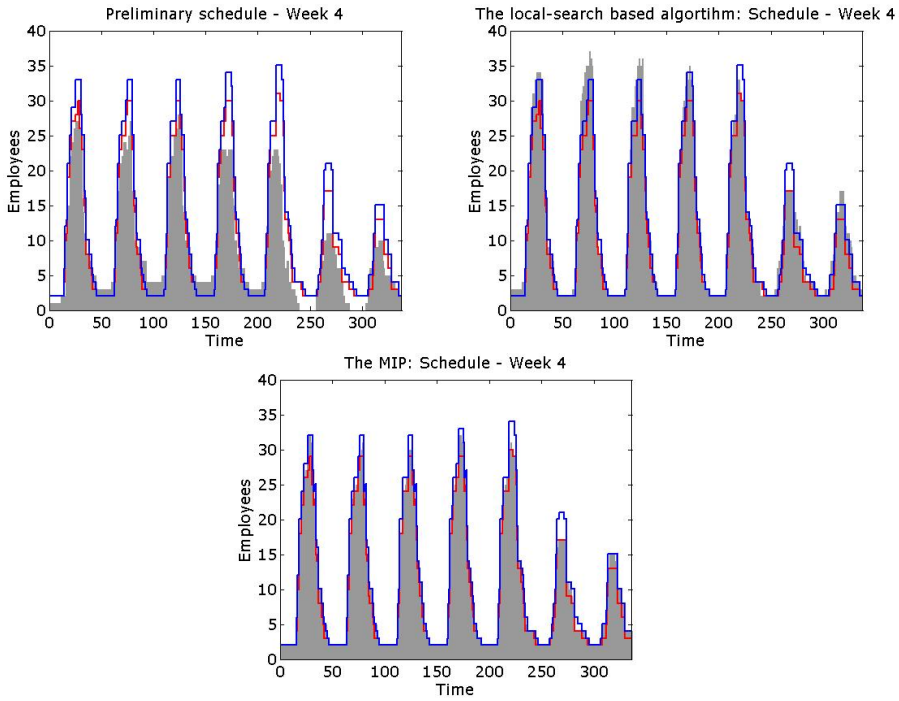

Figure 6.3: A comparison between the preliminary schedule, the schedule made from the results from the Ásgeirsson local-search based algorithm [16] and the schedule made from the results from the MIP model for call center 1. The gray area denotes the number of employees on duty, the blue line denotes the maximum required staff on duty at each time and the red line the minimum required staff.

6.2 Call center 1

The result generated by the MIP model is different than the ÁLS-algorithm for call center 1, see Figure 6.3. Understaffed man-hours in the result from the MIP model are 78 and man-hours overstaffed are 40 but understaffed man-hours in the result from the ÁLS-algorithm are 14 and overstaffed man-hours are 791, see Figure 6.4. 10,577.5 man-hours are scheduled in the MIP result but 11.920 man-hours scheduled in the ÁLS-algorithm. Call center 1 has 10 employees below the minimum on duty hours in the MIP result but the ÁLS-algorithm result has only 5 employees below. The ÁLS-algorithm result has overstaffed man-hours and is therefore able to fulfill more requests than the MIP model result that has understaffed man-hours, this difference is because of different emphases in these methods. The MIP model fulfills about 79% of the requests made by the employees but the ÁLS-algorithm about 96%.

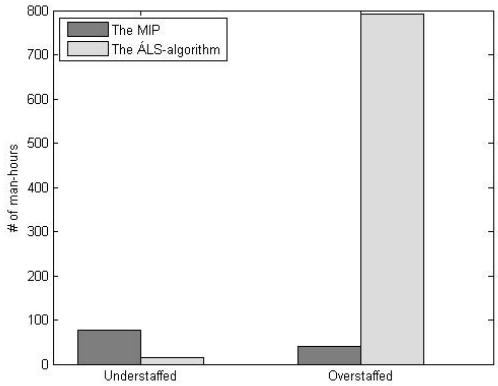

Figure 6.4: Call center 1: A comparison between the MIP model and the Ásgeirsson local-search based algorithm in [16] of the number of man-hours under- and overstaffed.

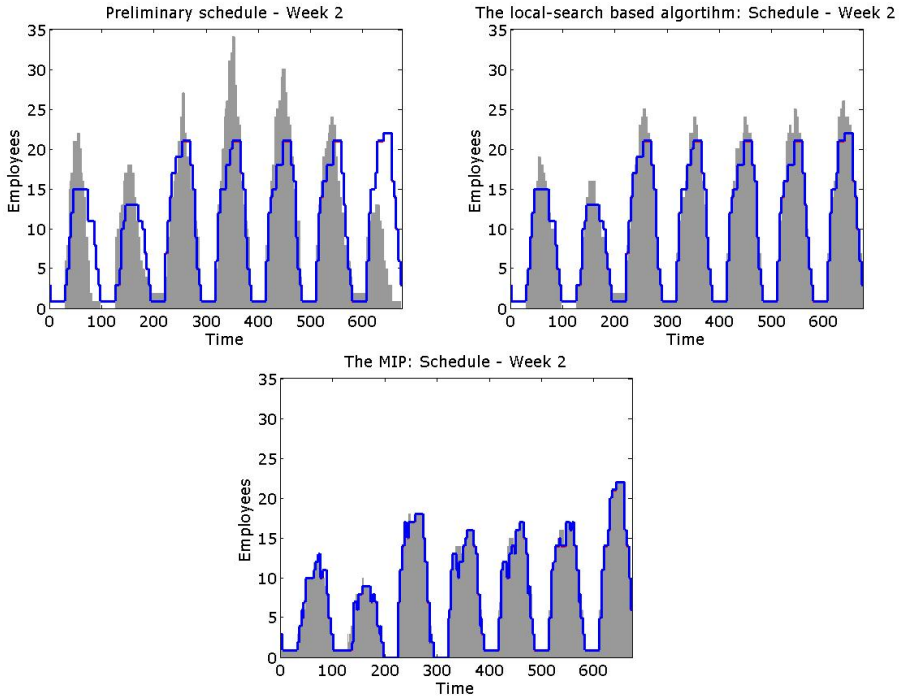

Figure 6.5: A comparison between the preliminary schedule, the schedule made from the results from the Ásgeirsson local-search based algorithm [16] and the schedule made from the results from the MIP model for call center 2. The gray area denotes the number of employees on duty, the blue line and the red line denotes the required staff on duty at each time.

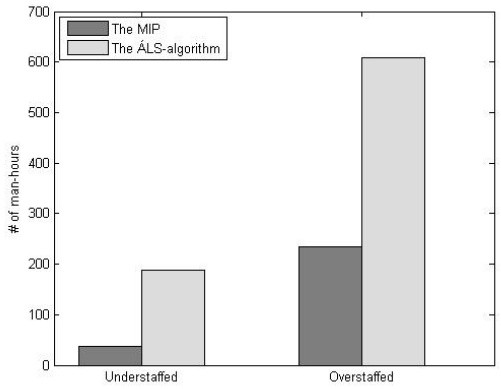

Figure 6.6: Call center 2: A comparison between the MIP model and the Ásgeirsson local-search based algorithm in [16] of the number of man-hours under- and overstaffed.

6.3 Call center 2

The MIP model generates a better result than the ÁLS-algorithm for call center 2, see Figure 6.5 and Figure 6.6. The MIP model result has 37 man-hours and 15 minutes understaffed and 234 man-hour overstaffed but the result generated by the ÁLS-algorithm has 189 man-hours understaffed and 609 man-hours overstaffed. Since the demand for manpower is an exact number instead of being minimum and maximum demand it is really difficult not to have any under- or overstaffed time periods. The MIP result has 6,071 scheduled man-hours but the result from the ÁLS-algorithm has 7,554 scheduled man-hours. The MIP model result has more employees below the minimum on duty hours than the ÁLS-algorithm or 17 employees versus 7 employees. The MIP model also fulfills fewer requests than the ÁLS-algorithm or 65% of all requests versus 86% of all requests. Different emphases in the methods can be seen in that the MIP model tends to have understaffed man-hours and employees below the minimum on duty hours but the ÁLS-algorithm tends to have overstaffed man-hours and employees above the maximum on duty hours. This tendency makes it possible for the ÁLS-algorithm to fulfill more requests than the MIP model.

6.4 The airport ground service

For the airport ground service the MIP model generates a better result than the ÁLS-algorithm, see Figure 6.7. The Ásgeirsson local-search based algorithm returns a result with 517 man-hours understaffed and 641 man-hours overstaffed but the MIP model returns a result with only 11 man-hours understaffed and 50 man-hours overstaffed, see

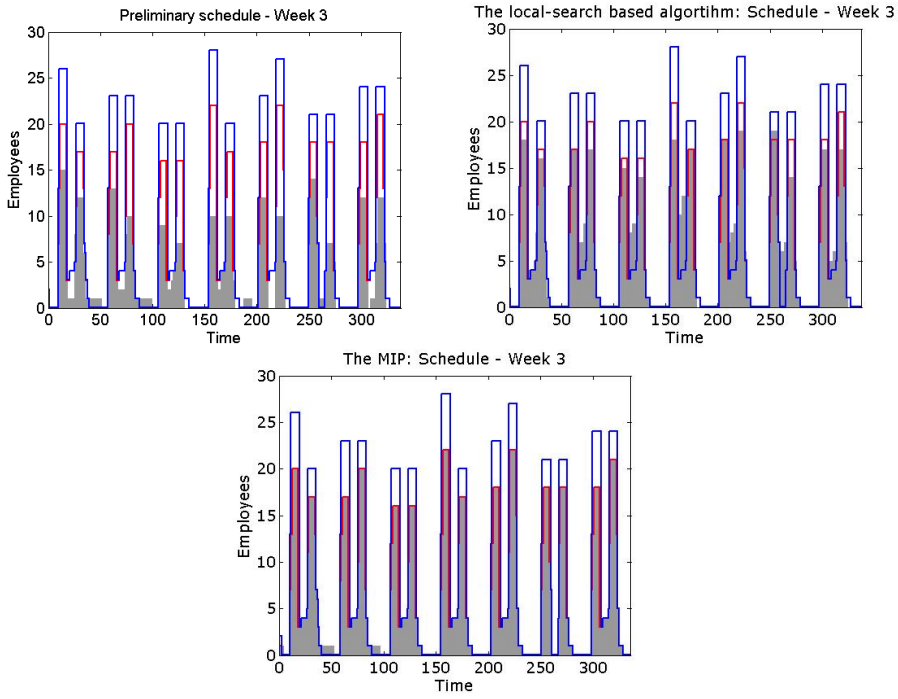

Figure 6.7: A comparison between the preliminary schedule, the schedule made from the results from the Ásgeirsson local-search based algorithm [16] and the schedule made from the results from the MIP model for the airport ground service.The blue line denotes the maximum required staff on duty at each time and the red line the minimum required staff, the gray area denotes the number of employees on duty.

Figure 6.8. Another interesting thing to compare is the scheduled hours, the ÁLS-algorithm has 6670 scheduled hours but the MIP model 6607.5, however the average percentage of requested hours met is 94% for the ÁLS-algorithm but 89% for the MIP model. Neither method has any employees under the minimum on duty hours.

6.5 Summary of the comparisons

The MIP model is able to generate a feasible staff schedule for all four companies and institutions just like the Ásgeirsson local-search based algorithm. There is an equilibrium between the minimum and maximum demand and being below or above the required on duty hours. The result from the MIP model show that the MIP model tends to have understaffed time periods and more employees below the minimum on duty hours but the ÁLS-algorithm goes in the other direction, its schedule tends to be overstaffed and have fewer employees below the minimum on duty hours. Neither method is more correct than the other, it depends on the companies policies and on the staff managers which one is more preferable. However the gaps between under- and overstaffed employees in

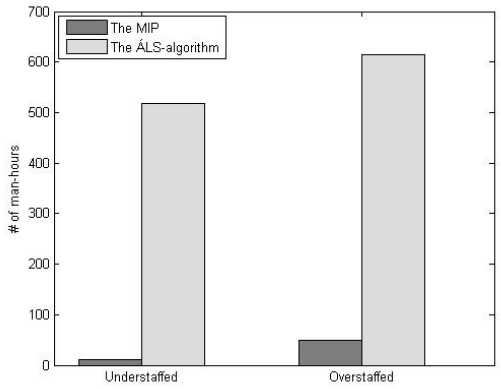

Figure 6.8: The airport ground service: A comparison between the MIP model and the Ásgeirsson local-search based algorithm in [16] of the number of man-hours under- and overstaffed.

the results from the MIP model are smaller than for the ÁLS-algorithm, showing that there might be a room for improvement in the ÁLS-algorithm. However it is not obvious that it is possible to improve the ÁLS-algorithm since employees needs to be able to get an explanation and justification for every change that the ÁLS-algorithm makes to the employee requests.

The methods have different emphasis on the requests made by the employees. Two of the companies, the nursing home and the airport ground service have comparable number of requests fulfilled but for the other two companies the MIP is much stricter. The MIP model discards employees requests rather than being overstaffed while the ÁLS-algorithm fulfills more requests at the cost of having more overstaffed hours. This could make the employee tolerance toward the ÁLS-algorithm more than toward the MIP model.

Chapter 7

Conclusion

In this paper we have introduced a model using the mathematical programming technique of mixed integer programming. The results from the four different Icelandic companies and institutions show that it is possible to achieve high quality schedules in a reasonable amount of time by using mixed integer programming. Our model allows flexibility in terms of shifts lengths and shifts starting times. The proposed model is able to handle all the constraints and requirements of personnel scheduling in today's complex environments.

A comparison between the proposed model and a local-search based algorithm [16] was executed. Both methods generated a feasible staff schedule for all four companies and institutions. The methods however have different emphasis and therefore generate different solutions. Neither method gives more correct solution than the other, it only depends on the staff managers and the companies policies which method is more preferable.

To be able to fully implement this model it is necessary to develop a graphical user interface, that would permit the employees to enter their requests and their availability and allow the staff manager to enter the minimum and maximum demand, the minimum and maximum on duty hours for each employee, maximum hours worked within a day, maximum amount of weekends each employee is allowed to work and the maximum amount of consecutive days for each employee. It is also necessary to consider changes to the model to take care of problems like fairness and boundary.

In this paper all employees are considered highly qualified so further research will be aimed at looking at employees with different qualifications where each time period would have a different manpower demand for each qualification.

Chapter 8

Appendix

The code written in MPL.

INDEX

$i := DATAFILE(\text{"}Name\ of\ the\ company/indexi.dat\text{"});$
$j := DATAFILE(\text{"}Name\ of\ the\ company/indexj.dat\text{"});$
$k := DATAFILE(\text{"}Name\ of\ the\ company/indexk.dat\text{"});$
$t := DATAFILE(\text{"}Name\ of\ the\ company/indext.dat\text{"});$
$l := DATAFILE(\text{"}Name\ of\ the\ company/indexl.dat\text{"});$
$w := DATAFILE(\text{"}Name\ of\ the\ company/indexw.dat\text{"});$
$p := DATAFILE(\text{"}Name\ of\ the\ company/indexp.dat\text{"});$
$c := DATAFILE(\text{"}Name\ of\ the\ company/indexc.dat\text{"});$

BINARY VARIABLES

$x[i,j,k]EXPORTTOSPARSEFILE(\text{"}variablex.dat\text{"});$
$y[i,t,k]EXPORTTOSPARSEFILE(\text{"}variabley.dat\text{"});$
$z[i,l,k]EXPORTTOSPARSEFILE(\text{"}variablez.dat\text{"});$
$d[i,k]EXPORTTOSPARSEFILE(\text{"}variabled.dat\text{"});$
$\omega[i,w]EXPORTTOSPARSEFILE(\text{"}variable\omega.dat\text{"});$
$p9[i]EXPORTTOSPARSEFILE(\text{"}varialbep9\text{"});$

VARIABLES

$p1[t,k]EXPORTTOSPARSEFILE(\text{"}variablep1.dat\text{"});$
$p2[t,k]EXPORTTOSPARSEFILE(\text{"}variablep2.dat\text{"});$
$p3[i]EXPORTTOSPARSEFILE(\text{"}variablep3.dat\text{"});$
$p4[i]EXPORTTOSPARSEFILE(\text{"}variablep4.dat\text{"});$

$p5[i,k]EXPORTTOSPARSEFILE(\text{"}variablep5.dat\text{"});$
$p6[i,t,k]EXPORTTOSPARSEFILE(\text{"}variablep6.dat\text{"});$
$p7[i,k]EXPORTTOSPARSEFILE(\text{"}variablep7.dat\text{"});$
$p8[i]EXPORTTOSPARSEFILE(\text{"}variablep8.dat\text{"});$

DATA

$dmin[t,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/dmin.dat\text{"});$
$dmax[t,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/dmax.dat\text{"});$
$tmin[i] := SPARSEFILE(\text{"}Name\ of\ the\ company/tmin.dat\text{"});$
$tmax[i] := SPARSEFILE(\text{"}Name\ of\ the\ company/tmax.dat\text{"});$
$tdmax[i,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/tdmax.dat\text{"});$
$maxw[i] := SPARSEFILE(\text{"}Name\ of\ the\ company/maxw.dat\text{"})$
$available[i,t,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/available.dat\text{"});$
$availables[i,j,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/availableshifts.dat\text{"});$
$shifts[j,k,t] := SPARSEFILE(\text{"}Name\ of\ the\ company/shifts.dat\text{"});$
$restshifts[l,k,t] := SPARSEFILE(\text{"}Name\ of\ the\ company/restshifts.dat\text{"});$
$requestshift[i,t,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/requestshift.dat\text{"});$
$weekend[\omega,k] := SPARSEFILE(\text{"}Name\ of\ the\ company/weekend.dat\text{"});$
$penalties[p] := SPARSEFILE(\text{"}Name\ of\ the\ company/penalties.dat\text{"});$
$maxday[i] := SPARSEFILE(\text{"}Name\ of\ the\ company/maxdays.dat\text{"});$
$days := DATAFILE(\text{"}Name\ of\ the\ company/days.dat\text{"});$

MACROS

$pen1 = SUM(t,k : p1[t,k]);$
$pen2 = SUM(t,k : p2[t,k]);$
$pen3 = SUM(i : p3[i]);$
$pen4 = SUM(i : p4[i]);$
$pen5 = SUM(i,k : p5[i,k]);$
$pen6 = SUM(i,t,k : p6[i,t,k]);$
$pen7 = SUM(i,k : p7[i,k]);$
$pen8 = SUM(i : p8[i]);$
$pen9 = SUM(i : p9[i]);$

MODEL

$MINpenalty = penalties[1] * SUM(t,k : p1[t,k]) + penalties[2] * SUM(t,k : p2[t,k]) + penalties[3] * SUM(i : p3[i]) + penalties[4] * SUM(i : p4[i]) + penalties[5] *$

$SUM(i,k:p5[i,k])+penalties[6]*SUM(i,t,k:p6[i,t,k])+penalties[7]*SUM(i,k:p7[i,k])+penalties[8]*SUM(i:p8[i])+penalties[9]*SUM(i:p9[i]);$

SUBJECT TO

$MinDemand[t,k]:SUM(i:y)>=dmin[t,k]-p1[t,k];$
$MaxDemand[t,k]:SUM(i:y)<=dmax[t,k]+p2[t,k];$

$TimeShift[i,t,k]:y[i,t,k]=SUM(j:shifts[j,k,t]*x[i,j,k]);$
$TimeFreeShift[i,t,l,k]WHERE(restshifts>0):y[i,t,k]<=1-z[i,l,k];$
$FreeShift[i,k]:SUM(l:z)=1;$

$AvailableAtTime[i,t,k]:y[i,t,k]<=available[i,t,k];$
$AvailableOnShift[i,j,k]:x[i,j,k]<=availables[i,j,k];$

$MinNr[i]:SUM(t,k:y)>=tmin[i]-p3[i];$
$MinNrb[i]:SUM(t,k:y)>=tmin[i]*(1-p9[i]);$
$MaxNr[i]:SUM(t,k:y)<=tmax[i]+p4[i];$
$MaxNrWithinDay[i,k]:SUM(t:y)<=tdmax[i,k]+p5[i,k];$

$Request[i,t,k]:requestshift[i,t,k]-y[i,t,k]<=p6[i,t,k];$
$OneShiftPerDay[i,k]:SUM(j:x)=d[i,k]+p7[i,k];$
$MaxDays[i,k]WHERE(k<days):SUM(c:d[i,k+c])<=maxday[i];$
$weekends[i,w,k]WHERE(weekend>0):d[i,k]+d[i,k+1]=2*\omega[i,w];$
$maxweekend[i]:SUM(w:\omega)<=maxw[i]+p8[i];$

BOUNDS

$p1[t,k]>=0;$
$p2[t,k]>=0;$
$p3[i]>=0;$
$p4[i]>=0;$
$p5[i,k]>=0;$
$p6[i,t,k]>=0;$
$p7[i,k]>=0;$
$p8[i]>=0;$

Bibliography

[1] W. J. Abernathy, N. Baloff, J. C. Hershey, and S. Wandel. A three-stage manpower planning and scheduling model-a service-sector example. *Operations Research*, 21(3):693–711, 1973.

[2] J. Bailey and J. Field. Personnel scheduling with flexshift models. *Journal of Operations Management*, 5(3):327–338, 1985.

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

[4] J. F. Bard, C. Binici, and A. H. deSilva. Staff scheduling at the united states postal service. *Computers and Operations Research*, 30(5):745–771, 2003.

[5] J. F. Bard and H. W. Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164(2):510–534, 2005.

[6] J. O. Brunner, J. F. Bard, and R. Kolisch. Flexible shift scheduling of physicians. *Health Care Management Science*, 12(3):285–305, 2009.

[7] E. K. Burke, J. Li, and R. Qu. A hybrid model of integer programming and variable neighborhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research*, 203(2):484–493, 2010.

[8] E.K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

[9] R. N. Burns and M. W. Carter. Work force size and single shift schedules with variable demands. *Management Science*, 31(5):599–607, 1985.

[10] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems - a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, 2003.

[11] L. C. Edie. Traffic delays at toll booths. *Journal of the Operations Research Society of America*, 2(2):107–138, 1954.

[12] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[13] B. Gopalakrishnan and E. L. Johnson. Airline crew scheduling: State-of-the-art. *Annals of Operations Research*, 140(1):305–337, 2005.

[14] H. S. Rowland and B. L. Rowland. In *Nursing administration handbook 4th ed.*, Gaithersburg, Maryland, 1997.

[15] D. M. Ryan. The solution of massive generalized set partitioning problems in aircrew rostering. *The Journal of the Operational Research Society*, 43(5):459–467, 1992.

[16] E. I. Ásgeirsson. Bridging the gap between schedules and feasible schedules in staff scheduling. *Practice and Theory of Automated Timetabling (PATAT)*, 2010.

[17] W. Townsend. An approach to bus-crew roster design in london regional transport. *The Journal of the Operational Research Society*, 39(6):543–550, 1988.

[18] D. M. Warner. Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, 24(5):842–856, 1976.

[19] D. M. Warner and J. Prawda. A mathematical programming model for scheduling nursing personnel in a hospital. *Management Science*, 19(4):411–422, 1972.

School of Science and Engineering
Reykjavík University
Menntavegur 1
101 Reykjavík, Iceland
Tel. +354 599 6200
Fax +354 599 6201
www.reykjavikuniversity.is
ISSN 1670-8539