Nurse-Rostering

Determination of a target for weekly working hours given different skills

B. van de Wiel

Technische Universiteit Delft

TU Delft

Nurse Rostering

Determination of a target for weekly working hours given different skills

by

B. van de Wiel

Student number: 4358597
Project duration: April 23, 2018 – July 5, 2019
Thesis committee: Dr. ir. J.T. van Essen, TU Delft, supervisor
Dr. J.L.A. Dubbeldam, TU Delft
Dr. B. van den Dries, TU Delft

TU Delft

Abstract

In this thesis we investigate an extension of the famous Nurse Rostering Problem. The Nurse Rostering Problem is the problem of finding an optimal assignment of nurses to shifts given some constraints. We extend the problem by considering that the nurses different sets of skills by which they can only perform certain tasks. We want the working hours of the nurses to be as close as possible to their contract hours, and moreover, we want the amount of work done on a specific task to be as close as possible to the demand for that task. It is important that the user of the model has control over the execution time.
We model this problem with an Integer Linear Program (ILP). The data we use to test the model is provided by ORTEC. ORTEC provided a random data generator which generates data such as contract hours and demands for certain skills, for a fictional company. The number of employees, skills and weeks can be determined by the user. We use small, medium and large data sets where large data sets consist of 100 employees, 52 weeks and 5 different skills. Our model is a linear approximation of a quadratic problem. It is based on minimizing the largest deviation in contract hours and demand.
To get control over the computation time of the model, a timer is implemented. The desired maximum computation time can be set by the user and the model provides the best solution found within that time boundary. The initial model is tested with three different solvers: COIN-OR, CPLEX and Gurobi. The computation times and consistency of the solvers are compared. We found that CPLEX was in our case the best solver with an average computation time of 7.2 seconds for data sets of 100 employees, 5 skills, and 52 weeks. Next, we use an iterative procedure, which runs the model several times in order to obtain a more accurate solution. In this procedure, certain deviations, and their contributing variables, are fixed in every iteration while the model runs on the remaining non-fixed variables. An extension to this iterative procedure is also presented. In this extension, multiple deviations are fixed in every iteration to decrease the execution time. The fixed percentage of maximum deviations per iteration is tested on accuracy and computation time. We found that for small data sets it is best to use 1%, which results in one fixed deviation per iteration. For medium data sets, the best range is from 1% to 10%. Lastly for large data sets we recommend to use a range of 10% to 20%.

Preface

This thesis is written on behalf of obtaining the Bachelors degree in Applied Mathematics at Delft University of Technology. In collaboration with and under guidance of E. van Veen, working for the company ORTEC, this project is done at the optimization department under supervision of J. T. van Essen. The project concerns an extension of the Nurse Rostering Problem. It consists of formulating an integer linear programming (ILP) model and constructing two algorithms for improving the model. These algorithms are compared on different performances such as accuracy and computation time. For more information on the used data and the individual result of each data set, there is the possibility to contact me.

I would like to thank Theresia van Essen for the support she gave me during this project. In rough times, Theresia was there with the right information and motivation to make sure I stayed on track. Also, I would like to thank Egbert van der Veen for providing this assignment, the knowledge needed for the assignment, and for the invitations to ORTEC for several meetings. Lastly, I would like to thank Johan Dubbeldam and Bart van den Dries for joining the presentation and being part of my thesis committee.

B. van de Wiel
Delft, June 2019

Contents

1 Introduction

For large companies with many employees, it can be difficult to manage a working schedule. In former times, this was all done by hand which made rostering a very slow process that was prone to mistakes and inefficiencies. Over the years, schedulers have started building software that can optimize a roster or help companies make decisions on how to schedule their workers. Especially health-care institutions, hospitals for example, are becoming more and more interested in this kind of software to support them in the rostering process. Due to the fact that the labour demand in hospitals can differ a lot over the year, this kind of problem is called the 'Nurse-rostering problem' (NRP) or 'Nurse-scheduling problem' (NSP).

A general description of the problem is as follows. There is a set of nurses and a set of time intervals the nurses can be assigned to. The goal is to assign nurses to specific time intervals in such a way that it suffices a set of constraints given by the scheduler. These constraints can be contract hours, laws or simply personal preferences of the nurses. The nurse-rostering problem has been discussed and examined for a very long time and although it seems like an easy problem, in practice, it is very difficult to obtain the optimal solution.

This thesis presents the investigation into extending an optimization software suitable for nurse-rostering problems. In addition to the nurses being assigned to different time intervals, the fact that different nurses have different skills, and therefore, can perform different tasks, is also considered. Moreover, we are using annualized hours to obtain more flexibility and to cope with fluctuating demand.

The thesis can be split into the following sections. The specific nurse-rostering problem that is dealt with is described is Chapter 3. In this chapter, we define all the variables and parameters necessary for this problem. Chapter 4 discusses the construction of the model that is used to approximate this problem. In Chapter 5, we introduce two iterative algorithms to provide more accurate solutions. In Chapter 6, the results of the different approaches are discussed. Finally we draw conclusions from this data and determine the success of this algorithm in Chapter 7. Some possible future extensions are also presented.

2

Literature Review

In this thesis, we investigate a possible extension of the classic Nurse Rostering Problem. The basic Nurse Rostering Problem consists of assigning nurses to shifts in an optimal way while satisfying certain constraints. We want to incorporate the ability to have a multi skilled workforce. Different nurses can only perform certain tasks and the difference between the actual working hours and the demand of these tasks has to be minimal. Moreover, the user of the model has to be able to control the calculation time. A lot of research has already been done on this topic. In this section, we are going to review literature concerning Nurse Rostering Problems.

Our goal is to determinate a target for weekly working hours, while taking contract hours of nurses and demand for certain tasks into account. To regulate working hours, three different methods are commonly used. These are measuring the overtime, annualized hours and Working Time Accounts (WTAs). In a contract with overtime, employees get paid extra when they work more than their weekly contract specifies. When annualized hours are used, the total working hours is measured over a longer period of time, allowing the employee to be more flexible and work a different number of hours per week. The idea behind working time accounts is that an employee is able to work longer or shorter hours than the contract specifies, over a certain period of time, and thereby, collect credits or debits in an individual working time account. These can later be compensated for by additional free time or work. An example of overtime can be found in Hasan et al. [11]. The overtime in working hours is seen as extra costs for the company. An example of WTAs can be found in Corominas et al. [4]. Annualized hours is widely discussed in van der Veen et al. [12] and in Keim [10]. Annualized hours is also what we consider in this thesis. The sum of the working hours of an employee have to be equal to the sum of the contract hours over that same period. We will also let the deviation from the contract hours be bounded in order to avoid extreme values.

In most annualized hours literature models, time intervals of one day or one week are considered with total periods of one year. Also, only one type of contract is considered. Multiple contract types are considered by [9] and [12], who distinguish between full-time employees, part-time employees and subcontractors. We will have random contract hours per employee, depending on what the data generator provides.

As already mentioned, in in Hasan et al. [11], overtime is seen as extra costs, and their goal is to minimize the costs. I other words, their goal is to minimize a specific objective function, which calculates the costs for the company. Different kinds of objective functions can be used for the same purpose. For example, Corominas et al. [3–5], van der Veen et al. [12] and Hasan et al. [11] all use an objective function that minimizes the cost. Keim [10] does not include any costs but only considers the overtime. Corominas [3] even uses a combination of cost and overtime in the objective function. While using different types of objective functions, the ultimate goal is the same: minimizing the over and under staffing and/or meeting certain demands. In this thesis, we are not including any costs. Moreover, in contrast to van der Veen et al. [12] and Hasan et al. [11] where constraints are added to meet

certain demands, we do not include these constraints. However, we do want the demand to be met as close as possible. Our objective function consists of deviations in contract hours and deviations in demand.

The most important feature is that the nurses can be assigned to different skills. This is also done in Attia et al. [1] and van der Veen et al. [12]. Certain constraints guarantee that an employee can only be scheduled for tasks if he is qualified to perform that task.

To make sure that the user of the model can control the computation time, we will test different solvers and develop a new algorithm. Their computation times will be compared. The computational complexity of the algorithms that we use is not a subject that is covered by this thesis. More information about this subject can be found in Keim [10] and J. van Leeuwen [13].

To make sure that the user of the model can control the computation time, we will test different solvers and develop a new algorithm. Their computation times will be compared.

The contribution of this thesis is that, to our knowledge, literature does not consider modeling annualized hours in combination with multi-skilled nurses without a fixed demand. While minimizing the difference between working hours and contract hours per week and per planning period, we also minimize the deviation in demand per skill. A new algorithm is developed to approximate a solution. The user of the model will have the ability to control the computation time using an implemented timer.

3

Problem Description

The problem that is studied in this thesis is a variation on the classical Nurse-rostering problem. We are using annualized hours and a given set of employees each possessing different skills. This allows every employee to perform various tasks. The objective of the selected problem is to select a set of number of working hours, per employee, per task, per week, in such a way that the under/overtime, compared to the employees weekly contract hours and the weekly demand per skill, is evenly distributed over the year. This has to be accomplished by creating a mathematical model which gives the user control over the calculation time.

The number of employees, the different skills and the time period in weeks can be determined by the user of the model who we call the employer. The work demand is given in hours per skill per week. Each employee has a contract which includes a number of hours they have to work per week. Since we are using annualized hours, it is possible that an employee works more or less than specified in the contract. How much under/overtime an employee is allowed to work per week is bounded and these bounds are also determined by the employer. It is possible that a given employee is not present in a specific week due to holidays or other circumstances. These weeks do not contribute to the annual working hours of an employee. The annual working hours are hence determined by an employees weekly contract hours times the number of present weeks.

Note that we are only looking for a weekly number of working hours. How these hours are divided over the days of the week is not to be determined by the model but by the employer. The model also does not take the salary of the employees into account. Determining or reducing costs is not an objective of this problem.

3.1. Sets, Parameters and Variables

In order to develop a model that can solve the problem described above it is necessary to divide all aspects of the problem into three categories: sets, parameters and variables. By doing so, we obtain a clear overview of the different aspects concerning this problem. A set is a collection of different objects. For example, $\{1, 2, 3, 4\}$ is the set of natural numbers less than 5 and $\{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday\}$ is the set of days in a week. Parameters are quantities that determine the output of a mathematical object and are determined by the employer. In the equation $y = x \cdot b$, we can determine the value of parameter b and thereby, influence the value of the outcome y. Variables are quantities which are unknown and are to be determined by the model we are going to create. In this problem, the following sets, parameters and variables are significant:

Sets

I Set of employees
J Set of all different tasks
T Set of weeks in a year

Parameters

d_{jt} Demand for task $j \in J$ in week $t \in T$
c_i Weekly contract hours of employee $i \in I$
l_i, u_i Min/Max hours employee $i \in I$ is allowed to work in one week

$$p_{it} = \begin{cases} 1, \text{ if employee } i \in I \text{ is present in week } t \in T \\ 0, \text{ if employee } i \in I \text{ is absent in week } t \in T \end{cases}$$

$$s_{ij} = \begin{cases} 1, \text{ if employee } i \in I \text{ can execute task } j \in J \\ 0, \text{ if employee } i \in I \text{ cannot execute task } j \in J \end{cases}$$

Variables

X_{ijt} Number of hours employee $i \in I$ works on task $j \in J$ in week $t \in T$

4

Model Construction

This section encompasses our modeling of the problem given in the previous section. In Section 4.1 we construct the initial quadratic model that we are trying to solve. Section 4.2 discusses the model type that is used to approximate the initial quadratic model and why it is chosen as our modeling technique. In Section 4.3 we explain the linearization process of the presented model. In Section 4.4 we introduce an extension to the model and finally, in Section 4.5, we present the final model.

4.1. Initial Quadratic Model

For this first model, we need a function that is desired to be minimized. This function is called the objective function. We want a solution in which the working hours are as close as possible to the contract hours of the employees and also as close as possible to the demand. This can be done by using the least squares method which provides the best 'fit' for the set of variables. We look at the difference between the actual working hours of an employee and his contract hours ($\sum_{j\in J} X_{ijt} - c_i$), and square this difference ($(\sum_{j\in J} X_{ijt} - c_i)^2$). If we do this for each employee in each week and we minimize the sum of all these squared numbers, we have found an assignment for working hours which has the least deviation from the contract hours. This specific assignment has the best 'fit'. If we do the same for the demand and combine these two components into one function, we obtain the following objective function:

$$\min \quad z = \lambda_1 \cdot \sum_{i\in I}\sum_{t\in T}\left(\sum_{j\in J} X_{ijt} - c_i\right)^2 + \lambda_2 \cdot \sum_{t\in T}\sum_{j\in J}\left(\sum_{i\in I} X_{ijt} - d_{jt}\right)^2 \tag{4.1}$$

The left component makes sure that the deviation in working hours and contract hours per employee is evenly distributed over the year. The right component makes sure that the deviation in supply and demand is evenly distributed over the year.
The importance of the two components may differ depending on the intention of the employer. To include this in our model, we have introduced two parameters λ_1 and λ_2 which are weights that indicate this importance. The values of λ_1 and λ_2 are also determined by the employer. We now have our objective function for this quadratic model complete.

Now, we need to add constraints to this problem. Firstly, constraint (4.3) implies that the sum of the total working hours of an employee is equal to the total contract hours of this employee over the whole period, in other words, the annualized hours. Secondly, constraint (4.4) implies that the number of working hours of an employee in a week stays within the boundaries given by the employer. Moreover, if an employee is absent in a certain week, no working hours can be assigned to this specific employee for this week. Constraint (4.5) ensures that no employees are assigned to tasks they can not perform. Lastly, constraint (4.6) ensures that the number of working hours can only be an integer. We now have our first model complete:

$$\min \quad z = \lambda_1 \cdot \sum_{i \in I} \sum_{t \in T} \left(\sum_{j \in J} X_{ijt} - c_i \cdot p_{it} \right)^2 + \lambda_2 \cdot \sum_{t \in T} \sum_{j \in J} \left(\sum_{i \in I} X_{ijt} - d_{jt} \right)^2 \tag{4.2}$$

$$\sum_{j \in J} \sum_{t \in T} X_{ijt} = \sum_{t \in T} c_i \cdot p_{it}, \qquad \forall i \in I, \tag{4.3}$$

$$l_i \cdot p_{it} \leq \sum_{j \in J} X_{ijt} \leq u_i \cdot p_{it}, \qquad \forall i \in I, \quad \forall t \in T, \tag{4.4}$$

$$X_{ijt} \leq u_i \cdot s_{ij}, \qquad \forall i \in I, \quad \forall j \in J, \quad \forall t \in T, \tag{4.5}$$

$$X_{ijt} \in \mathbb{N}, \qquad \forall i \in I, \quad \forall j \in J, \quad \forall t \in T. \tag{4.6}$$

4.2. Model Type

The model presented in the previous section is quadratic. Quadratic models can take a very long time to solve when combined with integer variables. Since our objective is to take control of the calculation time, we need a different approach. We are going to use approximate the quadratic model with an ILP model. ILP stands for Integer Linear Programming and is a special case of mathematical optimization. It consists of a linear objective function, subject to linear equality and or inequality constraints. Also, the variables contained in this problem can only attain the value of integers. It can be expressed in the following form:

$$\begin{aligned} \max \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{N}^n \end{aligned}$$

We have parameters A,b and c where A is a $m \times n$-matrix, b a m-dimensional vector and c an n-dimensional vector. Moreover, $c^T x$ is the objective function and $Ax \leq b$ are all the linear constraints. The variables are represented by the n-dimensional vector x.

The reason we choose to use this type of model is that ILP problems are generally much easier to solve. They take much less time. It is also preferable to let the working hours only be integers. This to prevent the optimal solution from containing non-integer working hours. For example, it is unnecessarily difficult to assign 36.78 working hours to a person for a week. In the next section, we convert the quadratic model to a linear model.

4.3. Linearizing the Quadratic Objective Function

We want to work with an ILP problem, therefore, we have to linearize the objective function. An important note here is that after linearizing, the optimal solution we find is not the exact optimal solution but an approximation.

We first consider the first component. This part makes sure that the under/overtime per employee is evenly distributed over the year. It uses the least-squares method to determine the best fit. Another way to make sure that the under/overtime is distributed evenly, is to find the specific person and week which account for the most under/overtime, and minimize this time.The under- or overtime of employee i in week t is given by equation 4.7.

$$\sum_{j \in J} X_{ijt} - c_i \cdot p_{it}. \tag{4.7}$$

Since we are considering the difference between the contract hours and the actual working hours, it does not matter if we have under- or overtime. From now on, we refer to under/overtime simply as deviation. We consider the absolute value of the deviation and obtain

equation 4.8.

$$\left| \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} \right| \tag{4.8}$$

Because every employee has a deviation of the contract hours in every week (this can be 0) there has to be a specific employee in a specific week who has the greatest deviation. In other words, there has to be an $i \in I$ and a $t \in T$ for which equation 4.8 has the largest value. Equation 4.9 expresses this maximum deviation.

$$\max_{\substack{t \in T \\ i \in I}} \left(\left| \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} \right| \right) \tag{4.9}$$

If we minimize this maximum deviation, the deviation of every employee has to be contained in a smaller and smaller interval, and therefore, the deviation gets more evenly distributed since there are no more outliers. If we apply this same principle to the second component of the objective function, we get a new objective function with the same goal but without squares in the equation.

$$\min \quad \lambda_1 \cdot \max_{\substack{t \in T \\ i \in I}} \left| \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} \right| + \lambda_2 \cdot \max_{\substack{t \in T \\ j \in J}} \left| \sum_{i \in I} X_{ijt} - d_{jt} \right| \tag{4.10}$$

Note that we changed the z from objective function (4.1) to z_1 since these are different values. z_1 is an approximation of z. We solved the problem of the squares but there are still absolute values and maximum values in the equation. Since these operations are not linear, we want to get those out as well. This can be done as follows. We can change the notation of the maxima to

$$\begin{aligned} \min \quad & \lambda_1 \cdot \max_{\substack{t \in T \\ i \in I}} \left(\max \left\{ \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} \quad , \quad c_i \cdot p_{it} - \sum_{j \in J} X_{ijt} \right\} \right) \quad + \\ & \lambda_2 \cdot \max_{\substack{t \in T \\ j \in J}} \left(\max \left\{ \sum_{i \in I} X_{ijt} - d_{jt} \quad , \quad d_{jt} - \sum_{i \in I} X_{ijt} \right\} \right) \end{aligned} \tag{4.11}$$

Note that the constraints we introduced in Section 4.1 are still valid. To make this objective function linear, we are going to add a few more variables and constraints. Since we want to minimize the maximum of $\sum_{j \in J} X_{ijt} - c_i \cdot p_{it}$ and $c_i \cdot p_{it} - \sum_{j \in J} X_{ijt}$, and $\sum_{i \in I} X_{ijt} - d_{jt}$ and $d_{jt} - \sum_{i \in I} X_{ijt}$, it also suffices to minimize a value equal to or greater than these values. The two new variables we are going to define are v and w and we put those under the following constraints:

$$v \geq \sum_{j \in J} X_{ijt} - c_i \cdot p_{it}, \qquad \forall i \in I, \qquad \forall t \in T, \tag{4.12}$$

$$v \geq c_i \cdot p_{it} - \sum_{j \in J} X_{ijt}, \qquad \forall i \in I, \qquad \forall t \in T, \tag{4.13}$$

$$w \geq \sum_{i \in I} X_{ijt} - d_{jt}, \qquad \forall j \in J, \qquad \forall t \in T, \tag{4.14}$$

$$w \geq d_{jt} - \sum_{i \in I} X_{ijt}, \qquad \forall j \in J, \qquad \forall t \in T. \tag{4.15}$$

By adding these variables and constraints we can write a new, simpler and linear objective function:

$$\min \quad \lambda_1 \cdot v + \lambda_2 \cdot w. \tag{4.16}$$

4.4. Extending the Objective Function

We have set up a model which minimizes two components. The first one is the largest deviation from the contract hours per skill and week over all employees. The second one is the largest deviation from the demand per skill and week. However, the current model does not consider any smaller deviations. Lets say for example that we have found an optimal solution in which the deviation of the contract hours of employee 1, 2, 3, and 4 in a certain week are 10, 6, 9, and 5 hours respectively. The largest deviation in this case is 10. But now suppose that there is another solution in which employees 1, 2, 3 and 4 have a deviation of respectively 10, 2, 3, and 0 hours. Then, the largest deviation is still 10 and this means that our model considers these two solutions to be equally good. But when assign these values to the original objective function (4.1), we see that the second solution gives a much lower value ($10^2 + 6^2 + 9^2 + 5^2 = 242$ and $10^2 + 2^2 + 3^2 + 0^2 = 113$). We clearly prefer the second solution over the first one.

We want to extend our model in such a way that it considers the second solution, with lower total deviation, as a better solution to the problem. The deviation of the contract hours per employee per week is given by $|\sum_{j\in J} X_{ijt} - c_i \cdot p_{it}|$. We want that the total deviation of all employees and over all weeks is minimized. Therefore, we add a new component to our objective function:

$$\min \quad \lambda_1 \cdot v + \lambda_2 \cdot w + \lambda_3 \cdot \sum_{i\in I}\sum_{t\in T}\left|\sum_{j\in J} X_{ijt} - c_i \cdot p_{it}\right|. \tag{4.17}$$

A new λ is introduced to reflect the importance of this component. The values of all λ's are to be set by the user. Since the first two components of the objective function are of more importance, it is logical to assign a lower value to λ_3 than to λ_1 and λ_2.

To get rid of the absolute value signs in our new objective function, we add new variables q_{it} (for $i \in I$ and $t \in T$) to our problem. These variables, q_{it}, are the deviations with the contract hours per person per week and it applies the same principle as variable v and w to eliminate the absolute value signs. We substitute the total overtime value $\sum_{i\in I}\sum_{t\in T} q_{it}$ in the objective function to obtain:

$$\min \quad \lambda_1 \cdot v + \lambda_2 \cdot w + \lambda_3 \cdot \sum_{i\in I}\sum_{t\in T} q_{it} \tag{4.18}$$

We also add the following constraints to the model:

$$q_{it} \geq \sum_{j\in J} X_{ijt} - c_i \cdot p_{it}, \qquad \forall i \in I, \qquad \forall t \in T, \tag{4.19}$$

$$q_{it} \geq c_i \cdot p_{it} - \sum_{j\in J} X_{ijt}, \qquad \forall i \in I, \qquad \forall t \in T. \tag{4.20}$$

(4.21)

To achieve an even better solution, we use the same method to minimize the total deviation in demand. The deviation in demand per skill per week is given by $|\sum_{i\in I} X_{ijt} - d_{jt}|$. The total deviation, given by $\sum_{j\in J}\sum_{t\in T} r_{jt}$, is added to the objective function with new variable r_{jt} (deviation in demand of skill $j \in J$ in week $t \in T$) to get rid of the absolute values. The new objective function becomes:

$$\min \quad \lambda_1 \cdot v + \lambda_2 \cdot w + \lambda_3 \cdot \sum_{i\in I}\sum_{t\in T} q_{it} + \lambda_4 \cdot \sum_{j\in J}\sum_{t\in T} r_{jt}. \tag{4.22}$$

The following constraints are added to the model:

$$r_{jt} \geq \sum_{i \in I} X_{ijt} - d_{jt}, \qquad \forall j \in J, \quad \forall t \in T, \tag{4.23}$$

$$r_{jt} \geq djt - \sum_{i \in I} X_{ijt}, \qquad \forall j \in J, \quad \forall t \in T. \tag{4.24}$$

4.5. Final Model

If we combine combine equation 4.22 with all previous constraints, we get the following final model:

$$\min \quad z_1 = \lambda_1 \cdot v + \lambda_2 \cdot w + \lambda_3 \cdot \sum_{i \in I} \sum_{t \in T} q_{it} + \lambda_4 \cdot \sum_{j \in J} \sum_{t \in T} r_{jt}$$

$$\begin{aligned}
s.t. \quad & \sum_{j \in J} \sum_{t \in T} X_{ijt} = \sum_{t \in T} c_i p_{it} && \forall i \in I \\
& l_i \cdot p_{it} \leq \sum_{j \in J} X_{ijt} \leq u_i \cdot p_{it} && \forall i \in I, \forall t \in T \\
& X_{ijt} \leq u_i \cdot s_{ij} && \forall i \in I, \forall j \in J, \forall t \in T \\
& v \geq \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} && \forall i \in I, \forall t \in T \\
& v \geq c_i \cdot p_{it} - \sum_{j \in J} X_{ijt} && \forall i \in I, \forall t \in T \\
& w \geq \sum_{i \in I} X_{ijt} - d_{jt} && \forall j \in J, \forall t \in T \\
& w \geq d_{jt} - \sum_{i \in I} X_{ijt} && \forall j \in J, \forall t \in T \\
& q_{it} \geq \sum_{j \in J} X_{ijt} - c_i \cdot p_{it} && \forall i \in I, \forall t \in T \\
& q_{it} \geq c_i \cdot p_{it} - \sum_{j \in J} X_{ijt} && \forall i \in I, \forall t \in T \\
& r_{jt} \geq \sum_{i \in I} X_{ijt} - d_{jt} && \forall j \in J, \forall t \in T \\
& r_{jt} \geq d_{jt} - \sum_{i \in I} X_{ijt} && \forall j \in J, \forall t \in T \\
& q_{it}, r_{jt}, v, w \geq 0 && \forall i \in I, \forall j \in J, \forall t \in T \\
& X_{ijt} \in \mathbb{N} && \forall i \in I, \forall j \in J, \forall t \in T
\end{aligned}$$

Values of Lambda The parameters λ_1 to λ_4 indicate the importance of the different components of the objective function. We consider the first two components, which indicate the largest deviations, to be the most important. When using larger data sets, the sum of all deviations, represented by component three and four, quickly exceeds the value of the largest deviations, represented by component one and two. To make sure that the first two components are considered as more important we have to assign higher values to λ_1 and λ_2. We have tried several values and decided to use the following assignments. λ_1 is equal to the maximum value that the sum of the deviations with the contract hours can attain. λ_2 is equal

to the maximum value that the sum of the deviation with the demand can attain. λ_1 and λ_2 are equal to 1.

$$\lambda_1 = \sum_{i \in I} \sum_{t \in T} (\max(c_i - l_i, u_i - c_i) \cdot p_{it} \tag{4.25}$$

$$\lambda_2 = \sum_{i \in I} \sum_{t \in T} d_{jt} \tag{4.26}$$

$$\lambda_3 = \lambda_4 = 1 \tag{4.27}$$

By choosing these values for λ_1 to λ_4, the first two components will always have a higher value than the third and fourth component, regardless of the values of $\sum_{i \in I} \sum_{t \in T} q_{it}$ and $\sum_{j \in J} \sum_{t \in T} r_{jt}$. Therefore, no matter what data set we choose, the first two components will always be considered more important by default. In the remainder of this thesis, we use these values for λ_1 to λ_4. However, we use λ_1 to λ_4 as our notation. This is for convenience and for the fact that the values are ultimately determined by the user and can be changed at any moment.

5

Iterative Procedure

The model we have used so far ensures that the largest value of deviation in contract hours or demand is minimized. This, as mentioned, means that the solution loses a bit of accuracy. We are going to describe a procedure in which the model is used multiple times in order to obtain a more accurate solution.

5.1. Iterative Procedure 1

The main idea is as follows. The first time the model provides a solution, we consider three aspects. The first one is the value of the largest deviation. The second one is for which $i \in I$, $j \in J$ and $t \in T$, the variables X_{ijt} contribute to this specific deviation. Lastly, we determine the values of these specific X_{ijt}'s. We then run a model equivalent to the original model, but we fix the values of these X_{ijt} variables. We also delete the constraints in which these X_{ijt} variables are included. Finally, we add a constraint which makes sure that the maximum deviation in the next iteration, is smaller or equal to current one. When solving this new model, the largest deviation in the solution of the new model, is essentially the second largest deviation in the original model. By replacing all old X_{ijt} variables with the newly found X_{ijt}'s, means that we have minimized the largest and second largest deviation of the original problem. When this principle is applied a second time, all X_{ijt}'s contributing to the second largest deviation are fixed, and again, constraints including these X_{ijt}'s are deleted from the model. Finally another constraint is added to make sure that the maximum deviation in the third iteration is smaller or equal to the current maximum deviation. By solving this new model, the third largest deviation can be found and minimized. By applying this principle a third time, the fourth largest deviation can be found and minimized, and so on. This is what is called Iterative Procedure 1. Next, we provide an example using a small data set.

Example Suppose we have only one employee ($I = \{1\}$) who has only one task ($J = \{1\}$) which he can execute ($s_{11} = 1$). His weekly contract hours are 10 ($c_1 = 10$), with a minimum of 5 ($l_{1t} = 5$, $\forall t \in T$), and a maximum of 15 ($u_{1t} = 15$, $\forall t \in T$) working hours. We consider a three week period ($T = \{1, 2, 3\}$). In week one, the demand for this task is 50 hours ($d_{11} = 50$). In week two and three, the demand for this task is 10 hours ($d_{12} = d_{13} = 10$). Assume that the employee is present in every week ($p_{it} = 1$, $\forall t \in T$). The model looks as follows:

This was our first iteration. For our second iteration, we take the value of the largest deviation in contract hours, which is $q_{11} = 5$ or $q_{12} = 5$, and the largest deviation in demand, which is $r_{11} = 35$. Since q_{11} and q_{12} have the same value, which one should we take? For now, we always go for the first one (This of course, may not be the best one to pick, but that will be fixed later on). So in this case we consider $q_{11} = 5$ to be the largest deviation in contract hours. Now we consider for which $i \in I, j \in J$ and $t \in T$, the variables X_{ijt} contribute to these specific deviations. In this case, this is only variable X_{111}. Therefore, $X_{111} = 15$ will be fixed

by making it a new constraint in our model. Furthermore, all constraints containing X_{111} will be removed from the model in order to not let the model take this variable into account. New variables, v_2 and w_2, are introduced and constraints $v \geq v_2$ and $w \geq w_2$ are added to prevent fluctuating behaviour between deviation in contract hours and deviation in demand. Note that now v and w have become fixed parameters. Finally, let T_2 be $T \setminus \{1\}$. Our model for the second iteration is this:

$$\min \quad z_2 = \lambda_1 \cdot v_2 + \lambda_2 \cdot w_2 + \lambda_3 \cdot \sum_{t \in T_2} q_{1t} + \lambda_4 \cdot \sum_{t \in T_2} r_{1t}$$

$$\begin{aligned}
s.t. \quad & \sum_{t \in T} X_{11t} = \sum_{t \in T} c_1 p_{1t} \\
& X_{111} = 35 \\
& l_{1t} \cdot p_{1t} \cdot s_{11} \leq X_{11t} \leq u_{1t} \cdot p_{1t} \cdot s_{11} && \forall t \in T_2 \\
& v \geq v_2 \\
& v_2 \geq X_{11t} - c_1 \cdot p_{1t} && \forall t \in T_2 \\
& v_2 \geq c_1 \cdot p_{1t} - X_{11t} && \forall t \in T_2 \\
& w \geq w_2 \\
& w_2 \geq X_{11t} - d_{1t} && \forall t \in T_2 \\
& w_2 \geq d_{1t} - X_{11t} && \forall t \in T_2 \\
& q_{1t} \geq X_{11t} - c_1 \cdot p_{1t} && \forall t \in T_2 \\
& q_{1t} \geq c_1 \cdot p_{1t} - X_{11t} && \forall t \in T_2 \\
& r_{1t} \geq X_{11t} - d_{1t} && \forall t \in T_2 \\
& r_{1t} \geq d_{1t} - X_{11t} && \forall t \in T_2 \\
& q_{1t}, r_{1t}, v_2, w_2 \geq 0 && \forall t \in T_2 \\
& X_{11t} \in \mathbb{N} && \forall t \in T_2
\end{aligned}$$

If we let CPLEX solve this model, we get the following result.

$X_{111} = 15$ $X_{112} = 8$ $X_{113} = 7$	$q_{11} = 5$ $q_{12} = 2$ $q_{13} = 3$	$r_{11} = 35$ $r_{11} = 2$ $r_{11} = 3$
	$v_2 = 3$	$w_2 = 3$
		$z_2 = 2527$
		$z = 1276$

Table 5.1: Solution to iteration 2

By looking at Table 5.1, we can see that new values for X_{112} and X_{113} are found. Also, since $v \geq v_2$ and $w \geq w_2$, it is logical that $z_1 \geq z_2$. To determine if we have found a better solution we have to plug in the new values of the X_{ijt} variables in objective (4.1). If we do so, we find that our new original objective value is 1276, which is less than the previous 1300. This means that we have found a better solution to the original problem.

This principle can be applied another time to find a z_3, a z_4, and so on. This so called Iterative procedure provides a better (or equally good) solution to the original problem with each iteration. On the downside, more iterations mean more total computation time. In Chapter 6, we compare the objective value of every iteration to the computation time. That way we can find a balance between the accuracy of the model and the time in which this answer can be provided.

5.2. Computation Time Management

For small data sets, like the one in the example of Section 5.1, computation time will probably not be an issue. This is because the optimal solution can be found relatively fast due to the small number of constraints, and also because of the small number of iterations that have to be done before all deviations have been checked. But, when we consider larger data sets, computation time may grow exponentially. There are tree methods that we have implemented in order to decrease or control the computation time.

The first one is stopping the iterative process when the largest deviation turns out to be zero. When the largest unchecked deviation is zero, it is not possible to find a better solution. Therefore, it is unnecessary to continue the iterative process.

The second method to control the computation time is the implementation of a timer. The user can set a maximum calculation time. When the computation time exceeds the maximum given time, the iterative process stops. The algorithm will present the best solution it has found within this time. In Chapter 6, we compare different time intervals to the accuracy of the found solution. This way we can see to what extend it is useful to give the algorithm more time.

A third way to reduce computation time is discussed in the next section. In this approach we fix multiple deviations per iteration. Thereby, the total number of iterations needed to check all deviations can be reduced substantially.

5.3. Iterative Procedure 2

In this procedure we are going to fix multiple deviations per iteration. Securing all maximum deviations at once will provide a solution in much shorter time. But it may come at the expense of the accuracy of the solution. Take for example the first solution found in the example in Section 5.1. There are two occurrences of the maximum deviation in contract hours, namely $q_{11} = 5$ and $q_{12} = 5$. If both of the deviations were fixed at once by adding $X_{111} = 15$ and $X_{112} = 5$ to the constrains, it would be impossible to find the better solution (where $X_{112} = 8$) in the second iteration. When we solve the model for a given data set, the maximum deviation v can have multiple occurrences. In Iterative Procedure 1, the first of these maximum occurrences is fixed and the model is solved a second time. Now two things can happen. We find a new solution in which $v > v_2$, or we find a solution in which $v = v_2$. When $v = v_2$, we have essentially found no new information on our second largest deviation, since this v_2 was already present in the first iteration and has not been improved. If we fix the variables that contributed to both deviations in the first iteration, we would have the same solution with less iterations. The ability to fix more than one variable is what is added in the second iterative procedure.

The user of the algorithm can determine, in advance, the percentage of maximum deviations that will be fixed in each iteration. This can be done separately for the deviation in contract hours and the deviation in demand. Suppose that in an arbitrary optimal solution, there are ten occurrences of a maximum deviation of contract hours. It takes the first iterative procedure at most ten iterations to confirm this, even though the first iteration already showed this. The second iterative procedure can fix all these deviations at once, and therefore, needs only one iteration to provide the same solution. The calculation time has then already been decreased by a factor of at most 10.

The relation between computation time, accuracy and percentage of fixed deviations per iteration, is explored in Chapter 6.

6

Results

In this chapter, we present various comparisons regarding computation time and accuracy between different solvers and the iterative procedures. We test our model on data sets obtained by a random data generator, created by E. van der Veen, provided to us by ORTEC.

The data sets we use when solving the model are provided to us by ORTEC. E. van der Veen created a random data generator which generates data such as contract hours and demand for a fictional company. We use this randomly generated data to test our model on.

6.1. Solvers

The model is implemented in Python 3.5 using a package named PuLP. PuLP is a free open source software which is used to describe optimization problems as mathematical models. External LP solvers (CPLEX, GUROBI, GLPK etc.) can be called by PuLP to solve this model and display the solution. We are going to use the open source-solver COIN-OR, and commercial solvers Gurobi and CPLEX to solve our model. Their computation times are compared in the next section. We use Gurobi and CPLEX because these are the best solvers for ILP problems.

6.1.1. Comparing Solvers

The three mentioned solvers each have their own way for calculating an optimal solution when dealing with an ILP problem. More information on how the solvers do this can be found the solver manuals [6–8]. It differs per problem which approach is most efficient. That is why we are comparing the solvers on three different aspects. Firstly, we have the time to set up the model. Secondly, we have the execution time. Lastly, we have the process time. Execution Time is the duration from when the process was started until the time it terminated. It can be seen as real 'wall clock time'. Process Time is the time that the CPU spent on computing the process. When other programs are running on your computer, the execution time may be higher or lower depending on how many programs your computer can run at the same time. The process time is trying to get rid of these inequalities by looking at the time the CPU spends on the calculations for this program.

We use three different data set sizes in this first comparison, small ($|I| = 3$, $|J| = 2$, $|T| = 5$) medium ($|I| = 10$, $|J| = 2$, $|T| = 52$) and large ($|I| = 100$, $|J| = 5$, $|T| = 52$). Here $|I|$ means the number of elements in the set I. In other words, $|I| = 3$ means that there are 3 employees in this data set.

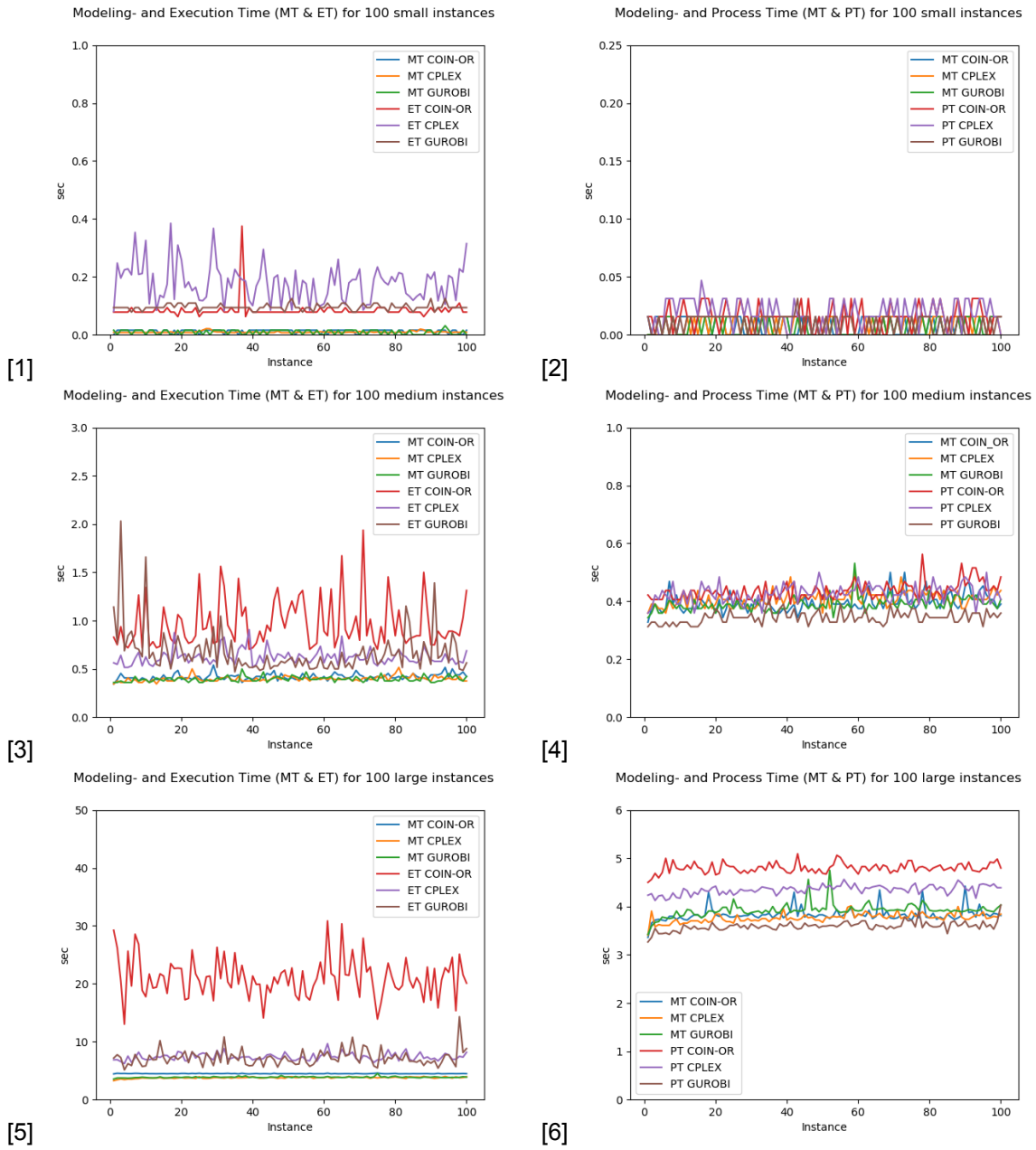

Figure 6.1: Modelling, Execution and Process Time of three different solvers and three different data set sizes.

When looking at Figure 6.1, a few things are noticeable. Firstly, the modelling time is similar for every solver in every data set. This is because setting up the model involves no calculations. Secondly, we look at the small data sets. There is almost no difference in processing time for all solvers. They vary from almost 0.0 to 0.05 seconds. However, there is a difference in execution time. CPLEX is, according to Table 6.1, on average almost 0.1 second slower. In the medium sized data sets, there is a small difference in the processing time. Gurobi is on average less than 0.1 seconds faster than COIN-OR and CPLEX. However, when looking at the execution time, we see that COIN-OR is starting to separate from CPLEX and Gurobi and takes 0.3 seconds longer to present a solution. We also see that COIN-OR and Gurobi are more fluctuating in execution time than CPLEX. This really becomes clear when we consider large data sets. Although the processing time for Gurobi is almost a second shorter than the one from CPLEX, for large data sets, the execution times are on average the same. Note that the fluctuation of the execution time of Gurobi is significantly larger than the one from CPLEX.

[1]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	0.00	0.01	0.02
mt-CPLEX	0.07	0.01	0.02
mt-Gurobi	0.00	0.01	0.03
et-COIN-OR	0.06	0.08	0.37
et-CPLEX	0.08	0.18	0.38
et-Gurobi	0.08	0.10	0.13

[2]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	0.00	0.01	0.02
mt-CPLEX	0.00	0.01	0.02
mt-Gurobi	0.00	0.01	0.02
pt-COIN-OR	0.00	0.02	0.03
pt-CPLEX	0.00	0.02	0.05
pt-Gurobi	0.00	0.01	0.03

[3]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	0.34	0.42	0.54
mt-CPLEX	0.34	0.40	0.52
mt-Gurobi	0.36	0.39	0.50
et-COIN-OR	0.70	0.96	1.94
et-CPLEX	0.52	0.62	0.91
et-Gurobi	0.47	0.67	2.03

[4]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	0.33	0.40	0.50
mt-CPLEX	0.36	0.41	0.50
mt-Gurobi	0.35	0.39	0.53
pt-COIN-OR	0.39	0.44	0.56
pt-CPLEX	0.36	0.43	0.50
pt-Gurobi	0.31	0.34	0.41

[5]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	4.42	4.48	4.58
mt-CPLEX	3.27	3.78	4.02
mt-Gurobi	3.56	3.87	4.51
et-COIN-OR	13.0	21.0	30.9
et-CPLEX	6.22	7.37	9.66
et-Gurobi	5.13	7.20	14.3

[6]

	t_{min}	$t_{average}$	t_{max}
mt-COIN-OR	3.36	3.83	4.42
mt-CPLEX	3.41	3.76	4.02
mt-Gurobi	3.42	3.92	4.75
pt-COIN-OR	4.50	4.80	5.09
pt-CPLEX	4.13	4.35	4.56
pt-Gurobi	3.27	3.59	4.03

Table 6.1

6.1.2. Choosing a Solver

For small and medium data sets, the differences per solver in processing and execution time are not significant in a practical sense. When working with larger data sets, it is clearly more preferable to use CPLEX or Gurobi as a solver since they are three times as fast as COIN-OR. One important objective of this thesis is to give the user control over the calculation time of the model. Because CPLEX has in our case a more constant execution time than Gurobi, we are going to use CPLEX as our solver for the remaining results of this thesis.

6.2. Iterative Procedure 1

We are going to apply the first iterative algorithm to large data sets to see what it does to the value of the original objective function. As stated in the previous section, we let CPLEX solve the model. The iterative algorithm is run on twenty different data sets of the same size. In Figure 6.2, the objective function is set out against the number of iterations. The implemented timer is set to thirty minutes. When thirty minutes have passed or when the maximum deviation reaches zero, the iterative procedure stops.

When we look at the upper plot of Figure 6.2, we see that there is definitely improvement in the original objective function. Most of the improvement takes place in the first 100 iterations. This becomes more clear when we zoom in on the middle section of the figure. This is displayed in the bottom plot of Figure 6.2.

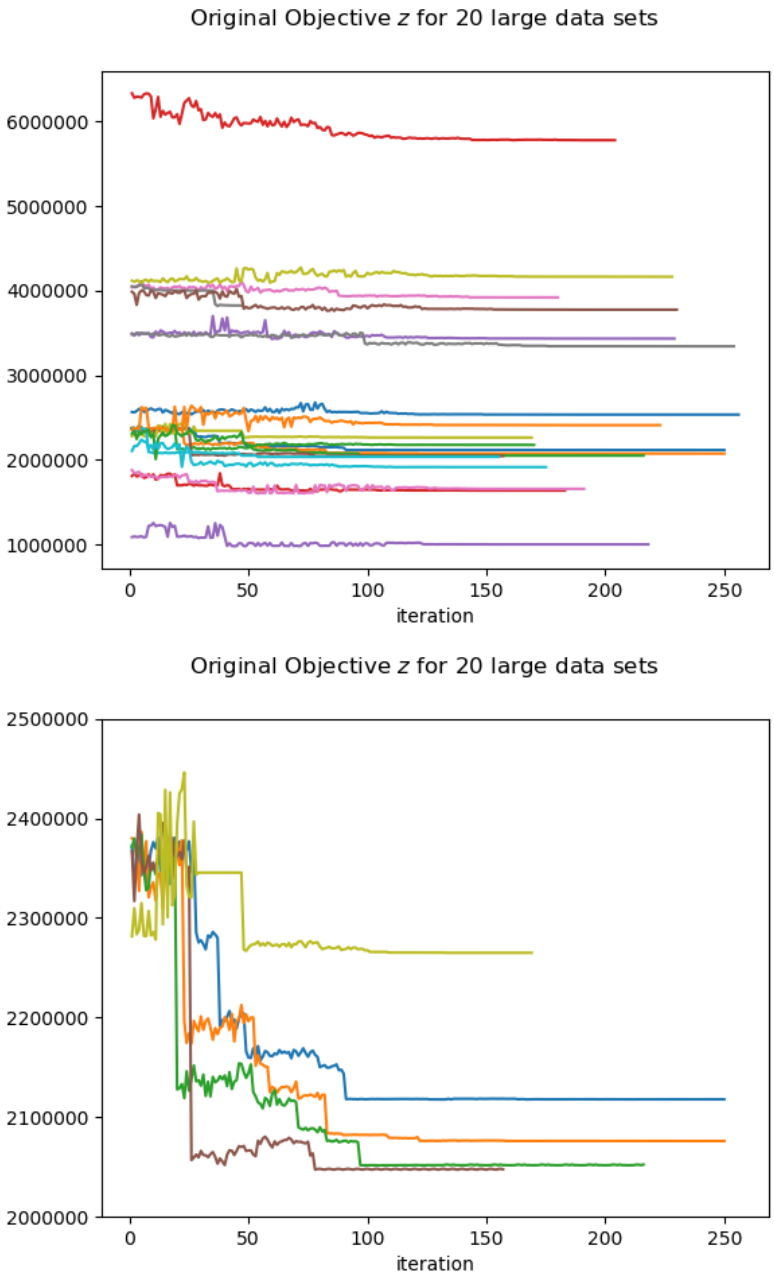

Figure 6.2

Every graph seems to fluctuate around a certain value before it gradually decreases in a step-wise manner. One would expect that the objective value should not increase. The fluctuations can be explained in the following way. The objective function of our model looks at the value of the sum of the deviations. Therefore, one deviation of size four or two deviations of size two will have the same contribution to the objective value (when these are not the largest deviations). However, in the original objective function we have $4^2 = 16$ and $2^2+2^2 = 8$, therefore, the deviation of size four has more 'weight'. Therefore, different solutions can have the same z_i value in our model, but other values of z, hence the fluctuations. Now we are going to plot the values of z_1, z_2, z_3, ...,z_n, for every iteration against the original objective value z to find a correlation between their decline. We do this for three random large data sets.

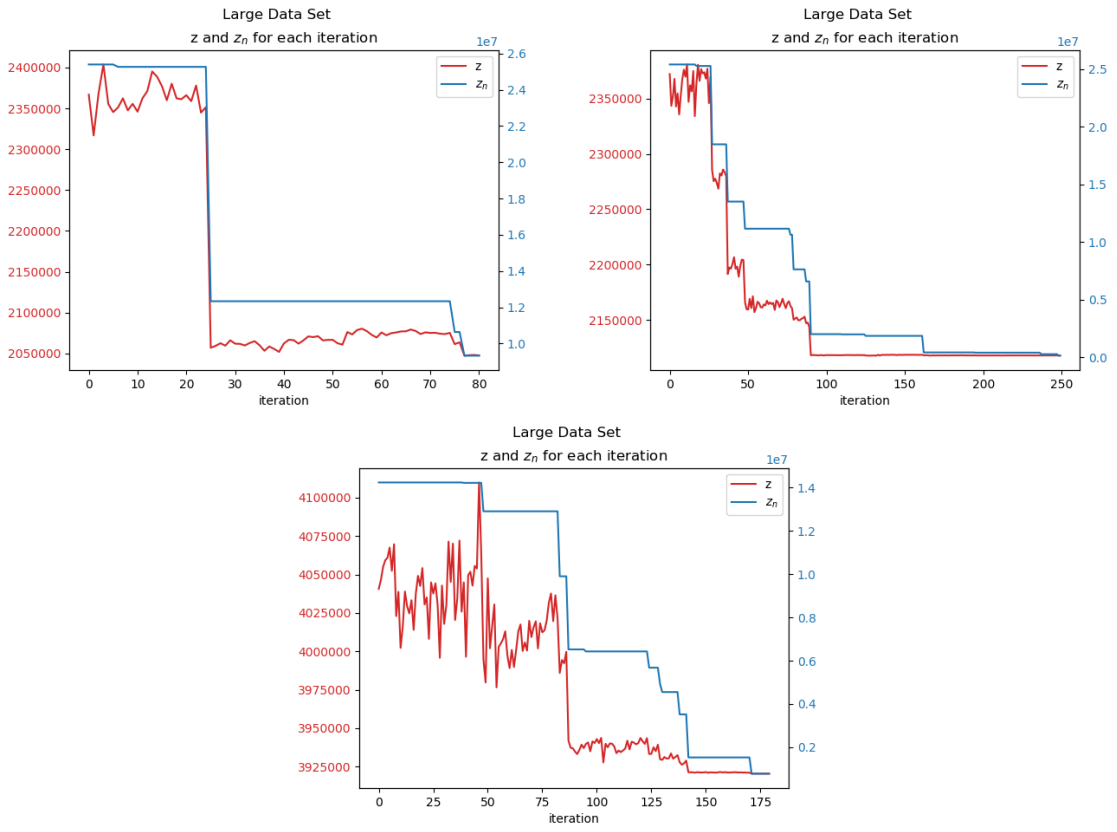

Figure 6.3: z and z_n comparison for three random large data sets.

In Figure 6.3, we see that every time when there is a significant drop in the value of the original objective z, there is also a drop in the value of z_i at the same iteration. We also see in the graph that the values of z_i can stay the same for multiple iterations. This means that the largest deviation, in contract hours and demand, occurs more often and can not be improved at multiple occurrences. In the third example of Figure 6.3, z_i has the same value for almost 50 iterations. This means that in this case the iterative procedure does not cause much improvement until iteration 50. In the next, section we see if this can be improved by fixing multiple deviations per iteration.

6.3. Iterative Procedure 2

In the previous section, we saw that the value of z_i can stay the same for many iterations. This is because there can be more occurrences of the maximum deviation. We are now going to fix multiple deviations in one iteration. We can select the percentage of maximum deviations that is fixed per iteration, which is denoted by FPPI (Fixed Percentage Per Iteration). For now, we let the FPPI be the same for the deviations in contract hours and the deviations in demand. The occurrences that are fixed are chosen at random. For example, we set FPPI to 20% and we get fifty occurrences of the maximum deviation in contract hours and twenty-one occurrences of the maximum deviation in demand in the first iteration. Then ten deviations in contract hours and five deviations in demand are fixed for the second iteration.

We compare three instances of the same size on accuracy, execution time and number of iterations. We do this for FPPI set to 1%, 5%, 10%, 15%, 20%, 50%, and 100%. Since we have seen that most of the improvement occurs within the first ten minutes, we set the timer to ten minutes. The results are presented in Table 6.2 to Table 6.4. The exact solution to each problem is also presented in the tables. To obtain this solution, we used the exact model in Bouwmeester [2]. There can be large differences in the exact solution and the approximation. This is a consequence of the fact that the data is randomly generated, and therefore, may not be representative for real companies. For example, a deviation of $d_{ij} = 150$ is not uncommon

in our data. However, it is unlikely to have a 150 hour deficiency per week in real life. Moreover, the computation time of the exact model is much lower. The average computation time for large data sets was twenty seconds. This will be discussed in Chapter 7.

Since the number of occurrences in maximum deviations is small for small data sets, we decided to consider only FPPI set to 1%, 50% and 100%. In Table 6.2, we see that fixing multiple deviations decreases the accuracy most of the times. Moreover, the difference in execution time is not more than three seconds. We do see an improvement in the original objective solution in the second example. This is a consequence of letting the iterative procedure choosing random maximum deviations to fix.

For medium data sets, shown in Table 6.3, increasing the FPPI, has a significant impact on the execution time. When the FPPI is set to 50%, for the second example, we obtain the final solution five times as fast as when using 1%, while their objective values are relatively close to each other, namely $96,870$ and $92,784$ respectively. However, in the first example, we see that the objective value for an FPPI of 50% is much higher than at 1%. Therefore, we can not say much about the accuracy of the objective value when using high FPPI. The optimal FPPI range when considering computation time and accuracy is 1% to 5% for medium data sets.

Lastly we look at the large data sets. Only for large data sets the time limit is reached in some cases. One remarkable point is that an FPPI of 10% or 15% provides better solutions in less or the same time. This is because too many maximum deviations have to be fixed in order to improve the objective function. When fixing 1% per iteration, ten minutes is not enough to cover all the maximum deviations, while when fixing 10% or 15%, it is enough. Moreover, in example three, we see that an FPPI of 20% provides a relatively accurate solution while the execution time is divided in half. FPPI's of 50% and 100% provide really fast solutions but are very inaccurate. Therefore the best range is 10% to 20%.

FPPI	1%	50%	100%
tijd in sec.	4.39	2.79	1.23
doelfunctie	8365	8457	8457
FPPI	1%	50%	100%
tijd in sec.	2.70	2.23	1.69
doelfunctie	650	650	618
FPPI	1%	50%	100%
tijd in sec.	4.50	4.50	1.25
doelfunctie	3229	3385	3571

Table 6.2: Results Iterative Procedure 2 on Small Data Sets

FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	318	290	248	159	163	58	17.4
obj.value	128,181	125,905	129,285	136,297	127,313	138,541	133,619
FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	257	204	197	156	68	39	8.4
doelfunctie	92,784	90,762	92,650	95,082	97,760	96,870	104,462
FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	67	109	58	116	33	29	18.8
doelfunctie	118,188	118,452	120,724	117,162	126,634	119,140	120,146

Table 6.3: Results Iterative Procedure 2 on Medium Data Sets

FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	600	566	600	437	257	103	51
doelfunctie	2.138.149	2.132.375	2.109.877	2.132.375	2.152.809	2.276.717	2.296.161
FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	600	600	600	558	455	188	65
doelfunctie	1,627,340	1,620,720	1,550,698	1,561,128	1,681,976	1,706,764	1,711,540
FPPI	1%	5%	10%	15%	20%	50%	100%
tijd in sec.	600	600	600	482	336	57	170
doelfunctie	2,113,973	2,102,273	2,036,265	2,040,513	2,070,843	2,168,009	2,221,167

Table 6.4: Results Iterative Procedure 2 on Large Data Sets

7

Conclusion

In this thesis our goal was to construct and investigate a model that is an extension of the famous Nurse Rostering Problem. This extension gave nurses different skills in order to perform certain tasks. It is important that the user has control over the execution time of the model. The model provides a working schedule for up to 100 employees, 5 skills and 52 weeks. It ensures that the maximum difference between working hours and contract hours are minimized and does the same for the demand per skill. The original quadratic objective function is linearized to formulate this problem as an ILP problem. Different solvers are compared to computation time and consistency. Lastly, two iterative procedures explained in Chapter 5 are compared on accuracy of the solution and their computation time.

We have found that Gurobi and CPLEX are the best solvers for this particular problem. On large data sets they both had an average computation time of 7.2-7.3 seconds. Since in our case, CPLEX was more consistent in computation time for different data sets, we choose CPLEX as our solver for the remainder of this thesis.

To get control over the computation time, the model has a built in timer that can be set to the preferences of the user. When using Iterative Procedure 1, we have seen that the accuracy of the model can be improved significantly. However, this comes at the cost of the computation time. After letting the model run for thirty minutes there is no more improvement to be made. Moreover, most of the improvement made by Iterative Procedure 1 takes place in the first ten minutes. Therefore, the timer is set to ten minutes when we consider Iterative Procedure 2.

When investigating Iterative Procedure 2, we discovered that the computation time can be decreased drastically. However, this may come at the cost of the accuracy. We have found that for small data sets, fixing multiple deviations per iteration has a negative impact on the accuracy of the model. Therefore, the advice is to set the FPPI to 1% when working with small data sets. Considering medium sized data sets, increasing the FPPI only has little benefits to the computation time. However, not much can be said about the accuracy. Our suggestion is to set the FPPI between 1% and 5% to ensure an accurate solution. For large data sets, we recommend to use an FPPI between 10% and 20%. A 10% FPPI can obtain a more accurate solution in the given ten minutes than 1% and 5%. However, a 20% FPPI is twice as fast. The importance of the accuracy, the computation time and the time limit are all determined by the user.

For medium and large data sets, our approach to solving this particular Nurse Rostering Problem was not as time efficient and accurate as the model used in Bouwmeester [2]. However, this approach may be of interest to other problems. Although this a good start in the investigation of this particular extension of the Nurse Rostering Problem, there are several ways in which it can be improved in further research. For example, our iterative procedure chooses random occurrences of largest deviations to fix. It would be interesting to investigate if there is a way to easily determine which deviations are best to be fixed. Another improvement that can be made is to let the FPPI differ on every iteration. For example, we can fix 50% of the maximum deviations in the first iteration, and 100% in the second iteration. In

that way we obtain a new maximum deviation every two iterations. Lastly, the model can be used on real data from real companies.

Bibliography

[1] El-Awady Attia, Philippe Duquenne, and Jean-Marc Le-Lann. Considering skills evolutions in multi-skilled workforce allocation with flexible working hours.

[2] M. Bouwmeester. Annualized hours: Comparing an exact optimization model with its approximation. 2019. URL http://resolver.tudelft.nl/uuid:2529862b-d90c-4ee7-a995-fcc54281e65a.

[3] A Corominas, A Lusa, and R Pastor. Using milp to plan annualised working hours. .

[4] Albert Corominas, Jordi Olivella, and Rafael Pastor. Capacity planning with working time accounts in services. .

[5] Albert Corominas, Amaia Lusa, and Rafael Pastor. Planning annualised hours with a finite set of weekly working hours and joint holidays. *Annals of Operations Research*, 128, 2004.

[6] IBM CPLEX Optimization Studio. Gurobi optimizer reference manual, 2019. URL https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/usrcplex.pdf.

[7] Computational Infrastructure for Operations Research. Coin-or documentation, 2019. URL https://www.coin-or.org/documentation.html.

[8] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019. URL http://www.gurobi.com.

[9] Alain Hertz, Nadia Lahrichi, and Marino Widmer. A flexible milp model for multiple-shift workforce planning under annualized hours.

[10] J.H. Keim. Annualised hours: Optimisation with heuristic algorithm. 2018. URL http://resolver.tudelft.nl/uuid:cff60809-75e8-4908-967f-1b93419b1079.

[11] Md Gulzar Ull Hasan, Irfan Ali, and SS Hasan. Annualized hours planning with fuzzy demand constraint. In *ProbStat Forum*, 2016.

[12] Egbert van der Veen, Erwin W Hans, Bart Veltman, Leo M Berrevoets, and Hubert JJM Berden. A case study of cost-efficient staffing under annualized hours. *Health care management science*, 2015.

[13] Jan van Leeuwen. Algorithms and complexity. handbook of theoretical computer science. vol. a, chapter 10. *Elsevier Science Publishers*, 22:81, 1990.