# Creating feasible schedules in the last step of the self rostering process

**Suzanne Uijland**

Master thesis

Industrial Engineering and Management,

Track Production and Logistic Management

Enschede, June 8, 2012

**Supervisor ORTEC**

E. (Egbert) van der Veen, MSc

**Supervisors University of Twente**

Dr. ir. J.M.J. (Marco) Schutten

Prof. dr. J.L. (Johann) Hurink

ORTEC

UNIVERSITY OF TWENTE.

Master thesis Industrial Engineering and Management

# Creating feasible schedules in the last step of the self rostering process

Suzanne Uijland

s0089028

Graduation committee:

Dr. ir. J.M.J. Schutten (University of Twente)

Prof. dr. J.L. Hurink (University of Twente)

E. van der Veen, MSc (ORTEC)

Universiteit Twente
Drienerlolaan 5
7522 NB Enschede
Tel: +31 (0)53 489 9111
Fax: +31 (0)53 489 2000
Email: info@utwente.nl

ORTEC
Groningenweg 6k
2803 PV Gouda
Tel: +31 (0)182 540 500
Fax: +31 (0)182 540 540
Email: info@ortec.com

# Summary

**Introduction** In service industries, such as healthcare and security services, people work around the clock. Considering the many preferences of employees and the labor legislations that are implied on the schedules, it is, both in theory (L. De Grano et al. (2009); Rönnberg and Larsson (2010)) and practice, often hard to come up with good schedules for these employees. A possible way to cope with employee preferences and to increase job satisfaction, is self rostering. The main idea in self rostering is that employees can propose their own schedule and if they do this in a 'good' way, they get to work most of their shifts as in their preferred schedule.

**Self rostering process** The self rostering process consist of 5 steps:

1. The organization defines the *staffing demand*, i.e., the number of employees that need to perform a shift is specified for each shift and day.
2. The employees propose their preferred schedules.
3. The employees' preferred schedules are matched to the staffing demand, from which information on understaffed and overstaffed shifts is derived.
4. The information of Step 3 is returned to the employees, after which employees can adjust their schedules.
5. The planner fulfills the understaffed shifts that remain after Step 4.

**Research objective** The goal of this research is to design a method that helps planners finalize the schedule in the last step of the self rostering process and that is widely applicable.

**Method** To fulfill the remaining understaffed shifts, we use an iterative method. Every iteration solves a linear program that mainly considers two things: the *score* of each employee's schedule and *swaps*.

The *score* of an employee's schedule is calculated as follows. First, we assign 'points' to shifts using a specified point system. For example, understaffed shifts are assigned 3 points, overstaffed shifts receive 1 point, and matching shifts (shifts where the staffing demand is exactly matched) receive 2 points. Second, we calculate each employee's score, by summing all points per employee, possibly multiplied by some factor, e.g., to take part-time percentages into account. Employees that have a high score work many relatively unpopular shifts, whereas for employees with a low score the opposite holds.

*Swaps* define a re-assignment of shifts. We consider two types of swaps: primary swaps and secondary swaps. A *primary swap* swaps shifts in the schedule of one employee. A *secondary swap* performs two swaps, where each swap is performed at another employee. So two employees are unassigned from a shift and assigned to a new shift. With secondary swaps it is possible to fulfill an understaffed shift by unassigning an employee from a matching shift and fulfilling the matching shift again by the second employee. This employee is swapped from an overstaffed shift to the matching shift.

Every iteration considers a subset of employees. The subset is based on the *scores* of the employees. For this subset of employees, we calculate the possible *swaps* per employee. Using a linear program, we select a subset of these swaps with at most one swap per employee. The linear program makes a trade-off between two factors. On the one hand, we want to minimize the number of understaffed shifts by applying swaps. On the other hand, we prefer to perform certain swaps (e.g., swaps that result in the highest score change) in the schedules of certain employees (e.g., employees that have a low score).

Per employee, we want to preserve a minimum fraction of his proposed schedule. For this, a constraint is included in the iterative method.

**Results** We applied our method to case studies from practice.

There are three stakeholders: the organization, the employees, and the planner. For each stakeholder we define a criterion to evaluate the method. These are respectively: Shortages, Remaining percentage, and computational Time. Furthermore, the proposed method has several input parameters. We study the effects on the outcomes when using different input parameter settings.

Two input parameters cause a trade-off between the number of understaffed shifts fulfilled (Shortages) and fraction of the preferred schedules that is preserved (Remaining percentage), these are *Swap strategy* and *Minimum percentage*. *Swap strategy* is the way of using primary and secondary swaps. *Minimum percentage* is the constraint on the minimum fraction of the proposed schedules we want to retain. Both input parameters cause a similar effect on the outcomes: when a change of an input parameter causes fewer shortages to remain, it also causes a lower remaining percentage of the preferred schedule, and the other way around. Fewer shortages are preferred by the organization, but lower remaining percentages are not preferred by the employees.

The input parameter *Initial employees* specifies the number of employees considered in the first iteration. This number of considered employees is increased each time no improvements are found in an iteration. The input parameter *Initial employees* has only an influence on the remaining percentage of the preferred schedules. Therefore, this parameter is only important from the employees' point of view.

The method has a maximum running time of 12 seconds for instances with a planning horizon of 28 days and about 70-80 employees. For instances with fewer employees (e.g., we have an instance with 15 employees), the

maximum running time is reduced to 2.6 seconds. As these maximum times are in the order of seconds and not minutes of hours, the method works well from the planners point of view.

**Conclusions** The method designed is shown to be a suitable method to advise the planner in the last phase of the self rostering process. Furthermore, we gained insight in which input parameter values work best for which type of preferred schedules. With this information, it is up to the user to decide which input parameter values to use.

# Preface

This master thesis is the result of my graduation project of the study Industrial Engineering & Management at the University of Twente. This research is conducted at ORTEC, one of the largest providers of advanced planning and optimization software solutions and consulting services. Using this opportunity, I would like to thank several people.

I thank ORTEC for giving me the opportunity to do this research. I thank my colleagues for providing me a comfortable, fun, and motivating work atmosphere. I thank the organizations who provided me cases and data. These gave me the opportunity to gain insight in the criteria from practice and apply my research on practical data. I thank the many friends and family who challenged, motivated, and helped me in all conversation I had with them. Finally, I want to thank the following people in particular.

Egbert van der Veen, I thank you for all your support, your input, and your critical view, which have really improved my thesis. I thank you for all our conversations where you challenged me to see what you probably already knew. Your enthusiasm and helpfulness have guided me through this research.

Marco Schutten and Johann Hurink, I thank you both for all your time and effort in this research. Your ideas, input, and constructive criticism have really improved this thesis on both the content as well as the structure. Thank you for your guidance.

Bernd, I thank you for always being their for me in so many ways, for your love and support. Finally, I thank my parents for always being there for me, for all chances you gave me, and your unconditional support. Thank you!

Suzanne Uijland
Enschede, June 2012

# Contents

# Chapter 1

# Introduction

Health care, security, and transport organizations are examples of organizations that provide services 24/7. In these organizations, shifts are defined around the clock. Some shifts are more popular than others. Popularity of shifts arises from employees' preferences: some employees like to work in the evening, while others prefer morning shifts.

At most organizations, employees report all their preferences to the planner, who is responsible for making the schedules of the employees. For the planner, it is very hard to make a schedule that covers both the staffing demand of the organization and the preferences of the employees, while meeting all labor legislations. Recently, more and more organizations study the possibility of using self rostering to solve this problem.

Self rostering is a scheduling process where employees are extremely involved. Employees balance their work hours with their personal responsibilities by creating their own preferred schedules. All schedules combined most likely do not match the staffing demand of the organization. Therefore, changes need to be made. First, employees get the chance to change their schedule and second, the planner makes the necessary changes. The planner needs to find a suitable strategy to complete the schedule in a fair way. In this research, we design a method to help the planner finalize the schedule.

This research is conducted at ORTEC. ORTEC is a company specialized in advanced planning and optimization software solutions. ORTEC offers software and consultancy for, among others, personnel planning and is therefore interested in new strategies of workforce scheduling, such as self rostering.

The next section (Section 1.1) briefly describes ORTEC and its workforce planning software ORTEC Harmony. Next, Section 1.2 describes the motivation for this research, followed by the problem description (Section 1.3), and the research objective and approach (Section 1.4).

## 1.1 Context description

### ORTEC

ORTEC is one of the largest providers of advanced planning and optimization software solutions and consultancy services. The products and services of ORTEC result in optimized fleet routing and dispatching, vehicle and pallet loading, workforce scheduling, delivery forecasting, and network planning. ORTEC has over 550 employees and offices in Europe, North America, Asia, and the Pacific Region. Moreover, it has over 1,450 customers worldwide in a large number of industries (ORTEC, 2011), for example:

- trade, transport, and logistics
- retail
- consumer packaged goods
- health care
- professional and public services
- manufacturing and construction
- oil, gas, and chemicals.

This research takes place within Product Delivery & Consultancy (PDC), which is part of Quality & Product Competence (QPC) of ORTEC Software Development (OSD) in Gouda, the Netherlands. PDC is responsible for transferring product knowledge and delivering complete products to the market units. This is accomplished by writing documentation (release papers, user manuals, etcetera), testing software, and providing training courses and product consultancy.

### ORTEC Harmony

ORTEC Harmony (from here on referred to as Harmony) is an advanced workforce scheduling software solution of ORTEC. Harmony is specifically useful in environments where work is carried out at irregular times or where the workload is fluctuating during operational hours. Typical customers are found in sectors such as health care (hospitals, nursing and caring homes, and ambulant care), transportation, security and field service workforce, oil and gas distribution, and the retail sector.

Harmony supports the entire workforce management process, from strategy development to evaluation. The possibilities of Harmony are enormous. To give an impression: at the beginning of the scheduling process, Harmony helps the user to define the organization within Harmony, determine labor demand, required workforce, and (company specific) legislation. After that, schedules can be created.

There are two ways to do this: (1) the user assigns the shifts to the employees manually or (2) use automatic planners. In both cases, Harmony checks legislation (e.g., collective labor and rest time regulation), company specific rules (e.g., required qualifications and skills), sociological criteria, and employee

preferences. Harmony also supports real-time decision making, which ensures that the user can quickly reallocate work if, e.g., last-minute absenteeisms or changes in work-load occur. Last but not least, Harmony automatically registers all kinds of information based on the employee schedules. Examples are: working hours overviews for pay-roll purposes, illness overviews, vacation entitlements per employee, and overviews of shifts that are not covered.

## 1.2 Research motivation

Since a couple of years, several Dutch organizations are interested in self rostering. ORTEC wants to keep up with these developments by extending their workforce planning software Harmony to facilitate self rostering. Through this, existing Harmony customers discover new possibilities in rostering, while organizations that have not been using Harmony yet may become interested in Harmony because of the self rostering functionality. To develop such a new functionality, the exact requirements for self rostering need to be sorted out. Moreover, algorithms need to be developed to complete the self rostering method.

Self rostering is also an interesting topic from a scientific point of view. In the last few years, numerous articles have been written about the impact of work-life balance on the health of employees. An imbalance may pose a threat to the health of employees. Sleep disturbances, fatigues, digestive problems, emotional problems, and stress related illnesses may all be consequences of an imbalance, and cause increased sick leave (Bambra et al., 2008). Furthermore, Thornthwaite (2004) shows that an imbalance may pose a threat to both the employee performance as well as to the levels of commitment and loyalty. Both are important factors at high-performance work systems. So the work-life balance of the employee is very important. Bambra et al. (2008) describe three interventions of self rostering that all have beneficial effects on the health and work-life balance of the employees. So from a scientific point of view, it is also interesting to further research self rostering.

## 1.3 Problem description

The assignment of this thesis is to help a planner finish the schedule at the end of the self rostering process. Self rostering consist of 5 steps, see Figure 1.1.

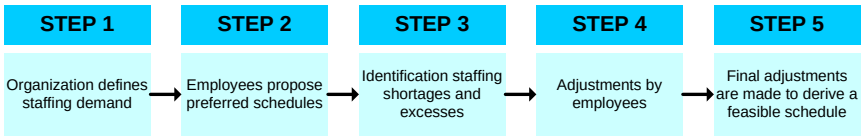



Figure 1.1: The self rostering process

In step 1, organizations define their staffing demand. They define how many employees with a certain skill have to be available at each time of each day within the planning period. In step 2, employees propose their preferred schedule for the next period. This schedule is individual, thus independent of the other employees or the preferences of the organization. Next, all proposed schedules

(from step 2) are compared to the staffing demand of step 1. In step 3, the excesses and shortages at certain shifts are identified. In step 4, these excesses and shortages are communicated to the employees, and employees have the opportunity to change their proposed schedule (by informal negotiations), to reduce the excesses and shortages. In step 5, the planner checks on violations and makes the final decisions to finish the schedule.

The assignment of this thesis is to develop a method that helps the planner finish the schedule in a way that retains most of the employees proposed schedules and that is transparent and thereby perceived as fair by the employees.

## 1.4 Research objective and approach

As described in Section 1.3, a method for the planner to finalize the schedule in the last step of self rostering is missing. This research focuses on developing such a method. The objective of this research is:

> *Design a method that helps planners finalize the schedule in the last step of the self rostering process and that is widely applicable*

To be able to achieve the research objective some questions need to be answered first. These questions are assigned to the chapters of this thesis. Below the questions per chapter, we describe our approach to answer these questions. In the introduction of each chapter we indirectly return to the corresponding questions and answer these in the conclusions of the chapter.

**Chapter 2: Self rostering in literature and practice**

1. *What is known about methods to finish the self rostering process?*
   (a) *What are the advantages and disadvantages of these methods?*
2. *Which organizations are interested in self rostering and why?*
3. *What criteria are important for the method according to the cases?*
   (a) *Which criteria should a method certainly meet?*

   We will answer these questions by the information found in literature, received from interested organizations, and received from ORTEC. We will describe the literature found about the self rostering process and self rostering in practice. From the interviews with the organizations, we create representative cases and determine the (most) important criteria for the method per case.

**Chapter 3: Self rostering process**

1. *How does ORTEC apply the self rostering process?*

   We will answer this question by the information received from ORTEC.

**Chapter 4: Method**

1. *What is a possible method?*
2. *What is a suitable mathematical approach for this method?*

   Based on the criteria determined by the organizations and the advantages of methods from literature, we will design a suitable method and a suitable mathematical approach for this method.

**Chapter 5: Experimental design and data**

1. *What is a suitable method to analyze the model outcomes?*
2. *What are suitable datasets to test the method?*

   To answer these questions, we will first describe what we want to analyze and search for a suitable method in literature. Second, we will gather datasets from the interested organizations and discuss whether these are suitable to test the method.

**Chapter 6: Experimental results**

1. *How does the method perform using different input data?*

   We will answer this question by applying the experimental design (described in Chapter 5) on various datasets and model parameter values, and discuss the results.

# Chapter 2

# Self rostering in literature and practice

After introducing the problem, research objective and research questions of this research in Chapter 1, this chapter discusses the theoretical and practical basis of this research. Section 2.1 explains the terminology that we use in this research. Section 2.2 describes literature found on self rostering processes. Section 2.3 describes self rostering in practice and is split up in two parts. The first part describes representative cases from practice. From interviews with the organizations, we describe the cases and determine per case the criteria for our method. The second part consists of examples of self rostering in practice found in literature, their advantages and disadvantages, and discusses how these are related to the cases. Finally, Section 2.4 describes our conclusions of this chapter.

## 2.1 Terminology

In this research a *shift* indicates a time period where work activities need to be fulfilled. With *planning period* we mean the time horizon for which we schedule shifts such that the staffing demand is fulfilled (e.g., one week, one month, one year). *Staffing demand* is the number of employees that is required for certain shifts on certain days by the organization.

In this research *shortage* and *understaffed shift* indicate that the number of employees assigned to this shift is less than the staffing demand. With *excess* and *overstaffed shift* we indicate that the number of employees assigned to a shift is larger than the staffing demand. A *matching shift* is a shift for which the number of employees scheduled matches the staffing demand.

Section 2.1.1 describes how we use the terms *scheduling* and *rostering*. Many different terms are used in literature for self rostering. Section 2.1.2 describes how we use the terms *preference rostering*, *self rostering*, and *individual rostering*.

### 2.1.1 Scheduling and rostering

Terms such as *scheduling* and *rostering* are often used in literature, both with multiple meanings that sometimes even overlap. In this research, scheduling is used for defining the shifts up to the assignment of the employees. This process consists of three steps:

1. *Define shifts*
   Define start and end times and required skills.
2. *Define staffing demand*
   Define the number employees needed per shift per day.
3. *Assign employees to shifts*
   Assign employees to shifts, such that the staffing demand is met.

In this research, rostering is used when only the last step of scheduling is meant, namely the assignment of employees to shifts. See Figure 2.1 for a clear overview of how we use the terms scheduling and rostering in this research.

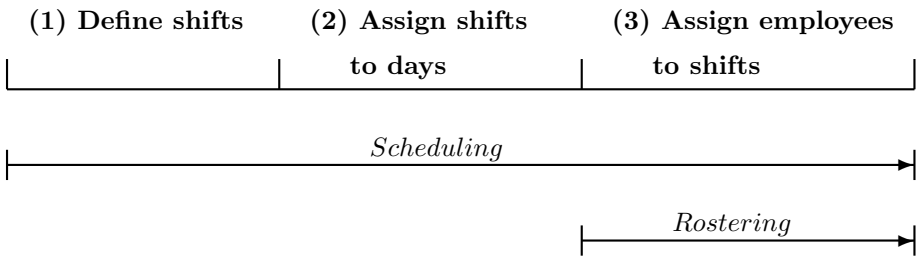



Figure 2.1: Scheduling and rostering used in this research

A *schedule* and a *roster* are both the result of their corresponding processes scheduling and rostering, respectively. Both processes end at the same moment. Thus in this research a schedule and a roster are seen as synonyms. Both terms refer to a timetable in which employees are assigned to shifts on certain times.

### 2.1.2 Preference, self and individual rostering

In this research, individual rostering is used when both the staffing demand and individual preferences of the employees are taken into account when creating a roster. This results in individual rosters for all employees. Preference rostering and self rostering are seen as two forms (out of the six) of individual rostering. To see these in perspective, the following list sorts all six forms of individual rostering from little control of the employee to a lot of control and from uniformity in working hours to diversity (NCSI, 2009):

1. *Swap*
   After a roster is published, employees have the opportunity to negotiate and swap shifts. This helps the employees with their incidental wishes.

2. *Repetitive rostering*
   In this concept, structural wishes are included in the scheduling process. For example, an employee wishes every Wednesday afternoon off in order to pick up the kids from school. Since this wish covers not just one Wednesday, but all Wednesdays, this a structural wish. The created rosters are cyclic and are used for an indefinite period.

3. *Preference rostering*
   For each planning period, the employees indicate their individual wishes. These are taken into account when creating the roster as long as the staffing demand is met. This roster is created for a couple of weeks or months.

4. *Shift picking*
   The employer presents an overview with shifts that are not yet assigned to employees. Employees who meet the qualifications of a shift are allowed to sign up for the shift. The planner makes the last decision of who is assigned to which shift. The planning period is a few weeks.

5. *Matching*
   Staffing demand is defined per time unit (e.g. hours, half hours). Employees subscribe themselves for a certain amount of time units per day when they want to work. The planner or a system matches the staffing demand with the preferences and determines the final roster. The planning period of this type is also a few weeks.

6. *Self rostering*
   Staffing demand is again determined by the employer. At self rostering, employees are fully responsible for making a roster within the restrictions of the organization, sometimes with help of software. The planning period ranges from 4 to 12 weeks.

This list is designed by Nederlands Centrum voor Sociale Innovatie (NCSI). NCSI is a Dutch knowledge center that stimulates sociological innovations in the Netherlands. Social innovations intend to improve performance, job satisfaction, and stimulate talent development (NCSI, 2011).

Different forms of individual rostering are often combined. For example, it is possible to first make a repetitive roster in which structural wishes are covered and secondly allow swaps, so the incidental wishes of the employees are also covered.

## 2.2 Swedish method

The 'Swedish method' is used as a guideline of the self rostering process described in Section 1.3. Self rostering is very popular and used successfully in Sweden since 1990. Therefore, different methods in Sweden became examples for other countries (Paralax BV, 2010). There is not just one Swedish method: many companies take over the idea of self rostering and adjust it to a form that works for their company. The main idea of self rostering is that teams, departments, or employees are responsible for finding a feasible roster (Lubbers, 2008). In general, employees first propose their individual schedules after which

they have to negotiate until a feasible roster is created. A feasible roster is a roster that meets the staffing demand and labor legislations are not violated. Rönnberg and Larsson (2010) are the only ones found in literature who describe the process of a popular form of self rostering in Sweden in detail. This process consists of 5 steps, see Figure 2.2.


Figure 2.2: Popular form of Swedish self rostering, described by Rönnberg and Larsson (2010)

In step 1, employees create and propose a preferred individual schedule. These schedules are independent of the other employees and the preferences of the organization, but have to satisfy all labor legislations. In this step employees usually are able to indicate strong preferences on which shifts they like and dislike (to work). In step 2, these proposals are compared to the staffing demand and excesses and shortages are identified. In step 3, the intention is that employees trade shifts and compromise through informal negotiations. This process helps finishing the roster, but might stagnate at a certain point in time. Then, in step 4, a scheduling group (consisting of a few employees of the department), takes over. They have two jobs. Their first job is to verify whether the schedule satisfies all labor legislations. First, during the trading it might be difficult for the employees to fulfill all labor legislations, and the scheduling group should identify and correct all violations that are of significance. Second, the scheduling group has to finish the schedule, so to eliminate the residual of shortages and excesses. For each adjustment, they contact the involved employee(s) and try to come to an agreement. When that is no longer possible, the scheduling group makes the necessary adjustments anyway. When the roster is finished, it is sent to the person who is responsible for this roster (step 5). To some extent, some violations, shortages, and excesses are permitted. However, if the number of violations is exceeded, the responsible person does not accept the roster and the scheduling group has to adjust the schedule until the responsible person approves the schedule.

The self rostering process described in Section 1.3 is in almost all steps similar to the steps described above. Step 1 of the Swedish method is in our process Step 2 (see Section 1.3). Rönnberg and Larsson (2010) did not define the step were the organization defines the demand. Step 4 and 5 of the Swedish method are not the same as in our self rostering process. Our process does not describe who finishes the roster, so this could be a scheduling group, but it is more likely that just one planner finishes the roster. Step 5, approval of the head of the department, is not mentioned in our self rostering process. It is most likely that the planner (who finishes the roster) is the responsible person and approves the schedule. This is not an explicit step.

## 2.3 Self rostering in practice

This section describes cases and literature examples from practice. Section 2.3.1 describes the cases. Section 2.3.2 discusses the important criteria from practice for our method. Section 2.3.3 discusses self rostering examples found in literature.

### 2.3.1 Cases from practice

The objective of this research is to design a method that helps planners finalize the schedule in the last step of the self rostering process, such that this method is widely applicable. This section considers the 'widely applicable' part. To determine which criteria from practice the method should consider, we interview customers of ORTEC from various industries that are interested in self rostering. We interview: Organization X (transport and logistic), Pompestichting (health care - forensic), and Westfriesgasthuis (health care). In addition, NedTrain provides a case from the service industry.

In this section we give some general information about the organizations, except for Organization X, they want to stay anonymous. Appendix A describes the case of each organization in detail (their current scheduling process, their goal for using self rostering, and their criteria for our method).

#### Services - Nedtrain

NedTrain is a Dutch company specialized in maintenance and services (cleaning and revision) on rolling material, mostly concerning trains. NedTrain has over 30 service sites from which it operates 24/7. (NedTrain, 2011).

#### Health care (forensic) - Pompestichting

Pompestichting is a private institution for forensic psychiatry. The main goal of the Pompestichting is to contribute to the safety of the society by offering treatment for people with a psychiatric disorder who are likely to commit or already committed a serious crime. (Pompestichting, 2011)

#### Health care - Westfriesgasthuis

The Westfriesgasthuis is a general hospital with a yearly output of 506 operational hospital beds and 258,000 polyclinic visits. The medical staff represents 26 specialisms. (Westfriesgasthuis, 2011)

### 2.3.2 Overview criteria from practice

The criteria that the method certainly has to meet are the method's hard constraints and referred to as *requirements*. The soft constraints are referred to as *wishes*. Tables 2.1 and 2.2 give per case an overview of the requirements and wishes, respectively.

Next, we discuss these requirements and wishes and determine which are important to implement directly in our method (basic functionalities), and which we optionally implement later on (optional functionalities) to our method and

which we do not implement. At the end of this section, we give an overview of these basic and optional functionalities for our method.

#### Requirement 1, full weekends

NedTrain specified a red weekend in their collective labor agreement. This means that in any period of 3 weeks each employee has to be free for at least one weekend. If employees only schedule themselves for one shift per weekend, valuable work hours in weekends are lost. Therefore, the employees and the method have to schedule a full weekend (Saturday and Sunday). This requirement is important for NedTrain, but none of the other cases stated this requirement. Therefore, we classify this as an optional functionality for our method.

#### Requirement 2, ground rules

Organization X still has to choose between three ranking rules (Seniority, Point system, and Groups). These rules rank the employees. Based on this ranking, employees have a higher chance of receiving a shift re-assignment. The rules 'Seniority' and 'Groups' are further discussed in Appendix A. Point systems are further explained in Chapter 3. Point systems encourage employees to take the unpopular shifts and unpopular shifts are divided more equally. In our opinion, this is important for the self rostering process. So, we take the use of a point system into account in our method as a basic functionality.

#### Requirement 3, transparency

Organization X wants the self rostering process to be transparent, such that employees know why they received certain shifts and are encouraged to propose their preferred schedules. Since the encouragement of employees is in general important for self rostering, we classify this requirement as a basic functionality for our method.

#### Requirement 4, working hours act and collective labor agreement

All cases require the method to satisfy the working hours act (WHA) and collective labor agreements (CLA). Two cases (Pompestichting and Westfriesgasthuis) also indicate that the method should respect the violations approved by the planner. The method does not have to try to fix these. This requirement is a basic functionality of our method.

#### Wish 1, bonus system

This wish, the use of a point system as a bonus system, mentioned by Westfriesgasthuis is the same as requirement 2 and therefore already included in the method.

#### Wish 2, not-work wish

Employees may indicate, for a certain amount of hours, when they wish not to work. This extra information may help the method to create higher quality

Table 2.1: Requirements from the cases from practice

| No. | Requirement | Organization X | NedTrain | Pompestichting | Westfriesgasthuis |
|---|---|---|---|---|---|
| 1 | Full weekends | - | Schedule shifts on Saturday and Sunday together | - | - |
| 2 | Ground rules | Select employee and swap option based on predefined rules | - | - | - |
| 3 | Transparency | Method is understandable for employees | - | - | - |
| 4 | WHA and CLA | | | | |
| | a. Input | Satisfy | Satisfy | - | - |
| | b. Swaps | Satisfy | Satisfy | Satisfy | Satisfy |

Table 2.2: Wishes from the cases from practice

| No. | Wishes | Organization X | NedTrain | Pompestichting | Westfriesgasthuis |
|---|---|---|---|---|---|
| 1 | Bonus system | - | - | - | Use of point system |
| 2 | Not-work wish | Per employee two wild cards per period to indicate when they prefer not to work | Employees can indicate when they prefer not to work | - | For every week, employees may indicate a weekday on which they prefer not to work |
| 3 | Number of changes | Minimize over all schedules | Retain, on average per year, 80% of the proposed schedule of an employee | Retain at least 80% of each proposed schedule | Retain at least 80% of each proposed schedule |
| 4 | Planning forward in rotation | Plan, per employee, forward or backward in rotation | Plan forward in rotation | - | Plan forward in rotation |
| 5 | Work hours | Minimum hours is contract hours $-40$ and maximum is contract hours $+60$ per period | Minimum hours is contract hours $-16$ and maximum is contract hours $+16$ per period | Satisfy contract hours annually, flexible per period | Contract hours with a minimum and maximum number of hours per period |

schedules. This is important for the employees, but not necessary for the method to create feasible schedules. However, this functionality improves the quality of the schedules and therefore, we see this wish as optional functionality for our method.

#### Wish 3, number of changes

All cases want to minimize the total number of changes as much as possible. Pompestichting and Westfriesgasthuis want a minimum of at least 80% remaining of the preferred schedule of an employee. NedTrain wants this minimum of 80% on average over a year. If many wishes are not retained, employees are discouraged to propose their preferred schedules. Therefore, we see this wish as basic functionality for our method.

#### Wish 4, planning forward in rotation

Planning forward in rotation means that a shift on the next day begins at the same time or later than the shift on the day before. The organizations indicate that if an employee plans his shift forward in rotation, the method should also do this. However, with self rostering we are not sure whether, and which, employees are going to use forward rotation in their preferred schedule. So, we see planning forward in rotation as optional functionality for our method.

#### Wish 5, work hours

Wish 5 is mentioned in all cases. Annual contract hours means that an employee has to work a certain amount of hours per year, but does not have an exact amount of hours per week or month. All cases use annual contract hours. Organization X, NedTrain and Westfriesgasthuis mention a minimum and maximum of hours per scheduling period. With this wish, we are allowed to give an employee extra shifts or delete shifts in his month schedule. These cause a lot more swap options per employee. Next to this, we also have to keep track of the hours worked and a strategy for when we assign extra shifts to employees. To keep the base of the method simple, we retain the number of work hours in the preferred schedule. We see this wish as an optional functionality.

#### Overview basic and optional criteria

Table 2.3 gives an overview of the wishes and requirement that we classified as basic and optional functionalities for our method.

### 2.3.3 Examples from literature

In the following, we describe two examples from literature. Each example describes a different form of self rostering (or a form close to self rostering) implemented in practice. One example is about transport and one is about nurse rostering. After each example, we describe the advantages, disadvantages and whether the method used is applicable to the cases from practice (see Section 2.3.1).

Table 2.3: Basic and optional functionalities from the cases from practice (Impl. = Implementation, Req. = Requirement)

| Impl. | No. | Rule | Explanation |
|---|---|---|---|
| Basic | Req. 2 | Ground rules | Method should select employee and swap option based on ground rules. |
| | Req. 3 | Transparency | Method is understandable for employees. |
| | Req. 4 | WHA and CLA | Each swap option should satisfy the working hours act and collective labor agreement. |
| | Wish 1 | Bonus system | Use a point system to select employees. |
| | Wish 3 | Number of changes | Minimize the number of changes and have the possibility of a minimum percentage of remaining schedule at all employees. |
| Optional | Req.1 | Full weekends | Schedule shifts on Saturday and Sunday together. |
| | Wish 2 | Not-work wish | Method takes not-work wishes into account. |
| | Wish 4 | Planning forward in rotation | Method should plan forward in rotation. |
| | Wish 5 | Work hours | Use the contract hours of the employee with a certain minimum and maximum number of hours an employee has to work per period. |

**Transport industry - Arriva Multimodaal**

Arriva Nederland exploits transport of different buses through the Netherlands and trains in the provinces Groningen and Friesland. NCSI (2009)

Employees are currently scheduled within a block system. There are 7 blocks each day and each block has a length of 10 hours. For example, block A starts at 4:00 am and ends at 2:00 pm, block B begins at 6:00 am and ends at 4:00 pm. The last block, block G, begins at 11:00 pm and ends at 7:00 am.

First, employees propose a schedule for a whole year. Then the planner can reassign employees to blocks where the staffing demand is not yet met. Each change implies negative points for the employee. The bigger the change the more negative points the employee receives, for example: if the employee preferred block A (4:00 am to 2:00 pm) on a specific day and he is assigned to B (6:00 am to 4:00 pm), he receives -1 point, if he is assigned to C (8:00 am to 6:00 pm), then he receives -2 points etc. The goal is that all employees have about the same amount of points at the end.

The advantage of this method is that two blocks that more overlap in work time receive fewer negative points. In this way, points indicate the 'size' of the change.

The disadvantage of this method is that employees with fewer contract hours or sick employees have fewer work hours in which they have to earn the same amount of points as the rest of the employees. Therefore, these employees have a higher ratio of unpreferred blocks in their schedule. The authors do not describe some kind of scale or compensation method.

Arriva uses a planning period of one year. We think that the method described is applicable to the cases from practice (see Section 2.3.1) if the planning period is reduced to one month and the point system is adjusted to the shifts of an organization. Also an extension that compensates for employees who work fewer hours should be in the system. The planning period needs to be reduced, because the employees do not know all their (incidental) wishes and the organization does not know its fluctuation in staffing demand a year in advance. The point system needs to be adjusted, because not all cases have such a standard schedule as public transport. At Arriva a new block starts every two hours, we expect that the shifts in the cases are not scheduled in such a structure.

The method used in this example, with the adjustments described before, forms a good basis for the main goals of the cases: (1) help employees to balance their personal and work responsibilities and (2) balance the staffing demand with the work demand.

#### Health care - Nurse scheduling 1

The main goal of the research of Rönnberg and Larsson (2010) is to create an automated system that is practically identical to the manual self rostering process described in Figure 2.2 (Section 2.2). Therefore, they define five kinds of requests that employees can use when they create their schedule, see Table 2.4.

Rönnberg and Larsson (2010) automated the system, therefore they excluded the trading step and lost information. Information such as the importance of a shift in the preferred schedule of an employee (in the first step of the process) is lost. Normally, this is clear in the trading step, when employees choose to trade or stay with their preferred schedule. To compensate for this loss, Rönnberg and Larsson (2010) created two other types of requests: a strong requests for working a shift and a strong request against working a shift (see Table 2.4).

Table 2.4: Types of requests

| **Color** | **Request** | **Extra information** |
|---|---|---|
| Purple | Individual task | Must be approved |
| Black | Holiday | Must be approved |
| Blue | Veto for working a shift | |
| Red | Veto against working a shift | |
| White | request for working a shift | |
| Green | Strong request for working a shift | Extra request |
| Yellow | Strong request against working a shift | Extra request |

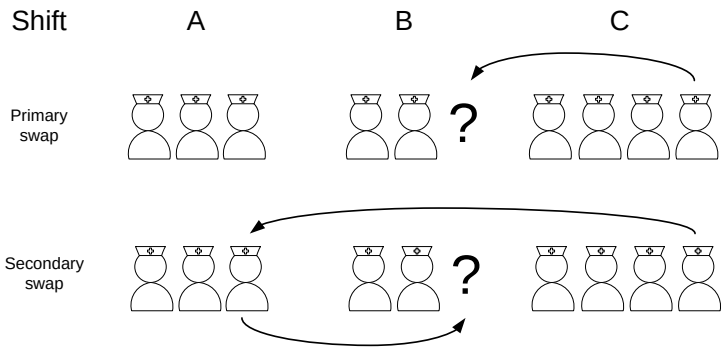



Figure 2.3: Secondary Swap (based on Figure 5 of Rönnberg and Larsson (2010))

Rönnberg and Larsson (2010) divided the requirements in hard and soft requirements. Purple, black, blue, and red request are considered as hard request and must be fulfilled. Green, white, and yellow requirements are considered soft requirements: these should be fulfilled if possible.

With these requirements employees are able to create their preferred schedule. After that, the automated system takes over and finalizes the schedules using an optimization model. This optimization model minimizes the number of shifts staffed by substitute employees and maximizes the fulfillment of requests in the proposed schedules in a fair way. The model also has constraints to satisfy. These can be summarized into 4 groups: staffing demand, scheduling rules, quality aspects, and auxiliary constraints.

The advantage of this method is in our opinion the use of secondary swaps. A secondary swap considers a situation where two employees have to swap shifts to fulfill the staffing demand for one shift on a certain day. For example, all shifts have a staffing demand of three employees (see Figure 2.3). Shift B has a shortage and C an excess, which can be solved with a primary or a secondary swap.

Despite the introduction of the two new requests, we think the disadvantage of this method is that the control of the employee is reduced. Employees can make the request and after that, they do not have any influence on their schedule.

If we look at the main goals of the cases, we think this method is applicable. Although, to let this system work in all cases, the criteria in the optimization model have to be generalized. In the model described in this example, many constraints are organization specific.

### Basic and optional functionalities from literature

The most important aspect of the two examples is in our opinion, the use of secondary swaps (Nurse scheduling). They allow a method to fulfill the staffing demand when we cannot fulfill it with only primary swaps. We do not see this as a necessary requirement for the method. Therefore, we classify this as optional functionality for our method.

We think that the point system that is based on the overlap in shifts (Arriva) is interesting. The use of a point system is already described and classified as a basic functionality for our method (see Table 2.3, Wish 1).

## 2.4 Conclusions

Requirements and wishes with respect to a self rostering method are provided by cases from the industries, health care, transport, and services. We see that the cases share a lot of requirements and wishes. For example, all cases require that the method should not create new violations with the working hours act and the collective labor agreements. Another example is that all cases wish that the number of changes in the preferred schedules is minimized. An overview of all requirements and wishes is found in Tables 2.1 en 2.2. In Table 2.3 we defined requirements that must be met in our method (basic functionalities), next to wishes that are regarded as optional functionalities.

In addition, literature provided two applications of self rostering in practice. From these, we determine additional functionalities for our method. One of the examples shows that secondary swaps may be useful (see Section 2.3.3). Hence, we take secondary swaps into account as extra optional functionality of our method.

# Chapter 3

# Self rostering process

After we derived the basic and optional functionalities from literature and practice for our method (see Chapter 2), this chapter describes the shortcomings of the current self rostering process (Section 1.3), and what ORTEC did to overcome these. First, Section 3.1 describes the shortcomings. Next, Section 3.2 describes and motivates the necessary process adjustments to overcome these shortcomings. Section 3.3 describes the working of point systems and finally, we describe our conclusions of this chapter in Section 3.4.

## 3.1 Shortcomings current self rostering process

This research is based upon the self rostering process described in Chapter 1 (see Figure 1.1). This process works well for small groups. For large groups, we identify four main shortcomings:

1. Negotiations between employees in step 4 are time consuming or impossible in large groups.

2. Employees who are not good at negotiations receive less favorable schedules.

3. Employees have an unfair feeling about the allocation of the unpopular shifts by the planner.

4. Violations of the scheduling rules are easy to miss.

## 3.2 Process adjustments

To overcome the four main shortcomings (described in Section 3.1) and to help the employees to find a suitable schedule, ORTEC implemented a point system in the process. To do this, adjustments to the process were needed. The steps of the self rostering process after adjustment, are shown in Figure 3.1.

The first adjustment is in step 3: next to identifying the shortages and excesses, points per shift are now defined and assigned to the employees with these shifts. Shifts with shortages receive most points and shifts with excesses

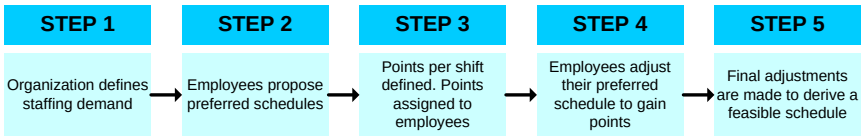



Figure 3.1: Self rostering process adjusted

receive fewest points. In this way, it is clear that an employee with many points has many unpopular shifts and the other way around.

The next adjustments are in step 4 and 5: next to the adjustments of the employees, employees gain more points for each adjustment. The points are defined per shift per day. These are fixed for the rest of the planning period (steps 4 and 5). The employees receive the sum of all points assigned to the shifts they have in their schedule. If employees are not satisfied with their total points, they can adjust their schedule in step 4. During this adjustment, the points of the old shift are subtracted and the points of the new shift are added to the total points of the employee. As mentioned before, points in step 4 and 5 refer to the fixed points defined in step 3. For example, if an employee swaps from an overstaffed shift (1 point) to an understaffed shift (3 points), the total points of the employee changes by -1+3 points.

#### Overcome the shortcomings

Employees can only swap from shifts with excesses to shifts with shortages. Therefore, this reassignment always leads to more points for the employee. More points means more unpopular shifts and thus decreases the chance of being swapped in the last step, which creates a more honest feeling about sharing the unpopular shifts (shortcoming 3).

The next advantage of this adjusted process is that time consuming negotiations are not needed anymore. By automating the system, employees can modify their schedule online. With this system, not only employees who are good in negotiations can create good schedules (shortcoming 1 and 2).

Another advantage of an automated system is that the system checks on scheduling rules. Therefore, it is no longer possible to propose a schedule that does not satisfy the scheduling rules. In the next steps these rules are checked at every action. This causes an extreme reduction of time needed by the planner and the chance of missed violations in the final roster decreases (shortcoming 4).

## 3.3 Point systems

This section gives details and a motivation for the point systems chosen.

ORTEC considered 6 different point systems. In this section, we describe systems 1 to 3. These are the main systems. Systems 4 to 6 are variants on the first three systems. In Appendix B we described examples of point system 1 to 3 and the three variants (point systems 4 to 6).

Every system takes the different amount of contract hours, sick leave, and vacation days of the employees into account by considering the average points

per shift per employee. We call this *the score* of an employee. We illustrate this in Example 1.

> **Example 1.** In Table 3.1 we see that De Jong has the fewest number of shifts assigned and the lowest total points. However, his average points is higher than the average points of Jansen. So, De Jong is in proportion assigned to more unpopular shifts than Jansen and receives a higher rank.

Table 3.1: Example points and score (TS = Total amount of shifts, TP= Total points, and Sc = Score)

| Employee | TS | TP | Sc | Rank |
|---|---|---|---|---|
| Jansen | 25 | 42 | 1.68 | 3 |
| De Vries | 23 | 45 | 1.96 | 1 |
| De Jong | 17 | 29 | 1.71 | 2 |

## Point system 1

Point system 1 is the most static and therefore transparent one. With this point system understaffed, matching, and overstaffed shifts receive a fixed amount of points. So, the size of the shortage or excess on the shift does not affect the points. In detail, all shifts with a shortage receive 3 points, all matching shifts 2, and shifts with excesses 1. An organization can adjust the points on condition that the unpopular shifts receive more points than the popular shifts.

A great advantage of this system is that it is simple and thereby transparent. Ratios between the points for understaffed, matching, and overstaffed shifts can be used to encourage employees to change their schedule in step 4.

The disadvantage of this system is that a slightly understaffed shift receives the same number of points as a highly understaffed shift. With this point system employees are not encouraged to choose the highly understaffed shifts first.

## Point system 2

Point system 2 is a bit more advanced. Points are defined as the difference between the number of employees who signed up for the shift (actual assignment) and the staffing demand. Negative points mean that more employees signed up for the shift than necessary.

The advantage of this system is that highly understaffed shifts receive more points than slightly understaffed shifts. So in step 4, employees are encouraged to choose the highly understaffed shifts instead of just an understaffed shift.

In discussion with a customer, ORTEC found out that negative points are hard to understand for the employees of the customer. Therefore, the negative points in this point system are a disadvantage. This system goes to zero, which also turned out hard to understand and calculate with. Another disadvantage is that the score of an employee can become lower instead of higher when he plans an extra shift in step 4. For example if an employee has planned shifts on Monday, Tuesday and Wednesday in the first step and each shift is 15 points worth, his total is 45 and his score is 15. Then in step 4 he sees that his

Table 3.2: Popularity classes - Point system 3

| **Popularity** | **Points** |
|---|---|
| 0% - 49% | 5 |
| 50% - 99% | 4 |
| 100% - 100% | 3 |
| 101% - 149% | 2 |
| >150% | 1 |

score is low compared to the other employees and want to increase his total points, so he plans an extra shift on Thursday, which was understaffed, but not as understaffed as the shifts he already has. This shift is 5 points worth, so his total points becomes 50 and his sore is now 12.5. By planning an extra understaffed shift on Thursday, which is good for the total schedule, he lowers his score and as a consequence he may not plan the shift.

## Point system 3

Point system 3 uses the popularity of a shift to determine the points. Popularity is calculated in percentages: total amount of employees who signed up for the shift (actual assignment) divided by the staffing demand, see equation 3.1.

$$\text{Popularity} = \frac{\text{Actual assignment}}{\text{Staffing demand}} * 100\% \tag{3.1}$$

To assign points to each shift, we define different classes of popularity and assign points to each class (for an example see Table 3.2).

A problem occurs when the staffing demand is zero, since we cannot calculate the popularity with equation (3.1). In this case, we look at the total amount of employees who signed up for the shift. When this is also zero, we assign the shift to the class where the actual assignment matches the staffing demand (100%). When this amount is greater than zero, we assign the shift to the highest popularity class.

The advantage of this system is that highly understaffed shifts receive more points than slightly understaffed shifts. So in step 4, employees are encouraged to choose the highly understaffed shifts instead of just an understaffed shift.

The disadvantage of this system showed up in a discussion between ORTEC and its customer. It turned out that this system is not transparent enough to the employees of the customer. Employees find it hard to work with several classes of popularity percentages. Another disadvantages of this system is that, just like point system 2, it is possible to get a lower score by planning an extra shift.

## Point systems used

The initial idea of ORTEC was to use point system 3, but after bringing this to practice it appeared to be too hard to understand for the employees. Point

system 2 was also too hard to understand because of the negative points, so ORTEC chose point system 1.

To study the influence of the point systems on the method, we test the method on all three point systems.

## 3.4 Conclusions

To overcome shortcomings such as time consuming negotiations, unfair feelings about the final adjustments, or missed scheduling violations by the planner, ORTEC implemented a point system in the self rostering process. This affects the process in steps 3 and 4 (see Figures 1.1 and 3.1). In this new process (see Figure 3.1), all employees receive points for each shift in their preferred schedule in step 3. Understaffed shifts are more valuable than overstaffed shifts thus, we know that employees with many points work many unpopular shifts. Within step 4, employees can now rearrange their own preferences to gain more points, this can be done online and without any negotiation with other employees. So, the time consuming negotiations are excluded and the unfair feeling is justified by the differences in points per employee. By extracting the negotiations, the process can be automated and the automated system can check on scheduling violations. So, the probability that the planner misses these becomes smaller. ORTEC considered 6 point systems and chose to use point system 1 (see Section 3.3 and Appendix B). We want to study the effects of the different point systems, therefore we test our method on point system 1, 2, and 3.

# Chapter 4

# Method

In Chapter 2, we defined basic and optional functionalities for our method. In Chapter 3, we described the shortcomings of our self rostering process and introduced an adjusted self rostering process which makes use of a point system. In this chapter we explain our method. First, this chapter repeats the basic functionalities for our method and describes the necessary assumptions (see Section 4.1). Next, we describe our method (Sections 4.2). Sections 4.3 to 4.5 describe the mathematical models we use in our method. Finally, we draw our conclusions of this chapter in Section 4.6.

## 4.1 Basic functionalities

The main goal of this research is to help the planner finalize the schedule in the last step of the self rostering process. To finalize a schedule, the planner must swap shifts in the schedules of the employees. After each swap, the new schedule has to satisfy the working hours act and collective labor agreements (see Table 2.3, requirement 4). Another objective is that the number of changes is minimized, with a constraint on the minimum percentage of the preferred schedule retained (see Table 2.3, wish 3). The method has to choose which employees to swap, based on their scores. An employee with a low score, has many popular shifts in his schedule, so he has to have a higher chance to be picked to swap shifts (see Table 2.3, requirement 2 and 3, and wish 1).

### 4.1.1 Assumptions

To translate the problem from practice to a mathematical method, we have to make several assumptions. These assumptions are necessary to keep the method from becoming too complex, without deviating too much from practice. The assumptions are:

1. *There are three shift types*
   We have three non-overlapping types of shifts: a day (A), evening (B,) and a night shift (C).
2. *All shifts have the same length (8 hours)*
   This implies that: if the model swaps an employee from a shift to another shift, his total working hours remain the same.

3. *Employees work complete shifts*
   Employees work the whole shift; they cannot be partly assigned to a shift.

4. *Employees propose a schedule in which they precisely cover their total work hours*
   This assumption combined with the assumption 2, gives us the ability to conclude that if we only swap a shifts for another shift, the total work hours are always covered.

5. *A swap can only be made from an overstaffed shift to an understaffed shift*
   If we swap a matching shift for another shift, this matching shift becomes understaffed. So, we need another swap to fulfill the staffing demand of this new understaffed shift. This causes extra swaps and a decrease of percentage of the remaining preferred schedule. The same holds for swaps from overstaffed to overstaffed or matching shifts and swaps from understaffed shifts to all other shifts. Therefore, we only swap overstaffed shifts for understaffed shifts. In Section 4.5 we relax this assumption and allow swaps from and to matching shifts.

6. *The process of scheduling is finished when:*

   (a) *all overstaffed shifts are eliminated, or*

   (b) *all understaffed shifts are eliminated, or*

   (c) *swaps are not possible anymore.*

   The planner wants to satisfy the staffing demand, so if the schedule meets the staffing demand, the schedule is finished. However, there are situations where the staffing demand and the employee availability do not match. Then the last understaffed or overstaffed shifts cannot be eliminated. In these cases, the schedule is also finished when there are still overstaffed or understaffed shifts. We assume that a swap can only be made from an overstaffed to an understaffed shift. So, when these swaps are not available anymore, the staffing demand cannot be fulfilled and the scheduling process is also finished.

7. *Collective labor agreements are not taken into account, the working hours act are taken partly into account*
   Collective labor agreements differ per organization. We want to design a suitable method that is widely applicable, so we implement only the working hours act (see Appendix C for an overview of the working hours act). We want to keep the scheduling process as realistic as possible, but since not all requirements are necessary for testing the method, we implement a selection of requirements from the working hours act. In consultation with ORTEC, we select the most important requirements, see Section 4.1.2.

### 4.1.2 Working hours act

In consultation with ORTEC we decided to implement 3 requirements from the working hours act, see Table 4.1. To cover these requirements we need to make the following additional assumptions.

1. *The maximum length of a workweek is 6 consecutive days, then at least one day has to be of non-work time.*
   This assumption implies that every week contains 24 consecutive hours of non-work time (weekly rest). This is in case the last shift of the series is a night shift (C) and the first shift of the next series is a day shift (A). Two other situations have a weekly rest of 32 hours and six other have a weekly rest of 40 hours or more. This way, we cover the requirement of 36 hours of weekly rest in the working hours act in most situations.

2. *Shift A and B are not allowed, after a night shift C.*
   The working hours act states that an employee is entitled to a rest time of 14 hours after a night shift. With the assumption we cover this requirement.

3. *It is not allowed to work two consecutive shifts and there is a maximum of one shift per day*
   The working hours act states that each employee is entitled to a daily rest of at least 11 hours per day. With this assumption and the previous assumptions we cover this in two of the three situations. The three situations are:

   (a) An employee has a day shift (A), the first allowed consecutive shift is a day shift (A) on the next day. This means a daily rest of 16 hours.

   (b) An employee has an evening shift (B), the first allowed consecutive shift is a day shift (A) on the next day. This means a daily rest of 8 hours.

   (c) An employee has a night shift (C), the first allowed consecutive shift is a night shift (C) on the next day. This means a daily rest of 16 hours.

4. *Breaks are included in the predefined shifts*
   The working hours act states that if a shift last longer than 5.5 hours, a break of 30 minutes is required. For the model, it is not necessary to specify the activities within a shift, as long as the begin and end time of a shift are known and the model knows whether it is a day, evening, or night shift (due to other requirements in the working hours act). That is why we assume that the 'break' is within the predefined shifts.

Table 4.1: Covered requirements

| **Category** | **Requirement** |
|---|---|
| [Resttime] | Daily rest: 11 hours |
| [Resttime] | Weekly rest: 36 hours (consecutive) |
| [Nightwork] | Resttime after night shift: 14 hours |

Using these assumptions, we cover more requirements of the working hours act. Table 4.2 gives an overview of these additional covered requirements.

Table 4.2: Additional covered requirements

| Category | Requirement | Explanation |
|---|---|---|
| [Breaks] | All | We assumed that all necessary breaks are within the predefined shifts. |
| [Night work] | Maximum working time per shift: 10 hours | We assumed that shift C, the night shift, is 8 hours. |
| [Night work] | Maximum length of series (with at least one night shift): 7 shifts | The maximum length of series is already set on 6 (see [Resttime] Weekly rest). |
| [Working time] | Per shift: max 12 hours | We assumed that all shifts (A,B, and C) have an equal length of 8 hours. |
| [Working time] | Per week: max 60 hours | The maximum length of series is set on 6 (see the implementation at [Resttime] Weekly rest). All shifts have the length of 8 hours. So, the maximum work time per week is $6 * 8 = 48$. |

## 4.2 Method

In this section, we describe our method to help the planner finalize the schedule in the last step of the self rostering process (see Section 4.1).

We use an method that modifies the schedules of the employees in an iterative process. However, before we start the iterations, we have to do some *preprocessing*, which is further explained in Section 4.2.1. After preprocessing, the method starts its iterations. Every iteration has three main phases: *employee selection, swap selection,* and *updating*. In Section 4.2.2, *employee selection*, we select employees and determine the possible swap options in their schedules. In Section 4.2.3, *swap selection*, we calculate which (combination of) swap option(s) are optimal. Finally, in Section 4.2.4, *updating*, we process all chosen swap options.

### 4.2.1 Preprocessing

The process we describe in this section takes place before the iterations start. At preprocessing, we load the preferred schedules and staffing demand and set the value for the 5 input parameter: *Point system, Number of initial employees, Minimum percentage, MILP,* and *Swap strategy*. For the input parameter *Point system* we consider the options: point system 1, point system 2, and point system 3 (see Chapter 3). We explain the functionality of the other input parameters in this chapter and define their possible settings in Chapter 5.

In this phase, we also calculate the scores of the employees based up on their preferred schedule and the chosen point system (see Section 3.3 for the different point systems and calculation of the scores).

### 4.2.2 Employee selection

At the start of each iteration, *employee selection* takes place. The method starts with selecting employees after which it creates all possible swap options in the schedules of these employees.

The selection of employees is based on the scores of the employees and the total number of employees, $z$ (input parameter: *Number of initial employees*), we want to take into account in this iteration. All employees are sorted in ascending order of their score and the first $z$ employees are selected.

If an employee is assigned to an overstaffed shift, it is possible to swap this employee from this overstaffed shift to an understaffed shift. However, this swap is only possible if (1) there is not already a shift for this employee on the day and time of the 'new' understaffed shift and (2) if the resulting schedule of the employee after the swap still satisfies the working hours act. All swaps that satisfy these rules, are called the 'swap options' of an employee.

The swap options in the schedule of an employee are only calculated when an employee is selected for the first time. If an employee was selected before, his swap options were calculated in a previous iteration and updated at the end of the iteration (see Section 4.2.4). In this way, the swap options do not have to be calculated again.

An extra functionality in our method is to retain at least $p$ percent (input parameter: *Minimum percentage*) of the preferred schedule of an employee. For this, we calculate whether a swap of a selected employee causes a percentage retained of his preferred schedule below $p$. If so, we delete all possible swap options of the employee such that no additional swaps are possible. If a swap does not cause a percentage retained below $p$, we do nothing and continue the method.

At the end of the *employee selection* phase, we have all possible swap options of the selected employees.

### 4.2.3 Swap selection

To finalize the schedule it may happen that we have to make multiple swaps in the schedule of one employee. However, if we select several swap options in the schedule of one employee simultaneously, we do not know whether shifts overlap and whether the schedule of the employees still satisfy the working hours act. To illustrate this, we look at Example 2.

> **Example 2.** In the preferred schedule (see Table 4.3), it is allowed to swap shift A from Wednesday for shift C on Monday. Another swap option is to swap shift A on Thursday for shift A on Tuesday. Both options are allowed separately, but not together: it is not allowed to work a day shift (A) directly after a night shift (C). Based on this consideration, we select at most one swap option per employee per iteration.

Table 4.3: Preferred schedule of an employee

| | **Monday** | **Tuesday** | **Wednesday** | **Thursday** |
|---|---|---|---|---|
| Employee | - | - | A | A |

So, given the fact that we select at most one swap option per employee per iteration, we have to determine which one. Note that if there is more than one employee selected, we can combine different swap options for various employees and that some combinations may fulfill more gaps than others. We show this with another example, Example 3.

**Example 3.** There are two employees selected, Figure 4.1 shows their preferred schedules. We have a day shift (A) and an evening shift (B). There is no demand for shift A on day 1, however on both day 2 and 3 the demand for shift A is 1. In Figure 4.1 we see that both employee 1 and employee 2 have shift A on day 1. Shift A on day 1 is therefore overstaffed. Employee 1 has one swap option: unassign shift A on day 1 and assign shift A on day 3. Employee 2 has two swap options: (1) unassign shift A on day 1 and assign shift A on day 2, or (2) unassign shift A on day 1 and assign shift A on day 3. When we select the swap option of employee 1, swap option 2 for employee 2 is not valid anymore because the staffing demand is already met. So, we can make two combinations. The first combination is swap option 1 at both employee 1 and 2. The second combination is: select no swap option at employee 1 and swap option 2 at employee 2. This first combination fulfills two gaps and the second just one gap, so, we prefer the first combination.

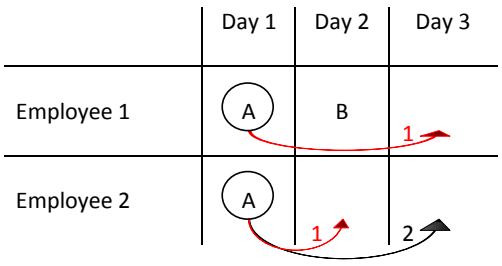



Figure 4.1: Example 3, combination of swap options

We use a mixed integer linear program (MILP) to find the optimal combination. A linear program (LP) is a mathematical model for optimization and consists of constraints and an objective. The constraints are linear equations of the form $\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad (i = 1, 2, \ldots, m)$ . The linear objective function $\left(\sum_{j=1}^{n} c_j x_j\right)$ defines what is to be optimized. An LP is of the form:

$$\text{Maximize or Minimize } \sum_{j=1}^{n} c_j x_j$$

$$\text{Subject to } \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, 2, \ldots, m)$$

$$x_j \geq 0 \qquad (j = 1, 2, \ldots, n)$$

$a_{ij}$, $b_i$, and $c_j$ are known coefficients. The decision variables $x_j$ have to be calculated. The problem is called a *mixed integer* linear program (MILP) if some of the variables have to take integer values, while others can take continuous values. See for further explanation of a (M)ILP: Chvátal (1983).

We start with the basic MILP in Section 4.3. The aim of this MILP is to minimize the difference between the staffing demand and the actual employee assignment as much as possible. After this, in Sections 4.4.1 and 4.4.2, we extend this basic MILP, such that we are also able to prefer swaps of employees with low scores. We use the input parameter *MILP* to define which of the MILP's we use: the basic MILP or the basic MILP with one of the extensions.

At the end of this phase we have calculated the optimal combination of swap options, given the selected employees.

### 4.2.4 Updating

From the swap selection phase we get the optimal combination of swap options, in this phase we process these selected swap options. The selected swaps are processed consecutively. Each selected swap option is first processed in the schedule of the employee. Then, the points and score of the employee are updated. Next, we update the swap options of the employee. It is possible that the employee had several swaps based on the same shift. If one of these swaps is chosen, the others become useless, so we delete them. Furthermore, some swap options now may lead to violations of the working hours act. We also delete these. In addition, new swap options may arise, we create these.

If the old or new shift from the swap now matches its staffing demand, the swaps at all other employees to or from these shifts are useless as well. So, we look at the swap options of *all* employees and delete these swap options. In the next section we discuss this process of updating the swap options in more detail.

#### Update swap options

There are two updates regarding the swap options. The first update is an update of the swap options of just one employee: the employee where two shifts were swapped. The second update is an update at all employees. This last one is only performed if the removed and/or added shift of the swap now matches the staffing demand.

*Update at one employee*

If we swap two shifts in the schedule of an employee, some swap options of the employee become invalid and new swap options may arise.

After performing the swap in the schedule, we perform 10 checks to see whether there are new swap options or whether some swap options became invalid. These 10 checks are divided in 2 times 5 checks. The first 5 checks are done on and around the day of the removed shift and the second set of 5 checks are performed on and around the added shift.

Within the first 5 checks we check the day of the old shift and days around the old shift. The day of the old shift becomes empty in the schedule. Therefore, we check whether there are still understaffed shifts on this day to create new swap options (check 1). If not, we are done. If so, we have to check the *day before* (check 2) and *after* the old shift (check 3) whether these contain shifts that revoke the possibility of placing the newly found understaffed shift in the schedule. When this is allowed, we check the *day before the series* (check 4) and *the day after the series of shifts* (check 5) whether we still meet the weekly rest when placing the newly found understaffed shift.

Next, we perform 5 checks on the day of and around the new shift in the schedule. The first check is whether there were other swap options to the day of the new shift (check 6). With this new shift placed, these are not possible anymore, so we delete these. Next, we check the *day before* (check 7) and *after* (check 8) the day of the new shift. We check these on having already a shift assigned. If not, we have to check whether swap options to these days exists, because some of these could be impossible due to the new shift. So, we have to delete these. For example, when the new shift is a night shift (C), then swap options to the morning shift (A) or day shift (B) of the next day are not allowed anymore. Another example is when our new shift is a morning shift (A), then a swap option to night shift (C) on the day before is not allowed anymore. So, we have to check whether these exist and delete them. The last two checks are performed on the *day before* (check 9) and *after* (check 10) the series of shifts. At these days, we check whether the weekly rest is still enough when we swap a shift to these days. When this is not enough, we have to delete these swap options.

*Update at all employees*

If a shift was overstaffed and matches the staffing demand after a swap, other swap options from this shift have to be deleted, otherwise we create a shortage of employees on an initially overstaffed shift, this contradicts assumption 5.

If a swap turns an understaffed shift into a matching shift, swaps to this shift are not needed any more, and are thus deleted.

## 4.3 Basic MILP

Our MILP selects for a given set of employees the optimal set of swap options. Optimal depends on the objective of the MILP. In this section, we define the parameters, indices, decision variables and define and explain the MILP. Appendix D summarizes the basic MILP.

### Parameters and indices

First, we define a set of employees, $I$, and a set of shifts, $K$, within a planning period. We define every shift as unique, so if for example shift A occurs on Monday and on Tuesday, these are seen as two different shifts $k \in K$. Next, we define a set of all swap options $J$. The set of swap options for employee $i \in I$ is given by $J_i \subset J$.

A swap option $j \in J_i$ is described by two parameters $j = (j_1, j_2)$. If swap option $j \in J_i$ is selected, we add $j_1 \in K$ and remove $j_2 \in K$ from the schedule of employee $i$. Note that $j_1$ and $j_2$ do not have to be on the same day.

The difference between the number of employees that is assigned to shift $k$ and the staffing demand for shift $k$, is denoted in the difference parameter $v_k$. The difference parameter is negative when the number of assigned employees is less than the staffing demand.

### Variables

Applying swap options changes the difference parameter $v_k$. The resulting difference parameter after applying a set of swaps is denoted by the variable $n_k$.

The decision variable $x_j$ denotes whether swap option $j \in J_i$ of employee $i \in I$ is applied:

$$x_j = \begin{cases} 1 & \text{if swap option } j \in J_i \text{ of employee } i \in I \text{ is applied} \\ 0 & \text{otherwise} \end{cases}$$

### Constraints

There are four constraint that have to be satisfied in the basic MILP. First, we may apply at most one swap option per employee (see Section 4.2.3):

$$\sum_{j \in J_i} x_j \leq 1 \qquad i \in I \tag{4.1}$$

The second constraint defines the new difference variable $n_k$. For this, we define two additional sets. The first set, $IN_i^k$, defines all swap options of employee $i \in I$ where the entering shift is equal to shift $k \in K$:

$$IN_i^k = \{j = (j_1, j_2) \in J_i | j_1 = k\} \tag{4.2}$$

The second set, $OUT_i^k$ analogously defines all swap options of employee $i \in I$ where the removing shift is equal to shift $k \in K$:

$$OUT_i^k = \{j = (j_1, j_2) \in J_i | j_2 = k\} \tag{4.3}$$

To calculate $n_k$, we add 1 to the difference parameter $v_k$ for each entering shift $\left(\sum_{j \in IN_i^k} x_j\right)$ and subtract 1 for each removed shift $\left(-\sum_{j \in OUT_i^k} x_j\right)$. We add and subtract the selected swap options of all employees $i \in I$. Hence, $n_k$ is defined as:

$$n_k = v_k + \sum_{i \in I} \left( \sum_{j \in IN_i^k} x_j - \sum_{j \in OUT_i^k} x_j \right) \qquad k \in K \tag{4.4}$$

To prohibit that overstaffed shifts become understaffed after applying a set of swaps, we introduce the set $OVER = \{k \in K | v_k > 0\}$ which represents the set of shifts that are currently overstaffed. The following constraint ensures that all shifts in $OVER$ do not become understaffed:

$$n_k \geq 0 \qquad k \in OVER \tag{4.5}$$

Analogously, the set $UNDER = \{k \in K | v_k < 0\}$ and the following constraint ensure that understaffed shifts do not become overstaffed:

$$n_k \leq 0 \qquad k \in UNDER \tag{4.6}$$

The objective of our basic MILP is to minimize the number of the overstaffed shifts. All swap options contain an overstaffed and understaffed shift. So, by minimizing the number of overstaffed shifts, the understaffed shifts are also minimized.

$$\text{Minimize} \quad \sum_{k \in OVER} n_k \tag{4.7}$$

## 4.4 MILP with employee preferences

The basic functionalities for our method (Table 2.3) require a bonus system for employees who work many unpopular shifts. As explained in Chapter 3, we use a point system. Points are assigned to the shifts. Employees with relatively many unpopular shifts in their schedule receive many points and therefore also a smaller chance of a reassignment of shifts in their schedule.

With the basic MILP a point system only affects the *employee selection* phase and not the selection of swaps in the MILP. Therefore, we introduce several extensions to our basic MILP that incorporate a point system for the selection of swaps. These are discussed in Sections 4.4.1 and 4.4.2.

### 4.4.1 Lowest Score

In this extension, we prefer swaps of employees with the lowest score. For this extension, we add parameter $s_i$, which is the current score of employee $i \in I$. To prefer employees with the lowest scores, we add the score of an employee to the objective function if a swap option of this employee is applied. We use the scaling constants $\lambda_1$ and $\lambda_2$ to define the trade-off between the first and second part of the objective function:

$$\text{Minimize} \quad \lambda_1 \sum_{k \in OVER} n_k + \lambda_2 \sum_{i \in I} s_i \sum_{j \in J_i} x_j \tag{4.8}$$

### 4.4.2 High score changes at low scores

In this extension, we prefer to apply swaps that cause a high score change of employees with a low score. The score change for swap $j \in J_i$ of employee $i \in I$ is the points from the entering shift, $j_1 \in K$, minus the points from the removed shift, $j_2 \in K$.

In this extension, we prefer employees who work most popular shifts to receive the most unpopular shifts. Note that this does not work with point system 1, since all overstaffed shifts receive the same points as well as all understaffed shifts, whereas point system 2 and 3 distinguishes between slightly and highly overstaffed and understaffed shifts.

We propose two alternative extensions in which we prefer high score changes in the schedules of employees with a low score, which we discuss next.

#### Multiplication

In addition to the score of the employee, $s_i$, we need to know the score changes of each swap option $j \in J_i$ for employee $i \in I$. For this, we introduce one additional parameter $w_j$.

The score change $w_j$ denotes the change in score $s_i$ of employee $i \in I$ that swap option $j \in J_i$ causes when it is applied. As explained in Section 3.3, scores are calculated as the average points per shift. So, the score change $w_j$ is calculated as the points received by assigning $j_1$ and subtracting the points of the unassigned shift $j_2$ divided by the total number of shifts.

To stimulate swaps of employees with a low $s_i$ we multiply $s_i$ and $w_j$ for all applied swap options. Using the scaling constants $\lambda_1$ and $\lambda_3$ the objective makes a trade-off between the number of overstaffed shifts and the combinations of employees' score and score change.

$$\text{Minimize} \quad \lambda_1 \sum_{k \in OVER} n_k + \lambda_3 \sum_{i \in I} s_i \sum_{j \in J_i} w_j x_j \tag{4.9}$$

#### Lower bound

In this extension, we stimulate to select swaps that cause a high score change of employees with a low score by maximizing the minimum 'new' score of all employees, $\tilde{w}$:

$$\tilde{w} \leq s_i + \sum_{j \in J_i} w_j x_j \qquad i \in I \tag{4.10}$$

The right hand side of Equation (4.10) denotes the 'new' score of employee $i$. In the objective of this extension, we use the scaling constants $\lambda_1$ and $\lambda_4$ to make a trade-off between the number of overstaffed shifts and the minimum 'new' score:

$$\text{Minimize} \quad \lambda_1 \sum_{k \in OVER} n_k - \lambda_4 \tilde{w} \tag{4.11}$$

Note that $\tilde{w}$ is most increased when the employees with the lowest scores receive the swaps that causes the highest score changes.

### 4.4.3 Combination basic MILP and extensions

It is possible to combine the basic MILP and all extensions in one MILP (for a complete overview, see Appendix D). The constraints are the constraints used in the basic MILP (see Section 4.3) combined with the constraint used for the

extensions (see Equation (D.7)). We control the extend of the objective of the combined MILP with the scaling constants $\lambda_1$ to $\lambda_4$:

$$\text{Minimize} \quad \lambda_1 \sum_{k \in OVER} n_k + \lambda_2 \sum_{i \in I} s_i \sum_{j \in J_i} x_j + \lambda_3 \sum_{i \in I} s_i \sum_{j \in J_I} w_j x_j - \lambda_4 \tilde{w} \quad (4.12)$$

## 4.5 Secondary swaps

The swap options we calculate in Section 4.2.2, are primary swaps. Primary swaps are swaps where an overstaffed shift is swapped for an understaffed shift. In literature we also found secondary swaps (Rönnberg and Larsson, 2010) (see Figure 4.2). In a secondary swap two swaps are performed to fill one gap. First, we swap an employee from an overstaffed shift to a matching shift, we refer to this as "secondary swap (1)". This matching shift becomes overstaffed and the second swap, a swap in the schedule of another employee from this shift to an understaffed shift, can be used to correct this overstaffed shift. We refer to this as "secondary swap (2)" (see Figure 4.3).

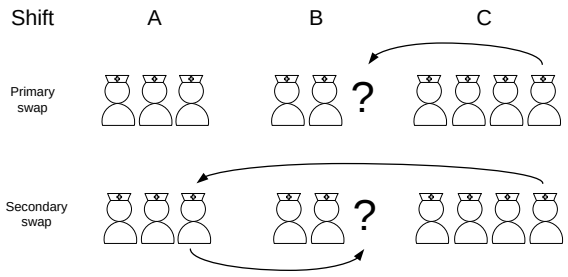



Figure 4.2: Secondary Swap (based on Figure 5 of Rönnberg and Larsson (2010))

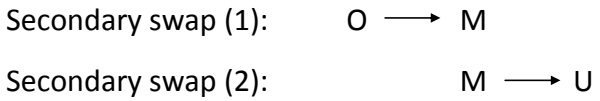



Figure 4.3: Secondary swap, O = overstaffed shift, M = matching shift, U = understaffed shift

We want to retain as much as possible of the preferred schedules of the employees, so primary swaps are preferred over secondary swaps. However, if the schedule does not satisfy the staffing demand when applying only primary swaps, secondary swaps might be helpful. Secondary swaps may also be useful to create new primary swaps, and to finalize the schedule. Applying primary and secondary swaps can be done in various ways, which we refer to as "swap strategies" (input parameter: *Swap strategy*). We consider the following swap strategies:

1. Use primary swaps until these are no longer available, next use secondary swaps. Stop when these are no longer available.
2. Use primary swaps until these are no longer available, next use secondary swaps until there are new primary swaps. Use these new primary swaps until these are no longer available, then again use secondary swaps until new primary swaps arise, etc.
3. Use primary swaps until X% of all understaffed shifts are fulfilled, then use secondary swaps until these are no longer available. We determine the value of X in Section 5.1.1.
4. Use primary swaps until Y% of all understaffed shifts are fulfilled, then use secondary swaps until new primary swaps arise, then use primary swaps until these are no longer available. We determine the value of Y in Section 5.1.1.
5. Use primary and secondary swaps at the same time. Stop when no swaps are available anymore.
6. Use secondary swaps until these are no longer available, then use primary swaps. Stop when these are no longer available.
7. Use primary swaps until these are no longer available, next use primary and secondary swaps at the same time until these are no longer available.

### 4.5.1 Implementation

To include secondary swap in our method (Section 4.2), we need to make some changes. We need to make changes in the phases: Employee selection, Swap selection, and Updating.

#### Employee selection

In Employee selection, we now also calculate the possible secondary swaps. We do this in a separate process as with the primary swaps and only when the secondary swaps are needed. Primary and secondary swaps per employee are stored separately. The swap options created for secondary swaps are of the form: *secondary swap (1)* and *secondary swap (2)*.

#### Swap selection

In our basic MILP we already defined a set for all overstaffed shifts, $OVER$, and a set for all understaffed shifts, $UNDER$. To ensure that when we apply *secondary swap (1)*, we also apply a *secondary swap (2)* and that a matching shifts still matches the staffing demand after applying secondary swaps, we define the set MATCH (Equation (4.13)) and one additional constraint (Equation (4.14)).

$$MATCH = \{k \in K | v_k = 0\} \tag{4.13}$$

$$n_k = 0 \qquad k \in MATCH \tag{4.14}$$

### Updating

If the secondary swaps are performed only in the final phase of the scheduling process, we solely have to update the secondary swaps. Otherwise we also update the primary swaps for each applied secondary swap and vice versa. So, besides the primary swaps, the updating phase now also updates the secondary swaps.

## 4.6 Conclusions

In each iteration of our proposed method, we select a number of employees (input parameter: *Number of initial employees*) and swap shifts in their schedules to fulfill the understaffed shifts. The employee selection is based upon the score of the employees (see Chapter 3). The higher this score, the lower the chance we select the employee. In this way, we are in control of which employees we select and which thus have the possibility of receiving a swap in their schedule. When we do not want to perform swaps on the subset of employees with the lowest score, we select another subset of employees.

We added an additional functionality to retain a minimum percentage $p$ (input parameter: *Minimum percentage*) of the preferred schedule of an employee. When several swaps have been executed within a schedule and the minimum percentage is reached for this employee, we do not allow any additional swaps within the schedule of this employee. In this way, we are in control of the minimum percentage retained of the preferred schedules.

We proposed a basic mixed integer linear program (MILP) to determine the optimal set of swap options for all selected employees. In this MILP we do not take the scores of the employees into account when applying swaps. Therefore, we introduced three extensions to the basic MILP. In the first extension, we prefer swaps of employees with the lowest scores. The other two extensions are alternatives where we prefer to apply swaps that cause a high score change for employees with a low score. In this way, employees with a higher score do not only have a smaller chance of being selected, but when they are selected they also have a smaller chance of receiving a swap. We use the input parameter *MILP* to define which of the MILPs we use: the basic MILP or the basic MILP with one of the extensions.

Finally, we considered secondary swaps next to the primary swaps. A secondary swap performs two swaps, where each swap is performed at another employee. Within a secondary swap, we swap one employee to a matching shift, where another employee is swapped from this matching shift. With secondary swaps we are able to fulfill understaffed shifts where primary swaps sometimes cannot. We use the input parameter *Swap strategy* to define how we use primary and secondary swaps.

Next to the four mentioned input parameters, our method is also flexible in the input parameter: *Point system*. This input parameter defines which point system we use: point system 1, point system 2, or point system 3 (see Chapter 3).

# Chapter 5

# Experimental design and Data

After we explained our method in Chapter 4, this chapter describes our approach to determine the settings for the input parameters. First, Section 5.1 describes our approach and key performance indicators we use to analyze the output. Next, Section 5.2 describes the data we use to test our method. Finally, we draw our conclusions of this chapter in Section 5.3.

## 5.1 Experimental design

The goal of this research is to design a method to finalize the schedule in the last step of the self rostering process, such that it is widely applicable. In Chapter 4 we explained our method, which considers five input parameters: 'Point system', 'Number of initial employees', 'Minimum percentage', 'MILP', and 'Swap strategy'. The goal of the analysis is to find a set of settings for these input parameters such that our method performs well in various situations.

First, in Section 5.1.1, we briefly discuss the input parameters and their possible settings. Next, in Section 5.1.2, we explain the method we use to analyze our data: the $2^k$-factorial design method.

### 5.1.1 Settings

We consider 5 input parameters: 'Point system', 'Number of initial employees', 'Minimum percentage', 'MILP', and 'Swap strategy' (see Sections 4.2 to 4.5). In principle, the input parameters 'Number of initial employees', 'Minimum percentage', and 'Swap strategy' have an infinite amount of possible settings. It is time consuming and not efficient to compare all these options to see which set of settings suites best. Therefore, we first determine a finite set of settings for these parameters.

#### Minimum percentage

The parameter 'Minimum percentage' sets a minimum to the percentage remaining of the preferred schedules of an employee. In principle, this parameter

can have all values between 0% and 100%. However we consider three settings: a minimum of 0%, 70%, and 80%. From practice we see that Pompesticht-ing and Westfriesgasthuis prefer a minimum percentage of 80% and NedTrain prefers a minimum of 80% on average per year. Therefore, we choose the setting with a minimum of 80%. Next, we are interested in the realized minimum percentage when the minimum percentage decreases. We are mainly interested in the magnitude of decrease between the realized minimum percentage when the minimum percentage is slightly lower than 80% (minimum percentage of 70%) and when there is no restriction at all (minimum percentage of 0%).

#### Number of initial employees

The 'Number of initial employees' is the number of selected employees at the *employee selection* phase within our method (see Section 4.2.2). This input parameter can take any positive integer value. For the 'Number of initial employees', we choose to compare the situation were *all* employees are selected from the start with the situations where we start with one, or three employees and iteratively add one employee when there are no possible swap options left. From practice, we see that Organization X wants to have a transparent scheduling process. The scheduling process is most transparent, when we start with one employee. Then, in each iteration, we select a swap option at one employee, e.g., with the lowest score. We like to compare this to a situation where we start with several employees. More employees and more selected swap options mean less transparency: employees do not know directly why they received a certain swap in their schedule. Therefore, we prefer a number that is as small as possible, but bigger than one. We expect that the situation where we start with two employees is likely to be similar to the situation where we start with one employee and iteratively add one employee. Therefore, for the second setting, we choose to start with three employees and iteratively add one employee. To study whether a less transparent setting results in better outcomes, we add the setting: start with *all* employees.

#### Swap strategy

The input parameter 'Swap strategy' contains two swap strategies for which we need to define the parameter values. These are:

3. Use primary swaps until X% of all understaffed shifts are fulfilled, then use secondary swaps until these are no longer available.

4. Use primary swaps until Y% of all understaffed shifts are fulfilled, then use secondary swaps until new primary swaps arise, then use primary swaps until these are no longer available.

We set X=95% since we do not want to use too many secondary swaps. Secondary swaps have a negative influence on the remaining percentages: schedules reach faster the minimum percentage and we may not apply any further swaps in these schedules. It is possible that no primary swaps are available anymore before we reach this 95%. Then the method stops. We set Y=80%. This parameter value is lower than X, since this strategy uses again primary swaps after the secondary swaps.

### 5.1.2 $2^k$-factorial design

The main goal of the analysis is to find a set of settings for which we receive the best outcomes in various situations. Therefore, we study the effect of each input parameter and the interaction effects among the input parameters on the outcomes. For this, we use the $2^k$-factorial design method, which is designed to find and analyze effects of parameters as well as interaction effects (Law, 2007).

#### Method

Law (2007) refers to input parameters as *factors* and to output parameters as *responses*. There are two kinds of factors: *Quantitative* and *Qualitative*. Quantitative factors can be specified by numerical values, while qualitative factors represent structural assumptions that cannot be quantified.

Within the method, each factor has two settings, which are called *levels*. We determine the responses of all possible combinations of all levels, which are called the *design points*. When $k$ is the number of factors, we thus have $2^k$ design points.

Using the $2^k$-factorial design, we can compare the combinations where the level of factor $i$ varies and all other factors are fixed. The average change in response is then caused by the change of factor $i$. This effect of factor $i$ is denoted by $e_i$. We illustrate this in Example 4.

**Example 4.** Consider two factors, A and B. We denote the levels of these factors by (-) and (+). The signs are arbitrarily assigned to a level. Since, $k = 2$, we have $2^2 = 4$ design points, see Table 5.1

Table 5.1: Design matrix $2^2$-factorial design

| Design point | Factor A | Factor B | Response |
|---|---|---|---|
| 1 | - | - | $R_1$ |
| 2 | + | - | $R_2$ |
| 3 | - | + | $R_3$ |
| 4 | + | + | $R_4$ |

The differences between the responses of combination 1 and 2 ($R_2 - R_1$) and combination 3 and 4 ($R_4 - R_3$) are due to the change in level of factor A. The *main* effect of factor A, is the average of these two differences, see Equation (5.1).

$$e_A = \frac{(R_2 - R_1) + (R_4 - R_3)}{2^{2-1}} \tag{5.1}$$

Equation (5.2) shows the calculation for the main effect of B.

$$e_B = \frac{(R_3 - R_1) + (R_4 - R_2)}{2^{2-1}} \tag{5.2}$$

If the effect of factor A depends on the level of factor B, A and B are said to interact. To measure this interaction, we calculate the interaction

effect of factor A and B. This effect can be seen as the difference between the average effect of factor A when the level of factor B is (+) ($\frac{R_4-R_3}{2^{2-1}}$) and the average effect of factor A when the level of factor B is (-) ($\frac{R_2-R_1}{2^{2-1}}$). Equation (5.3) shows the calculation for the interaction effect of factor A and B.

$$e_{AB} = \frac{(R_4 - R_3) - (R_2 - R_1)}{2^{2-1}} \tag{5.3}$$

Note that the interaction effect of A with B ($e_{AB}$) is equal to the interaction effect of B with A ($e_{BA}$).

In our case, we use the input parameters described in Section 5.1.1 as factors. However, all input parameters have more than two settings. So, to be able to use the $2^k$-factorial design we need to select two settings per input parameter. Therefore, we perform a preliminary test. (see Section 6.1).

### 5.1.3 Key performance indicators

In this section we describe the key performance indicators (KPIs) on which we evaluate our method. We distinguish three KPIs: Shortages, Remaining percentage, and Computational time. In the following, we discuss each indicator and explain how we calculate the response for this indicator.

**Shortages** The main goal of our method is to help the planner to finalize the schedule. Therefore, the method minimizes the number of overstaffed shifts (and thus also the number of understaffed shifts). To measure this, we define our first indicator as the number of understaffed shifts left: Shortages. The stakeholder of this indicator is the organization. The fewer the number of shortages left, after the execution of our method, the better it is for the organization.

The number of shortages left after the execution of our method does not always indicate the performance of the method correctly. When there are fewer overstaffed shifts than understaffed shifts before our method starts, the method cannot cover all understaffed shifts. Therefore, we need to correct the number of shortages left. We define the number of shortages as the difference between $U_a$, the number of understaffed shifts left after the execution of our method, and the number of understaffed shifts that cannot be covered. If all understaffed shifts can potentially be covered then $Shortages = U_a$. Equation (5.4) shows how we calculate the responses for this first indicator.

$$\text{Shortages} = U_a - \max(0, U_b - O_b) \tag{5.4}$$

**Remaining percentage** A wish from practice is to minimize the number of changes in the schedules of the employees and retain at least (on average) 80% of the schedules. Therefore, the second indicator is: Remaining percentage. The stakeholder of this indicator is the employee. The higher

| | |
|---|---|
| $U_a$ | Number of understaffed shifts after our method |
| $U_b$ | Number of understaffed shifts before the start of our method |
| $O_b$ | Number of overstaffed shifts before the start of our method |

the remaining percentage, the more of the preferred schedule is retained, the better it is for the employee.

The Remaining percentage is the total number of days where the assignment of the employees is unchanged divided by the total number of shifts of all employees and multiplied with 100%:

$$\text{Remaining percentage} = \sum_{i \in I} \left( \frac{D_i}{S_i} \right) * 100\% \tag{5.5}$$

| | |
|---|---|
| $D_i$ | number of days in the schedule of employee $i$ where the assignment is unchanged |
| $S_i$ | number of shifts in the schedule of employee $i$ |

**Computation time** This indicator is not explicitly mentioned before. However, we consider the computation time as important. A computation time of minutes to hours is not desirable, since this may imply that the planner himself is even faster than our method. When the computation time is already less than a minute, we are satisfied and do not have to consider this indicator.

Computation time is measured as the total time MATLAB needs to finish our method. This is the time between the start of the *employee selection* phase and when our method is finished.

Since we discovered, during the runs, that CPLEX needs a lot of computational time for a few combinations and 'the best solution found so far' in CPLEX did not change after a couple of seconds, we specified a time limit of 60 seconds for CPLEX to solve the swap selection problem. When 'the best solution found so far' is feasible, we use this solution of applied swaps in this iteration. In case CPLEX did not find a feasible solution within the 60 seconds, we do not apply any swaps. Then, our method changes the input for the next iteration by adding one employee to selected employees.

## 5.2 Data

After Section 5.1 described our approach to analyze the outcomes of our method, this section describes the input data we use to test our method.

First, this section describes the data we gathered to test our method (Section 5.2.1). Next, we describe the modification methods we used on these datasets

to be able to use these datasests to test our method (Section 5.2.2). Finally, in Section 5.2.3, we describe which modification method we use per dataset.

### 5.2.1 Practical data

We received one dataset from the Pompestichting, two datasets from Westfriesgasthuis, and three datasets from Organization-X. Table 5.2 shows the type of the schedule (e.g. cyclical schedule) and planning horizon, and the number of employees per case.

Table 5.2: Description of received schedules per department (PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, WFG-2 = Westfriesgasthuis-2, X-1 = Organization X-1, X-2 = Organization X-2, and X-3 = Organization X-3

| **Case** | **Type and planning horizon** | **No. of employees** |
|---|---|---|
| PS | One-year schedule | 14 |
| WFG - 1 | One-year schedule | 49 |
| WFG - 2 | One-year schedule | 73 |
| X - 1 | 21-weeks cyclical schedule | 84 |
| X - 2 | 24-weeks cyclical schedule | 72 |
| X - 3 | 24-weeks cyclical schedule | 72 |

### 5.2.2 Modification methods

Since we have no data on the staffing demand we assume that the given schedules fulfill the staffing demand. However, if all shifts match the staffing demand we cannot test our method. Thus, to test our method, we have to modify the individual schedules, such that these do not match the staffing demand. We do this in two ways, to make the results less dependent on the modification method we use. The two modification methods are:

1. *Shuffle - Rearranges intervals of days in the schedules of the employees*
   The input for this method are year-schedules of all employees. In each year-schedule we select 12 non-overlapping intervals of 28 days and consider these as our 'month' schedules.

   First, we consider the month schedules of employee 1, $a_1 \dots a_{12}$. We assign them to the first month of the new schedules of the employees. So, the first month schedule of employee 1 is $a_1$ and the first month schedule of employee 2 is $a_2$, etcetera. After we assign the month-schedules of employee 1, we assign the month schedules of employee 2, $b_1 \dots b_{12}$, etcetera.

   When we have more months than employees, we cannot assign all month-schedules of employee 1 to the first month of all employee schedules. Then, we place the remaining month schedules in the second period. For example, if we have 10 employees, we assign $a_{11}$ to the second period of employee 1 and $a_{12}$ to the second period of employee 2. In this case, we assign $b_1$ to the second period of employee 3, $b_2$ to the second period of employee 4, etcetera. We illustrate this example in Table 5.3.

Table 5.3: Shuffle method

| | Month | | | | | |
|---|---|---|---|---|---|---|
| Employee | 1 | 2 | 3 | 4 | $\cdots$ | 12 |
| 1 | $a_1$ | $a_{11}$ | $b_9$ | $c_7$ | $\vdots$ | $j_3$ |
| 2 | $a_2$ | $a_{12}$ | $b_{10}$ | $c_8$ | $\vdots$ | $j_4$ |
| 3 | $a_3$ | $b_1$ | $b_{11}$ | $c_9$ | $\vdots$ | $j_5$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| 10 | $a_{10}$ | $b_8$ | $c_6$ | $d_3$ | $\vdots$ | $j_{12}$ |

2. *Disturbance*
   Again, the input for this method are year-schedules of all employees. In these schedules we again select 12 'month' schedules. Instead of rearranging the month schedules we apply a disturbance to the staffing demand of each month.

   A disturbance in staffing demand can be created in several ways. We choose to divide each month in 4 blocks of 7 days. The disturbance in staffing demand is chosen to be different in each block. In block 1 and 3 the staffing demand is increased and in block 2 and 4 the staffing demand is decreased. In this way, we create periods where too many employees prefer to work and in other periods too few employees (e.g., employees with children in school, school vacations).

   Each day contains a staffing demand for each shift on that day. On each combination of day and shift, we determine with a certain probability whether this staffing demand receives predefined disturbance. This probability is the same for each shift within a week. Since Pompestichting and Westfriesgasthuis-1, *group (1)*, have fewer employees and a lower staffing demand than Westfriesgasthuis-2 and Organization X-1, Organization X-2, and Organization X-3, *group (2)*, the disturbance in group (1) is also chosen to be smaller than in group (2). Tables 5.4 and 5.5 show the disturbance together with the associated probabilities we applied for group (1) and group (2), respectively.

   Table 5.4: Disturbance and probability of disturbance of group (1)

| **Day** | **1-7** | **8-14** | **15-21** | **22-28** |
|---|---|---|---|---|
| Probability | 50% | 50% | 66.67 % | 66.67% |
| Disturbance | +2 | -2 | +1 | -1 |

   In Tables 5.4 and 5.5, we see that the probability of a disturbance of block 1 is equal to the probability of block 2 and the same holds for block 3 and 4. In this way, in expectation the number of employees we need in the first block equals the number of employees we do not need in the second block. Therefore, we create an approximately equal amount of understaffed and

Table 5.5: Disturbance and probability of disturbance of group (2)

| **Day** | **1-7** | **8-14** | **15-21** | **22-28** |
|---|---|---|---|---|
| Probability | 50% | 50% | 66.67 % | 66.67% |
| Disturbance | +3 | -3 | +1 | -1 |

overstaffed shifts. In case these probabilities are not equal, we would create more overstaffed than understaffed shifts or the other way around. Our method cannot fulfill these extra understaffed or overstaffed shifts, so we also do not want to create these. The probability for the disturbance in the third and fourth interval is chosen to be higher than in the first and second interval. Note that we choose these values since we think that the larger the deviation from the actual staffing demand, the lower the chance this actually happens in practice. However other values for the parameters might also work.

### 5.2.3 Datasets

This section discusses how we generate datasets for our method using the schedules discussed in Section 5.2.1 and the modification methods discussed in Section 5.2.2.

Per case we create two datasets: Shuffle and Disturbance. However for Organization X we only apply the Disturbance method. From Organization X we received 21-week and 24-week cyclical schedules instead of one year-schedule. To apply the Shuffle method we need year-schedules. Multiplying the cycle to a year-schedule results in 12 similar month schedules. Using the Shuffle method would again result in 12 similar month schedules. For the Disturbance method we create one month schedule from each of the three cyclical schedules. To compare these with the other cases we need 12 schedules. Therefore, we apply the Disturbance method 4 times on each of the 3 month schedules and combine these in one case. From here on, we refer to this case as 'Organization X'.

## 5.3 Conclusions

We described the $2^k$-factorial design method to analyze the outcomes of our method in order to determine the settings for the input parameters. With this $2^k$-factorial design method we calculate the effect of each input parameter and the interaction effect between input parameters. The $2^k$-factorial design method requires exactly two settings per input parameter. Since we have more settings at each input parameters, we need to do a preliminary test to analyze which settings we want to use in the $2^k$-factorial design. We describe and discuss this test in Chapter 6.

We gathered data from Pompestichting, Westfriesgasthuis, and Organization X. We assumed these schedules to be the preferred schedules of the employees and by lack of knowledge about the staffing demand we assumed that all schedules together fulfill the staffing demand. When all preferred schedules match the staffing demand, the planner, and thereby our method, do not have to change anything. Therefore, we created two modification methods: 'Shuffle'

and 'Disturbance'. Within the Shuffle method, we create preferred schedules by rearranging intervals within the year schedule over all employees. Within the Disturbance method, we apply a disturbance in the staffing demand. We designed two modification methods, to make the results less dependent on the modification method we use.

Finally, we use both modification methods on each case, except for the case of Organization X. This organization provided 3 similar cyclical schedules. Multiplying the cycle to a year-schedule and shuffle all month schedules, would result in several similar schedules. Therefore, we choose to apply only the Disturbance method. However, due to the similar cyclical schedules we received, applying 12 times a disturbances to one schedule would lead to 12 similar schedules. Therefore, we applied 4 times a disturbances on each cyclical schedule and combined these to a set of 12 schedules, we call this set of schedules the case of Organization X.

# Chapter 6

# Experimental results

After we explained our approach to generate and analyze the outcomes of our method, in order to determine the settings for the input parameters (Chapter 5), this chapter discusses the experimental results. We implemented our method, described in Section 4.2, in MATLAB 2011b (MathWorks, 2012) and solved the MILP, described in Sections 4.3 and 4.4, with CPLEX 12.2 and its connector to MATLAB (IBM, 2012). We generated all results on a DELL LATITUDE E6410 with Windows XP (Sp3 RIS), an Intel(R) Core(TM) i7 CPU processor (M 640 @ 2.80GHz), and a physical memory of 3.24 GB (1.17 GHz).

Section 6.1 discusses the results from the preliminary test and concludes which two settings of each factor we select for the $2^k$-factorial design. Next, Section 6.2 discusses the results of the $2^k$-factorial design. In Section 6.3 we justify the assumptions made in Section 6.2. Finally, we draw our conclusions in Section 6.4.

## 6.1 Preliminary test

This section describes the preliminary test and discusses the results. In Section 6.1.1, we explain what we do within the preliminary and in Section 6.1.2 we explain and discuss the choices we make per factor.

### 6.1.1 Setup

For the preliminary test we use the schedules of the Pompestichting case created with the Shuffle method. The Pompestichting has the least number of employees and therefore we expect to reduce the computation time of MATLAB by using this case.

For all 12 schedules of the Pompestichting case and all possible combinations of levels (864, Appendix E) we generate the outcomes. We refer to the combination of a schedule and a combination of levels as 'instance'. So, in total we have 10368 instances. We analyze the outcomes of all instances per factor and per level. Hence, we analyze the outcome of an instance at each factor, but for different levels.

We analyze the outcomes on three KPIs (described in Section 5.1.3): Remaining percentage, Shortages, and Computation time. We use a table and

figures to analyze the levels of each factor. The table shows the average of the outcomes per level for each KPI. From these, we first indicate per KPI which two levels provide the best outcomes. Each figure shows a moving average for one of the KPIs. We use these to observe whether the two best levels per KPI are indeed in general the best levels. Based on this, we select two levels.

We use a moving average, since we have many outcomes to analyze and we are looking for a general solution. We use an moving average over 100 points, since this causes the frequent changes in response to disappear, while the general effects remain visible. With a moving average over 100 instances our first point in the graph is the average over the outcomes of instance 1 to 100. The second point is the average over the outcomes of instance 2 to 101, etcetera.

We illustrate the effect of a moving average in Figure 6.1. In this figure, the upper graph shows 10368 outcomes divided over three levels (3456 outcomes per level). The lower graph represent the moving average per level. From the upper graph we cannot draw any clear conclusion, while in the lower graph we observe that level 3 is in general higher than level 2. A more specific example is visible around instance per level 2305. In the upper graph we expect level 1 to be in general the highest, while we observe in the lower graph that level 1 is in general the lowest and level 3 is in general the highest.

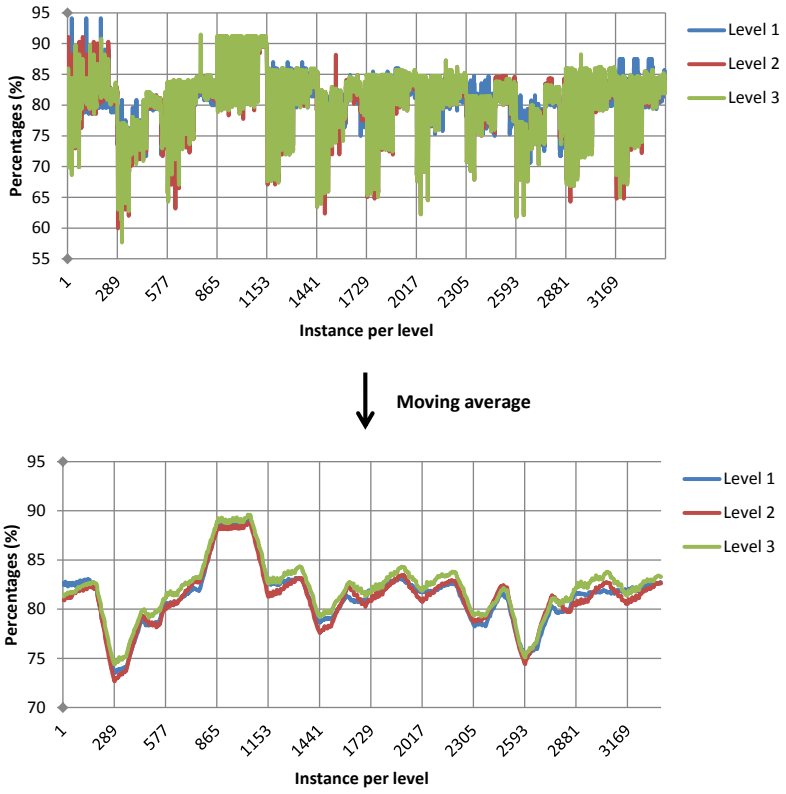



Figure 6.1: Effect of a moving average

### 6.1.2 Analysis preliminary test

In this section, we analyze the different levels of each factor per KPI. Table 6.1 shows the levels per factor we select based on the preliminary test. The levels are arbitrarily assigned to (-1) and the (1).

Table 6.1: Selected levels per factors

| Factor | Explanation | Level | Explanation |
|---|---|---|---|
| 1 | Point system | (-1) | Point system 1 |
| | | (1) | Point system 3 |
| 2 | MILP | (-1) | Basic MILP |
| | | (1) | Lowest Score |
| 3 | Swap strategy | (-1) | Only primary swaps |
| | | (1) | Primary and secondary swaps: first use primary swaps until these no longer exist, then use secondary swaps until there are new primary swaps. Use these new primary swaps until these no longer exist, then again use secondary swaps until new primary swaps arise etc. |
| 4 | Initial employees | (-1) | Start with 3 employees (with the lowest score), when these do not cause a change, add 1 employee with the lowest score of the unselected employees. |
| | | (1) | Start with 1 employee (the employee with the lowest score), when this do not cause a change, add 1 employee with the lowest score of the unselected employees. |
| 5 | Minimum percentage | (-1) | Use a minimum of 70%. |
| | | (1) | Use a minimum of 80%. |

Table 6.2 shows the maximum computation time, the number of times CPLEX reached the time limit, and the number of times MATLAB encountered memory problems. Since the maximum computation time for all combinations of levels is less than 34 seconds (see Table 6.2), we do not consider this indicator any further. While generating all responses, MATLAB did encounter some memory problems at 9 of the 864 combinations (see Table 6.2) and could not finish our method. Therefore, we could not take these responses into account and we do not want to select these combinations of levels for the $2^k$-factorial design.

Table 6.2: Overview of time issues per factor and level, where L = Level, MT = Maximum Time (seconds), TL = number of runs that reached the Time Limit, and MP = number of runs that caused Memory Problems.

| | **Factor 1** | | | **Factor 2** | | | **Factor 3** | | | **Factor 4** | | | **Factor 5** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **L** | **MT** | **TL** | **MP** | **MT** | **TL** | **MP** | **MT** | **TL** | **MP** | **MT** | **TL** | **MP** | **MT** | **TL** | **MP** |
| 1 | 5.84 | 12 | 0 | 4.10 | 0 | 0 | 18.28 | 0 | 0 | 33.36 | 38 | 0 | 33.36 | 16 | 6 |
| 2 | 18.28 | 12 | 9 | 5.84 | 28 | 0 | 7.73 | 24 | 0 | 18.28 | 1 | 4 | 6.83 | 14 | 2 |
| 3 | 33.36 | 16 | 0 | 33.36 | 7 | 0 | 17.64 | 0 | 0 | 11.95 | 1 | 5 | 7.75 | 10 | 1 |
| 4 | – | - | - | 18.28 | 5 | 9 | 18.23 | 5 | 0 | – | - | - | – | - | - |
| 5 | – | - | - | – | - | - | 17.77 | 3 | 0 | – | - | - | – | - | - |
| 6 | – | - | - | – | - | - | 11.95 | 1 | 8 | – | - | - | – | - | - |
| 7 | – | - | - | – | - | - | 33.36 | 2 | 1 | – | - | - | – | - | - |
| 8 | – | - | - | – | - | - | 17.62 | 5 | 0 | – | - | - | – | - | - |

**Factor 1: Point system**

*Shortages:* level 1 and 3
*Remaining percentage:* level 1 and 3

Figure 6.2 and 6.3 also show that level 1 and 3 are in general the best. Therefore, we select level 1 and 3.

Table 6.3: Average responses per level (factor 1)

| Level | Shortages | Remaining percentage (%) |
|---|---|---|
| 1 | 3.79 | 81.67 |
| 2 | 4.11 | 81.41 |
| 3 | 3.96 | 81.50 |

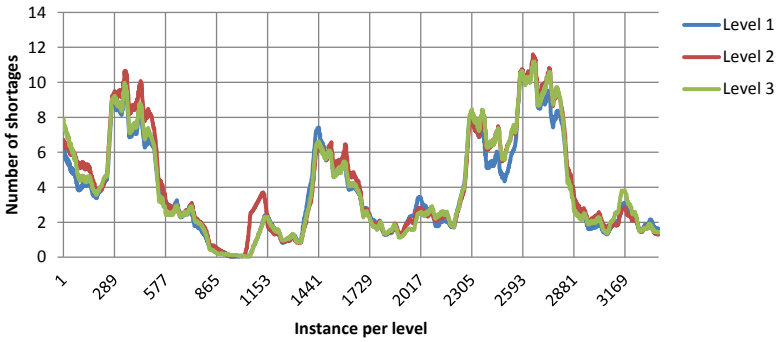



Figure 6.2: Factor 1 - Shortages

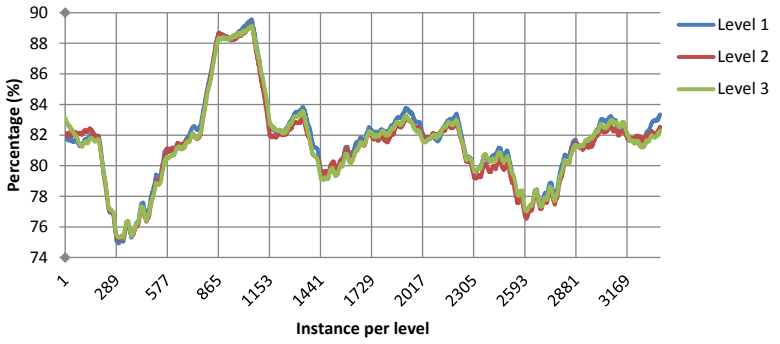



Figure 6.3: Factor 1 - Remaining Percentage

**Factor 2: MILP**

*Shortages:* level 2, 1, and 4
*Remaining percentage:* level 2 and 1

Table 6.4: Average responses per level (factor 2)

| Level | Shortages | Remaining percentage (%) |
|---|---|---|
| 1 | 3.89 | 81.81 |
| 2 | 3.88 | 81.93 |
| 3 | 4.06 | 81.27 |
| 4 | 3.98 | 81.09 |

Figure 6.4 and 6.5 show that all levels follow the same trend and the level 1 and 2 are in general the best. Therefore, we select level 1 and 2.

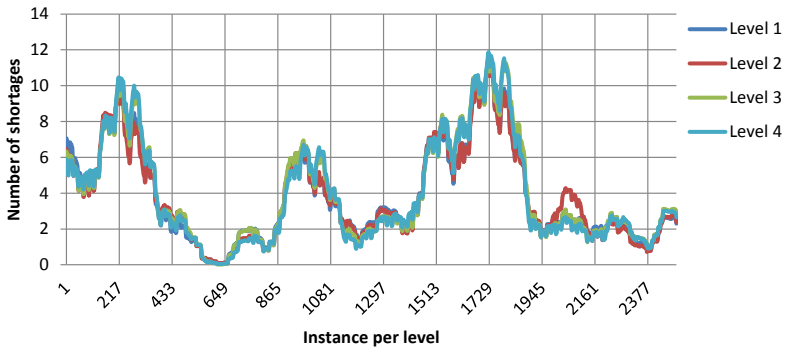



Figure 6.4: Factor 2 - Shortages

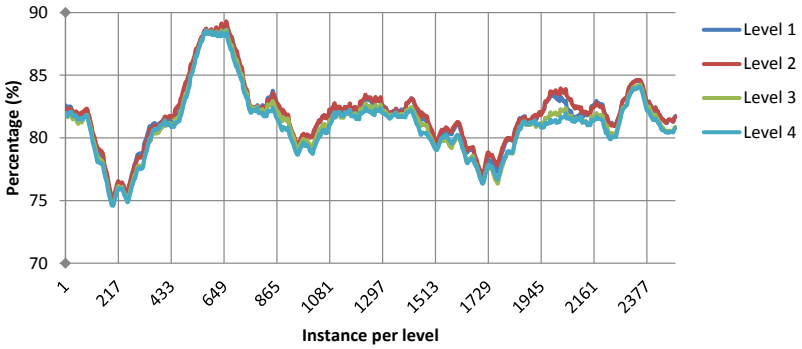



Figure 6.5: Factor 2 - Remaining Percentage

### Factor 3: Swap strategy

*Shortages:* level 4, 7, and 5
*Remaining percentage:* level 1 and 3

First, we select level 1 (only primary swaps), because we want to compare this to the best level of the hybrid forms to study the effect of secondary swaps.

Table 6.5: Average responses per level (factor 3)

| Level | Shortages | Remaining percentage (%) |
|---|---|---|
| 1 | 2.49 | 82.62 |
| 2 | 17.34 | 77.43 |
| 3 | 2.45 | 82.51 |
| 4 | 1.55 | 81.90 |
| 5 | 1, 81 | 82.05 |
| 6 | 2.25 | 81.89 |
| 7 | 1.81 | 81.76 |
| 8 | 1.94 | 82.07 |

The best hybrid form is the one with the fewest number of shortages, because we expect that using primary and secondary swaps fulfills more understaffed shifts than when we use only primary swaps. Figure 6.6 an 6.7 show that all levels follow the same trend and therefore the averages do not depend on outliers. Level 4 has the lowest average on Shortages therefore we select level 4.

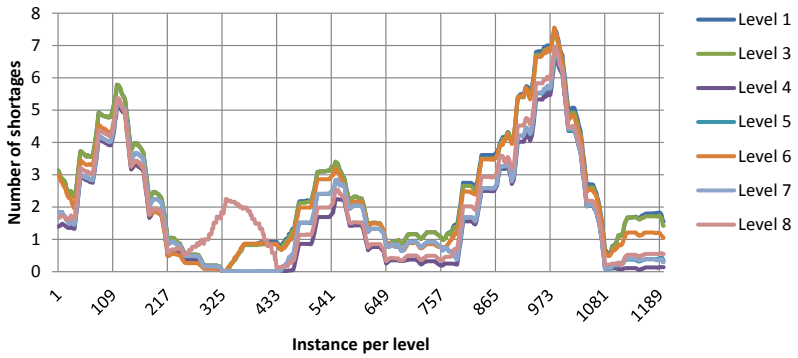



Figure 6.6: Factor 3 - Shortages

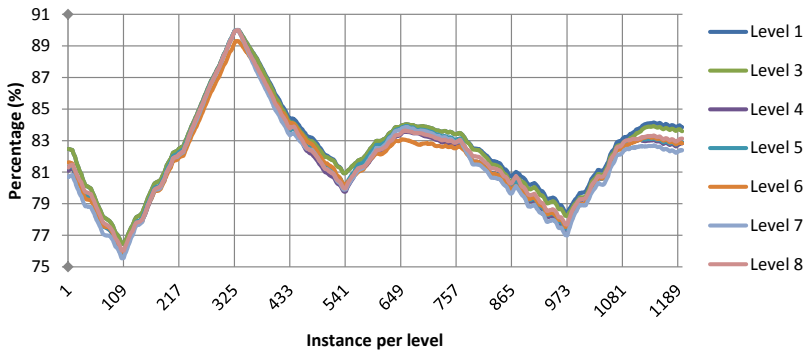



Figure 6.7: Factor 3 - Remaining Percentage

### Factor 4: Number of initial employees

*Shortages*: level 3 and 2
*Remaining percentage:* level 3 and 1

Figures 6.8 and 6.9 show the same results for Shortages and Remaining percentage. First, we select level 3. Next, we know from the cases from practice (see Section 2.3.2) that most organizations prefer (an average of at least) 80% at Remaining percentage. This is accomplished at both level 1 and 2. Since, level 2 fulfills almost 2 more shortages than level 1, we select level 2.

Table 6.6: Average responses per level (factor 4)

| Level | Shortages | Remaining percentage (%) |
|---|---|---|
| 1 | 5.08 | 81.37 |
| 2 | 3.56 | 81.20 |
| 3 | 3.22 | 82.01 |

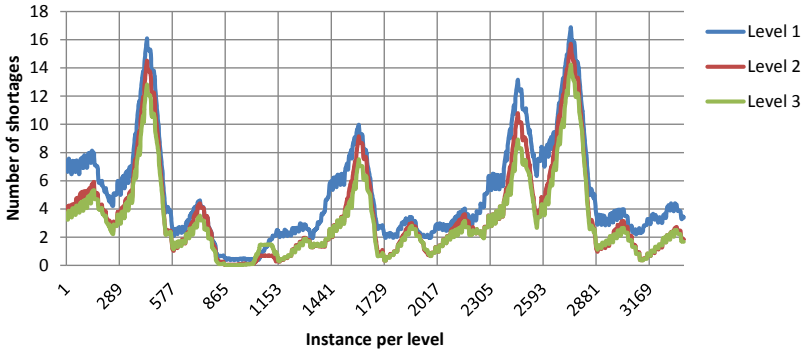



Figure 6.8: Factor 4 - Shortages

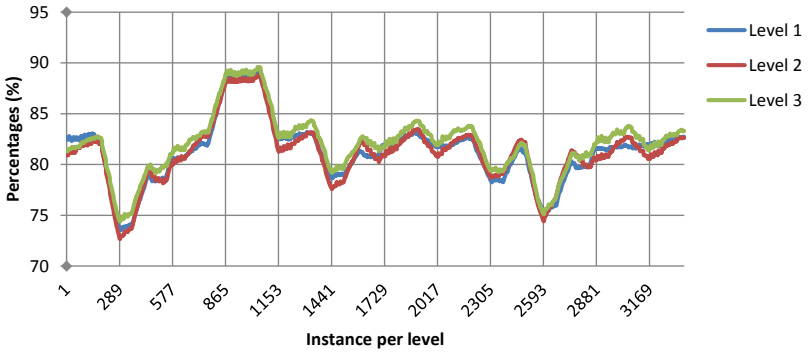



Figure 6.9: Factor 4 - Remaining Percentage

**Factor 5: Minimum percentage**

*Shortages*: level 1 and 2
*Remaining percentage:* level 3 and 2

First, we select level 3 (minimum percentage of 80%), because as described in Section 5.1.1, we want to compare this to a level that is slightly lower (level 2, minimum of 70%) or when there is no restriction at all (level 1, minimum of 0%). Figure 6.10 an 6.11 show that all levels follow the same trend and therefore the averages do not depend on outliers. Level 2 is most promosing for both Shortages and Remaining percentage. Therefore, we select level 2 and 3.

Table 6.7: Average responses per level (factor 5)

| Level | Shortages | Remaining percentage (%) |
|---|---|---|
| 1 | 2.10 | 80.33 |
| 2 | 3.17 | 81.09 |
| 3 | 6.59 | 83.16 |

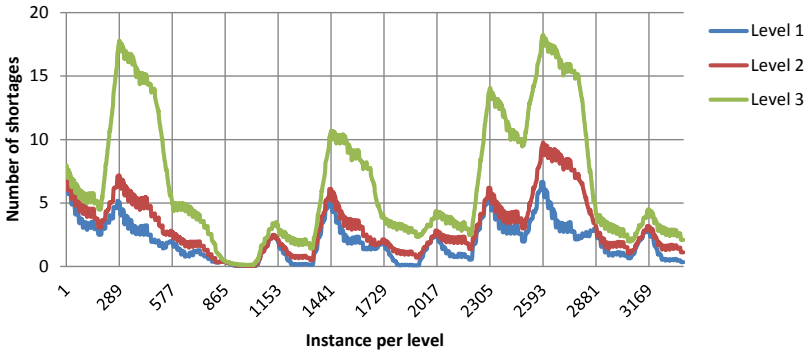



Figure 6.10: Factor 5 - Shortages

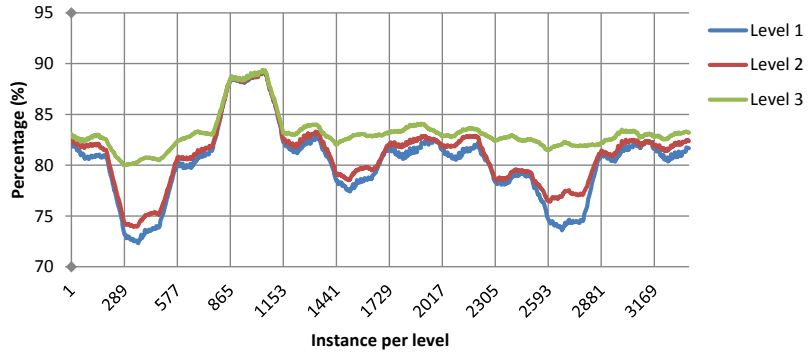



Figure 6.11: Factor 5 - Remaining Percentage

## 6.2 $2^k$-factorial design

The goal of using the $2^k$-factorial design method is to identify the main and interaction effects of all factors. In the preliminary test we selected two levels per factor that we want to compare (see Table 6.1). So, we have 5 factors each with 2 levels, thus 32 combinations, i.e., design points. See Appendix E for the full factorial design matrix.

In this section, we show the general results per case and modification method (Section 6.2.1). Next we describe one additional KPI we use at the $2^k$-factorial design and whether we consider Computation time (Section 6.2.2). Next, we explain the method we use to analyze the significance of the found (interaction) effects (Section 6.2.3). In Section 6.2.4, we apply this method and discuss the results. To calculate the effects, we made some assumptions, which we discuss in Section 6.3.

### 6.2.1 General results

This section shows the general results of each experiment. Table 6.8 shows the average input: number of employees, shortages, and excesses. Table 6.9 shows the output: the average number of shortages left, the average remaining percentage over all preferred schedules, the maximum computation time, and the number of primary swaps and secondary swaps used.

In these tables, we see that most of the shortages are fulfilled by our method, the remaining percentage is on average above 83%, and the maximum computation time is 12 seconds. We also see that we mostly use primary swaps to fulfill the shortages. Secondary swaps are mostly used in the schedules of Organization X. We explain this in section 6.2.4.

### 6.2.2 Key performance indicators

Next to the KPIs discussed in Section 5.1.3, we introduce one additional KPI: Minimum remaining percentage. We introduce this, since almost all cases want to retain at least 80% of each proposed schedule. With this KPI we get an indication whether the minimum over all individual remaining percentages is above or close to the 80%.

Table 6.9 shows the maximum computation time per case and modification method. As with the preliminary test, the computation time needed is again in the order of seconds. Hence, the KPI Computation time is not discussed in the remainder of this chapter.

### 6.2.3 Method

We performed a $2^k$-factorial design on the responses of each schedule. For all schedules we found main and interaction effects. To find out whether these effects are significant for the case and not due to sampling fluctuations in the preferred schedules, we need to determine whether the effects are significant.

We use confidence intervals to determine the statistical significance of an effect. Within each combination of case and modification method we have 12 schedules. At each schedule we calculate the effects for all factors. So, per

Table 6.8: Input (Em = Number of employees, Sh = Average number of shortages, Ex = Average number of excesses, PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, WFG-2 = Westfriesgasthuis-2, and X = Organization X)

| **Modification method** | **Case** | **Em** | **Sh** | **Ex** |
|---|---|---|---|---|
| Shuffle | PS | 14 | 45.08 | 45.08 |
| | WFG-1 | 49 | 102.58 | 102.58 |
| | WFG-2 | 73 | 128.33 | 128.33 |
| Disturbance | PS | 14 | 21.25 | 21.42 |
| | WFG-1 | 49 | 34.25 | 36.08 |
| | WFG-2 | 73 | 38.17 | 40.50 |
| | X | 76 | 48.00 | 46.75 |
| Average | | 50 | 59.67 | 60.11 |

Table 6.9: Output (Sh = average number of shortages, R% = Average remaining percentage, MT = Maximum computation time (seconds), P = number of primary swaps used, S = number of secondary swaps used, PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, WFG-2 = Westfriesgasthuis-2, and X = Organization X)

| **Modification method** | **Case** | **Sh** | **R%** | **MT** | **P** | **S** |
|---|---|---|---|---|---|---|
| Shuffle | PS | 2.02 | 83.4 | 2.57 | 38.14 | 0.12 |
| | WFG-1 | 0.42 | 91.8 | 11.88 | 68.86 | 0.05 |
| | WFG-2 | 0.14 | 92.8 | 10.18 | 89.82 | 0.04 |
| Disturbance | PS | 1.10 | 89.7 | 1.63 | 18.78 | 0.56 |
| | WFG-1 | 0.25 | 95.3 | 4.24 | 31.77 | 0.23 |
| | WFG-2 | 0.12 | 96.3 | 5.67 | 37.27 | 0.10 |
| | X | 2.27 | 96.2 | 7.35 | 37.33 | 1.82 |
| Average | | 0.90 | 92.2 | 6.22 | 46.00 | 0.42 |

(interaction) factor we have 12 effects. When we calculate a $100(1-\alpha)$ confidence interval of the expected effect and this interval does not contain zero, we conclude that the effect is statistical significant.

We use a t-distribution to calculate the confidence interval per group of 12 effects, see Equation 6.1 (Law, 2007). To use a t-distribution the sample data needs to be normally distributed. To test for normality we use the Shapiro-Wilkinson test. Not all groups passed this test at a confidence interval of 95%. However, the central limit theorem (CLT) states that when a set is large enough its sample mean is normally distributed. Therefore, with the CLT we assume that all groups, of 12 effects, are normally distributed and hence we are allowed to use the t-distribution to calculate the confidence intervals of each group.

$$\bar{X}(n) \pm t_{n-1,1-\alpha/2} \cdot \sqrt{\frac{S^2(n)}{n}} \tag{6.1}$$

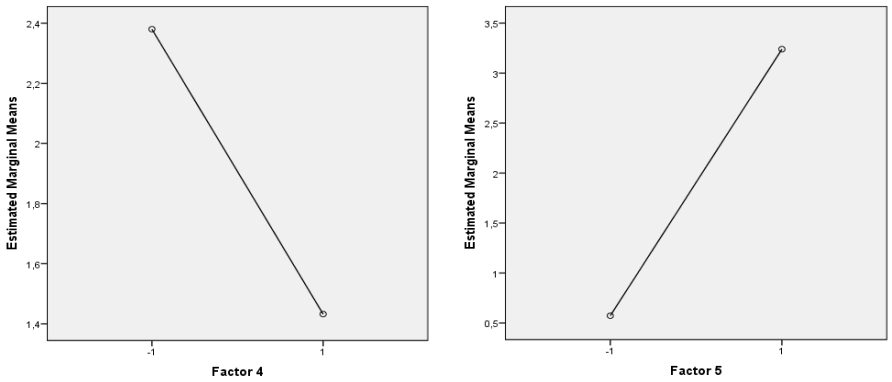
| | |
|---|---|
| $\bar{X}(n)$ | Sample mean |
| $t_{n-1,1-\alpha/2}$ | The upper $1-\alpha/2$ critical point for the t-distribution with $n-1$ degrees of freedom |
| $S^2(n)$ | Sample variance |

When we analyze the (interaction) effects, we have to bear in mind that when a significant main effect is also within a significant interaction effect, the main effect is limited. Let us explain this with an example. In Table 6.10 at Pompestichting - Shuffle method - Shortages, we see that factor 4 has a negative effect (confidence interval below zero), while factor 5 has a positive effect and the interaction factor 45 is negative. Figure 6.12 illustrates the main effects of factor 4 and 5. In Figure 6.12a we see that when only the level of factor 4 changes, the number of shortages decreases from 2.4 to 1.4. The opposite holds for factor 5. Figure 6.13 illustrates the interaction effect. In this figure, we observe the effect of factor 4 by the difference between level -1 and 1 when the level of factor 5 is fixed. The average over these two differences (level -1 and 1 of factor 5) is the main effect of factor 4. The difference is much larger when factor 5 is on level 1. So, the main effect of factor 4 is mainly accomplished by the interaction with factor 5. So, when a main effect is also within a interaction effect, we need to look at this before we can say anything about the main effect.

### 6.2.4 Results

In this section we discuss the significant effects. Tables 6.10 and 6.11 show the confidence intervals for each significant effect per KPI and modification method. In these tables, we indicate a main effect by the number of the factor and an interaction effect between two factors by the numbers of both factors. For example, 345 indicates an interaction effect of factor 3, 4, and 5.

First, we describe and explain the most important observation per modification method. For each method we also recommend which settings are 'best'. Finally we discuss the differences in recommendations between the methods.




(a) Main effect factor 4 (b) Main effect factor 5

Figure 6.12: Main effects of factor 4 and 5

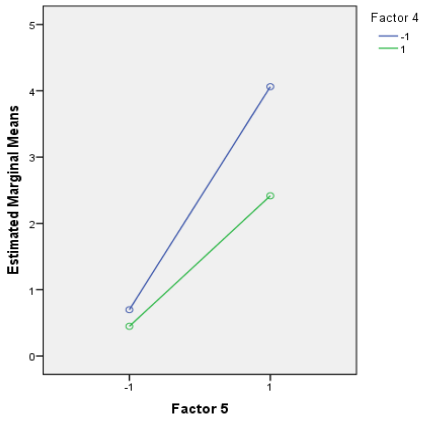



Figure 6.13: Shortages - Interaction effect factor 4 and 5

### Shuffle

The significant (interaction) effects of the Shuffle method are summarized in Table 6.10. The most important observation is:

*Factors 4, 5, and 45 have the largest influence on all KPIs for all cases.*
Since each case also has an interaction effect, we first have to look at this effect. The interaction effect differs for (Minimum) Remaining percentage and Shortages. First, we explain the interaction effect for (Minimum) Remaining percentage, after which we explain the interaction effect for Shortages.

In Figure 6.14 we see the interaction effect of 45 for Pompestichting for Remaining percentage. The figure of the Pompestichting for Minimum remaining percentage and the figures of the Westfriesgasthuis cases at both these indicators are similar. In this figure, we see that both main effects are less when both factors move from level -1 to 1. Nonetheless, the average response when both

Table 6.10: Confidence intervals - Shuffle (SE = Significant effect, LB = Lower bound, UB = Upper bound, PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, and WFG-2 = Westfriesgasthuis-2)

| **KPI** | **Case** | **SE** | **LB** | **UB** |
|---|---|---|---|---|
| Remaining percentage | PS | 5 | 0.4233 | 2.5938 |
| | | 4 | 0.9513 | 1.8139 |
| | | 45 | −0.7240 | −0.1983 |
| | | 3 | −0.3244 | −0.0971 |
| | | 134 | 0.0068 | 0.1164 |
| | WFG-1 | 4 | 0.5997 | 1.2457 |
| | | 45 | −0.1047 | −0.0199 |
| | | 125 | 0.0036 | 0.0535 |
| | WFG-2 | 4 | 0.5057 | 0.9936 |
| | | 5 | 0.0035 | 0.1569 |
| | | 45 | −0.1316 | −0.0289 |
| Shortages | PS | 5 | 0.7161 | 4.6172 |
| | | 4 | −1.4908 | −0.4051 |
| | | 45 | −1.0921 | −0.3038 |
| | | 3 | −0.3749 | −0.1043 |
| Minimum remaining percentage | PS | 5 | 6.0560 | 9.9038 |
| | | 4 | 0.4365 | 1.7584 |
| | | 45 | −1.2662 | −0.3335 |
| | WFG-1 | 5 | 2.6493 | 7.5441 |
| | | 4 | 0.3288 | 1.8289 |
| | | 45 | −1.2539 | −0.1598 |
| | | 12 | −0.6342 | −0.0355 |
| | WFG-2 | 5 | 4.7211 | 9.2675 |

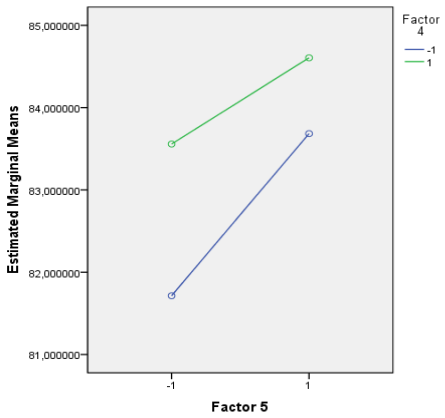



Figure 6.14: Remaining percentage - Interaction effect factor 4 and 5 at Pompestichting

factors are at level 1 is the highest and therefore the best option. This means that starting with only one employee (level 1, factor 4) combined with a minimum percentage of 80% (level 1, factor 5) is most profitable for the (minimum) remaining percentage. We can explain this: if we start with one employee, we do not have to make a combination of swap options. So, we just select one swap. In this case, CPLEX selects, if possible, a swap on the same day. A swap on the same day causes less decrease of the remaining percentage, because we calculate this as the number of days where we changed the assignment. In case three employees are considered, CPLEX needs to select a combination and selects more often swap options that make swaps on different days. This causes less remaining percentage for these employees. By setting a minimum percentage of 80% (level 1, factor 5), we perform fewer swaps in some schedules than at a minimum of 70% (level -1, factor 5). Hence, the employees have a higher (minimum) remaining percentage. With this information, we can explain the interaction effect for (Minimum) Remaining percentage. When we start our method with 3 employees (level -1, factor 4), more employees receive a remaining percentage lower than 80% than when we start with 1 employee (level 1, factor 4). Therefore, a minimum percentage of 80% (level 1, factor 5) has a greater effect on the average remaining percentage when we start with 3 employees (level -1, factor 4). The effect of factors 4, 5, and 45 is different for Shortages than for (Minimum) Remaining percentage. Figure 6.13 shows that the combination of factor 4 level 1 (start with 1 employee) and factor 5 level -1 (minimum of 70%) causes the least number of shortages left. By setting a minimum of 70% instead of 80%, more swaps are made in the same employee schedule and therefore more shortages are fulfilled. Starting with three employee instead of one causes CPLEX to select more often swaps on two different days. This causes a lower remaining percentage per swap than a swap on the same day. Therefore, the minimum percentage is reached faster. So, starting with one employee, causes employees to reach the minimum percentage less quick. Therefore, with this setting it is possible to perform more swaps in the schedules and fulfill more shortages. The lower the minimum percentage is, the more shortages we are able to fulfill.

**Recommendation** For (Minimum) Remaining percentage it is most profitable to start with one employee (level 1, factor 4) and set a minimum of 80% (level 1, factor 5). For Shortages it is most profitable to start with one employee (level 1, factor 4) and set a minimum of 70% (level -1, factor 5). The most profitable level at factor 5 contradicts for both KPIs. The lower the minimum percentage, the more shortages we expect to be fulfilled.

### Disturbance

The significant (interaction) effects of the Disturbance modification method are summarized in Table 6.11. The most important observation is:

*Factors 3 and 5 have the largest influences on all KPIs for all cases.*
First, we explain the effect of factor 3 (swap strategy). At the Disturbance method we created shortages in one week and excesses in another. We did this despite the fact that all employees had full schedules for both weeks. Due to the working hours act and the few overstaffed and understaffed shifts, there is a small chance that an employee has an overstaffed shift in the first week and is able to swap this shift for an understaffed shift in the second week (primary swap). Therefore, when there are no primary swaps available anymore (level -1, factor 3), secondary swaps take over (level 1, factor 3). With these, we fulfill more shortages, but also cause lower remaining percentages.

Table 6.11 shows a high confidence interval for Organization X on Shortages: factor 3. Table 6.8 shows that the average number of secondary swaps performed at Organization X is relatively higher than for the other cases. In the dataset of Organization X, the preferred schedules are created from 21 and 24-week cyclical schedules. Our planning period is 4 weeks. So, every shift sequence within a week appears at least 4 times in the schedule. The schedules have on average 76 employees, so we start the cycle 3 to 4 times. So, each shift sequence within a week is repeated about 16 times in one schedule. Thus, when in one of those sequences primary swaps cannot be performed, it affects multiple shift sequences in the schedule. In contrast to a non-cyclical schedules, we use relatively more secondary swaps to finalize a cyclical schedules.

Factor 5 (minimum percentage) affects the KPIs as discussed at the Shuffle method. When there are relatively many swaps needed and few employees available, schedules are most likely to reach the minimum percentage. A change of this minimum causes a different (minimum) remaining percentage and number of shortages left.

**Recommendation** For the (Minimum) Remaining percentage it is most profitable to set a minimum of 80% (level 1, factor 5). While, for the shortages left it is most profitable to set a minimum percentage of 70% (level -1, factor 5) and use primary and secondary swaps (level 1, factor 3). Again, the settings of factor 5 contradict for the different KPIs.

### Overall discussion

Now we have discussed the most important observations within each modification method, we are able to compare the results for both methods. The main difference is that the Shuffle method causes significant (interaction) effects, of

Table 6.11: Confidence intervals - Disturbance (SE = Significant effect, LB = Lower bound, UB = Upper bound, PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, WFG-2 = Westfriesgasthuis-2, and X = Organization X)

| **KPI** | **Case** | **SE** | **LB** | **UB** |
|---|---|---|---|---|
| Remaining percentage | PS | 3 | −1.4580 | −0.6445 |
| | | 5 | 0.0053 | 0.2959 |
| | WFG-1 | 3 | −0.2031 | −0.0376 |
| | | 1345 | −0.0072 | −0.0002 |
| | WFG-2 | 3 | −0.0696 | −0.0031 |
| | | 134 | 0.0025 | 0.0220 |
| | X | 3 | −1.0700 | −0.2752 |
| | | 5 | −1.7E-14 | −3.3E-16 |
| Shortages | PS | 3 | −1.5651 | −0.6641 |
| | | 5 | 0.0366 | 0.5676 |
| | WFG-1 | 3 | −0.7771 | −0.1396 |
| | WFG-2 | 3 | −0.3895 | −0.0168 |
| | | 134 | 0.0156 | 0.1198 |
| | X | 3 | −5.7162 | −1.5545 |
| Minimum remaining percentage | PS | 5 | 2.5479 | 6.0831 |
| | WFG-1 | 5 | 0.1872 | 1.3753 |

factor 4, 5, and 45, over all KPIs, while the Disturbance method causes significant effects of factor 3 and 5. Next, we explain the difference in the effect of factor 3 and 4 between the modification methods.

In the schedules created by the Disturbance method we have either periods of overstaffed or understaffed shifts. When we start with one employee (level 1, factor 4), CPLEX selects mostly swaps on the same day if possible. This is not possible in the schedules of the Disturbance method and therefore this factor has only an effect at the schedules of the Shuffle method.

We already explained the effect of factor 3 at the Disturbance method. At the Disturbance method we create periods with either overstaffed or understaffed shifts. All employees already had full schedules for both weeks, so with satisfying the working hours act and relatively few overstaffed and understaffed shifts, it is hard to find possible swap options. Therefore, using secondary swaps in these schedules effects the number of shortages left and (minimum) remaining percentage in these schedules.

## 6.3 Response analysis

In Section 6.2.3 we assumed, using the CLT, that all sample data is normally distributed to conduct the confidence intervals. In this section, we show that this assumption does not lead to inconsistent conclusions.

We use graphs of the responses per instance, KPI, and design point to identify the effects. We have 12 schedules per instance, so, we have 12 responses at each design point. Next, we calculate the average over all responses and the average over the responses per design point. We show these also in the graphs, see for example Figure 6.15.

We know at each design point what the settings of the factors are. Therefore, we can determine the effects by analyzing the graph. Table 6.12 shows when each factor changes level in the design matrix. For example, factor 5 changes level after each 16 design points. There are 32 design points, so this happens once per design. If we see a transformation between the first 16 and the second 16 dots, the choice of level at factor 5 has an effect on the responses. We use this information to show that the significant effects per factor are indeed present.

Table 6.12: Level changes per factor

| **Factor** | **Number of design points after which the level changes** |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 4 |
| 4 | 8 |
| 5 | 16 |

#### Pompestichting - Shuffle

In this section we analyze the graphs per KPI of the responses created by the dataset: Pompestichting - Shuffle. Appendix F shows the analyses of the other

datasets. At the end of this section we summarize the results of all analyses.

In Figure 6.15 we clearly see a movement between each group of 8 large dots. Within each group of 16 large dots, the first 8 dots are lower than the second 8 dots. This indicates the effect of factor 4 (initial employees). Starting with one employee (level 1) has a positive effect on the Remaining percentage. Over all 32 points we see that the first 16 large dots are lower than the second group of 16 large dots, this indicates the effect of factor 5 (minimum percentage). Setting a minimum of 80% has a positive effect on the Remaining percentage. Finally, we see that the difference between the first 8 and second 8 large dots is larger than the difference between the third and fourth group of 8 large dots. This indicates the negative interaction effect between factor 4 and 5.

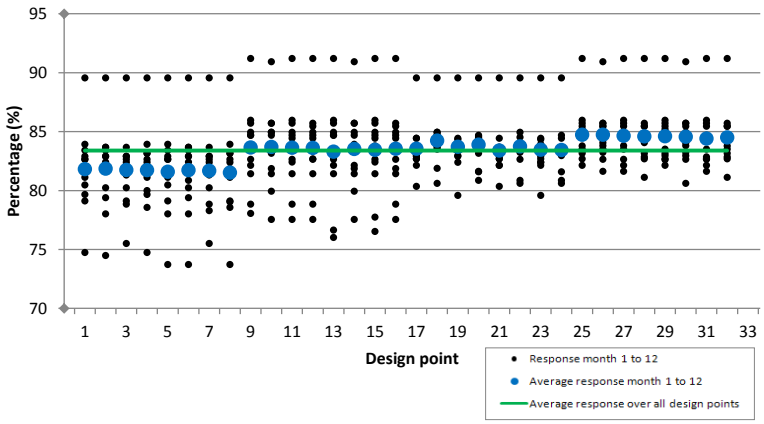



Figure 6.15: Remaining percentage

In Figure 6.16 we see a difference between the first 16 large dots and the second 16 large dots. This indicates the effect of factor 5 (minimum percentage): setting a minimum of 70% (level -1) has a positive effect on the number of shortages left. In the second group op 16 large dots, we see that the first 8 large dots are in general higher than the second 8 large dots. In the first 16 large dots, we see similar behavior, but much less. This indicates the interaction effect between factor 4 and 5. The effect of factor 4 (initial employees) is mainly caused by the interaction effect, which we already discussed in Section 6.2.4.

In Figure 6.17 we again see a difference between the first 16 large dots and the second 16 large dots. This indicates the positive effect of factor 5 (minimum percentage). Within the first 16 large dots we see that the second 8 are higher than the first 8, but we do not see this within the second 16 large dots. Again, this is the interaction effect of factor 4 and 5. As showed in Section 6.2.4 the effect of factor 4 is mainly established by interaction effect 45.

### Summary

Table 6.13 shows all noticeable effects from the analyses of the cases. In this table, we see that the effects of factors 4, 5, and 45 are still the most important at all cases and KPIs with modification method Shuffle. However, there are some differences:

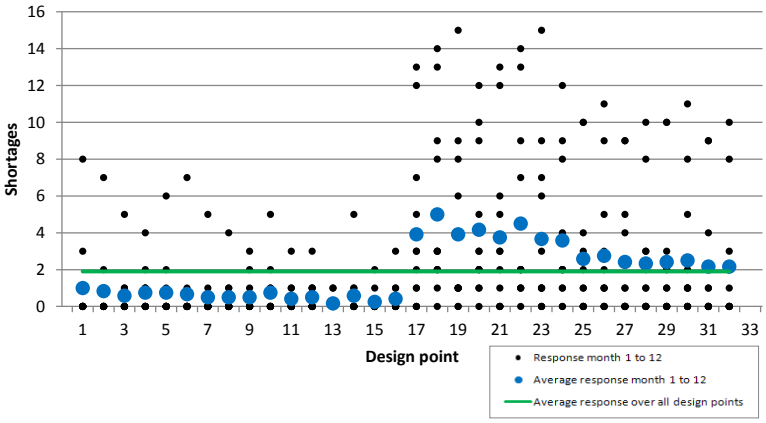



Figure 6.16: Shortages

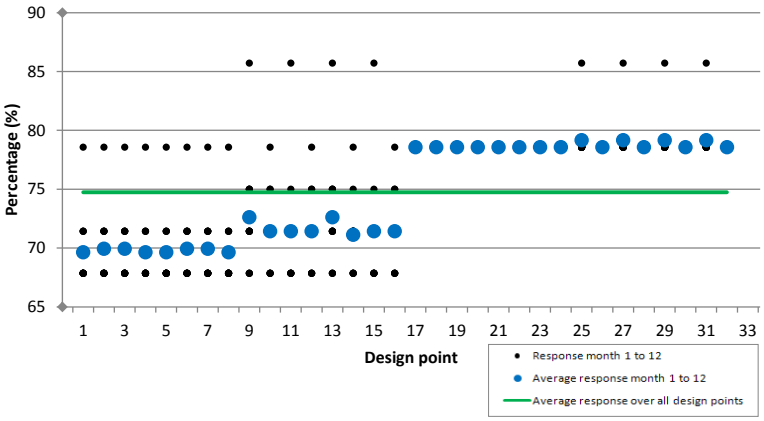



Figure 6.17: Minimum remaining percentage

- Although the effect of factor 5 for Westfriesgasthuis-1 and Westfriesgasthuis-2 with the Shuffle method were not found significant, they were present in the graphs. This strengthens our conclusion about the importance of factor 5 for the schedules created with the Shuffle method.
- Although factor 4 was found significant, it was not always visible in the graphs of Minimum remaining percentage for the Shuffle method. In Figure 6.14 we saw that effect 4 is mainly caused by the interaction effect between factor 4 and 5 (the figures of the other cases are similar). This interaction effect is present in the graphs, which leads us to conclude that the effect of factor 4 is also present.

For the modification method Disturbance, we still see that factor 3 is clearly present at the key performance indicators Remaining percentage and Shortages and factor 5 is most important at the Minimum remaining percentage.

So, overall we see that the assumption of normality did not lead to inconsistent conclusions.

Table 6.13: Expected effects (PS = Pompestichting, WFG-1 = Westfriesgasthuis-1, WFG-2 = Westfriesgasthuis-2, and X = Organization X)

| **Modification** | **Case** | **Remaining percentage** | **Shortages** | **Minimum remaining percentage** |
|---|---|---|---|---|
| Shuffle | PS | 4, 5, 45 | 5, 45 | 5, 45 |
| | WFG-1 | 4 | 5 | 5, 45 |
| | WFG-2 | 4 | 5 | 5 |
| Disturbance | PS | 3 | 3, 5 | 5 |
| | WFG-1 | 3 | 3 | 5 |
| | WFG-2 | 3 | 3 | 5 |
| | X | 3 | 3 | - |

## 6.4 Conclusions

We performed a preliminary test to indicate which levels we want to compare in the $2^k$-factorial design. Table 6.1 shows the selected levels per factor.

Using the $2^k$-factorial design we calculated the significant effects schedule per dataset. From these, we conclude that the choice of level at factor 4 (initial employees) and factor 5 (minimum percentage) at the schedules created by the Shuffle method are most important to keep track of. Factor 3 (swap strategy) and factor 5 (minimum percentage) are most important to keep track of at the schedules created with the Disturbance method. So, we specify two situations:

*Situation 1: Relatively often understaffed and overstaffed shifts on the same day (Shuffle method)*

**Organization** From the organization's point of view it is most preferable to use a minimum of 70% (level -1, factor 5) instead of a minimum percentage of 80% (level 1, factor 5). It does not matter which levels are chosen for the other factors.

**Employee** From the employees' point of view it is preferable to start our method with one employee (level 1, factor 4) and set a minimum of 80% (level 1, factor 5). It does not matter which levels are chosen for the other factors.

**Planner** All combinations of levels result in a total computation time of less than 11 seconds. Therefore, from the planner's point of view it does not matter which levels are chosen.

*Situation 2: the preferred schedules of all employees result in periods of either understaffed or overstaffed shifts (Disturbance method)*

**Organization** In this situation, only the choice of level at factor 3 is important. The choice for primary and secondary swaps (level 1, factor 3) fulfills significantly more shortages than using only primary swaps.

**Employee** From the employees point of view it is most preferable to use only primary swaps (level -1, factor 3). Next, it is preferable to use a minimum percentage of 80% (level 1, factor 5). It does not matter which levels are chosen for the other factors.

**Planner** Again, the maximum computation time is less than 11 seconds. Therefore, from the planners point of view it does not matter which levels are chosen.

# Chapter 7

# Conclusions and further research

As outlined and motivated in Chapter 1, the goal of this thesis is to design a method that helps planners finalize the schedule in the last step of the self rostering method and that is widely applicable. In Section 7.1 we describe our conclusions. In Section 7.2 we describe the possible directions for further research.

## 7.1 Conclusions

In industries such as health care, security, and transport, employees have to work shifts around the clock. To balance their work with their private responsibilities, employees often have specific shift preferences. It is hard for planners to make a schedule that covers both the staffing demand and the preferences of the employees, while meeting all labor requirements. Self rostering is a method that helps the planner creating such a schedule.

The self rostering process consists of several steps that starts with employees creating their own preferred schedule. All preferred schedules together most likely do not match the staffing demand. Therefore, changes need to be made. First, employees get the chance to change their schedule and second, the planner makes the necessary changes. To help the planner, we designed a method that swaps shifts to match the actual assignment with the staffing demand as much as possible.

From cases from practice, our method has to satisfy the labor legislation and retain as much as possible of the preferred schedules, at least (on average) 80%.

In every iteration of our method, we select a number of employees and swap shifts in their schedules to fulfill the understaffed shifts. The employee selection is based on the scores of the employees. The score of an employee is calculated by summing all points per employee, multiplied by some factor, e.g., to take part-time percentages into account. Employees receive points for each shift in their schedule. Employees with a high score work many relatively unpopular shifts, whereas for employees with a low score the opposite holds. The higher this score, the lower the chance for the employee of being selected and receiving a swap in his schedule. In this way, we are in control of which employees are

in the iteration and have the possibility of receiving a swap in their schedule. When we do not want to perform swaps on the subset of employees with the lowest score, we can easily select another subset of employees.

We considered two types of swaps: primary swaps and secondary swaps. A primary swap swaps shifts in the schedule of a single employee. A secondary swap performs two swaps, where each swap is performed at another employee. Secondary swaps are able to fulfill understaffed shifts where primary swaps sometimes cannot. Swap options are only created if they satisfy the labor legislation. We check all possible swap options on labor legislation in a separate process. This makes our method flexible: we can easily adjust, add or delete rules from the labor legislation.

All selected employees and their swap options are input for the mixed integer linear program (MILP), which selects the optimal subset of swaps. The MILP selects at most one swap per employee. Doing this, it makes a trade-off between two factors. On the one hand, we want to minimize the number of understaffed shifts by applying swaps. On the other hand, we prefer to perform certain swaps (e.g., swaps that result in the highest score change) at certain employees (e.g., employees that have a low score).

To make sure employees retain at least (on average) 80% of their preferred schedule, we added a check to our method. This check, forbids swaps in the schedule of an employee if less than X% (where X has to be predetermined) of the preferred schedule is retained after performing another swap.

We applied our method to cases from practice and used 3 KPIs to evaluate our method: Shortages, Remaining percentage, and Computation time. We observed similar results for all cases. On average we started each time with 59.67 shortages and we finished with 0.90 shortages left. Our method retained on average 92% of the preferred schedules and the maximum computation time is 12 seconds for a planning horizon of 28 days, and about 70-80 employees (see Tables 6.8 and 6.9).

To fulfill more shortages or to retain a higher fraction of the preferred schedules, we studied the (interaction) effects of our input parameter settings. We have 5 input parameters for our method, each with two settings (see Table 6.1). We observed that when we use primary and secondary swaps combined with a constraint that 70% of the preferred schedules should be retained, on average most shortages are fulfilled. This is preferred by the stakeholder Organization. When we use only primary swaps, start our method with only one selected employee, and use a constraint that 80% of the preferred schedules should be retained, we observed that the percentages ratained of the preferred schedules are on average the highest. This is preferred by the stakeholder Employees. So, the only difference between the preferred input parameters settings per stakeholder are: the minimum percentage that has to be retained of the preferred schedules and whether to use only primary swaps or primary and secondary swaps.

Concluding, we designed a suitable method to help the planner finalize the schedule and that is widely applicable. We gained insight in the influence of different input parameter settings for our method, which is valuable information for the user in practice.

## 7.2 Further research

Our method already showed some promising results. This section discusses interesting topics left for further research.

**Quality of schedules** In our analysis we measured the quality of the schedules with the retained fraction of the preferred schedule. However, certain swaps can result in a higher retained fraction, but a lower quality of the schedule. During interviews for the cases, we discovered that schedule preferences differ per employee. For example, some employees prefer periods were they either have work or do not have not work. So, if our method swaps one shift from the middle of a work period into the middle of the not work period, the employee receives a schedule in which the work and not work days are mixed. In this case it is possible that a second swap, thus a lower retained fraction, creates periods of work and not work again, which is more preferable. We have two suggestions to take the quality of the schedules further into account.

Our first suggestion is to add a functionality to our method that takes not work wishes into account. This is possible by punishing the situation were we select a swap option to a day with a not work wish in the MILP. The punishment must be not too large, so that we do select the option when nothing else is available (it is still a wish), but large enough to not prefer this option.

Our second suggestion is to analyze whether the employees use forward or backward rotation and permit our method to disrupt these rotations. Forward rotation means that the start time of the next shifts is at least the start time of the previous shift. At backwards rotation is the start time of the next shift at most the start time of the previous shift. If employees use a rotation in their schedule, it is good for the quality of the schedules to retain this rotation. We already implemented this by permitting the creation of swap options that interrupt the rotation. Due to time issues we did not analyze the effects of this constraint on our method.

**Additional functionalities** A point of improvement could be the order in which we add employees to the number of selected employees and use secondary swaps. In the current situation we only use secondary swaps when all employees are selected and there are no primary swaps left. Our motivation is that primary swaps causes fewer changes in the preferred schedules than secondary swaps. So, we first try to fulfill all shortages by performing primary swaps. We expect that the strategy: 'when no primary swaps are selected use secondary swaps, if then no secondary swaps are selected add one employee and use primary swaps', could influence the number of shortages left and the remaining percentage of the employees and is therefore worthwhile to investigate.

Another functionality is to be more flexible in the amount of work hours per planning period. We assumed that employees proposed schedules in which they precisely cover their contract hours. We only swapped shifts for other shifts and all shifts have equal lengths. Therefore, the total amount of work hours in a schedule remains the same. However, all cases wish more flexibility in the amount of planned work hours. They wish

that employees have a minimum and maximum amount of work hours per period and satisfy their work hours annually. Our suggestion is to implement this by adding a constraint that causes employees to plan at least the minimum amount of work hours and that they cannot plan more than the maximum amount. Besides this, the method has to keep track of the hours worked per employee and we suggest further research on a strategy for out method to plan extra shifts or remove shifts from a schedule of an employee.

**Benchmark** It might be interesting to benchmark our method against other methods. To our knowledge, these do not exist or are not accessible. However, it is possible to benchmark our method against the work of a planner.

Another option is to benchmark against an optimal method. We expect that it is possible to design an integer linear program that fulfills all shortages at once and retains the highest fraction of all preferred schedules. This method is not preferable, because it is not flexible in, among others, the labor legislation, and it is not transparent to the employees. It is not possible to tell afterwards why certain swaps took place. Finally, we expect this method to take a lot of computation time, which is also not preferable. However, the schedules are optimal and from a theoretical point of view, we could benchmark our method.

**Mathematical adjustment** At 'Benchmark' we discuss the option of using only an integer linear program to solve our problem. However, as discussed at 'Benchmark' this method is not preferable. In contrast to this, our method is a greedy method: each iteration it chooses a local optimum. An alternative to our method could be a method that calculates several iterations ahead before choosing a set of swaps in the current iteration. Then, we are able to climb out of a local optimum and have a higher chance to find the global optimum.

**Exploring further industries** It is not in the scope of our research, but we think it is interesting to further explore the options of using self rostering outside the industrial business, e.g., volunteer planning, and whether our method is also applicable to these cases.

# Bibliography


Clare L. Bambra, Margaret M. Whitehead, Amanda J. Sowden, Joanne Akers, and Mark P. Petticrew. Shifting schedules: The health effects of reorganizing shift work. *American Journal of Preventive Medicine*, 34(5):427 – 434.e30, 2008.

Vašek Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.

Melanie L. De Grano, D. Medeiros, and David Eitel. Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Management Science*, 12:228–242, 2009.

Averill M. Law. *Simulation Modeling and Analysis*. McGraw-Hill, 4th edition edition, 2007.

Arthur Lubbers. Het zweedse wondermiddel: Innoveren met zelfroosteren. *IntermediairPW*, 2008. derived from `http://www.intermediairpw.nl/artikel/vakinformatie/170437/het-zweedse-wondermiddel.html`, May 10, 2011.

Ministry of Social Affairs and Employment. The working hours act: information for employers and employees, 2010. derived from `http://www.rijksoverheid.nl/onderwerpen/werktijden/documenten-en-publicaties/brochures`, May 25, 2011.

NCSI. *Individueel roosteren – Kansen voor werkgevers en werknemers*. Nederlands Centrum voor Sociale Innovatie, 2009.

Paralax BV. Participerend plannen: Het nederlandse model voor zelfroosteren. `http://www.paralax.nl/whitepaper`, Retrieved on: Octobre 15, 2010.

Elina Rönnberg and Torbjörn Larsson. Automating the self-scheduling process of nurses in swedish healthcare: a pilot study. *Health Care Management Science*, 13:35–53, 2010.

Louise Thornthwaite. Working time and work-family balance: A review of employees' preferences. *Asia Pacific Journal of Human Resources*, 42(2): 166–184, 2004.

# Websites

IBM. IBM ILOG CPLEX Optimizer - Software. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, March 16, 2012.

MathWorks. MATLAB - Software. http://www.mathworks.nl/products/matlab/, March 16, 2012.

NCSI. http://www.ncsi.nl/nl/ncsi/organisatie, May 23, 2011.

NCSI. Zelfroosteren bij NedTrain - werken als er werk is. http://www.ncsi.nl/nl/kennis/kennisbank/zelfroosteren-bij-nedtrain------werken-als-er-werk-is/795, July 4, 2011.

NedTrain. http://www.nedtrain.nl/nl/2/organisatie.html, July 4, 2011.

ORTEC. http://www.ortec.com, April 28, 2011.

Pompestichting. http://www.pompestichting.nl/index.php?option=com_frontpage&Itemid=93, July 6, 2011.

Westfriesgasthuis. http://www.westfriesgasthuis.nl, July 6, 2011.

# Appendix A

# Cases from practice

In this appendix we describe the cases in more detail. We use information provided in interviews by contact persons of each organization (see Table A.1) to describe the cases.

Table A.1: Contact persons per organization

| **Organization** | **Industry** | **Contact person(s)** | **Function** |
|---|---|---|---|
| Organization X | Transport & Logistic | - | - |
| NedTrain | Services | Edwin Arts | Location manager at Binckhorst |
| Pompestichting | Health care (forensic) | Nanco Janssen | Application manager Personnel planning system |
| Westfriesgasthuis | Health care | Carola Stoutjesdijk | Unit Head, Obstetrics, Gynecology, and Transfer |
| | | Fred Beemsterboer | Former executive |

The first organization wants to stay anonymous, so we call it Organization X. The contact persons and their functions are not mentioned and, in the description of the case, we do not give general information about this organization. At all other cases, we first give some general information about the organization. Second, we describe at all cases the current scheduling process of the organization. In the last few years, some organizations had a self rostering pilot. At these organizations we describe the scheduling process prior to the pilot. Next, we describe the goal that the organization wants to achieve by using self rostering. Finally, we describe criteria per case. These criteria are categorized by the organization. We call the criteria that method certainly has to meet *requirements* and we call the criteria that are nice to have *wishes*.

## Transport and logistic - Organization X

### Scheduling process

Organization X wants to implement individual rostering in two maintenance departments. Both departments have around 100 employees that are divided in approximately 6 teams. Each team consists of employees with different positions (there are about five different positions). Each employee can have different qualifications, even employees with the same function. The teams are now scheduled in a cyclic roster. The staffing demand varies per hour, per day, per month and per season. So the use of a cyclic roster, which results in a fixed amount of staff present at each moment of the day and week, is not always efficient.

In the past, self rostering pilots are performed in the organization. From the pilots, they learned that it is important to have a large enough group of non-homogeneous employees, to use user friendly IT support, that there has to be enough space in the collective labor agreement, and above all, the reasoning behind the changes in the schedules of the employees have to be clear.

### Goal

The main goal of Organization X is to shift the control over work and rest times towards the employees. This way, the employees receive more diversity in their schedule and colleagues to work with. The employees are also in a better position to balance their personal life with their work responsibilities. Besides the advantages for the employees, the organization wants to have more flexibility in defining its staffing demand.

### Requirements:

1. *Transparency*
   According to Organization X, it is most important that the process to the final schedule is transparent. With this, we mean that the planner can explain each step in the scheduling process to the employees, such that the employees understand the changes in their schedules.

2. *Ground rules*
   Organization X wants to use two phases in the method, *employee selection* and *swap selection*. In employee selection they determine a ranking order of the employees. The employees on top receive most likely a swap in their schedule. Organization X defined three ranking strategies and has still to choose which one they want to use. These strategies are:

   (a) *Seniority*
   This rule sorts employees by their age or length of employment. Older employees or the employees who work the longest are at the bottom of the list. The rule aims to first use the employees at the top of the list to fulfill the staffing demand.

   (b) *Point system*
   Employees who choose a shift that nobody else wants to work (an unpopular shift), receive many points for this shift. Employees who

choose a shift that everyone wants to work (a popular shift), receive only a few points for this shift. Employees are ranked up on their total points. Employees with the least points have the highest chance of being selected to fulfill the staffing demand in the last step of the self rostering process.

(c) *Groups*
All employees are divided in groups. In the first period, all groups are ranked arbitrary. In the next periods, the groups rotate. So, if we have 4 groups and group 1 is the group with the highest ranking in the first period, this group becomes group 4 in the next period. In the third period they become group 3, etcetera. Every scheduling period the employees in the top group try to fulfill the staffing demand first, next the second group does an attempt, etcetera. So, the employees in the last group have a high chance of receiving their preferred schedule.

After the employee selection, the swap selection phase starts. For this, Organization X did not defined an entire strategy yet. They know that they want to perform the most unpleasant swaps in the schedules of the employees on top of the list. However, they still have to determine what the most unpleasant swap is.

3. *Working hours act (WHA) and collective labor agreement (CLA)*
The final schedule may not violate the working hours act or the collective labor agreement.

**Wishes:**

1. *If possible, the method should take the following factors into account*
If there are several similar changes that can be made in the schedule of the selected employee and the method should choose one, it has to take the following constraints into account:

   (a) *Planning forward in rotation*
   Planning forward in rotation means that a shift on a next day begins at the same time or later than the shift of the day before. If an employee has planned himself forward or backwards in rotation, the method has to take this into account.

   (b) *Not-work wish*
   Per period, the employee receives two wild cards. The employee can assign these on free hours in his schedule or on shifts. He cannot collect them over different scheduling periods and cannot trade them with co-workers. These wild cards indicate when an employee prefers not to work or to work a specific shift. The method has to take these into account, if possible.

   (c) *Number of changes*
   The total number of changes over all schedules should be minimized, so that a maximum number of wishes in the preferred schedules remains.

   (d) *Work hours*
   The method has to satisfy the contract hours of an employee per

year. Per period, there is a maximum and a minimum of hours. The maximum of hours is the contract hours of the employee +60 hours and the minimum is the contract hours of the employee −40 hours.

## Services - Nedtrain

NedTrain is a Dutch company that is specialized in maintenance and services (cleaning and revision) on rolling material, mostly concerning trains. NedTrain has over 30 sites from which it takes care of its customers' railroad cars and locomotives, 24/7. (NedTrain, 2011).

### Scheduling process

Den Haag Binckhorst is one of the maintenance locations. Shunters, (control) mechanics, drivers, and cleaners work at this location. All these employees are scheduled in a cyclic roster, although it is known that some periods offer less work. Fluctuation in demand is even possible within a day, e.g., during peak hours in the morning all trains are busy, but before and after the peak hours NedTrain has to check the trains and if needed they carry out maintenance.

Fluctuating demand combined with a cyclic roster results in too many scheduled employees at some times. Therefore, some employees have nothing to do, which is not efficient. Next to a less efficient schedule, 65% of the employees is dissatisfied with their rosters. This is why location Binckhorst started a pilot study self rostering in June 2009. (NCSI, 2011)

### Goal

The main goal is to improve the work-life balance of the employees. Self rostering provides the opportunity for employees to combine personal matters with work responsibilities. A secondary goal is to increase efficiency, i.e., to create a better balance between workload and staff present. With self rostering, the staffing demand can vary per shift per day, so shifts are only scheduled when there really is work to do.

### Requirements:

1. *Full weekends*
   NedTrain specified a red weekend in their collective labor agreement. This means that in any period of 3 weeks each employee has to be free for at least one weekend. So if an employee planned himself for two morning shifts on consecutive Saturdays, he knows that he is free the next weekend. If employees only schedule themselves for one shift per weekend, valuable work hours in weekends are lost. Therefore, the method has to schedule a full weekend (Saturday and Sunday), vacations and holidays are scheduled separately.

2. *Working hours act (WHA) and collective labor agreements (CLA)*
   All proposed schedules have to satisfy the working hours act and the collective labor agreement. Also, the method should satisfy these in order to create final schedules without any violation.

**Wishes:**

1. *Not-work wish*
   The method takes 'not-work-wishes' into account. It has to be possible for employees to specify a certain amount of hours per period which they really cannot or do not want to work, e.g., due to sports, kids at the daycare, or hobbies. When the method changes the schedule of an employee, it has to consider the not-work wishes.

2. *Number of changes*
   The method should retain on average (over the whole year) at least 80% of preferred schedules. It is possible that one period the percentage that remains is 60% and another period it is 100%, but the average has to be at least 80%.

3. *Planning forward in rotation*
   Planning forward in rotation means that a shift on a next day begins at the same time as the shift of the day before or later. The method has to take this into account when it changes shifts. So, for example, it is only possible to swap a night or a day shift after a day shift, a morning shift is not allowed.

4. *Work hours*
   The method has to satisfy the contract hours of an employee per year. Per period, NedTrain uses a minimum and maximum number of hours. The minimum per period is the contract hours $-16$ and the maximum is the contract hours $+16$. If an employee works extra hours in a month, he has to use these in the next month. This prohibits employees from saving many hours throughout the year, such that they are free in December.

## Health care (forensic) - Pompestichting

The Pompestichting is a private institution for forensic psychiatry. The main goal of the Pompestichting is to contribute to the safety of the society by offering treatment for people with a psychiatric disorder who are likely to commit or already committed a serious crime. (Pompestichting, 2011)

### Scheduling process

Department De Niers is a high secured forensic department. This department needs at least 9 FTE of socio therapists allocated over 24 hours 7 days a week. The staffing demand is known and stable, so they defined a standard staffing demand per period. Every month employees are rostered manually by a team member who knows all the wishes and preferences of the employees. If there are problems with finalizing the schedule, the manager of the department decides which employee should be assigned to the understaffed shift. Pompestichting wants to centralize their planning process, i.e., the schedules are made by a central planning department who do not know the employee preferences and wishes. To keep employees satisfied with their roster, Pompestichting wants to see whether self rostering is a solution. This is why a pilot study took place at Department De Niers from October 2009 until December 2010.

### Goal

The goal of the pilot was to see whether self rostering helps the employees to balance their personal responsibilities with their work responsibilities if rosters are not made by a team member. They wanted to see whether this way of planning works for employees regarding the opportunity to indicate their wishes and preferences.

### Requirement:

1. *Working hours act (WHA) and collective labor agreement (CLA)*
   The method should allow schedules that contain violations concerning the working hours act and collective labor agreement, but the method should not create new violations. If some proposed schedules contain violations that are approved by the planner, the method does not have to try to fix these violations. All changes made by the method should satisfy the working hours act and the collective labor agreement.

### Wishes:

1. *Number of changes*
   After the method has finished the schedule, at least 80% of each schedule has to be unchanged. So, 80% of the wishes of each employee remains.

2. *Work hours*
   The working hours per employee are flexible within a planning period, but fixed per year. The method should take this into account. The contract hours that an employee has to fulfill per period is flexible. So, it is possible to work more than the defined contract hours in a month and work less in another month. The contract hours per year are fixed, thus after a year the total number of contract hours have to meet the required number of contract hours.

## Health care - Westfriesgasthuis

The Westfriesgasthuis is a general hospital with a yearly output of 506 operational hospital beds and 258,000 polyclinic visits. The medical staff represents 26 specialisms. (Westfriesgasthuis, 2011)

### Scheduling process

For many years, some departments have worked with a form of self rostering. One of these departments is Longgeneeskunde (EN: pulmonary medicine). At Longgeneeskunde, nurses, student nurses, secretaries, flex employees, unit heads, and service employees are working. However, only the nurses and student nurses participate in self rostering. Each (student) nurse has access to Harmony, where they can assign shifts to themselves. Harmony shows the staffing demand per shift per day and all (student) nurses know the agreements, e.g., about the amount of shifts they have to plan and the ratio between student nurses and nurses present per type of shift. If a nurse wants a shift that already meets its staffing demand, then the nurse can contact the other employees who signed

up for the shift and try to come to an agreement. After all (student) nurses completed their schedule, the planner finishes the schedule and checks it on the aforementioned agreements with the nurses. In the past, Westfriesgasthuis worked with a fixed staffing demand for the summer season and another fixed staffing demand for the winter season. Currently, they work with a minimum and maximum staffing demand for each shift, so the amount of nurses can vary between these bounds.

**Goal**

The goal of Westfriesgasthuis is to keep their employees satisfied and involved with their rosters. Self rostering provides extra opportunities for employees to indicate their wishes and preferences, which helps employees to balance their personal responsibilities with their work.

**Requirement:**

1. *Working hours act (WHA) and collective labor agreements (CLA)*
   The method should allow employees to propose schedules that contain violations concerning the working hours act and the collective labor agreement, but the method should not create new ones. Initially, Westfriesgasthuis aims for zero violations in the schedule. If there are violations in the preferred schedules of the employees and the planner approves these, the method does not have to try to fix these violations.

**Wishes:**

1. *Bonus system*
   Employees who want to work the unpopular shifts should receive some kind of bonus.

2. *Not-work wish*
   Employees should have the possibility to specify (part of) a weekday that they wish not to work, e.g., because they have a fixed evening to sport or they do not have a babysitter on Wednesday afternoon. This is only on weekdays, so not on Friday night, Saturday and Sunday. There should be a maximum number of not-work wishes, approximately one per week.

3. *Number of changes*
   The method should remain at least 80% of the preferred schedule of an employee.

4. *Planning forward in rotation*
   For health reasons, the method should plan shifts forward in rotation. Forward in rotation means that an employee begins with day shifts, then a certain amount of evening shifts, where after a certain amount of night shifts follows. So the method should not plan a night shift in between two day shifts.

5. *Work hours*
   Employees have to fulfill their contract hours per year, so the method should also do this. The work hours they have to fulfill per period are

more flexible, there is a certain minimum and maximum number of hours that an employee has to fulfill. The method should take this into account.

# Appendix B

# Point systems

This appendix first describes examples for the three point systems described in Chapter 3. Next, this appendix describes three variants on the three point systems.

## Point system 1

Tables B.1, B.2, and B.3 show an example of step 1, 2, and 3 of the self rostering process with point system 1. The example shows 3 employees in a planning period of 7 days. Table B.1 shows the staffing demand defined by the organization. Table B.2 shows the proposed schedules of the employees and the points they receive for each shift. We present these points per shift per day in Table B.3. In this example, understaffed shifts receive 5 points, matching shifts receive 4 points, and overstaffed shifts receive 1 point. The points per shift in Table B.3 are again used in Table B.2 to calculate the (total) points per employee.

In detail, in Table B.1 we see that the staffing demand for shift A on day 7 is 0. In Table B.3 we see that the actual assignment is 2 employees, so the shift is overstaffed and receives 1 point. In Table B.2 we see that De Vries and De Jong both receive 1 point for shift A on day 7 in his schedule.

## Point system 2

For point system 2, we defined the points as the difference between the number of employees who signed up for the shift (actual assignment) and the staffing demand. Negative points mean that more employees signed up for the shift than necessary.

We use the same example as for point system 1. The staffing demand (step 1) and the proposed schedules of the employees (step 2) stay the same. The points assigned to the shift and the employees are given in Table B.4 and Table B.5.

In detail, the staffing demand for shift A on day 6 is 0 (see Table B.1). The actual assignment is 1 employee (see Table B.5). Therefore, the points for shift A on day 6 are $0 - 1 = -1$. In Table B.4 we see that De Jong receives -1 points for shift A on day 6 in his schedule.

Table B.1: Staffing demand (Step 1)

| **Shift\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| B | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| C | 1 | 1 | 1 | 1 | 2 | 2 | 2 |

Table B.2: Chosen shifts and corresponding points (Steps 2 and 3) - Point system 1 (TS = Total number of shifts, TP = Total points, S = Score, and R = Ranking)

| **Employee\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **TS** | **TP** | **S** | **R** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Jansen | | | | | C | C | C | 3 | 15 | 5.00 | 1 |
| Points | | | | | 5 | 5 | 5 | | | | |
| De Vries | | | C | B | | B | A | 4 | 10 | 2.50 | 2 |
| Points | | | 4 | 1 | | 4 | 1 | | | | |
| De Jong | B | C | | | B | A | A | 5 | 11 | 2.20 | 3 |
| Points | 4 | 4 | | | 1 | 1 | 1 | | | | |

Table B.3: Actual assignment per shift per day and assigned points (Step 3) - Point system 1

| **Shift\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| Points | 4 | 4 | 4 | 5 | 5 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| Points | 4 | 5 | 5 | 1 | 1 | 4 | 5 |
| C | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Points | 5 | 4 | 4 | 5 | 5 | 5 | 5 |

Table B.4: Chosen shifts and corresponding points (Steps 2 and 3) - Point system 2 (TS = Total number of shifts, TP = Total points, S = Score, and R = Ranking)

| **Employee\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **TS** | **TP** | **S** | **R** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Jansen | | | | | C | C | C | 3 | 3 | 1.00 | 1 |
| Points | | | | | 1 | 1 | 1 | | | | |
| De Vries | | | C | B | | B | A | 4 | -3 | -1.33 | 3 |
| Points | | | 0 | -1 | | 0 | -2 | | | | |
| De Jong | B | C | | | B | A | A | 5 | -4 | -1.25 | 2 |
| Points | 0 | 0 | | | -1 | -1 | -2 | | | | |

Table B.5: Actual assignment per shift per day and assigned points (Step 3) - Point system 2

| **Shift\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| Points | 0 | 0 | 0 | 1 | 1 | -1 | -2 |
| B | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| Points | 0 | 1 | 1 | -1 | -1 | 0 | 1 |
| C | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Points | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

## Point system 3

Point system 3 uses the popularity of a shift to determine the points. Popularity is calculated in percentages: total amount of employees who signed up for the shift (actual assignment) divided by the staffing demand, see equation 3.1.

To illustrate this system, we again use the example for point system 1. The staffing demand is again the same (see Table B.1). To assign points to each shift, we define different classes of popularity and assign points to each class (see Table B.6). Table B.8 shows, according to the popularity, the points per shift. Next, we assign the points to the shifts of the employees, calculate their total points, and rank the employees, see Table B.7.

In detail, the staffing demand for shift B on day 1 is 1 employee (see Table B.1). The actual assignment is 1 employee (see Table B.8). The popularity of this shift is $\frac{1}{1} * 100\% = 100\%$. In Table B.6 we see that a popularity of 100% receives 1 point. In Table B.7 we see that De Jong receives 1 point for shift B on day 1 in his schedule.

## Variant 1

The first variant is the option to use the fixed points in step 3 and a fixed percentage of these fixed points in steps 4 and 5. For example, points in step 4 are 50% of the points used in step 3. Or the other way around, in step 4 points are 150% of the points used in step 3. This way, employees are more encouraged to change their schedule in step 4. This can be applied to all point systems. The advantage is that the organization has extra control on the two steps (step 3 and 4). The disadvantage is again the negative effect on the transparency of the system. According to customers of ORTEC, the values of shifts are hard to estimate for the employee within this system.

## Variant 2

This variant normalizes all points on shifts of 8 hours. This way, shorter shifts receive less points. With this variant employees are encouraged to choose long shifts first. Eventually all shifts have to be covered, but with the encouragement a larger time period of staffing shortage is fulfilled first. The disadvantage again turned out, in discussions between ORTEC and their customer, to be the negative effect on the transparency for the employees of the customer.

## Variant 3

In this variant the ranking is not determined by the average points per shift (the score), but the average points per worked hour. The total points are divided by the total worked hours.

Shorter shifts with the same points as longer shifts, contribute more to the average of the employees. This way, employees are encouraged to choose the shorter shifts first. It is up to the organization to see this as an advantage or disadvantage.

Table B.6: Popularity classes - Point system 3

| **Popularity** | **Points** |
|---|---|
| 0% - 49% | 5 |
| 50% - 99% | 4 |
| 100% - 100% | 3 |
| 101% - 149% | 2 |
| >150% | 1 |

Table B.7: Chosen shifts and corresponding points (Steps 2 and 3) - Point system 3 (TS = Total number of shifts, TP = Total points, S = Score, and R = Ranking)

| **Employee\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **TS** | **TP** | **S** | **R** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Jansen | | | | | C | C | C | 3 | 12 | 4.00 | 1 |
| Points | | | | | 4 | 4 | 4 | | | | |
| De Vries | | | C | B | | B | A | 4 | 8 | 2.00 | 2 |
| Points | | | 3 | 1 | | 3 | 1 | | | | |
| De Jong | B | C | | | B | A | A | 5 | 9 | 1.80 | 3 |
| Points | 3 | 3 | | | 1 | 1 | 1 | | | | |

Table B.8: Actual assignment per shift per day and assigned points (Step 3) - Point system 3

| **Shift\Day** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| Points | 3 | 3 | 3 | 5 | 5 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| Points | 3 | 5 | 5 | 1 | 1 | 3 | 5 |
| C | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| Points | 5 | 3 | 3 | 5 | 4 | 4 | 4 |

# Appendix C

# Working Hours Act

Source: Ministry of Social Affairs and Employment (2010)

| | | Standard |
|---|---|---|
| **Working time** | per shift | 12 hours |
| | per week | 60 hours |
| | per week in a 4-week period | 55 hours on average* |
| | per week in a 16-week period | 48 hours on average |
| **Rest times** | daily rest | 11 hours (consecutive)<br>(1x per week: 8 hours, if necessary because of nature of work or business circumstances) |
| | weekly rest | 36 hours (consecutive), or<br>72 hours per 14 days<br>(split into periods of at least 32 hours) |
| **Breaks** | in the event of > 5.5 hours per shift | 30 minutes (possibly 2 x 15 minutes) |
| | in the event of > 10 hours per shift | 45 minutes (possibly 3 x 15 minutes) |
| | in the event of > 5.5 hours per shift | 15 minutes* |

* By collective arrangement

| | | Standard |
|---|---|---|
| **Sunday rest** | Sunday work | no work on Sunday except if:<br><br>- in accordance with the type of work and stipulated or<br>- necessary in connection with the type of work or business circumstances<br>- agreed with works council (in absence of such: with employees involved)<br>- individual consent |
| | free Sundays | 13 (per 52-week period)<br>any other number*, provided:<br>- individual consent if fewer than 13 free Sundays per year |
| **Night work**<br>Night shift: $>$ 1 hour of work between midnight and 6 am | working time per shift | 10 hours<br><br>12 hours, provided:<br>- shift is followed by 12 hours of rest<br>- 5 x per 2 weeks<br>- maximum 22 x in 52-week period |
| | working time per week | 40 hours (per 16 weeks),<br>if $\leq$ 16 night shifts per 16 weeks |
| | rest time after night shifts<br>*applies for night shifts ending after 2 am* | 14 hours<br>(1 x per week: 8 hours, if necessary in connection with type of work or business circumstances) |
| | rest time after $\leq$ 3 night shifts | 46 hours |
| | maximum length of series<br>*applies if at least one of the shifts is a night shift* | 7 or 8* |
| | maximum number | - 36 night shifts per 16 weeks, or<br>- 140 night shifts per 52 weeks, or<br>- 38 hours between midnight and 6 am per 2 consecutive weeks |

| | | Standard |
|---|---|---|
| **On-call duty** | ban on on-call duty | - 14 days free of on-call duty per 4 weeks<br><br>- 2x2 days per 4 weeks no on-call duty and no work<br>- no on-call duty 11 hours before or 14 hours after a night shift |
| | working time per 24 hours | 13 hours |
| | working time per week in the event of on-call duty at night<br><br>*applies if on-call duty is assigned between the hours of midnight and 6 am 16 times or more in a 16-week period* | - average 40 hours (per 16 weeks), or<br>- average 45 hours (per 16 weeks), provided:<br>◇ there is an uninterrupted 8 hours of rest before starting the new shift (in the event of last call between midnight and 6 am), or<br>◇ 8 hours of uninterrupted rest in the 18 hours following 6 am (if the last call took place between midnight and 6 am and was immediately followed by a new shift). |

# Appendix D

# Combined MILP

In this appendix, we give a complete overview of the basic MILP (Section 4.3) combined with all described extensions (Section 4.4). This combined MILP is reducible to the basic MILP by setting $\lambda_2$ to $\lambda_4$ to zero. In that case $\tilde{w}$ is not in the objective and constraint (D.7) becomes redundant.

**Combined MILP**

*Parameters*

| | |
|---|---|
| $I$ | Set of employees |
| $J_i \subset J$ | Set of all swap options for employee $i \in I$ |
| $K$ | Set of unique shifts |
| $j = (j_1, j_2) \in K^2$ | Entering $j_1$ and removed $j_2$ shift in swap option $j$ |
| $v_k \in \mathbb{Z}$ | Difference vector |
| $IN_i^k = \{j = (j_1, j_2) \in J_i \mid j_1 = k\}$ | Set of all swap options of employee $i \in I$ where the entering shift is equal to a shift $k \in K$ |
| $OUT_i^k = \{j = (j_1, j_2) \in J_i \mid j_2 = k\}$ | Set of all swap options of employee $i \in I$ where the removed shift is equal to a shift $k \in K$ |
| $OVER = \{k \in K \mid v_k > 0\}$ | Set of all overstaffed shifts |
| $UNDER = \{k \in K \mid v_k < 0\}$ | Set of all understaffed shifts |
| $s_i \in \mathbb{R}$ | Score of employee $i \in I$ |
| $w_j \in \mathbb{R}$ | Score change of swap option $j \in J_i$ for employee $i \in I$ |

*Decision variables*

$$x_j = \begin{cases} 1 & \text{if swap option } j \in J_i \text{ of employee } i \in I \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

*Variables*

$$\text{Minimize} \quad \lambda_1 \sum_{k \in OVER} n_k + \lambda_2 \sum_{i \in I} s_i \sum_{j \in J_i} x_j + \lambda_3 \sum_{i \in I} s_i \sum_{j \in J_I} w_j x_j - \lambda_4 \tilde{w} \quad \text{(D.1)}$$

| | |
|---|---|
| $n_k \in \mathbb{Z}$ | New difference vector |
| $\tilde{w} \in \mathbb{R}$ | Lowest new score $(s_i + w_i)$ of all selected employees |

*Constraints*

$$\sum_{j \in J_i} x_j \leq 1 \qquad i \in I \qquad \text{(D.2)}$$

$$n_k = v_k + \sum_{i \in I} \left( \sum_{j \in IN_i^k} x_j - \sum_{j \in OUT_i^k} x_j \right) \qquad k \in K \qquad \text{(D.3)}$$

$$n_k \geq 0 \qquad k \in OVER \qquad \text{(D.4)}$$

$$n_k \leq 0 \qquad k \in UNDER \qquad \text{(D.5)}$$

$$x_j \in \{0, 1\} \qquad j \in J_i \qquad \text{(D.6)}$$

$$\tilde{w} \leq s_i + \sum_{j \in J_i} w_j x_j \qquad i \in I \qquad \text{(D.7)}$$

# Appendix E

# Full factorial design matrix

In Section 6.2 we discuss the results of the $2^k$ factorial design. Table E.1 shows the design matrix of the full factorial design we use. Table E.2 shows the selected levels per factor.

Table E.1: $2^5$-factorial design matrix

| **Design point** | **1** | **2** | **3** | **4** | **5** | **Design point** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | 17 | - | - | - | - | + |
| 2 | + | - | - | - | - | 18 | + | - | - | - | + |
| 3 | - | + | - | - | - | 19 | - | + | - | - | + |
| 4 | + | + | - | - | - | 20 | + | + | - | - | + |
| 5 | - | - | + | - | - | 21 | - | - | + | - | + |
| 6 | + | - | + | - | - | 22 | + | - | + | - | + |
| 7 | - | + | + | - | - | 23 | - | + | + | - | + |
| 8 | + | + | + | - | - | 24 | + | + | + | - | + |
| 9 | - | - | - | + | - | 25 | - | - | - | + | + |
| 10 | + | - | - | + | - | 26 | + | - | - | + | + |
| 11 | - | + | - | + | - | 27 | - | + | - | + | + |
| 12 | + | + | - | + | - | 28 | + | + | - | + | + |
| 13 | - | - | + | + | - | 29 | - | - | + | + | + |
| 14 | + | - | + | + | - | 30 | + | - | + | + | + |
| 15 | - | + | + | + | - | 31 | - | + | + | + | + |
| 16 | + | + | + | + | - | 32 | + | + | + | + | + |

Table E.2: Input parameters and settings

| **Input parameter** | **Settings** |
|---|---|
| Point system<br>Section 3.3 | 1. Point system 1<br>2. Point system 2<br>3. Point system 3 |
| MILP<br>Section 4.3 and 4.4 | 1. Basic MILP<br>2. Lowest Score<br>3. High score changes at low Scores: Multiplication<br>4. High score changes at low Scores: Lower bound |
| Swap strategy<br>Section 4.5 | 1. Only primary swaps<br>2. Only secondary swaps<br>3. First, use primary swaps until 95% of all understaffed shifts are fulfilled, then use secondary swaps until these no longer exist.<br>4. First, use primary swaps until these no longer exist, then use secondary swaps until there are new primary swaps. Use these new primary swaps until these no longer exist, then again use secondary swaps until new primary swaps arise etc.<br>5. First, use primary swaps until 80% of all understaffed shifts are fulfilled, then use secondary swaps until new primary swaps arise, then use primary swaps until these no longer exist.<br>6. Use primary and secondary swaps simultaneously.<br>7. First, use primary swaps until these no longer exist, then use primary and secondary swaps at the same time.<br>8. First, use primary swaps until these no longer exist, then use secondary swaps until these no longer exist. |
| Number of initial employees | 1. Begin with *all* employees.<br>2. Begin with 3 employees (with the lowest score), when there are not any swap options available, add 1 employee with the lowest score of the unselected employees.<br>3. Begin with 1 employee (the employee with the lowest score), when there are not any swap options available, add 1 employee with the lowest score of the unselected employees. |
| Minimum percentage | 1. Use a minimum of 0%.<br>2. Use a minimum of 70%.<br>3. Use a minimum of 80%. |

# Appendix F

# Analysis of effects per case

In this appendix, we shortly discuss each key performance indicator per case, except for the Pompestichting - Shuffle case, this is already discussed in Section 6.3. Per case we determine the whether the significant effects are noticeable.

**Westfriesgasthuis-1 - Shuffle**

In Figure F.1 we see a movement between every 8 large dots. Within each 16 large dots, the second 8 are higher than the first 8. This is the effect of factor 4 (initial employee), where starting with one employee has a positive effect over the average remaining percentage.

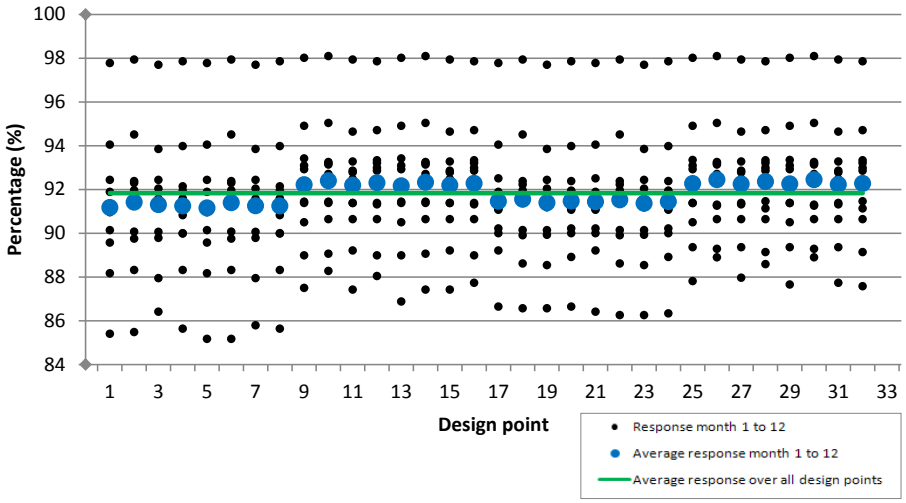



Figure F.1: Remaining percentage - Westfriesgasthuis-1 - Shuffle

In Figure F.2 we see a difference between the first group and second group of 16 large dots. The second group of large dots are (almost) all above the line that indicates the overall average, while the first group of dots are all below this line. This indicates an effect of 5 (minimum percentage).

From Figure F.3 we see the effect of 5 (minimum percentage). Within the 32 dots, we see a difference between the first and second group of 16 large dots.

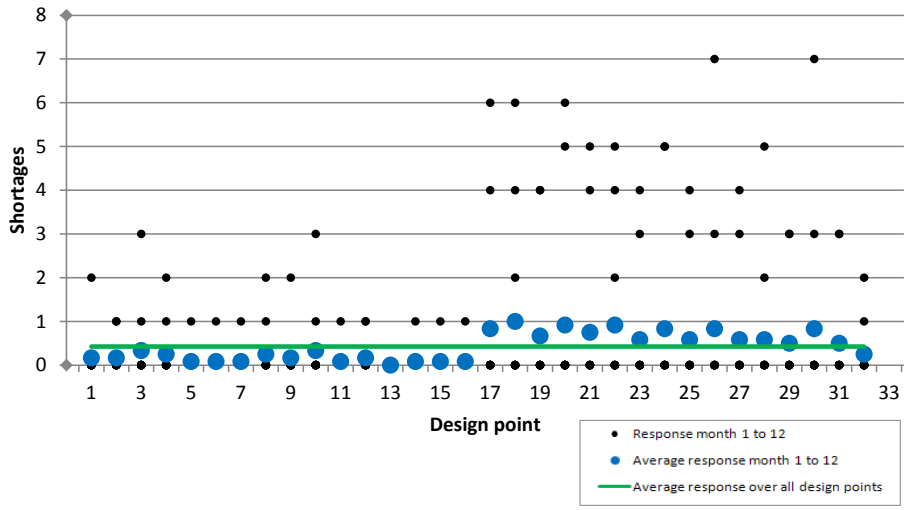



Figure F.2: Shortages - Westfriesgasthuis-1 - Shuffle

Next, we see interaction effect 45. The difference between the first and second group of 8 large dots within the first 16 dots is larger than in the second group of 16 large dots. As with the Pompestichting, the effect of factor 4 is mainly established through this interaction.

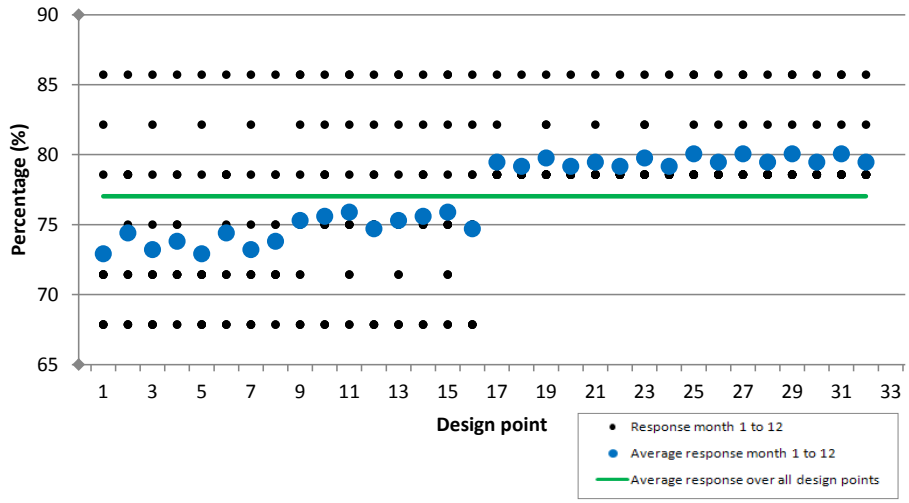



Figure F.3: Minimum remaining percentage - Westfriesgasthuis-1 - Shuffle

### Westfriesgasthuis-2 - Shuffle

In Figure F.4 we only see the effect of factor 4, a movement between the first and second group of 8 large dots within each group of 16 large dots. This movement is not very large, around 1%, but clearly present. This provides a higher remaining percentage when we start with only one employee (level 1)

instead of three employees (level -1).

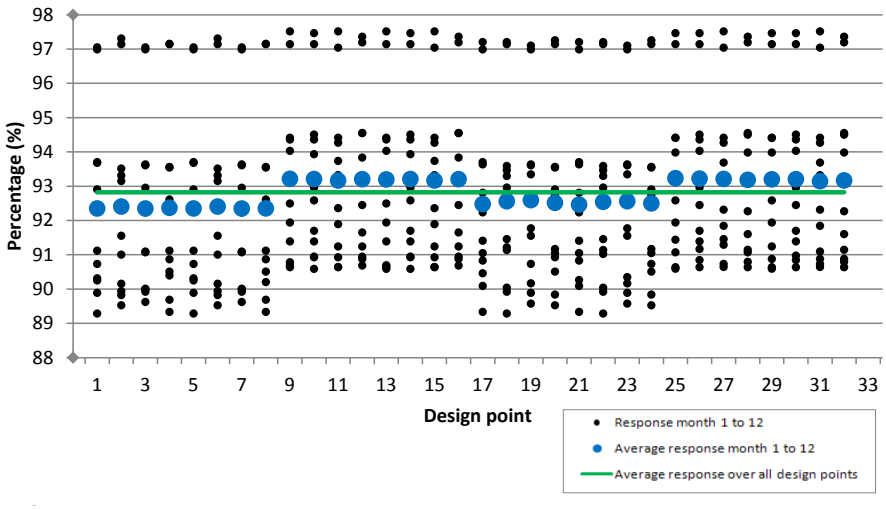



Figure F.4: Remaining percentage - Westfriesgasthuis-2 - Shuffle

In Figure F.5 we an effect of factor 5. This is the difference between the first and second group of 16 large dots. This movement is not very large, that is probably why we did not found this effect to be significant.

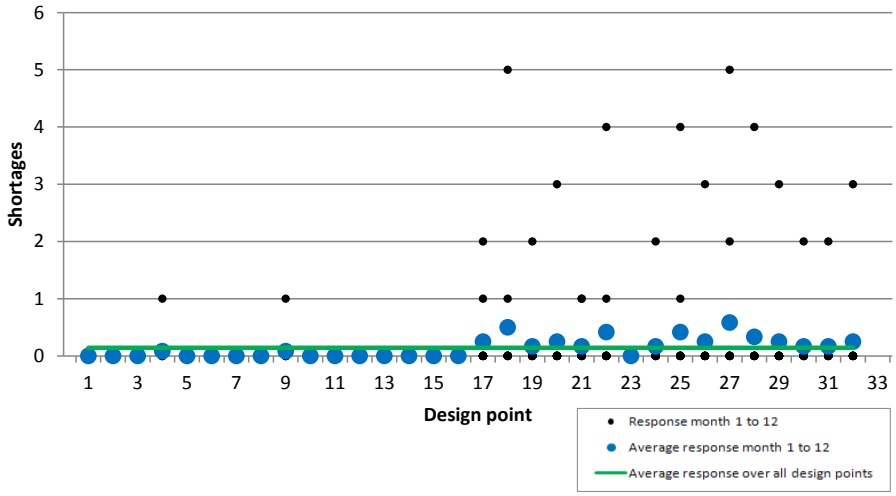



Figure F.5: Shortages - Westfriesgasthuis-2 - Shuffle

In Figure F.6 we clearly see a movement between the first and second group of 16 large dots: the effect of factor 5 (minimum percentage).

### Pompestichting - Disturbance

In Figure F.7 we see the effect of factor 3 (swap strategy). From each group of 8 large dots, the first 4 dots are above the line that indicates the overall average

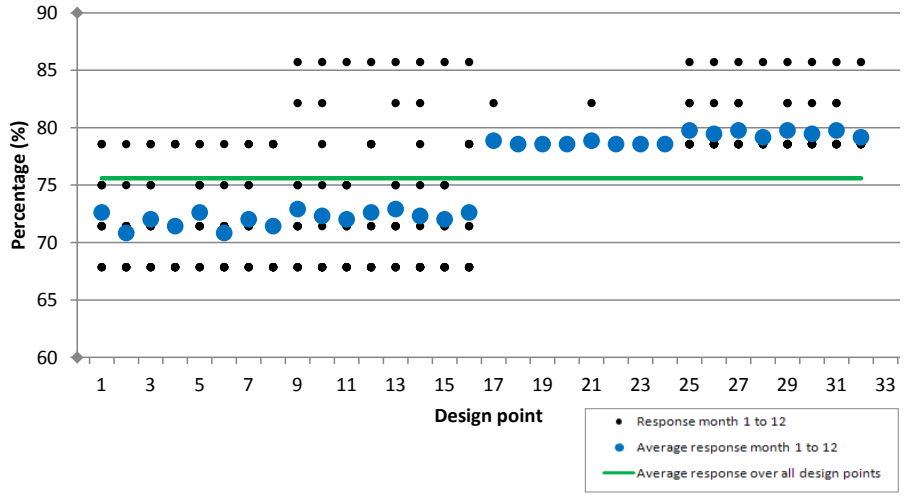



Figure F.6: Minimum remaining percentage - Westfriesgasthuis-2 - Shuffle

and the second group is below this line, which indicates the effect.

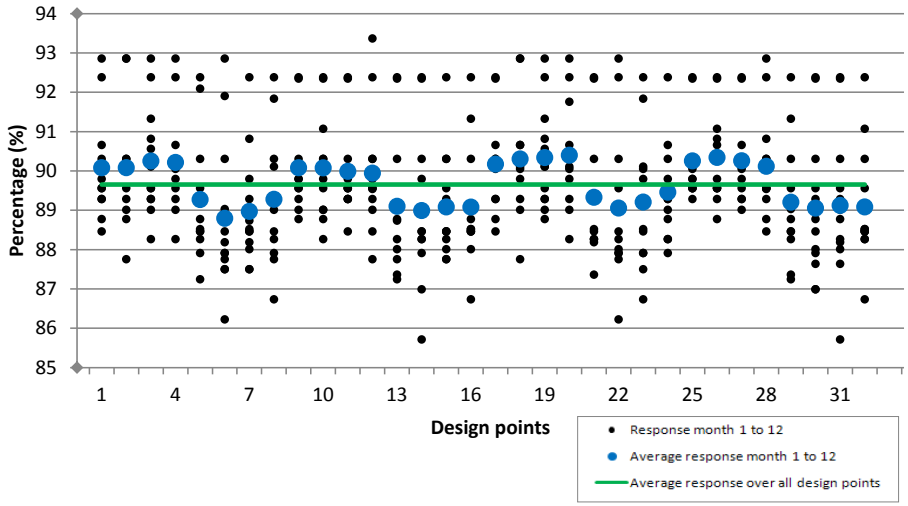



Figure F.7: Remaining percentage - Pompestichting - Disturbance

In Figure F.8 we again see a movement between every 4 large dots, which indicates the effect of factor 3 (swap strategy). Next, we also see the small effect of factor 5. The second group of 16 large dots is slightly higher than the first group of 16 large dots.

The effect of factor 5 is clearly present in Figure F.9. The first 16 large dots are below the line that indicates the overall average, while the second group of 16 dots is above this line.

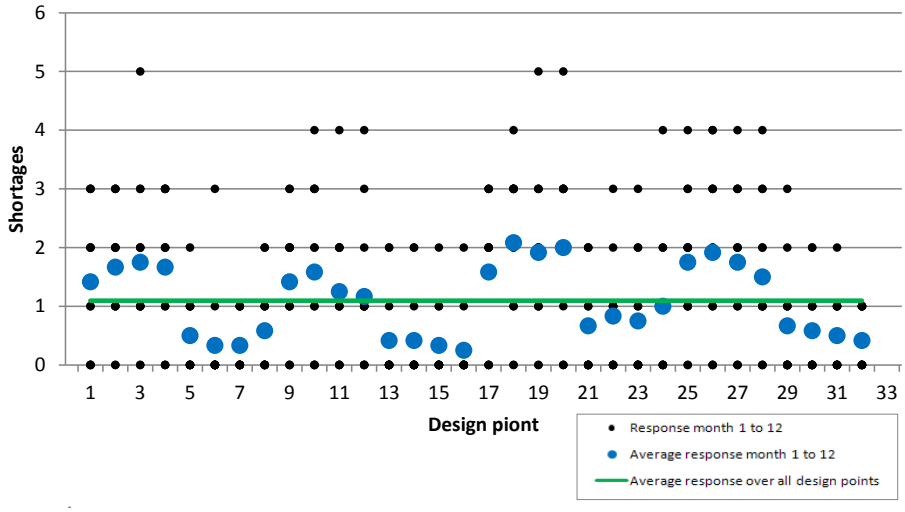



Figure F.8: Shortages - Pompestichting - Disturbance

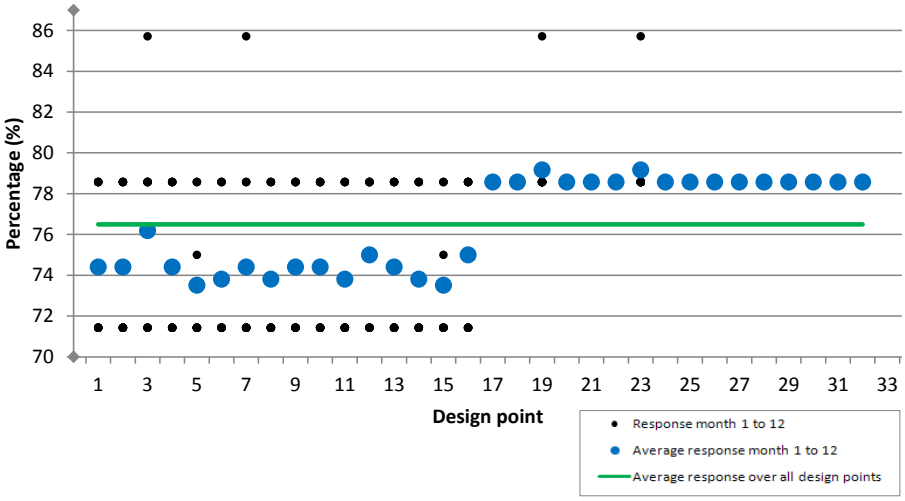



Figure F.9: Minimum remaining percentage - Pompestichting - Disturbance

### Westfriesgasthuis-1 - Disturbance

Although the difference is small, the effect of factor 3 (swap strategy) is clearly present in Figure F.10. This is indicated by the transformation every 4 design points.

In Figure F.11 we see that in each group of 8 large dots, the first 4 are above the line that indicates the overall average and the second 4 are below this line. This indicates the effect of factor 3.

In Figure F.12 we see that the first 16 large dots are mainly below the overall average, while the second group of 16 large dots are mainly above the overall average. This indicates the effect of factor 5 (minimum percentage).

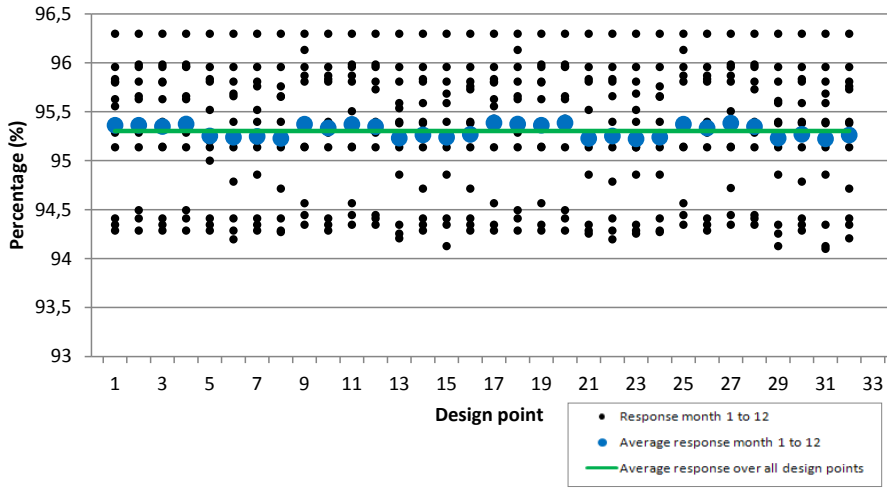



Figure F.10: Remaining percentage - Westfriesgasthuis-1 - Disturbance

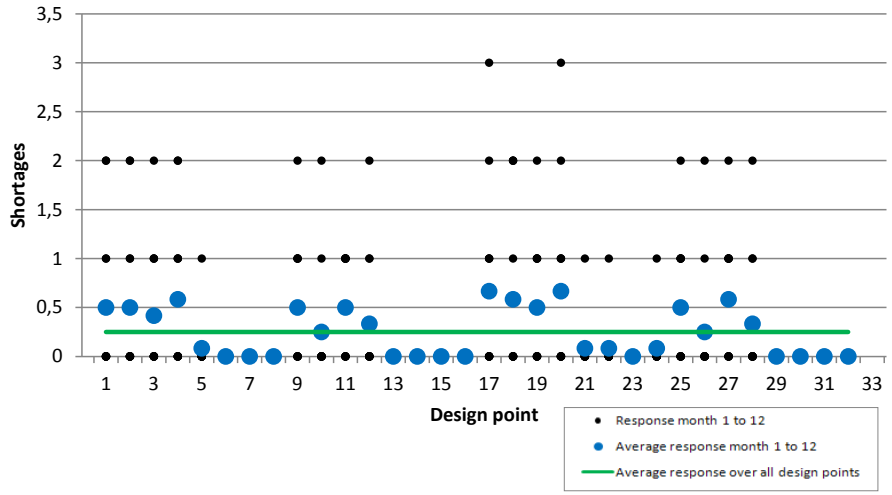



Figure F.11: Shortages - Westfriesgasthuis-1 - Disturbance

### Westfriesgasthuis-2 - Disturbance

Although the differences are small, the effect of factor 3 is present in Figure F.13. In every group of 8 large dots, the first 4 are at the upper side of the line, while the second 4 are just below the line.

In Figure F.14 we see that from each group of 8 large dots, the second group of 4 large dots are mainly lower than the first group. This indicates the effect of factor 3 (swap strategy).

In Figure F.15 we see that the first 16 large dots are mainly below the overall average and the last 16 large dots are mainly above this line, this indicates the effect of factor 5 (minimum percentage).

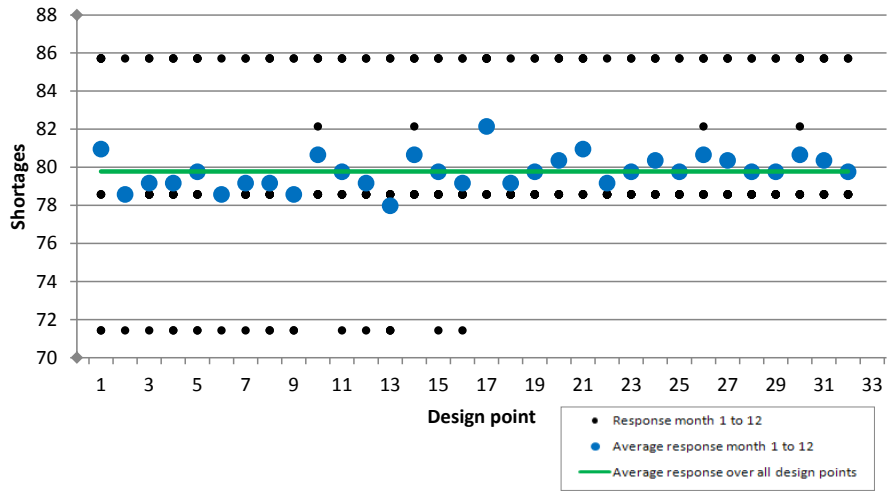



Figure F.12: Minimum remaining percentage - Westfriesgasthuis-1 - Disturbance

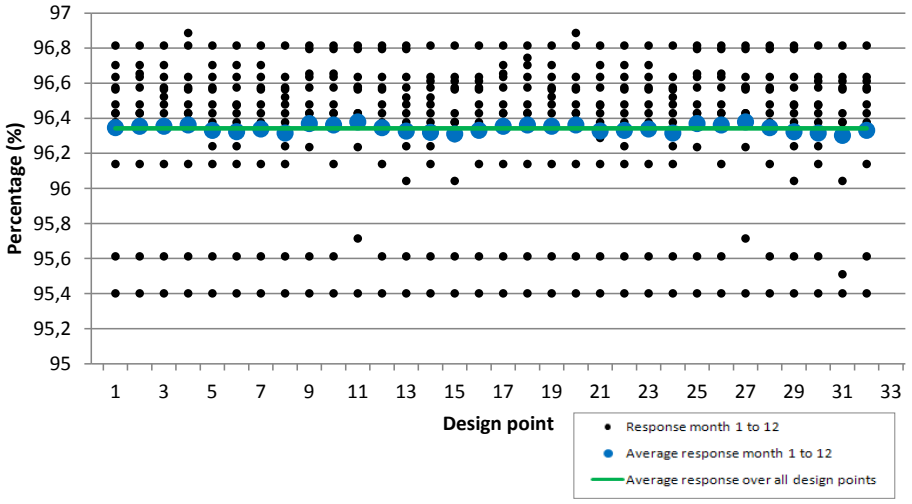



Figure F.13: Remaining percentage - Westfriesgasthuis-2 - Disturbance

### Organization X - Disturbance

In Figure F.16 we see a clear movement in each group of 8 large dots between the first 4 and second 4 dots. This indicates the effect of factor 3 (swap strategy).

The effect of factor 3 (swap strategy) is also clearly present in Figure F.17. The movement between the first and second group of 4 large dots in every group of 8 large dots indicates this effect.

In Figure F.18 we do not see any transformation, therefore, we do not expect any effect.

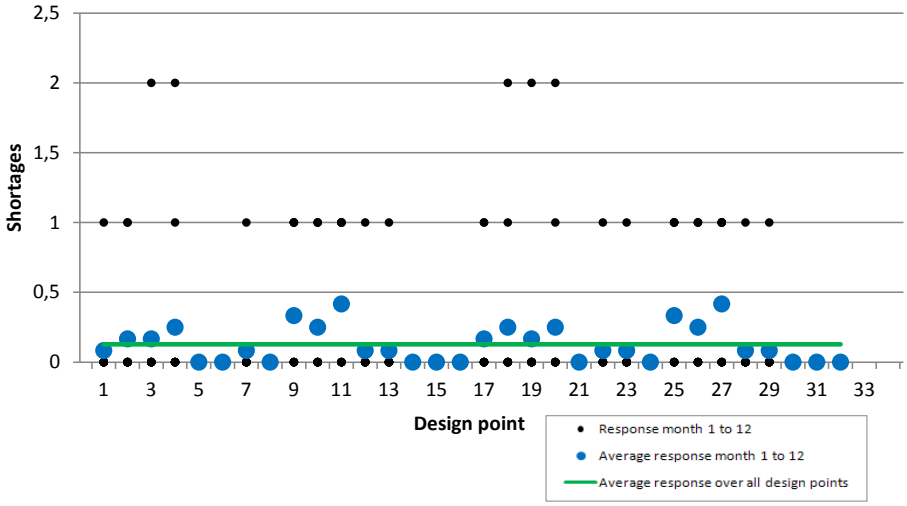



Figure F.14: Shortages - Westfriesgasthuis-2 - Disturbance

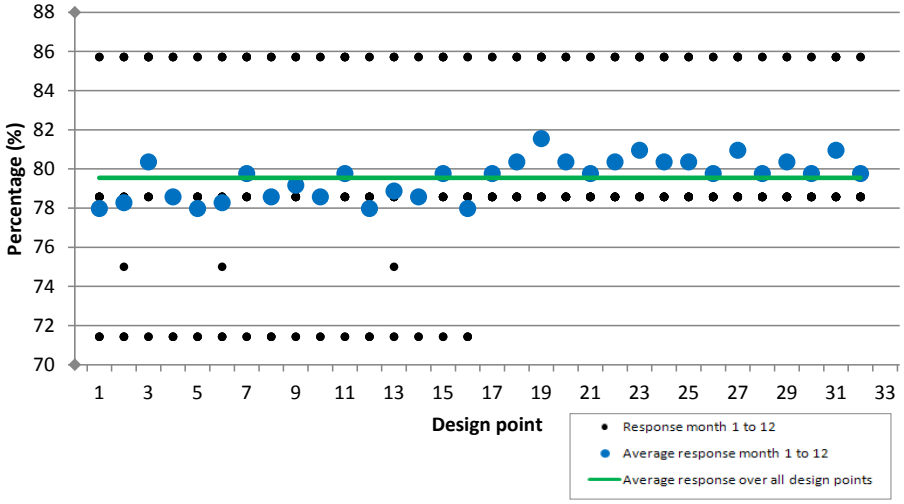



Figure F.15: Minimum remaining percentage - Westfriesgasthuis-2 - Disturbance

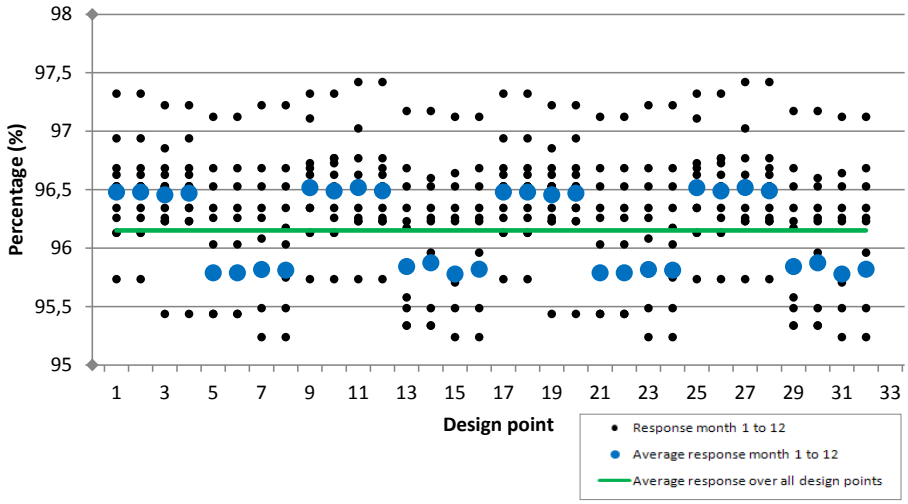



Figure F.16: Remaining percentage - Organization X - Disturbance

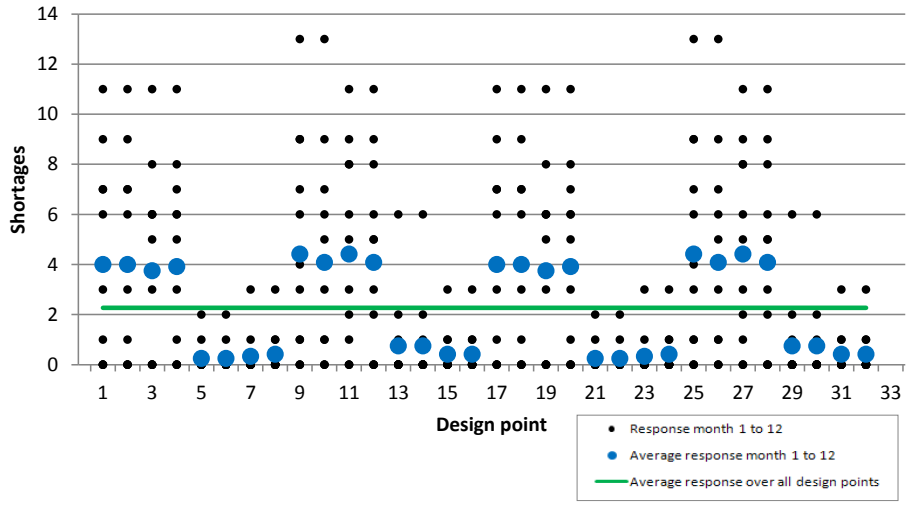



Figure F.17: Shortages - Organization X - Disturbance

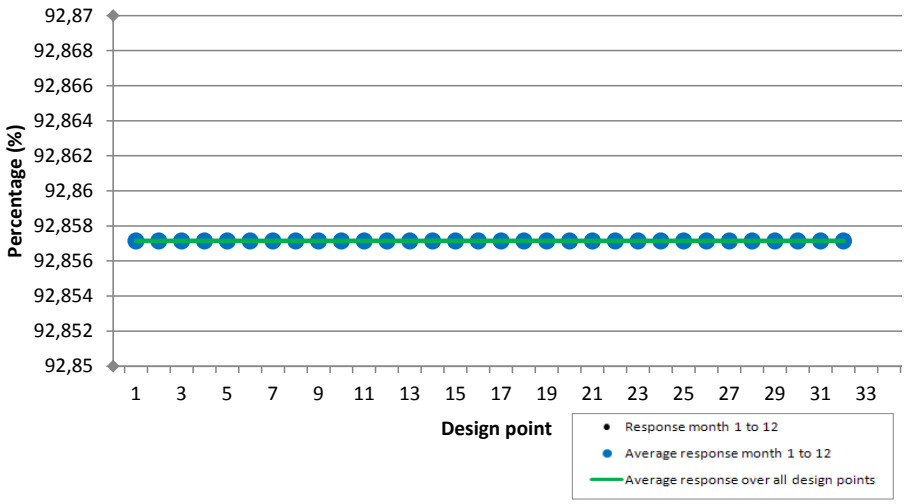



Figure F.18: Minimum remaining percentage - Organization X - Disturbance