Master thesis

Work Distribution for a Heterogeneous Library Staff - A Personnel Task Scheduling Problem

Claes Arvidson and Emelie Karlsson

LiTH-MAT-EX–2016/05–SE

Work Distribution for a Heterogeneous Library Staff - A Personnel Task Scheduling Problem

Optimeringslära, Linköpings Universitet

Claes Arvidson and Emelie Karlsson

LiTH-MAT-EX–2016/05–SE

Exam work: **30 hp**

Level: **A**

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

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

Linköping: **June 2016**

Abstract

The distribution of tasks to a heterogeneous work force at libraries and other service institutions is a time consuming task for manual schedulers. In this thesis, we study the possibility of making the assignment using operations research techniques. The problem studied concerns seven days per week, five types of tasks, two types of staff qualifications and around 100 tasks per week to be assigned to the staff. Staff member satisfaction is also taken into account in the scheduling process.

The main objective is to create an optimal ten week rotating schedule, in which the stand-in staff members are evenly distributed. Such a schedule is considered to be robust, since stand-in staff can replace the regular staff when there is unforseen absence.

A mathematical model is formulated for the problem and is solved using the commercial solver CPLEX. We also present two different large neighbourhood search heuristic implementations for this problem. The first heuristic assigns complete week blocks to the staff members, while the second one distributes one task at a time. The latter heuristic works better than the former and achieves results comparable to those of the commercial solver. Our conclusion is that the second heuristic works better because it focuses on finding a good weekend distribution before creating the rest of the schedule. A conclusion from our work is that the weekend-worker constellation is the most significant degree of freedom in the problem.

Keywords: Optimization, Scheduling, Task distribution, LNS, Weekend Scheduling, Heterogeneous workforce

URL for electronic version:

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-77777

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Torbjörn Larsson, at Linköping University, who has helped us find our way in moments of confusion. Thank you for the time and effort you have spent on our thesis. We would also like to thank our examiner, Elina Rönnberg, who has provided encouraging words and valuable insights throughout the project.

Furthermore, we would like to thank Elisabeth Cserhalmi and Ingrid Loeld Rasch at Norrköpings Stadsbibliotek for providing us with an interesting topic and for answering all of our questions many times over.

We thank our opponents, Akdas Hossain and Emma Miléus, for their comments and thoughts on this report.

Lastly, we would like to thank our families and loved ones, who have continuously supported us throughout our studies.

Nomenclature

The most reoccurring terms and abbreviations are described here.

Optimization terms

Heuristic	An algorithm designed to find good, but not necessarily optimal, solutions.
Large neighbourhood search	Heuristic search method which alternately destroys and respairs the solution in order to move through the solution space.
Simulated annealing	Heuristic search method which allows for the acceptance of poorer solutions with a certain probability in order to diversify the search. The probability decreases with time.

Library related terms

Fetch list	The task of collecting reserved books from shelves.
Library on wheels	The library bus providing books to remoter areas of town.
Outer tasks	The scheduled tasks including Exp, Info, PL, HB and BokB.
Inner tasks	The tasks which are not outer tasks are inner tasks. These include less visible work such as answering emails or attending group meetings.
Week rest	The period of rest granted a staff member after weekend work. This normally includes one or two extra days off.

Abbreviations

LNS	Large neighbourhood search
SA	Simulated annealing
Exp	Service counter (sv. expeditionsdisken)
Info	Information counter (sv. informationsdisken)
PL	Fetch list (sv. plocklistan)
BokB	Library on wheels (sv. bokbussen)
HB	Hageby library

Contents

List of Figures

List of Tables

Chapter 1

Introduction

At a library there exist a number of tasks, which are to be distributed over the staff members. Poor task distribution to staff members can cause problems and result in a shortage of staff at certain shifts. If a staff member becomes unavailable at such shifts due to, for example, illness, a qualified stand-in is required to fill the vacancy. Therefore, it is of high priority to a library to create schedules with as many skilled stand-ins as possible in order to handle such unexpected disturbances.

The problem addressed in this thesis work concerns the library staff at Norrköpings Stadsbibliotek (en. the Library of Norrköping), Sweden. This library currently has 39 employees and the renown building from the 1950's is a central gathering point in Norrköping. The library is open during weekdays from 8 a.m. to 8 p.m. and from 11 a.m. to 16 p.m. during weekends. The generous opening times also create a scheduling challenge for the library as they require a large pool of well coordinated personnel to keep the library open. In addition, the library also provides its services to one other smaller library, which adds further complexity to the problem of task distribution among the staff.

This chapter contains a problem description, where the scheduling problem is described in detail, a section describing the goals and aims of the thesis work, a section about the methods used to solve the problem and an outline of this report.

1.1 Problem description

The main objective of the studied problem is to create a ten week schedule for all staff members at the library. Each staff member's ten week schedule should consist of two five-week schedules, which should be as similar as possible, although not identical, as some tasks are performed only on even or odd weeks. Furthermore, the ten week schedule should be rotational, meaning that it can be reused after ten weeks.

The schedule should be created so that the stand-ins are distributed evenly over all days. In order to do this, the library provided information about when staff members are available for task assignment and how and when these tasks can be assigned. These constraints are discussed in detail in this section.

1.1.1 Description of the daily tasks at the library

The highest priority at a library, as with any service institution, is to provide good service to visitors. This includes book loan services, as well as being able to give visitors helpful information about the resources at disposal at the library. This type of work is referred to as "outer tasks" in this thesis. In addition to these, "inner tasks" such as sorting books, ordering new books and answering emails also exist and are part of the every day library tasks as well. The problem of assigning the outer tasks, while taking inner tasks into consideration, is studied in this thesis.

Three main types of outer task can be identified at the Library of Norrköping: working at the service counter (sv. expeditionsdisken), working at the information counter (sv. informationsdisken) and assembling books which are to be sent to other libraries according to the "fetch list" (sv. plocklista). The fetch list is a task where the staff member is scheduled during the whole day (evenings excluded), while the other tasks are scheduled for only one shift. The outer tasks can be performed either exclusively by librarians or by both librarians and assistants, as described in Table 1.1.

Table 1.1: Outer tasks can be performed either exclusively by librarians or by both librarians and assistants.

Task	**Description**	**Qualification**
Service counter (Exp)	Administering loans, library cards and managing returned books	Librarian or assistant
Information counter (Info)	Handling questions about the library's resources	Librarian
Fetch list (PL)	Fetching books that are to be sent to other libraries	Librarian or assistant
Hageby (HB)	Handling librarian tasks at the filial Hageby during weekends	Librarian
Library on wheels (BokB)	Driving the library on wheels to different areas of town	Librarian

As in the case with most libraries, Norrköpings stadsbibliotek also has responsibilities that fall outside of its normal daily activities. One such responsibility is weekend staffing of a smaller library subsidiary in Hageby, situated in the suburban area of Norrköping. Only librarians are qualified for this task. While some librarians prefer to work only at HB during weekends, others prefer never to work there.

Mostly, weekend work entails working three consecutive days: Friday (evening), Saturday and Sunday. However, staff members who are scheduled to work in HB during a weekend only work Saturday and Sunday. The weekend work is compensated by days free from work, often placed at the week following upon weekend work, in this thesis referred to as the week rest week.

Since the number of visitors at the library varies throughout a day, so does the demand for personnel for the outer tasks. This demand is illustrated in Table 1.2, which is constructed according to figures given by the library.

A library task not included in this Table 1.2 is the "library on wheels" (sv. bokbussen), for which only a handful of librarians are qualified. This work involves driving a library bus, which provides citizens in remoter areas of the

Table 1.2: Staff demand during a week. PL is marked as one shift, but it is performed during the whole day.

	Exp	Info	PL	HB
Monday				
Shift 1:	2	2	1	0
Shift 2:	3	3	0	0
Shift 3:	3	3	0	0
Shift 4:	3	3	0	0
Tuesday				
Shift 1:	2	2	1	0
Shift 2:	3	3	0	0
Shift 3:	3	3	0	0
Shift 4:	3	3	0	0
Wednesday				
Shift 1:	2	2	1	0
Shift 2:	3	3	0	0
Shift 3:	3	3	0	0
Shift 4:	3	3	0	0
Thursday				
Shift 1:	2	2	1	0
Shift 2:	3	3	0	0
Shift 3:	3	3	0	0
Shift 4:	3	3	0	0
Friday				
Shift 1:	2	2	1	0
Shift 2:	3	3	0	0
Shift 3:	3	3	0	0
Shift 4:	3	3	0	0
Saturday				
Shift 1:	3	3	0	1
Sunday				
Shift 1:	3	3	0	1

city with books. The library on wheels only operates a few times per week and the schedule differs between odd and even weeks, as is illustrated in Table 1.3.

Apart from the outer tasks described above, there are also inner tasks at the library which sometimes need to be scheduled. One such type of inner task is meetings, which concerns a large number of staff members. There exist both library meetings, scheduled for the whole work force, and group meetings, scheduled only for a part of the staff members. The library meetings are fixed in time and take place every fifth week on Monday morning from 8 a.m. to 10 a.m. Also group meetings take place every fifth week but are flexible in time. For group meetings, all group members must be available. Only three groups have scheduled meetings: the child, adult and media group.

Table 1.3: Demand of staff for library on wheels

Odd Week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**
08:00-10:00	1	0	1	1	1
16:00-20:00	1	0	1	1	0
Even Week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**
08:00-10:00	1	0	1	1	0
16:00-20:00	0	0	1	1	0

1.1.2 Personnel attributes

The two types of library staff members that are to be assigned tasks are librarians and assistants. Librarians can perform all the tasks listed above, while assistants can perform a subset of these tasks.

Norrköpings stadsbibliotek currently has 39 library staff members, 23 of which are librarians and 16 of which are assistants. The staff members have different availabilities for performing tasks, depending on their working hours and the amount of inner work they have. In the standard case, each staff member is assigned one evening per week and once per five weeks he or she is assigned to work during the weekend. This is normally compensated the week after with two extra days free from work, placed according to the wishes of the staff member.

Let us consider a sample staff member who is a librarian, works full time and is also assigned to evening work on Wednesdays. The staff member is assigned to work weekend on the fourth week and has chosen to take out its days off on Thursday and Friday during the week rest week. The availability for such a staff member is illustrated in Table 1.4. The schedule repeats itself after five weeks and illustrates only the availability for outer tasks. Thus, it does not show if the staff member has been assigned any tasks during these weeks.

All staff members have a five week schedule in the same manner as the sample staff member. However, in order to meet the weekend demands, illustrated in Table 1.2, staff members have to be assigned to weekend work at different weekends. Thus, a staff member's schedule has to be rotated in order to make sure that all weekend demand is fulfilled.

Considering again the sample staff member's schedule in Table 1.4, where the person is available throughout all days, five basic constraints regulate how tasks can be assigned. Firstly, staff members are only allowed to take at a maximum one task per day. This guarantees that staff members also have time for inner tasks. Secondly, staff members are only allowed to perform a maximum of four tasks per week, in order for them to have a day completely free from tasks. Thirdly, staff members can only be assigned to PL at most once a week. Fourthly, there is a limit on how many PL tasks can be assigned to a staff member in total over all ten weeks which varies between different staff members, but is at most four. Lastly, a staff member is not allowed to have the same shift more than twice per week. This guarantees that there is fairness in the schedule, so that no staff member has to work too many times at a shift which is considered to be "bad". An example of a feasible week for the sample staff member is provided in Table 1.5.

The availability of all staff members is provided by the library. Some staff members work differently even and odd weeks, others never work during evenings,

Table 1.4: Availability schedule for a sample staff member. Yellow signifies that the person is available. In parenthesis, the weekend shift.

Week 1	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Week 2	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Week 3	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Week 4	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Week 5	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							

and some only, or never, work weekends. This data was used in order to solve the scheduling problem.

Table 1.5: Example of a feasible week for a staff member.

Week 1	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)		Exp		PL			
10:00-13:00	Exp			PL			
13:00-16:00				PL			
16:00-20:00			Info				

1.1.3 Scheduling objectives: stand-in maximization and five week repetition of the schedule

When we were first presented with the scheduling problem, it was explained to us that the main challenge for the library is to find a good number of stand-ins for all days. After studying their schedule, we observed that only one stand-in was present at the worst day. Therefore, the main objective of this thesis work

is to create a feasible schedule, according to the constraints explained in this section, with as many stand-ins as possible at the worst day (or days).

A staff member is as stand-in for a certain day if he or she is available for outer tasks during the first three shifts of the day and if the person is not scheduled for any shift during that day. Both librarians and assistants can be stand-ins, however, only librarians are qualified to be a stand-in for all tasks. Since the library tasks during regular hours are the most crucial, there is no need for stand-ins during evenings and weekends. Similarly, there are no assigned stand-ins for the library on wheels or Hageby.

Apart from maximizing the number of stand-ins for each day at the library, it is desirable for the sake of the personnel to create schedules that are repeated according to a five week pattern. Thus, the ten week schedule should ideally consist of two five week schedules.

1.2 Aims and goals

The goal of this thesis is to investigate how the ten week scheduling problem for the library staff members at Norrköpings stadsbibliotek can be solved using optimization methods. The goal includes:

1. To investigate what has previously been done in the area of task distribution.
2. To find a mathematical model for the given problem.
3. To implement the model in AMPL.
4. To implement two heuristics for the problem using a large neighbourhood search framework, one using a week block scheduling approach and one using a task scheduling approach.
5. To evaluate and compare the models used in the three implementations as well as their performance.

1.3 Method

The method used for solving the problem, was to first formulate a mathematical model for the problem, which then was solved using two different approaches. The first approach involved modelling the problem in the AMPL programming language and solving it by using the commercial optimization solver CPLEX. The second approach involved creating an independent solver in C++ and solving the problem again using a heuristic. Worker availability data, used to solve the problem, was imported from Excel sheets to data files using Visual Basic.

Two different heuristics were tested in the second approach. The first heuristic distributes week blocks to staff members. All blocks are pre-generated and feasible according to the constraints described in the previous section. In the second heuristic, individual tasks are distributed to the staff members. In order to move through the solution space and find better solutions, both heuristics were based on a large neighbourhood search algorithm.

1.4 Outline

The thesis is divided into six different chapters. The first chapter provides a description of the problem studied as well as the context of the study. In Chapter 2, previous work done in the area is presented. The mathematical model constructed for solving the problem is presented and explained in Chapter 3. In Chapter 4, the different implementations of the model are presented and the results from these implementations are given in Chapter 5. Lastly, conclusions from the thesis work and suggestions for further development are presented in Chapter 6.

1.5 Contributions of the authors

The authors of this thesis have implemented some parts of the work individually and some together. In particular, the mathematical model was constructed and implemented by both authors, whereas the two heuristics were constructed and implemented separately. The week block scheduling heuristic was implemented by Claes while the task distribution heuristic was implemented by Emelie.

In the report, Chapter 1 was mainly written by Emelie. Chapter 2 was divided so that Claes wrote 2.1 and 2.2 Emelie wrote 2.3 and 2.5. Other parts in the chapter were written together. The third chapter was written by Claes and in Chapter 4 and 5, the authors wrote the parts which concerned their heuristic. Chapter 6 was written by both authors.

Chapter 2

Literature review

Staff and machine scheduling problems are mathematical optimization problems which have been studied since the 1950s and concern creating feasible and satisfactory schedules for workers or machines performing tasks. One of the most extensive overviews of the area of staff scheduling is provided by Ernst et al. (2004). They state that, although the complexity of scheduling problems has not increased in recent years, the mathematical models used to solve scheduling problems have become more realistic and refined. Modern staff scheduling problems often concern the distribution of tasks and the creation of worker shifts, also taking softer values into account, such as worker satisfaction and worker fatigue. Due to this modelling refinement, as well as the development of more powerful computational methods, it is possible today to solve staff scheduling problems in a more satisfactory way than before.

In this chapter, staff scheduling problems are classified into different subcategories which are related to the thesis work. The relevant areas for our work include Personnel Task Scheduling Problems (PTSP), Shift Minimization Task Scheduling Problems (SMTSP), Tour Scheduling Problems (TSP), and a few variations of these. Also problems featuring soft constraints will be studied in an additional section. Models and solution methods of these problems will be discussed under each headline and a summary will be provided at the end, together with a discussion about the relevance of the studied articles to our thesis work.

2.1 Personnel task scheduling problem

In many real life situations production managers will face the Personnel Task Scheduling Problem (PTSP) while scheduling service operations. Krishnamoorthy and Ernst (2001) write in their article that the PTSP occurs when the rosterer or shift supervisor need to allocate tasks with specified start and end times to available personnel who have the required qualifications. It also occurs in situations where tasks of fixed times shall be assigned to machines. Each machine have a certain amount of maintenance workers assigned to keep it in operation. Decisions will then have to be made both regarding which machine the workers are assigned to, as well as the optimal amount of maintenance workers needed at each machine.

There are numerous variants of the PTSP. Studies of these have been made by Krishnamoorthy and Ernst (2001), who give a list of attributes that commonly appear in a PTSP, which are listed in Table 2.1. Furthermore, there are traits that always appear in a PTSP; tasks with fixed start and end time are to be distributed to staff members that possess certain skills, allowing them to perform only a subset of the available tasks. Start and end time of their shifts are also predetermined for each day.

One variant, which also is the most simple, is mentioned in Krishnamoorthy and Ernst (2001) and is called the *Feasibility Problem*, where the aim is to merely find a feasible solution. This requires that each task is allocated to a qualified and available worker. It is also required that a worker cannot be assigned more than one task simultaneously, as well as tasks cannot be pre-empted, meaning that each task has to be completed by one and the same worker.

In Table 2.1 attributes of PTSP variants are shown. The nomenclature of the attributes T, S, Q and O refer to the *Task type*, *Shift type*, *Qualifications* and *Objective function* respectively.

Table 2.1: PTSP variants

Attribute	**Type**	**Explanation**
T	F	Fixed contiguous tasks
	V	Variable task durations
	S	Split (non-contiguous) tasks
	C	Changeover times between consecutive tasks
S	F	Fixed, given shift lengths
	I	Identical shifts which are effectively of infinite duration
	D	Maximum duration without given start or end times
	U	Unlimited number of shifts of each type available
Q	I	Identical qualification for all staff (homogeneous workforce)
	H	Heterogeneous workforce
O	F	No objective, just find a feasible schedule
	A	Minimize assignment cost
	T	Worktime costs including overtime
	W	Minimize number of workers
	U	Minimize unallocated tasks

Many of the most basic problems and a few more complex ones can be described with this definition of PTSP attributes. It is, however, not possible to describe all of the numerous types of PTSP using these nomenclatures according to Krishnamoorthy and Ernst (2001).

By combining attributes it is possible to obtain more complex variants of the PTSP. An example would be the PTSP[F;F;H;A-T-W] mentioned in Krishnamoorthy and Ernst (2001), where multiple objectives are used. This problem has fixed contiguous tasks, fixed shift lengths, heterogeneous workforce and three objective functions (A-T-W), which represent assignment costs, work time with overtime included and requirements to minimize the number of workers, respectively. The objective function for this problem is then a linear combination with parameters used to weigh (prioritize) them against each other.

Given the nomenclature above, our problem would be most related to the PTSP[F;F;H;F]. The difference is that the objective function is not empty. We

are looking to maximize the number of qualified stand-ins each day as well as maximize employee satisfaction by meeting their recommendations. We have a fixed number of workers, no costs and no unallocated tasks when a feasible solution is found. Therefore, none of the objective function types given in Table 2.1 are relevant in our case.

Some variants of the PTSP are given specific names in the literature, which is stated by Krishnamoorthy and Ernst (2001). An example is when the shifts and qualifications are identical (S=I, Q=I) and the objective function is to minimize the number of workers that are used (O=W). This variant, PTSP[F;I;I;W], has been published as the Fixed Job Schedule Problem and is described in Section 2.4.

2.1.1 Applications

An example where the PTSP can be found is when developing a rostering solution for ground personnel at an airport. This is mentioned in the article by Krishnamoorthy and Ernst (2001). This problem can be dealt with by first assigning the workers to days in order to satisfy all the labour constraints, followed by assigning the tasks to the scheduled workers.

In the article mentioned above, three further problems of type PTSP, all related to airplanes, are mentioned. They occur when scheduling for either airport maintenance staff, planes to gates or staff that do not stay in one location, such as airline stewards. Scheduling for airport maintenance staff can lead to either PTSP[F;I;H;U-A] or PTSP[F;I-U;H;W], which are similar problems but given two different names: Operational Fixed Interval Scheduling Problem and Tactical Fixed Interval Scheduling Problem respectively. These are both described further in Section 2.4.

Another application, which has been frequently studied, is classroom assignments and is discussed in Krishnamoorthy and Ernst (2001). Based on specifications such as the amount of students in a class or the duration of a class, different classrooms have to be considered. Requirements of equipment, e.g. for a laboratory, may also greatly limit the available classrooms to choose from. A majority of the complications of this problem is due to the fact that lessons can span over multiple periods.

Worth noting for classroom assignment problems is that there are no start or end times for the shifts, as they represent the rooms. The aim in this problem would be to simply find a feasible assignment of classrooms. Therefore, the nomenclature of the problem would be PTSP[S;I;H;F], with the possibility of adding preferences to the objective function. An example of a preference would be to assign the lessons as close to each other as possible on a day, preventing travel distances for teachers and students.

2.2 Shift minimization personnel task scheduling problem

A close relative to the PTSP is the Shift Minimization Personnel Task Scheduling Problem (SMPTSP), which is a variant where the aim is to minimize the cost occuring due to the number of personnel that are used. The same common traits for this problem are mentioned in the article Krishnamoorthy et al. (2012)

as for the PTSP: Workers with fixed work hours are to be assigned tasks with specified start and end times, for which they are qualified.

In the same article as above the authors "... concentrate mainly on a variant of the PTSP in which the number of personnel (shifts) required is to be minimized". In doing so, it is possible to determine the lowest number of staff and the mix of staff a company needs in order to be able to complete the tasks to be operational. They also presumed that the pool of workers is unlimited for each skill group, which is not the case in our problem due to the limitations of librarians and assistants.

Furthermore, it is said in Krishnamoorthy et al. (2012) that SMPTSP can be applied when there is a large number of workers available with different qualifications and it must be ensured that the tasks for that day are performed. The PTSP and SMPTSP are therefore useful day-to-day operational management tools and commonly occur in practical instances where tasks are allocated on a daily basis.

It is shown in Kroon et al. (1995) that SMPTSP is a complex problem even if the preemption constraint were to be removed. However, it is stated in Krishnamoorthy et al. (2012) that if the qualifications of the workers were identical it would be an easily solvable problem. The difference in publication year between the articles indicates that this problem has become less challenging in recent years.

During the last decade, a couple of heuristics have been implemented to deal with the SMPTSP. One method introduced by Krishnamoorthy et al. (2012) is a Lagrangean relaxation approach that combines two problem specific heuristics: Volume Algorithm (VA) and Wedelin's Algorithm (WA). These heuristics exploit the special structure of the SMPTSP by relaxing some of the harder constraints into the objective function, thus being a problem specific heuristic. What remains after the relaxation is a problem decomposed into several problems which can be solved independently. One way to solve these decomposed problems is discussed further in Krishnamoorthy et al. (2012).

Another way to solve the SMPTSP is to use a very large-scale neighbourhood search algorithm, as was presented by Smet and Vanden Berghe (2012). The main purpose of their implemented hybrid local search is to repeatedly fix and optimize to find neighbouring solutions. Using this method on 137 benchmark instances introduced by Krishnamoorthy et al. (2012), Smet and Vanden Berghe managed to find 81 optimal solutions compared to Krishnamoorthy et al. who only managed to find 67. Though, both methods found feasible solutions for 135 out of the 137 problem instances. This comparison is presented in Smet et al. (2014).

Smet et al. (2014) introduced a third and most effective method to solve the benchmark instances related to the SMPTSP. By using a versatile two-phase matheuristic approach, solutions to all 137 benchmark instances could be found for the first time. The procedure used in their implementation is to first generate an initial solution by using a constructive heuristic, followed by improving the solution using an improvement heuristic.

2.3 Tour scheduling problem

The Tour Scheduling Problem (TSP) is described by Loucks and Jacobs (1991) as a combination of shift scheduling and days-off scheduling. Shift scheduling refers to creating sets of contiguous hours during which a worker is assigned for work. The need for days off scheduling typically occurs when the time horizon for scheduling is weekly or more and when weekend staff is needed. Using the notation in Section 2.1, this would be be classified as a variant of the PTSP[F;D;I;W] or PTSP[F;D;H;W], depending on if the workforce is homogeneous or heterogeneous.

According to Loucks and Jacobs (1991), the vast majority of all tour scheduling problems up to 1991 involve a homogeneous workforce, that is, any worker can perform any assigned task. One such early study of a tour scheduling problem is provided by Thompson (1988). The problem studied concerns only homogeneous workforces and the task assignment part is not considered.

In the article by Loucks and Jacobs (1991), the authors study a tour scheduling problem with a heterogeneous workforce. The problem both involves tour scheduling and task assignment, where the latter part is most interesting to us. The problem is studied in the context of fast food restaurants, where certain personnel is qualified only for certain stations in the restaurant. In such industries, the demand of staff differs between weekdays and times of the day. Two worker attributes are considered, their availability for work and their qualifications to perform different tasks. The problem concerns finding shifts for all workers which are to be assigned a length between a minimum and maximum number of hours per day.

The main problem studied in the article involves creating a one-week schedule for 40 workers in a fast food restaurant, available for eight different tasks with a seven-day and 128-hour workweek. Several synthetic problems are studied in the article, all, however, with a minimum shift length of three hours, a maximum shift length of eight hours and a maximum of five work days.

A similar problem to the one descibed by Loucks and Jacobs is studied by Choi et al. (2009). They focus on a particular fast food restaurant in Seoul, which is made a representative of fast food chains in general. In this study, only two types of workers are available, fulltime and part time workers, with no reference to difference in skill. The different shifts are already given by the restaurant managers and the task is to combine them into a tour. The task assignment aspect is lacking in this article.

In both articles the main objective is to minimize both overstaffing and understaffing, which will both have economic consequences for the fast food chain. This is done by reducing or increasing the workforce. For a problem with a fixed workforce, such as ours, this objective function is not relevant. In the example studied by Loucks and Jacobs there is also a goal to meet staff demand on total working hours. This is modeled as a secondary goal which is similar to ours and somehow models a "soft" value, which is of interest to us.

A more recent TSP concern monthly tour scheduling, as opposed to most literature which concerns only weekly scheduling. Such a study was done by Rong (2010). The main advantage of monthly scheduling over shorter time periods, as stated in the article, is the possibility to plan a schedule with respect to fairness and balance over a longer period of time. The problem concerns workers with different skills, where each worker also can possess multiple skills.

This is referred to as a mixed-skill problem. Thus, the problem is similar to our problem, where mixed-skill is also present. In the study, workers have individual weekend-off requirements. The problem does not involve task assignment, which makes it less relevant for us.

The solution methods used to solve the TSP differ greatly between the articles studied. In the older articles, such as Thompson (1988) and Loucks and Jacobs (1991), custom made algorithms very similar to the methods used in manual scheduling are proposed to solve the problem. These solution methods involve classifying staff and distributing them according to some rule (for example, the staff with the most scarce skill is assigned first). General commercial solvers are not proposed, due to their lack of efficiency during these times.

In Hojati and Patil (2011), the same model and data is used as in Loucks and Jacobs (1991). Also this article states that commercial solvers are insufficient for solving the problem. Instead two methods are proposed, one which decomposes the problem into two problems solvable by commercial solvers and one customized heuristic method based on a Lagrangian relaxation method. This method solves the problem with more satisfactory results than in Loucks and Jacobs (1991), as explained in the article. In Choi et al. (2009), a pure integer programming method is used.

2.4 Other similar problems

In this section a couple of other problems, similar to our own, will be described. The focus will not be on the variety of methods used to solve these problem; instead, the focus is to give clarity to how closely related many of these problems are.

2.4.1 Fixed job schedule problem

Variations of the task assignment problem relevant to our problem include for example the Fixed Job Schedule Problem (FJSP). The FJSP has been studied since the 1970s in the context of task assignment in processors. According to Krishnamoorthy et al. (2012), the problem concerns the distribution of tasks with fixed starting and ending times over a workforce with identical skills, such as processing units. Such problems have been solved by Gertsbakh and Stern (1978) and Fischetti et al. (1992).

In the article by Gertsbakh and Stern (1978) a situation is studied where n jobs need to be scheduled over an unlimited number of processors. The objective function of such a problem becomes to minimize the number of machines needed to perform all tasks. Fischetti solves a similar problem, but adds time constraints, saying that no processor is allowed to work for more than a fixed time T during a day as well as a constraint forcing tasks to spread out with time gaps S over a processor.

2.4.2 Tactical fixed interval scheduling problem

The Tactical Fixed Interval Scheduling Problem (TFISP) is a problem very closely related to the SMPTSP, with the sole difference being that the TFISP concerns workers which always are available, such as industrial machines or

processors. The problem is studied in Kroon et al. (1995). A typical TFISP can be expressed using the nomenclature in Table 2.1 and written as PTSP[F;I-U;H;W].

Opposed to the FJSP, the TFISP deals with a heterogeneous workforce. Two different contexts are studied by Kroon et al. (1995). One of them concerns the handling of arriving aircraft passengers at an airport. Two modes of transport from the airplane to the airport are investigated, directly by gate or by bus. The two transportation modes thus correspond to two processing units, which can only handle a number of jobs at the same time.

2.4.3 Operational fixed interval scheduling problem

The Operational Fixed Interval Scheduling Problem (OFISP) is a close relative to the TFISP. Both OFISP and TFISP are restricted by the following: Each machine (worker) cannot handle more than one job at a time, it can only handle a subset of the jobs and preemption of jobs is not allowed. The difference between OFISP and TFISP occurs in the objective function, as described in Kroon et al. (1995). TFISP tries to minimize the number of workers, while OFISP tries to minimize the operational costs and the number of unallocated tasks using priority indices. In the present nomenclature this would give rise to the problem PTSP[F;I;H;U-A].

Given the problem definition above, working shifts are to be created for the workers and tasks have to be allocated on a day-to-day basis. The OFISP, studied by Kroon et al. (1995), can therefore be seen both as a job scheduling problem and a task assignment problem.

2.4.4 Stochastic job problems

What differs mostly between the problem types described above and the problem studied in this thesis work, is the objective function. The main objective function is often to minimize staff for a fixed number of jobs, not taking stand-in assignment into account.

An area where the need for stand-in personnel appears is in the maintenance industry, where some jobs can be foreseen and other (emergency) jobs are of a stochastic nature, that is, there is a probability that such jobs will occur a certain hour. The problem combining both foreseen and stochastic maintenance worker scheduling was studied by Duffuaa and Al-Sultan (1999), as a continuation of the work by Roberts and Escudero (1983).

In the former article, a fixed heterogeneous workforce consisting of electricians, plumbers and mechanics is studied. The shifts of the personnel are predetermined by their given work times and thus the problem becomes a pure task assignment problem. The goal is to maximize the number of planned and unplanned jobs performed by the workers, by taking into account the probability of unplanned work to occur. Thus, certain workers will be left at the station as stand-ins in the case an unplanned job arises. The commercial solver LINDO was used to solve the problem.

2.5 Modelling soft constraints

For most scheduling problems, the main objective is to minimize worker-related costs by reducing the number of workers needed to perform a task, or by reducing the working time for part-time employees. Recently, however, many studies have started to focus more on softer goals such as worker satisfaction as an objective function. Such goals are usually considered when scheduling is done manually, but have been forgotten or set aside in mathematical modelling.

In an article by Akbari et al. (2013) a scheduling problem for part-time workers with different preferences, seniority level and productivity is investigated. In this article, these aspects are reflected in the objective function and weighted against each other. A similar problem was also studied by Mohan (2008), but for a workforce of only part-time workers.

Other factors which may affect worker satisfaction, and in the long run efficiency and presence at work are fatigue, fairness and boredom. These are discussed by Eiselt and Marianov (2008). Repetitiveness of a job as well as the level of challenge can cause boredom of workers. The variance in the schedule is increased by Eiselt and Marianov (2008) through providing an upper bound of how many tasks can be performed in a given time span. The article suggests a sort of measurement of the distance between the task requirements and the worker abilities used. This will then be minimized in the objective function.

Another modelling method which is relevant specifically for scheduling problems featuring soft constraints is fuzzy goal programming. The method is discussed by Shahnazari-Shahrezaei et al. (2013), who model soft constraints as "fuzzy goals". These goals can become contradictory, for example could a preference of high seniority level workers come in conflict with a preference in working hours by an employee. The article uses fuzzy set theory, and a solution approach involving Li's two-phase method (Li, 1990). Soft constraints are modeled as trapezoid functions and an optimal solution with the best average value of all functions is found.

The solution methods proposed by Akbari et al. (2013) to solve a scheduling problem with soft constraints are two matheuristics: Simulated Annealing (SA) and Variable Neighbourhood Search (VNS). According to Akbari et al. (2013), SA has been studied as a solution method for the scheduling problem since the 1990s and many studies have shown that it is capable of providing near-optimal solutions in a short time compared to optimizing integer programming models, for a variety of problems. An exponential cooling time was used for the algorithm and it was concluded that it was faster than the commercial solver LINGO in finding a solution.

VNS is the other proposed method by Akbari et al. (2013). A big difference between this and other methods is that VNS requires very little parameter tuning, while often providing good solutions. Larger movement sizes are used at higher temperatures, and smaller at lower temperatures. The method uses both a random and a systematic phase. In the random phase, worker schedules are regenerated randomly and better solutions are saved. In the systematic phase, two shifts are swapped. In the article, it was concluded that VNS could solve the problem faster than the implemented SA heuristic.

In the article by Eiselt and Marianov (2008), a commercial solver was used. This was also the case in Mohan (2008), as the article compares these commercial solver results with the results obtained from a branch-and-cut algorithm. Also

Shahnazari-Shahrezaei et al. (2013) finds results using a commercial solver.

2.6 Summary

In order to get an overview over the problems discussed in this chapter it is a good idea to look at general historical trends among the problems. One clear trend is the shift from problems concerning homogeneous to heterogeneous workforces. As stated by Krishnamoorthy et al. (2012), the problem of heterogeneous workforces was trivial at the publishing year, while in Loucks and Jacobs (1991), it is introduced as a rather new and challenging concept. Furthermore, the articles concerning the different PTSPs are mainly from the 1980s and 1990s and were challenging in their structure during these times. Some of them, such as FJSP, are related to the scheduling in processors, which was a hot topic at the start of the computer era. All the articles about modelling of soft constraints are presented after the year 2000, probably as a result of better computational powers.

The various solution methods found in the studied articles include commercial solvers, heuristics and matheuristics. Heuristics which have been studied in the articles include SA, VNS and Lagrangian relaxation, with and without VA. Also some local search-based heuristics have been used. Commercial solvers are used increasingly in newer articles such as Hojati and Patil (2011) and Mohan (2008), probably due to the improvement in performance in such solvers. Similarly, matheuristics are also discussed mostly in more recent articles, such as Akbari et al. (2013).

2.7 Relevance to our problem

As described in Section 2.1, many different types of personnel tasks scheduling problems exist. Only a few of them have been discussed in this chapter, of which some are closer related to our problem than others. In order to get a better understanding for how they are related to our problem, a few connections to them will be presented in this section.

Using Table 2.1, the closest classification of the problem studied in this thesis work would be the PTPS[F;F;H;F]. This describes a Personnel Task Scheduling Problem with fixed tasks, fixed shift lengths, a heterogeneous work force and an empty objective function. As stated in Section 2.1, the main difference lies in the objective function, as ours is not empty. Thus, many problem types discussed in this chapter concern problems which are similar to ours but with a different objective function. This includes the shift minimization personnel task scheduling problem, the tour scheduling problem, the fixed job schedule problem, the tactical fixed interval scheduling problem and the operational fixed interval scheduling problem. The first two are relevant to our problem as they concern task assignment and, in the case of tour scheduling problem, days off scheduling. However, both involve shift scheduling while we have fixed shifts. The other three problems concern only task assignment, but are otherwise further from our problem since the tasks types are different from ours.

As stated in Section 2.1, our problem has an objective function in which the minimal number of stand-in personnel is to be maximized. Such problems

often arise in the maintenance job scheduling problem, which is discussed in Section 2.4.4. As a secondary objective function, we are also interested in making the schedule varied and with a repeating pattern over a cycle of five weeks. Modelling of such constraints is discussed in Section 2.5.

Chapter 3

The mathematical model

In this chapter the mathematical model implemented to solve the ten week scheduling problem described in Section 1.1 will be presented. Prior to the objective function and constraints, the most significant sets and variables will be provided. Section 3.2 presents the objective function and gives a short description of what it represents. In Section 3.3, the essential constraints will be presented and explained. A complete model with all definitions and the full set of constraints can be found in Appendix A.

3.1 Set and variable definitions

To solve the problem, many sets and variables have to be declared. As mentioned before, there are many unique and individual requirements given by the library that have to be met. An example is that some staff members want a day free from outer tasks to be able to attend meetings or to perform their inner tasks. Another example is that some staff members have two alternating schedules for odd and even weeks. These specific cases have to be modelled and result in a variety of sets and variable definitions. Below, we focus on the common and general cases in order to keep the presentation simple and easily accessible. A complete list of the definitions can be found in Appendix A.

I	Set of staff members
I_{lib}	Set of librarians ($I_{lib} \subseteq I$)
I_{ass}	Set of assistants ($I_{ass} \subseteq I$)
$I_G\{g\}$	Set of workers in the group g
W	Set of all ten weeks
W_5	Set of first five weeks
D	Set of all days in a week
D_5	Set of all five weekdays
S_d	Set of shifts available on day d
S_3	Set of first three shifts on a weekday
J_d	Set of task types available on day d
G	Set of groups
V	Set of possible week rotations (shifts the week by 0-9 steps forwards)

In order to further define the problem we introduce the following variables.

$$x_{iwdsj} = \begin{cases} 1, & \text{if staff member } i \text{ is assigned to a task } j \text{ in week } w, \text{ day } d, \text{ shift } s \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$H_{iwh} = \begin{cases} 1, & \text{if staff member } i \text{ works weekend } h \text{ in week } w \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

$$r_{iw} = \begin{cases} 1, & \text{if staff member } i \text{ has its schedule rotated } w\text{-}1 \text{ steps forwards} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

$$l_{iwd} = \begin{cases} 1, & \text{if librarian } i \text{ is a stand-in week } w, \text{ day } d \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

$$a_{iwd} = \begin{cases} 1, & \text{if assistant } i \text{ is a stand-in week } w, \text{ day } d \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

$$y_{iwds} = \begin{cases} 1, & \text{if staff member } i \text{ works at task type E, I or P in week } w, \text{ day } d, \text{ shift } s \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

$$W_{iwd} = \begin{cases} 1, & \text{if a staff member } i \text{ is working a shift in week } w, \text{ day } d \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

$$b_{iw} = \begin{cases} 1, & \text{if staff member } i \text{ works at HB in week } w \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

$$f_{iw} = \begin{cases} 1, & \text{if staff member } i \text{ is assigned to work Friday evening in week } w \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

$$M_{wds} = \begin{cases} 1, & \text{if a big meeting is placed in week } w, \text{ day } d, \text{ shift } s \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

$$m_{wdsg} = \begin{cases} 1, & \text{if a meeting is assigned for group } g \text{ in week } w, \text{ day } d, \text{ shift } s \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

$$d_{iwds} = \begin{cases} 1, & \text{if there is a difference for a staff member } i \text{ in assignment of tasks at} \\ & \text{shift } s, \text{ day } d \text{ between weeks } w \text{ and } w\text{+}5 \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

$$l^{min} = \text{lowest number of stand-in librarians found (integer)} \tag{3.13}$$

$$a^{min} = \text{lowest number of stand-in assistants found (integer)} \tag{3.14}$$

$$s^{min} = \text{weighted sum of numbers of stand-in librarians and assistants} \tag{3.15}$$

$$\delta = \text{total number of shifts that differ for all staff members (integer)} \tag{3.16}$$

Based on the variables defined above it has been possible to solve our scheduling problem. The variables l^{min} and a^{min} are the ones of most significance, as they represent the number of stand-ins found after a run.

3.2 Objective function

As the problem contains multiple objective functions, it has been necessary to weigh them against each other using parameters. These are shown in Equation 3.17, as M and N.

$$\text{maximize} \quad M \cdot s^{min} - N \cdot \delta \tag{3.17}$$

The first part of the objective function represents the lowest amount of stand-in librarians and assistants found, while the second part is a preference from the library that two weeks with a five-week interval shall be as similar as possible (for example week 1 and 6 or 2 and 7). The parameter N prioritizes the similarity of weeks compared to the number of stand-ins. Based on the information given by the library, it is a much higher priority to have many stand-ins. Hence, $M \gg N$, where $N > 0$ holds in our case. The exact relation between M and N is not of high importance, as we have seen that multiple optimal solutions exist, regarding the stand-ins. In our case, we set M to be a hundred times bigger than N.

3.3 Constraints

To model the problem it is of relevance to divide many of the constraints into weekend and weekday constraints. Several auxiliary constraints and variables are also added to avoid multiplication of two variables, which would make the problem non-linear. These auxiliary constraints are needed as the solver CPLEX can not handle non-linear constraints, as it displays error messages in case they occur. The auxiliary constraints are left out of this chapter for simplicity reasons. Instead, they can be seen in Appendix A.

3.3.1 Demand and assignment constraints

The most crucial constraint is to ensure that the demand of staff members is met each day. This can be modelled as

$$\sum_{i \in I} x_{iwdsj} = demand_{wdsj}, \; w \in W, d \in D, s \in S, j \in J_d \tag{3.18}$$

Here, $demand_{wdsj}$ is an integer representing the number of staff members required in week w, day d, shift s for a task j.

The following constraint says whether or not a staff member is assigned a task during a weekday. Evening shifts and library on wheels tasks are excluded.

$$y_{iwds} = \sum_{j \in J_d \setminus \{L\}} x_{iwdsj}, \; i \in I, w \in W, d \in D_5, s \in S_3 \tag{3.19}$$

This variable assignment is used to simplify a couple of constraints later on.

To ensure that no staff members are assigned more than one task the following constraint is implemented.

$$\sum_{s \in S} \sum_{j \in J_d} x_{iwdsj} \leq 1, \ i \in I, w \in W, d \in D \tag{3.20}$$

However, if we allow a person to have two shifts at the library on wheels on a day, Equation 3.20 has to be slightly modified. This is left out of this chapter, as it is allowed in the complete model.

It is preferred to allow only one PL per week and a maximum of three PL per ten weeks. These are easily modelled with the following constraints.

$$\sum_{s \in S_d} \sum_{d \in D} x_{iwdsP} \leq 1, \ i \in I, w \in W \tag{3.21}$$

$$\sum_{w \in W} \sum_{s \in S_d} \sum_{d \in D} x_{iwdsP} \leq 3, \ i \in I \tag{3.22}$$

The long duration of a PL (in the equations named "P"), is the reason for these preferences. Some staff members are required to have some time free from outer tasks, to be able to perform their inner tasks.

Another preference is to have varying work hours when it comes to the assignment of tasks. Then the more and less desired shifts will be fairly distributed. The following equation is implemented to meet such requirements.

$$\sum_{d \in D_5} y_{iwds} \leq 2, \ i \in I, w \in W, s \in S_3 \tag{3.23}$$

Equation 3.23 allows a staff member to have at most two tasks starting at the same hour every week. Worth noting is that library on wheels is disregarded in this constraint as the variable y_{iwds} is used; see Equation 3.19 for the definition.

It is desirable to avoid assigning too many tasks to a staff member. The reason for this is, as mentioned previously, to let the staff member have some time to allot for inner services during weekdays. The equation below models this preference.

$$\sum_{d \in D_5} \sum_{s \in S_d} \sum_{j \in J_d} x_{iwdsj} \leq 4, \ i \in I, w \in W \tag{3.24}$$

Equation 3.24 allows a staff member at most four weekday shifts per week. The exception, which is one of the BokB staff members, is left out of this equation for simplicity reasons. To model that, a new subset of I needs to be created where that staff member is left out.

3.3.2 Weekend and rotation constraints

Every staff member's schedule is able to rotate up to nine times, where the decision variable r_{iv} decides the rotation. Therefore, the parameter $qualavail_{iwdsj}$ has to align with the rotation so that all staff members are assigned tasks only when available. This is provided in the following equation.

$$x_{iwdsj} \leq \sum_{v \in V} r_{iv} * qualavail_{i\omega(v,w)dsj}, \ i \in I, w \in W, d \in D, s \in S_d, j \in J_d \tag{3.25}$$

The function $\omega(v, w)$ represents the modulus function $mod_{10}(w - v + 10) + 1$, which is a function that takes rotations into the account when calculating the week to look for in the availability matrix. Table 3.1 gives a better understanding of how this modulus function is used.

The variable *w*, in Table 3.1, represents the current week in the initial schedule, and *v* represents *v-1* rotations to the right from the initial schedule. With initial schedule, we refer to the schedule, where all staff members' weekends occur in the first week. The resulting week, when rotations have been taken into the account, can be seen inside the cells. When scheduling, it is the resulting week ω that is of interest, when checking if a staff member is available for a task assignment.

Table 3.1: Resulting table of the function $\omega(v, w) = mod_{10}(w - v + 10) + 1$. Here, w is the week in the initial schedule, v is the rotation variable. Inside the cells, the resulting week is shown when rotations have been made.

		w =									
		1	2	3	4	5	6	7	8	9	10
v =	1	1	2	3	4	5	6	7	8	9	10
	2	10	1	2	3	4	5	6	7	8	9
	3	9	10	1	2	3	4	5	6	7	8
	4	8	9	10	1	2	3	4	5	6	7
	5	7	8	9	10	1	2	3	4	5	6
	6	6	7	8	9	10	1	2	3	4	5
	7	5	6	7	8	9	10	1	2	3	4
	8	4	5	6	7	8	9	10	1	2	3
	9	3	4	5	6	7	8	9	10	1	2
	10	2	3	4	5	6	7	8	9	10	1

An example to better understand the function: imagine that we are looking at an unrotated initial schedule where everyone is assumed to work weekend the first week. Say that we look at a staff member's first week, $w = 1$. If this schedule is rotated two times to the right ($v = 3$), then the first week in the new rotated schedule represents the previous ninth week, that is $w = 9$ from the initial schedule. The availability to look at, in the initial schedule is, therefore, week nine.

Initially, all staff members are assigned weekend work the first week. Then, a week rotation variable is introduced that rotates the staff members' availability matrices so that the demand constraints on evenings and weekends can be met. The most basic constraints, regarding which week a staff member's weekend work occurs and its week rotation, can be seen below.

$$\sum_{w \in W} r_{iw} = 1, \ i \in I \tag{3.26}$$

$$\sum_{w \in W} H_{iwh} \leq 1, \ i \in I, h \in \{1, 2\} \tag{3.27}$$

$$r_{iw} \geq H_{iw1}, \ i \in I, w \in W \tag{3.28}$$

Equation 3.26 provides all staff members with a rotation of their schedule regardless if they are working weekends or not. Equation 3.27 allows a staff member a maximum of two weekends, for $h = 1$ and $h = 2$, per ten weeks.

Once a staff member is assigned a new rotation, the availability matrix has to be correctly rotated. This is done using Equation 3.28. In case a staff member is never due for weekend work, that is, $H_{iwh} = 0,\ \forall i \in I, w \in W, h \in \{1, 2\}$, then the rotation r_{iw} is free, as its availability matrix is identical for all weeks.

A staff member is supposed to work weekends with a five-week interval. However, in case there are enough staff members to satisfy the demand on weekends, it may be enough for some to only work one weekend per ten weeks. To avoid problems with the rotation if only the second weekend is assigned to a staff member, the following equation is used.

$$r_{i(mod_{10}(w+4)+1)} \geq H_{iw2},\ i \in I, w \in W \tag{3.29}$$

Worth noting is that if a staff member is assigned both weekends, then Equation 3.29 provides the same information as Equation 3.28.

A staff member is supposed to work with the same task both Saturday and Sunday when working a weekend. This can be modelled with the following constraints.

$$\sum_{j \in J_d} x_{iw61j} + \sum_{j \in J_d} x_{iw71j} = 2 * \sum_{h=1}^{2} H_{iwh},\ i \in I, w \in W \tag{3.30}$$

$$x_{iw61j} = x_{iw71j},\ i \in I, w \in W, j \in J_d \tag{3.31}$$

Equation 3.30 ensures that the staff member will work Saturday and Sunday consecutively, whenever he or she is due for weekend work. To make sure it is the same task as well, Equation 3.31 is implemented.

Friday evening is also included in an extension to working Saturday and Sunday, unless the staff member is assigned to HB. It is, however, not a necessity to perform the same task Friday evening as during the weekend, thus Fridays are not included in Equation 3.31. Equation 3.32 below adds Fridays to the weekend.

$$\sum_{j \in J \setminus \{L\}} x_{iw54j} = f_{iw},\ i \in I, w \in W \tag{3.32}$$

Here, "L" refers to the task type library on wheels. The variable f_{iw} is, as mentioned in the variable declaration, equal to one if and only if a staff member is working weekend as well as not being assigned to HB.

It is of interest to implement a constraint to prevent staff members from being assigned to HB more than once every ten weeks in order to avoid unfairness. Why this is preferable is described in Section 1.1.1. This constraint can be modelled as.

$$\sum_{w \in W} \sum_{d=6}^{7} x_{iwd1B} \leq 2,\ i \in I \tag{3.33}$$

Here, "B" refers to HB. Equation 3.31 and 3.33 ensure both that a staff member only can be assigned to HB two days per ten weeks, and that those days are consecutive. The exception is in case a staff member only works in HB, which is disregarded in this simplfied model.

3.3.3 Objective function constraints

The variable δ in the objective function, Equation 3.17, is defined by the following equation.

$$\delta = \sum_{i \in I} \sum_{w \in W_5} \sum_{d \in D_5} \sum_{s \in S_3} d_{iwds} \tag{3.34}$$

This equation gives an integer value, which represents the total number of shift differences that occur for all workers, through all weekdays, in the library. The variable d_{iwds}, defined in Equation 3.36 below, compares task assignments between two five-week separated weeks.

To calculate the variable s^{min} used in the objective function, 3.17, the following equation is used.

$$s^{min} \leq L \cdot \sum_{i \in I_{lib}} l_{iwd} + A \cdot \sum_{i \in I_{ass}} a_{iwd},\ w \in W, d \in D_5 \tag{3.35}$$

Here, l_{iwd} and a_{iwd} are binary variables stating whether a staff member is a stand-in on a day or not. Since s^{min} is being maximized in the objective function, it will assume the lowest value of the sum of stand-in librarians and assistants for any day during the ten weeks. It will, therefore, represent the worst solution found regarding stand-ins. Just as for previous equations, auxiliary constraints have been left out for simplicity reasons. Here, it is left out how l_{iwd} and a_{iwd} are determined.

If $L < A$ holds in the model, the solver would prioritize assistants over librarians as stand-ins. Librarians are, however, more desired as stand-ins, due to their ability to perform all types of tasks. Therefore, it is desired to let $L \geq A$.

As stated in Section 3.2, two weeks with a five-week interval should be as similar as possible. Equation 3.36, together with Equation 3.34 and the objective function equation, 3.17, provides this preference.

$$d_{iwds} = |y_{iwds} - y_{i(w+5)ds}|,\ i \in I, w \in W_5, d \in D_5, s \in S_3 \tag{3.36}$$

The decision variable d_{iwds} states if there is a difference in assignment between two tasks at the same hour and day for the two weeks, w and $w + 5$. The differences occur if, say, an Exp task is assigned on Monday week one at 8 a.m. to 10 a.m. and no task is assigned the same shift Monday week six. Thus, a variable that is minimized in the objective function.

3.3.4 Meeting constraints

At the library there are both library meetings and group meetings, which both occur with a five-week interval. Library meetings are set to take place from 8 a.m. to 10 a.m. on Mondays, whereas group meetings are more freely distributed. A few staff members are not assigned library meetings, as they are needed in the library to keep it running. The set I_{big} in the equation below consists of all staff members who are to be assigned library meetings. The constraints modelling library meetings are as follows.

$$\sum_{w \in W_5} M_{w11} = 1 \tag{3.37}$$

$$M_{(w+5)11} = M_{w11},\ w \in W_5 \tag{3.38}$$

$$\sum_{s=1}^{3} \sum_{j \in J_1 \setminus \{L\}} x_{iw1sj} \leq 1 - M_{w11},\ i \in I \setminus I_{big}, w \in W \tag{3.39}$$

Equation 3.37 and 3.38 assign two library meetings with a five-week interval during the ten-week scheduling period. Equation 3.39 makes sure that the staff members, that do not attend library meetings, are not assigned any other task during the day of the meeting. This implicitly force the staff members, that do not attend the library meetings, to meet the library demand constraints during the meeting.

The constraints added to model the group meetings are somewhat similar to the library meeting constraints. Just as for library meetings, they take place two times during the ten weeks with a five-week interval, which is described by Equations 3.40 and 3.41.

$$\sum_{w \in W_5} \sum_{d \in D_5} \sum_{s \in S_3} m_{wdsg} = 1,\ g \in G \tag{3.40}$$

$$m_{(w+5)dsg} = m_{wdsg},\ g \in G, w \in W_5, d \in D_5, s \in S_3 \tag{3.41}$$

$$m_{wdsg} + x_{iwdsj} \leq 1,\ g \in G, i \in I_G\{g\}, w \in W, d \in D_5, s \in S_3, j \in J_d \tag{3.42}$$

$$m_{wdsg} \leq \sum_{v \in V} r_{iv} * qualavail_{i\omega(v,w)dsE},\ g \in G, i \in I_G\{g\}, w \in W, d \in D_5, s \in S_3 \tag{3.43}$$

Equation 3.42 prohibits a staff member from multitasking, that is, to attend a meeting and work with a task simultaneously. Equation 3.43 enables group meetings only when everyone in that group is available. The rotation is also taken into the account.

In Equation 3.43 ω is the week calculation when rotations are taken into the account, see Section 3.3.2. The task type "E" stands for Exp, and is used due to it is the only task type everyone is available for, as *qualavail* is dependent on the task variable j. A better way to model this would be to create a new parameter, say $avail_{iwds}$, that is not dependent on staff member qualifications. As this way of modelling works, this was not implemented in order to to save us some time.

Chapter 4

Two heuristic solution methods

Three different approaches to the problem were studied. The first part of the thesis was to implement the problem in AMPL, according to the model described in Chapter 3, and the supplementary details given in Appendix A. Running this model in CPLEX gives an optimal solution to the problem. The other two approaches are classified as heuristics and are implemented in C++. These are implemented in order to solve the problem without using expensive software, such as CPLEX, although they do not guarantee that the optimal solution is found.

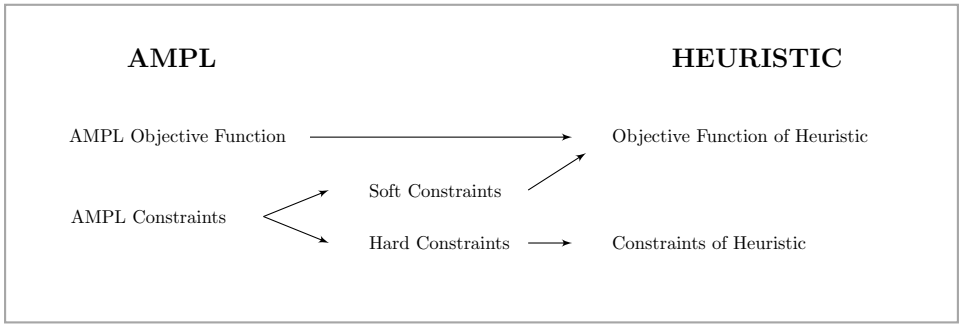
The first heuristic is a week block scheduling heuristic, which creates a solution from distributing complete week blocks to staff members. This heuristic first creates a large pool of unique week appearances. In this pool, there is one block for every possible combination of tasks for any of the staff members. After the pool is created, each staff member is assigned one block from this pool for each scheduled week.

In the second heuristic, weekends are first assigned to staff members, creating a weekend constellation that is aiming for an even distribution of stand-ins. After this, the algorithm enters a second phase, where weekday tasks are allocated according to a greedy heuristic.

In order to implement the heuristics, some of the mathematical model's constraints are softened. Softened constraints are constraints that are considered as hard in the original model, but are placed in the objective function in the new model. In the new model, violating a softened constraint gives a penalty cost and having no such penalty cost is equivalent to having a feasible solution. This process of constraint softening is illustrated in Figure 4.1.

Due to a lack of time, the two heuristics were implemented using a simplified model, where some constraints from the original model have been relaxed or removed. How the heuristics differ from the original model is described in Tables 4.1 and 4.2, together with the soft constraints. Although the problem is simplified in both heuristics, its most fundamental structure is preserved, since the most significant degree of freedom in the problem is given by the weekend allocation. The weekend allocation affects where the staff members have their week rest, which in turn affects the distribution of stand-ins in the schedule.

Figure 4.1: Illustration of the differences between the model used in AMPL and the heuristic model



Therefore, the main focus of both heuristics is the weekend allocation.

Relaxed constraints are constraints which have been slightly altered from the original model. In both heuristic implementations, these are identical. The relaxed constraints concern the even and odd week availability issue. In the heuristics, the model becomes independent of the even and odd week constraints by reducing the BokB schedule to a fixed schedule, and removing differences in the odd and even week availability for the staff members in question. In addition, solving a five week scheduling problem rather than a ten week problem simplifies the problem further.

As implementation time was lacking, meetings were removed from the model in the heuristic implementations. This was a natural choice, as meetings are not currently in the library's schedule and since an extra feature added to the mathematical model upon the library's request.

Both heuristic implementations are based on a Large Neighbourhood Search (LNS) framework. This method, which is classified as a metaheuristic, works by alternately destroying and repairing parts of solutions, in order to move through the solution space. Typically, the parts of the solution that are considered poor are destroyed. How much that is destroyed is regulated by the destroy degree. The repair function rebuilds the destroyed part of the solution using some type of greedy heuristic. The LNS method is described more in detail by Pisinger and Ropke (2010).

The second heuristic also uses Simulated Anneahling (SA), which allows the algorithm to accept solutions which are poorer than the current one. This makes it possible for the search to move out of a local optima. A variable called a "temperature" is used in order to decide the probability of accepting a poorer solution. Usually, this temperature decreases over time, and thus the probability of accepting a poorer solution also decreases. The method is described more thoroughly by Pisinger and Ropke (2010).

4.1 Week block scheduling approach

The first heuristic approach creates initially a large pool of week block appearances. These are then filtered for every staff member based in its qualities, such

Table 4.1: Showing changes in the model for the week block scheduling approach. The soft, relaxed and removed constraints are shown.

Soft Constraints	
Affected constraints	Constraint description
3.18	The number of staff members needed for every shift and task type in the library.
3.22	Maximum number of PL per staff member and week cycle.
Relaxed Constraints	
Affected constraints	Constraint relaxation
Several. $W = W_5$ in all constraints.	Five week scheduling instead of ten week scheduling.
BokB-constraints	BokB manually assigned, due to low degree of freedom.
Availability data	Even and odd week staff members have availability at each shift according to the stricter of the two sets.
Removed Constraints	
Affected constraints	Constraint description
3.36	Any two weeks w and $w+5$ shall be as similar as possible.
3.37 - 3.43	Meetings are not implemented.

as availability and qualification. Then, a large neighbourhood search is applied by alternating between a destroy and repair on a given number of staff members. In this case three staff members are destroyed per iteration. A critical step during a destroy and repair phase is to be able to swap rotations between staff members. To swap rotation means that staff members are scheduled for weekend work another week.

The implemented heuristic can be seen in Figure 4.2. The LNS framework is shown in the middle of the figure.

In case a run does not find a feasible solution after a given number of iterations, the run is discarded and a new iteration is executed. This is shown on the right side of Figure 4.2. In case a run is close enough to enter the final phase, a check is made if there are enough stand-ins to be assigned the remaining tasks. If not, all staff members will be assigned empty week blocks, so that a new try of finding a feasible solution can be made.

Every staff member's information such as availability and qualification is inserted into an Excel table. It is then written to a text file using a Visual Basic code, which in turn is read and used by the heuristic.

In the following sections the implemented week block approach heuristic will be explained step by step.

4.1.1 Block creation

A major part of this heuristic is to create the large pool of possible week block appearances. These are then filtered for each staff member based on their availabilities. The staff members' availability are generalized into three categories:

Table 4.2: Showing changes in the model for the task allocation approach. The soft, relaxed and removed constraints are shown.

Soft Constraints	
Affected constraints	Constraint description
3.20	At most one task per day.
3.24	At most four tasks per week.
3.21	At most one PL per week.
3.22	Maximal number of PL per ten weeks.
3.23	Not more than two tasks at the same shift in a week restriction.
Relaxed Constraints	
Affected constraints	Constraint relaxation
Several. $W = W_5$ in all constraints.	Five week scheduling instead of ten week scheduling.
BokB-constraints	BokB placed according to constraints, but the schedule is fixed.
Availability data	Even and odd week staff members have availability at each shift according to the stricter of the two sets.
Removed Constraints	
Affected constraints	Constraint description
3.36	Any two weeks w and $w+5$ shall be as similar as possible.
3.37 - 3.43	Meetings are not implemented.

weekend, week rest and weekday week. Worth noting is that the weekday week occurs three times during a five week period, see Table 1.4.

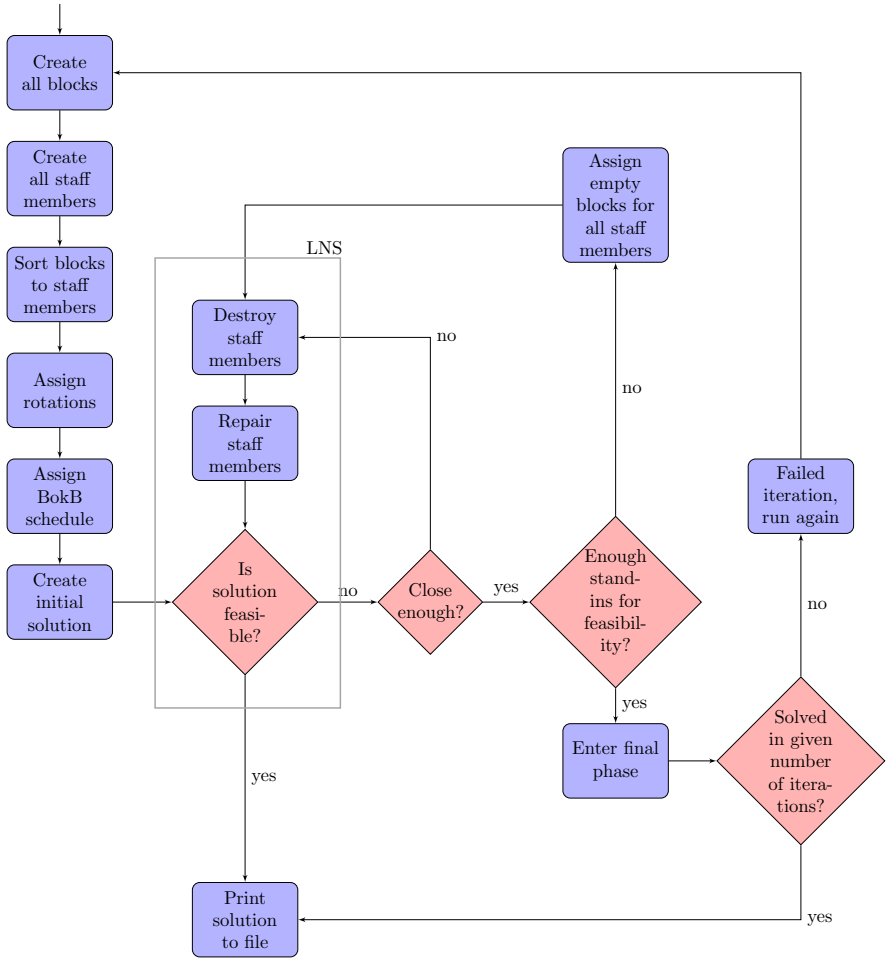
One week block contains seven days and up to four shifts where each shift can contain at most three different tasks. Table 4.3 below is a representation of a general week block with all possible tasks for each day d and shift s. In the table, I represents that "No task" is assigned that day, D represents "Desk task" meaning either Exp or Info, PL represents "Fetch list" and HB represents "Hageby".

Table 4.3: A general week block with all possible tasks

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00	I,D,PL	I,D,PL	I,D,PL	I,D,PL	I,D,PL	I,D,HB	I,D,HB
10:00-13:00	D	D	D	D	D		
13:00-16:00	D	D	D	D	D		
16:00-20:00	D	D	D	D	D		

Every day must contain exactly *one* task from either of the four shifts when creating a week block. The tasks I and PL are ranging over more than one shift. The duration of a PL is three shift and I refers to the entire day. They are both placed in the first shift to simplify the complete task representation. When creating the combinations of block appearances there are additional conditions that have to be met. These are:

Figure 4.2: A flow chart of the implemented heuristic with week block scheduling



1. At most two tasks per week can be assigned the same shift.
2. At most one evening is allowed per week, if Friday evenings are excluded.
3. At most one PL is allowed in a week block.
4. Saturday and Sunday shall always contain the same type of task.
5. If Saturday and Sunday contain Desk tasks, then so shall Friday afternoon (fourth shift).
6. No more than four tasks are allowed during the weekdays, leaving at least one day free from tasks.

To illustrate the growth of the number of block combinations when more tasks are added, consider the following example. If items 1, 2, 3, 5 and 6 are disregarded there exists $6^5 * 3 = 23,328$ unique week blocks. In contrast, if

Exp and Info were to be considered separately, instead of using the combination of the two, the possible combinations would be $10^5 * 4 = 400,000$. By applying all conditions above the total number of unique block appearances for this implementation are only 4,175.

An illustration of one of the 4,175 existing blocks can be seen in Table 4.4 below.

Table 4.4: Illustration of one of the unique block appearances

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00		PL	I		I	HB	HB
10:00-13:00							
13:00-16:00				D			
16:00-20:00	D						

This block contains five tasks; two of them are weekend tasks and three are weekday tasks. Which weeks that this block can be assigned to is dependent on the staff member's rotation. Since Hageby is assigned to the week block, one can conclude that only a librarian can have this block assigned to itself. Due to this fact, the Desk tasks can mean either Exp or Info, as librarians are qualified for both. In case it was an assistant, the Desk tasks can only refer to Exp.

4.1.2 Block filtering

After creating all possible week block appearances, they are filtered for each of the staff members based on their availability in each of the three categories, mentioned in Section 4.1.1. Table B.1 in Appendix B shows the number of available week block appearances with respect to the number of possible week blocks, after the filtering has been made for all staff members.

All of the values in the table are fractions of the total number of 4,175 blocks. Knowing the structure of the problem, one can deduce that the number of available week rest blocks shall always be less than or equal to the available weekday blocks. The reason is because the only difference between the two mentioned block categories, is when a staff member is free from work due to its week rest. Hence, they are equally many when a staff member is never working weekends.

Looking at a typical staff member's availability, shown in Table 4.5, one might think that there shall be more weekend blocks available than weekday blocks, as the availability, almost in all cases, is higher for weekend blocks. This is not the case, however, as all blocks without weekend tasks are filtered out from the weekend block category. This means that all combinations with "No task" on weekends are excluded in that category. Furthermore, the case shown in Table 4.6 decreases the number of existing weekend blocks further. This case occurs when a staff member is assigned weekend Desk tasks and, therefore, can not be assigned any other task that Friday, as only one task is allowed per day.

The notion "X" in Table 4.6 can represent any task, and shifts colored in black mean that no other task can be assigned that shift.

Table 4.5: Typical availability for a five-week period for a staff member. Yellow signifies that the staff member is available during the stated hours. In parenthesis, the weekend shift hours are stated.

Weekend week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Week rest week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Weekday week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Weekday week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							
Weekday week	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
08:00-10:00 (11:00-16:00)							
10:00-13:00							
13:00-16:00							
16:00-20:00							

Table 4.6: A weekend block with Desk tasks preventing any other tasks on Fridays.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00	X	X	X	X		D	D
10:00-13:00							
13:00-16:00							
16:00-20:00					D		

4.1.3 Rotation assignment

The rotation assignment is an early step in the destroy and repair process. Every staff member has a rotation assigned, stating which week the weekend work occurs. In the rotation assignment this rotation is assigned to a staff member based on a few constraints. In a destroy and repair iteration the three randomly chosen staff members can have their rotations swapped. The constraints and an illustration of a swap will be presented here.

There are 35 weekend staff members available, out of which 21 are librarians and 14 are assistants. The demand for weekend staff members each week is

seven, that is, the demand for five weeks is exactly $7 * 5 = 35$ staff members. Another requirement is, in addition to the seven staff members each weekend, that at least four of them have to be librarians. The reason is because three librarians are needed in the Information desks and one in Hageby. Because of the exact balance between supply and demand, it deems reasonable to swap rotations between staff members in the destroy and repair loop, so that there always are enough qualified staff members every weekend.

The swaps are done using a random generator in the assignment of rotations, such that the above described weekend staffing requirements always are met. However, all of the BokB-staff members have fixed weekends; hence, they are not given new rotations in the destroy and repair loop. This is more thoroughly described in Section 4.1.4 below.

Table 4.7 shows a destroy and repair iteration regarding rotation assignments. What happens is that the three staff members will be able to be assigned a new weekend week, as long as the demand constraints mentioned above are met.

Table 4.7: An iteration in the destroy and repair iteration showing a swap of weekends when three staff members' week rotations are destroyed.

Initial assignment:

Week	1	2	3	4	5
Librarians	4	4	5	4	4
Assistants	3	3	2	3	3

After destroy:

Week	1	2	3	4	5
Librarians	3	4	4	4	4
Assistants	3	2	2	3	3

After repair:

Week	1	2	3	4	5
Librarians	4	5	4	4	4
Assistants	3	2	3	3	3

Yellow indicates that a staff member, either librarian or assistant, has had its week rotation destroyed and green indicates that a staff member's week rotation has been repaired. Comparing the "Initial assignment" with "After repair", a swap can be seen between week 2 and 3. Worth noting is that a swap can occur, even if the number of qualified staff members remains the same after a repair. To understand this, consider the case when two librarians, with different rotations, have them destroyed. Then two cases can occur; either they are assigned the same rotation as before or they swap.

4.1.4 Assignment of library on wheels

In order to avoid creating enormous number of block combinations, BokB (library on wheels) tasks are assigned manually. This leads to a fix week rotation for the five BokB-staff members, which slightly reduces the degrees of freedom of the problem, as the BokB-staff members' rotation will remain unchanged. However, as there only exist a couple of different sets of feasible BokB assignments, it should not be the deciding factor for the quality of the solution.

Without a manual assignment of BokB, the number of possible tasks in a week block would increase significantly. The number of tasks would increase from 36 to 43, as there are seven unique BokB tasks during a week, which would result in a lot more than 4,175 possible week appearances.

4.1.5 Initial solution

At first, when creating the initial solution, all staff members will have five empty week blocks assigned to them. Then, based on a greedy heuristic, the best week block for a random staff member and random week is found and inserted. This is done until every staff member has one weekend block, one week rest block and three weekday blocks assigned to them. The initial solution is created in a similar fashion to this heuristic's repair function, described in Section 4.1.8 below.

In order to find the best week block, several costs are introduced to measure week block insertions. Say, if the library demands two librarians at the Information desk on Monday at 8 a.m. and currently there is only one assigned, then it shall be profitable to assign another one. Such an assignment will, therefore, be rewarded using a cost contribution. Good assignments will provide negative costs and bad assignments will provide positive costs to the objective function value.

Table 4.8 together with Figure 4.3 show an example of week block evaluation using costs. Here, different cost contributions have to be considered before assigning it to a staff member. In order to calculate demand costs for the PL assume that the block, which is being evaluated, is to be inserted for week three.

Table 4.8: A block example to be evaluated using costs. Only the PL task will be evaluated for simplicity reasons.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00	PL	I		I		I	I
10:00-13:00					D		
13:00-16:00			D				
16:00-20:00							

Task to be evaluated according to Figure 4.3

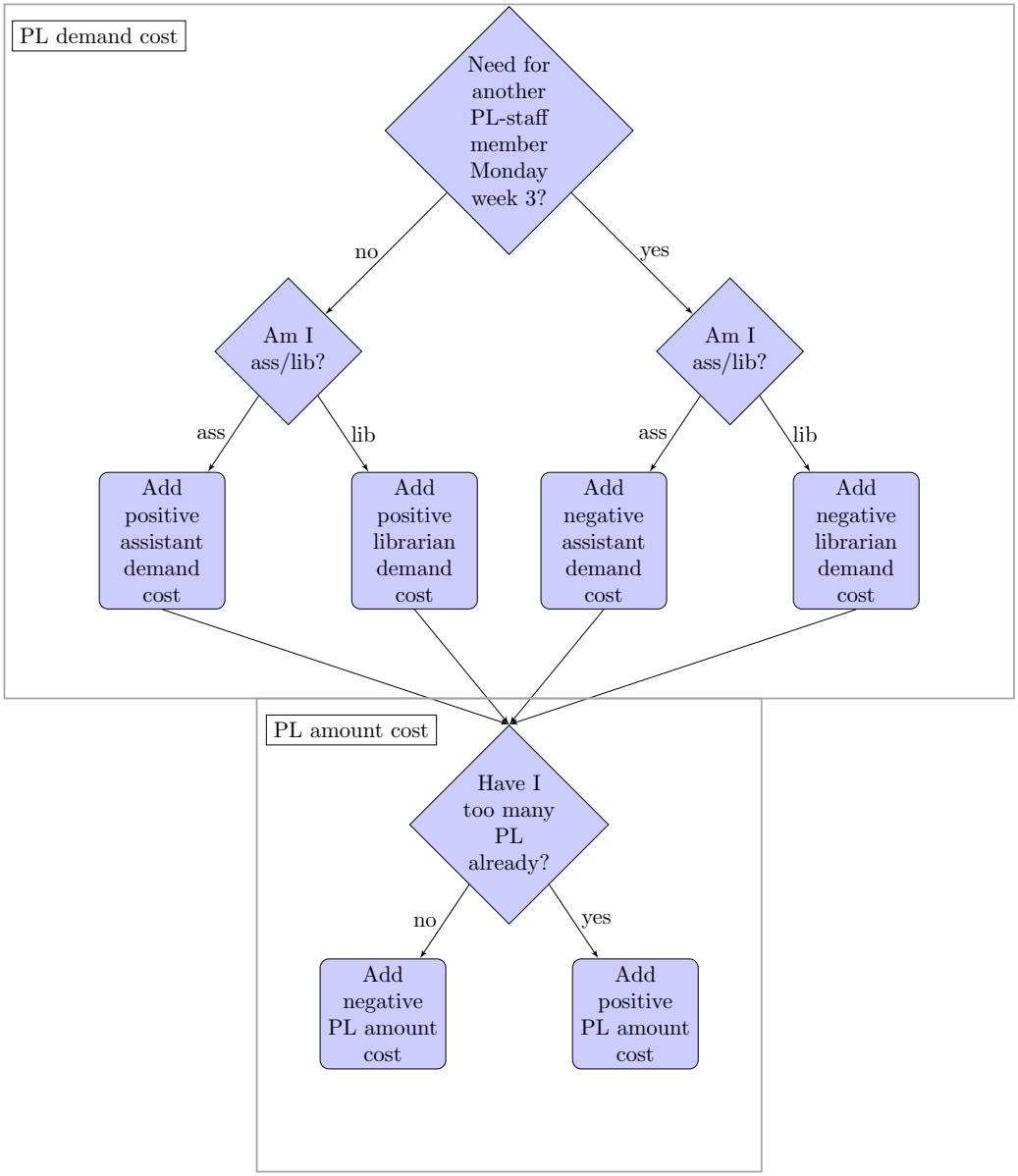
	Mon
08:00-10:00	PL
10:00-13:00	
13:00-16:00	

4.1.6 Costs

In order to find a feasible solution the implemented costs must be carefully chosen. Presently, there are 16 different costs, where some of them are correlated with each other.

Six of the costs are presented in Figure 4.3. They have all been assigned unique values so that, for example, the positive assistant demand cost differs from the negative assistant demand cost in absolute value. To explain why, consider a case where four staff members are demanded, out of which two have

Figure 4.3: A flow chart of costs appearing when a PL is assigned a block on a Monday, third week relative to the library schedule.



to be librarians, see Table 4.9. The first case, where three assistants and one librarian have been assigned, is infeasible, as there is a need of two librarians at the Information desks. The second case, where one assistant and three librarians have been assigned, is feasible. It is, however, not optimal due to the exceeding use of librarians. The optimal solution is, therefore, Case 3. Hence, the first case must be punished more than case 2 and 3, as having more assistants than necessary are redundant assignments.

The complete list of the 16 costs with descriptions can be seen in Table 4.10.

Table 4.9: Library demand at a shift and solution qualities.

Demand:	4 staff members ($\geq$ 2 librarians)	
Case 1:	3 assistants, 1 librarian	(infeasible)
Case 2:	1 assistant, 3 librarians	(feasible)
Case 3:	2 assistants, 2 librarians	(optimal)

Table 4.10: List of all costs with description.

Demand costs	
Cost name	Description
Demand_few_ass	In need of more assistants to fill the demand.
Demand_few_lib	In need of more librarians to fill the demand.
Demand_many_ass	More assistants assigned than needed.
Demand_many_lib	More librarians assigned than optimal.
Demand_few_total	Incorrect number of staff members assigned a task.
Demand_many_total	Incorrect number of staff members assigned a task.
Demand_evening_cost	Incorrect number of staff members assigned an evening task.
Demand_PL_good_ass	Assigning a PL to an assistant, when not already assigned to another.
Demand_PL_good_lib	Assigning a PL to a librarian, when not already assigned to another.
Demand_PL_bad_ass	Assigning a PL to an assistant, when already assigned to another.
Demand_PL_bad_lib	Assigning a PL to a librarian, when already assigned to another.
PL costs	
Cost name	Description
PL_good_amount	Assigning a PL to a staff member still in need of more for feasibility.
PL_violate_amount	Assigning more PL than allowed to that staff member.
Weekend costs	
Cost name	Description
HB_amount	Either no or more than one HB staff member assigned the same weekend.
No_weekend	No weekend blocks available for assignment.
Stand-in costs	
Cost name	Description
Stand_in_cost	Occuring when a possible stand-in is ruined due to an assignment.

Whenever a new week block is to be inserted in a repair all available blocks for that staff member have to be given costs. The cost will be based on the current number of staff members assigned to the tasks in the library. The block with the lowest total cost will be inserted in the repair. An illustrative example can be seen in Table 4.11. The signs of the cost contributions are made up, as we have no current demand in the library.

4.1.7 Destroy

The destroy consists of two contiguous steps: Randomly choosing three staff members and then destroying their assigned blocks. In case a staff member that works with BokB is chosen in the destroy, neither their BokB tasks nor their rotation are destroyed.

Table 4.11: Cost evaluation for two blocks. Negative cost contributions indicate task assignments that are needed in the library. The lower the total cost is, the more desired it is to insert the block in the repair.

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00	PL	I		I		I	I
10:00-13:00					D		
13:00-16:00			D				
16:00-20:00							

$$Totalcost = -Demand\_PL\_good\_lib + PL\_violate\_amount + Demand\_many\_lib - Demand\_few\_lib = 400$$

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
08:00-10:00	I	D	I		D	I	I
10:00-13:00							
13:00-16:00							
16:00-20:00				D			

$$Totalcost = -Demand\_few\_lib - Demand\_evening\_cost + Demand\_many\_lib = -20,260$$

4.1.8 Repair

Initially in the repair function, the destroyed staff members are assigned new rotations, see Table 4.7. Afterwards, five new week blocks are assigned. Which order the blocks are inserted in is randomly generated. For each repair iteration, one of the three staff members is randomly chosen, followed by a randomly chosen block that has not already been reassigned, until all 15 destroyed weeks have been repaired. Which week block that is chosen was discussed in Section 4.1.6.

4.1.9 Evaluation of solution

After each destroy and repair loop the new solution is evaluated. It is common for the new solution to be of worse quality than the previous one. However, all new solutions are accepted regardless of this.

All costs are taken into account in the evaluation of the solution. If there are any demand violations in the library, or in the number of PL assigned to a staff member, these are given the respective costs. Furthermore, in case there are stand-ins, these are provided with a negative stand-in cost.

4.1.10 Final phase

At the end of a script iteration, there are usually a few infeasibilities left that do not get solved. To fix this, a final phase was implemented. Whenever a run is "relatively close" and it is likely to be able to find a feasible solution, the final phase is initiated. "Relatively close" means in this case that the only infeasibility left is the lack of staff members at a handful of weekday shifts. A handful would refer to an estimate of 3-8 tasks violations. It is likely to be able to find a solution, when there are more stand-ins available than demand violations for all days. The reason is because the stand-ins are expendable and can be used to create a feasible solution.

The final phase destroys one random week for a random staff member and immediately repairs it, resulting in most likely the same or a better solution. This is repeated until a feasible solution is found.

With well chosen cost parameter values, the run will always get relatively close to feasibility and reach the final phase eventually. In case there are fewer stand-ins available any day where infeasibility occurs, when reaching the final phase, empty blocks will be assigned to all staff members and the solution enters the destroy and repair loop again, see Figure 4.2.

4.2 Task allocation approach

The second heuristic approach tested allocates individual tasks to staff members and greatly resembles the process of manual scheduling. The implemented algorithm undergoes two phases. In the first phase, weekends, as well as some weekday tasks, are allocated and in the second phase the rest of the weekday tasks are allocated.

The primary heuristic method used in this approach is a large neighbourhood search in combination with a simulated annealing accept function. Destroying and repairing the solution, which is the main feature of LNS, leads the search towards better solutions in the local vicinity of the current solution. The SA function, on the other hand, leads the search out of the current neighbourhood by accepting poorer solutions.

4.2.1 Costs

The search algorithm is guided using penalty costs, both for staff member schedules and for the whole library schedule. Each staff member has a number of cost contributions, as displayed in Table 4.12. Each of these costs is caused by a badly allocated task, resulting in an infeasible schedule or unused stand-in potential. The costs have different weights, as illustrated in the table. How the weights should be set is discussed in Chapter 5.

For the library schedule, costs are weighed together in three different ways. These are referred to as the three library objective functions, as is illustrated in Table 4.13. These costs partly overlap with the individual staff member costs.

The weekend objective function, as referred to in the table, is associated with the weekend allocation phase of the problem. There are three types of costs in this objective function, each measuring a certain aspect of a weekend schedule. Since we want to increase the number of stand-ins at the most critical days, the worst shift or day with respect to these aspects is to be maximized. This is referred to as the min cost. Also, the average of each aspect is considered, in order to distinguish between solutions for which the worst shift or day are identical. These costs are calculated using the formula

$$C_{type} = W_{lib} * num\_of\_lib + W_{ass} * num\_of\_ass. \tag{4.1}$$

The weights W_{lib} and W_{ass} are constants used in all weekend objective function costs. Typically W_{lib} is larger than W_{ass} since librarians can perform more tasks than assistants. The weekend objective function becomes the weighted sum of the costs displayed in Table 4.13.

Table 4.12: Individual staff member costs and cost weights.

Stand-in cost		
Cost	Weight	Cost description
C_{SI}	W_{s_SI}	Cost from having a task on a day where the staff member instead can be a stand-in.
Infeasibility costs		
Cost	Weight	Cost description
C_{Task_D}	$W_{s\_Task\_D}$	Cost for allocating too many tasks in a day.
C_{Task_W}	$W_{s\_Task\_W}$	Cost for allocating too many tasks in a week.
C_{PL_D}	$W_{s\_PL\_W}$	Cost for allocating to many PL tasks in a week.
C_{PL_Tot}	$W_{s\_PL\_Tot}$	Cost for allocating too many PL tasks in total.
C_{SShift_W}	$W_{s\_SShift\_W}$	Cost for allocating to many tasks at the same shift in a week.

Table 4.13: Library objective functions with their cost components and weights.

Weekend Objective Function		
Cost	Weight	Cost description
C_{SI_m}	W_{SI_m}	Cost of day with minimum number of stand-ins.
C_{S_m}	W_{S_m}	Cost of shift with minimum number of staff members.
C_{D_m}	W_{D_m}	Cost of day with minimum number of staff members.
C_{SI_a}	W_{SI_a}	Cost of average number of stand-ins per day.
C_{S_a}	W_{S_a}	Cost of average number of staff members at a shift.
C_{D_a}	W_{D_a}	Cost of average number of staff members at a day.
Staff Member Objective Function		
Cost	Weight	Cost description
C_{Task_D}	W_{Task_D}	Cost for allocating too many tasks in a day for all staff members.
C_{Task_W}	W_{Task_W}	Cost for allocating too many tasks in a week for all staff members.
C_{PL_W}	W_{PL_W}	Cost for allocating to many PL tasks in a week for all staff members.
C_{PL_Tot}	W_{PL_Tot}	Cost for allocating too many PL tasks in total for all staff members.
C_{SShift_W}	W_{SShift_W}	Cost for allocating to many tasks at the same shift in a week for all staff members.
Weekday Objective Function		
Cost	Weight	Cost description
C_{SI_m}	(No weight)	Cost of day with minimum number of stand-ins.

After a certain number of iteration the weekend phase is finished and the weekday allocation phase is entered. Here, the staff member objective function is used to identify infeasibilities in the staff member schedules, and is simply a sum of the staff member costs for all staff members. When this cost is zero, the schedule is feasible and the weekday objective function value is calculated. This objective function contains the minimum stand-in cost, which is calculated in the same way as in the weekend objective function. The weekday objective function corresponds to the objective function in the mathematical model in Chapter 3 and is to be maximized.

4.2.2 Weekend phase

The weekend phase is the first phase the algorithm enters and its flow is described in Figure 4.4. The LNS component consists of the "destroy and repair loop", while the simulated annealing step is marked as the decision to accept a solution which is worse than the current one.

The reason for implementing the weekend phase separately from the allocation of other tasks is that the number of stand-ins in the final schedule depends to a large extent on the weekend-staff member constellation. In particular, it is the week rest following upon weekend work which determines the number of stand-ins at a particular day.

The outer loop in the flow chart is performed a specified number of iterations. This loop is guided by the weekend objective function described in the previous subsection. Better solutions are always accepted and the globally best found solution is saved. The SA component is implemented using exponential cooling, so that a solution which is worse than the current one is accepted with a probability P, given by

$$P = exp(-\Delta E/T) \tag{4.2}$$

where

$$\Delta E = Wend_t - Wend_{t-1} \tag{4.3}$$

and

$$T = T_0 \alpha^t. \tag{4.4}$$

Here, T is the temperature of the SA accept function, which has an initial value of T_0 and which cools down with the iteration counter t at a rate α $(0 < \alpha < 1)$. The objective function value at iteration t is written $Wend_t$.

The inner loop of the algorithm destroys and repairs the solution. The loop checks if the current solution is infeasible, that is if the difference between the number of available staff members at each shift and the number of required staff members is negative. Such schedules are discarded and a destroy and repair is performed. The number of available staff members per shift is calculated as the sum of all staff members who are available at that shift and who do not have any tasks scheduled at that day.

The algorithms for allocating, destroying and repairing weekends is simply a random function, since it is very hard to predict what is a good or bad placement of a specific weekend, without creating the whole schedule. Since the weekend

objective function is only an estimate of the current number of stand-ins and available staff members, this number will be greatly reduced when placing the weekday tasks.

Table 4.14: Staff member availability when only placing weekends. The intensity of red is proportional to the number of available staff members.

	Number of available assistants						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	8	10	10	10	6	0	0
Shift 2:	8	9	9	9	6	0	0
Shift 3:	9	8	9	7	5	0	0
Shift 4:	3	2	2	3	0	0	0
	Number of available librarians						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	16	15	16	13	12	0	0
Shift 2:	18	15	17	14	13	0	0
Shift 3:	17	14	18	18	13	0	0
Shift 4:	3	4	4	3	1	0	0
	Number of available BokB-librarians						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	2	0	1	1	1	0	0
Shift 2:	0	0	0	0	0	0	0
Shift 3:	0	0	0	0	0	0	0
Shift 4:	1	0	2	2	0	0	0

Table 4.15: Staff member availability after placing weekends as well as evening tasks and BokB for the same week.

	Number of available assistants						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	7	10	9	9	8	0	0
Shift 2:	7	9	8	8	8	0	0
Shift 3:	7	7	8	6	6	0	0
Shift 4:	0	0	0	0	0	0	0
	Number of available librarians						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	14	11	13	11	12	0	0
Shift 2:	14	11	14	11	12	0	0
Shift 3:	13	11	14	13	13	0	0
Shift 4:	0	0	0	1	1	0	0
	Number of available BokB-librarians						
	Mo	Tu	We	Th	Fr	Sa	Su
Shift 1:	0	0	0	0	0	0	0
Shift 2:	0	0	0	0	0	0	0
Shift 3:	0	0	0	0	0	0	0
Shift 4:	0	0	1	0	0	0	0

In order to get a better estimate of the number of available staff members during the weekend phase, evening tasks and BokB tasks are allocated after each weekend repair. The evening tasks are allocated using the same method as is described in Section 4.2.3, while BokB tasks are placed according to a fixed schedule. The effect of this allocation is shown in Tables 4.14 and 4.15. High intensity of red in a cell indicates a higher number of staff members. It can be seen that the first table is generally redder and thus there are fewer available staff members in the second table. This state reflects better the real number of available staff members during the week days for a certain weekend constellation.

4.2.3 Weekday phase

When entering the weekday phase, all weekends, evenings and BokB tasks are already allocated. In the weekday phase, the remaining tasks, referred to as "weekday" tasks are to be allocated. The staffing demand for a week is illustrated in Table 4.16 and it is identical for all weeks.The algorithm for allocating weekday tasks is found in Figure 4.5, and greatly resembles the algorithm for allocaing weekends.

The outer loop of the weekday phase, or the "weekday task allocation loop", is performed a specified number of iterations and calculates the weekday objective function for each iteration. The best solution found is saved, others are discarded.

The inner "destroy and repair loop" destroys and repairs the solution until the staff member objective function is zero, that is, until all staff member schedules are feasible. If this loop cannot find such a schedule within a maximum number of iterations, the whole solution is discarded as infeasible, and the outer loop is reentered.

When destroying and repairing tasks, these are first sorted according to qualification requirement, meaning that librarian tasks are placed first and assistant tasks second. This guarantees that not all librarians will be used up for assistant tasks. Secondly, in each subgroup, the tasks are sorted according to the difference between the number of available staff members and the demand at each task. This makes sure that the tasks which are most critical in this respect are allocated first.

The tasks are then staffed one at a time. The process starts with temporarily placing the task at all available staff members. This will generate a cost for each staff member according to Table 4.12. The cheapest staff members are then chosen and permanently placed on the task.

The destroy function identifies the staff member with the highest cost. The worst week of this staff member is then identified and destroyed. Also, a few random staff members are chosen, their number depending on the destroy amount specified, and for these the same week is destroyed. This makes it possible for the first staff member to obtain a better worst week, by exchanging tasks with the other destroyed staff members in the repair function.

Table 4.16: Staffing demand during a week when entering weekday phase.

	Exp	Info	PL	HB	BokB
Monday					
Shift 1:	2	2	1	0	0
Shift 2:	3	3	0	0	0
Shift 3:	3	3	0	0	0
Shift 4:	0	0	0	0	0
Tuesday					
Shift 1:	2	2	1	0	0
Shift 2:	3	3	0	0	0
Shift 3:	3	3	0	0	0
Shift 4:	0	0	0	0	0
Wednesday					
Shift 1:	2	2	1	0	0
Shift 2:	3	3	0	0	0
Shift 3:	3	3	0	0	0
Shift 4:	0	0	0	0	0
Thursday					
Shift 1:	2	2	1	0	0
Shift 2:	3	3	0	0	0
Shift 3:	3	3	0	0	0
Shift 4:	0	0	0	0	0
Friday					
Shift 1:	2	2	1	0	0
Shift 2:	3	3	0	0	0
Shift 3:	3	3	0	0	0
Shift 4:	0	0	0	0	0
Saturday					
Shift 1:	0	0	0	0	0
Sunday					
Shift 1:	0	0	0	0	0

Figure 4.4: A flow chart over the weekend phase.

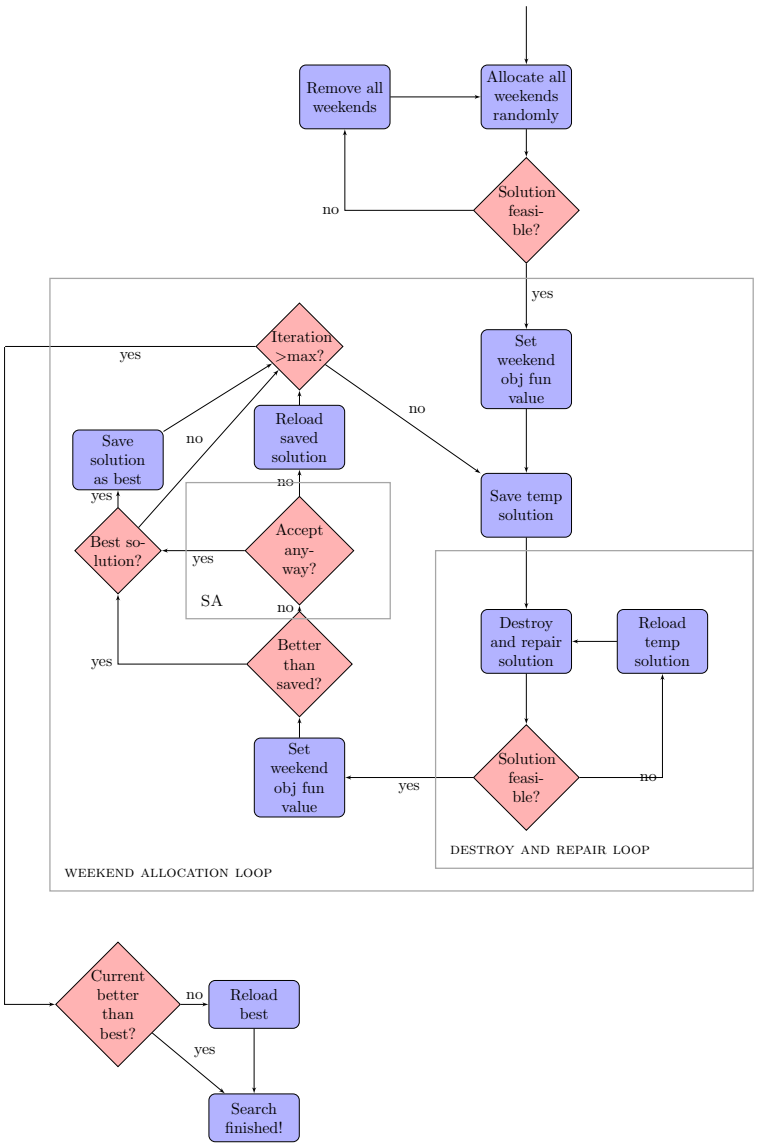
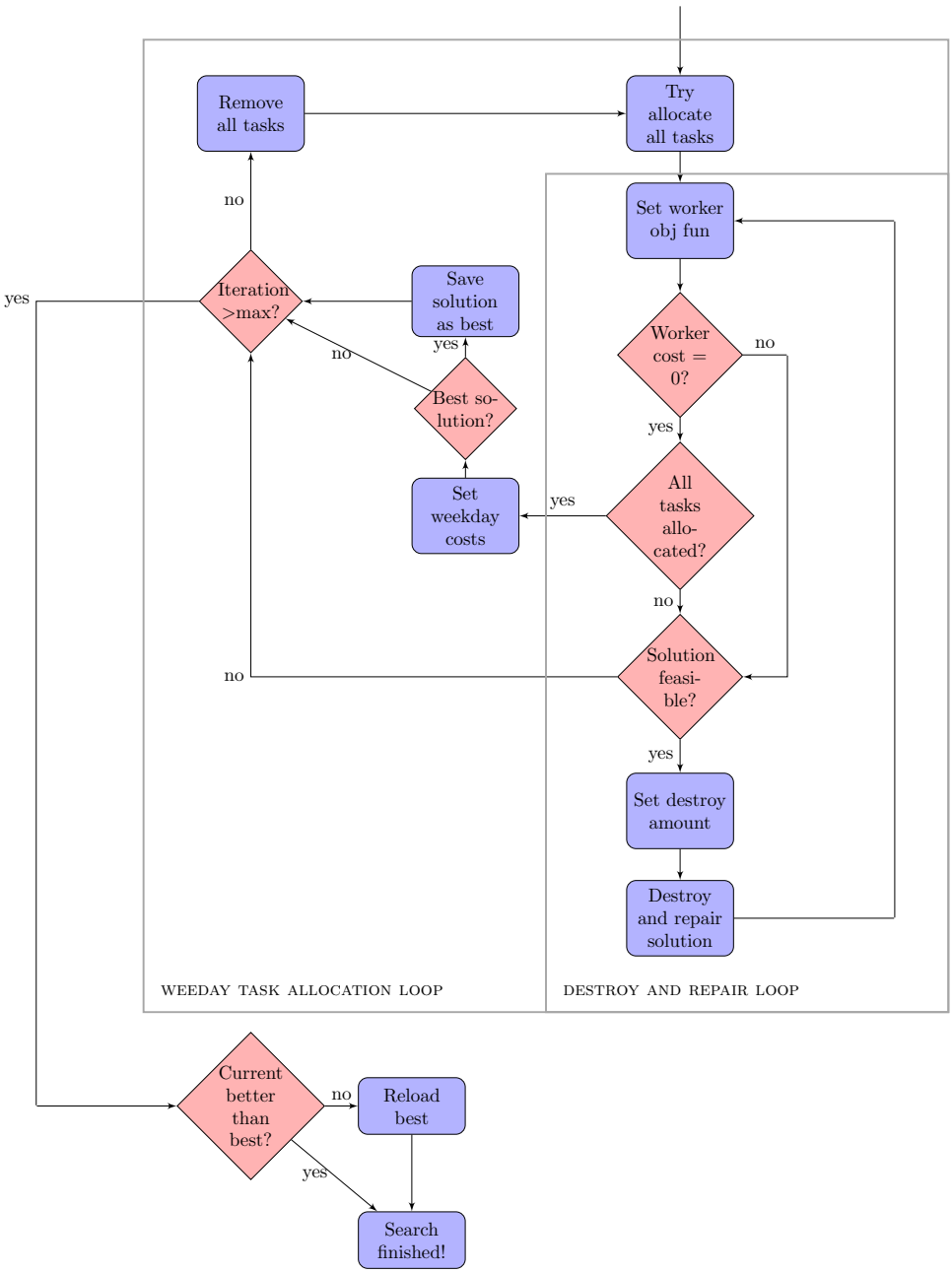


Figure 4.5: A flow chart for the weekday phase.



Chapter 5

Results and discussion

5.1 Results

In the following sections, results from running the AMPL implementation and both heuristics will be presented. The parameter values used in the runs will also be given.

5.1.1 AMPL implementation

By implementing the given mathematical model in AMPL and solving it with CPLEX 12.5, provides the values displayed in Table 5.1. The weights in Equation 3.17 and 3.35 were set to $L = 2$, $A = 1$, $M = 100$ and $N = 1$, thus making librarians twice as valuable as assistants and stand-ins 100 times more valuable than no shift changes.

Table 5.1: Results from solving the mathematical model with CPLEX solver.

	Min num stand-ins (lib, ass)	**Stand-in cost**	**Solution time**
Result	(3,0)	6	19 min

Only the stand-in part of the objective function is presented, since this value can be used as a benchmark in measuring the performance of the heuristics. The number of shift changes, which is the second part of the objective function in Equation 3.17, is zero.

5.1.2 Week block scheduling approach

As most of the costs are correlated with each other, some parameter tuning was required. The final result of the parameter tuning are shown in Table 5.2. Their respective descriptions can be seen in Table 4.10.

Worth noting is that these values are most certainly not the best possible, as they have mostly been assigned using intuition. However, a few relations were determined by either running tests or using reasoning. An example is if $PL\_good\_cost > Demand\_PL\_good\_ass$ or $PL\_good\_cost > Demand\_PL\_good\_lib$. If one of those holds, all staff members that is able to be assigned another PL,

Table 5.2: List of all costs used in the week block scheduling approach and their respective values.

Demand costs	
Cost name	Cost value
Demand_few_ass	350
Demand_few_lib	300
Demand_many_ass	200
Demand_many_lib	40
Demand_few_total	800
Demand_many_total	700
Demand_evening_cost	20,000
Demand_PL_good_ass	1,200
Demand_PL_good_lib	800
Demand_PL_bad_ass	1,200
Demand_PL_bad_lib	1,600
PL costs	
Cost name	Cost value
PL_good_amount	1,000
PL_violate_amount	1,500
Weekend costs	
Cost name	Cost value
HB_amount	15,000
No_weekend	5,000
Stand-in costs	
Cost name	Cost value
Stand_in_cost	5

without violating the individual PL assigment constraint, will be. Thereby, leaving the library overstaffed with staff members assigned to PL. The reason is because the net of $-PL\_good\_cost + Demand\_PL\_good\_ass/lib$ will be negative, meaning it is preferable to assign yet another PL to a staff member, regardless of how many there already are assigned. This leads to an overstaffed library on some days, and lack of staff members on other days.

Table 5.3 shows the success rate of finding a solution during a script execution. A run is considered to fail, if no feasible solution is found during the final phase. This script execution is using the cost parameters shown in Table 5.2.

Table 5.3: Number of successful runs. A failed run is when no feasible solution is found during the final phase.

Successful runs	420
Runs (total)	638
%	∼66

Table 5.4 shows statistics of how many stand-ins were found after 420 successful runs. In case "No stand-ins" are found after a run, then a feasible solution is found, however, there are no stand-ins on the worst day (or days). The results "1 assistant" or "1 librarian" mean that one stand-in was found at

best on the worst day (or days).

Table 5.4: Number of stand-ins found in 420 successful runs.

Stand-in spread	Times
No stand-ins	351
1 assistant	33
1 librarian	36

Figure 5.1 shows a plot of the objective function value after destroy and repair iterations. When the value on the y-axis reaches zero, a feasible solution is found. Whenever an iteration is below the final phase boundary, the entering criterion is checked. If the final phase is entered, the few existing demand violations will be taken care of, one by one, by assigning the stand-ins where the violations occur. A feasible solution is found whenever an objective function value of zero is reached. The final phase boundary is set as $1 \cdot 10^4$.

Every time the value skyrockets, a solution has been discarded in the final phase. Empty blocks are, thereafter, assigned to all staff members and the run is once again entering the destroy and repair loop. Destroy and repair iterations are being executed until a solution is likely to be found, that is, there are enough stand-ins available to take care of the demand violations.

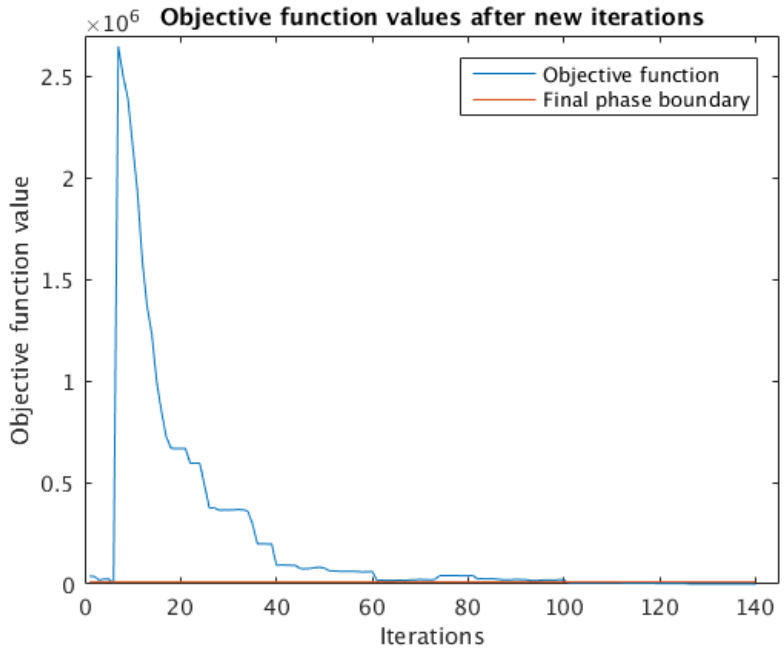


Figure 5.1: Plot of the objective function value after destroy and repair iterations; one time without enough stand-ins, when trying to enter the final phase.

As seen in Figure 5.1 there are not enough stand-ins one time during the run. What happens is that the objective function gets below the final phase boundary, but does not meet the stand-in criterion and, therefore, destroys every workers' schedule.

Figure 5.2 illustrates a run when there are enough stand-ins to enter the final phase.

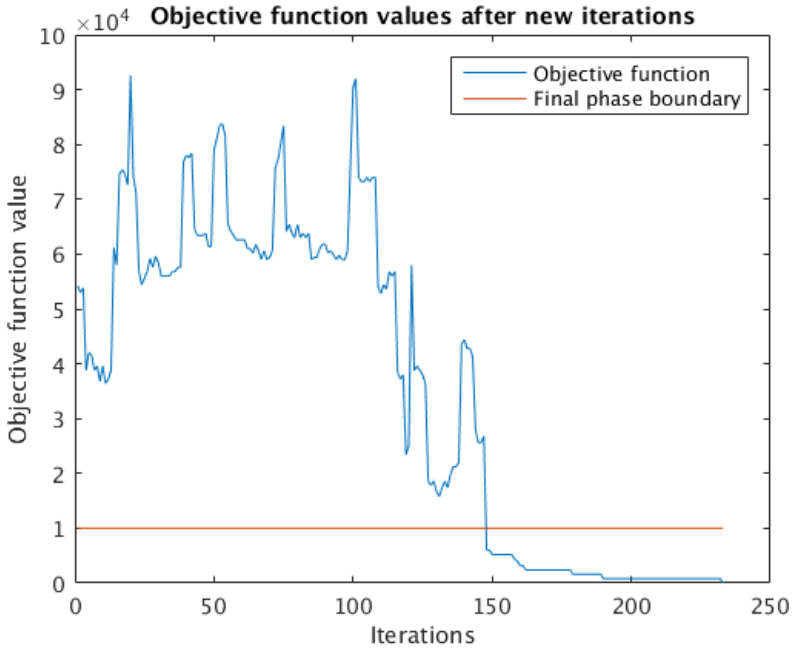


Figure 5.2: Plot of the objective function value after destroy and repair iterations with enough stand-ins when trying to enter the final phase.

To illustrate why the solution gets worse after a destroy and repair iteration; consider the case when two librarians with the same rotation are being repaired, where one only can work at HB during weekends and the other one can work at either HB or with Desk tasks. If the more versatile librarian is repaired first and being assigned to HB, due to the cost parameters, then it will be impossible for the second librarian to be assigned any weekend tasks. This results in a high cost contribution to the objective function value.

5.1.3 Task allocation approach

In order to find a good value for the weights described in Section 4.2.1, different tuning tests were performed. These tests resulted in the weights presented in Table 5.5. These weights are used in all test in this section unless stated otherwise.

The first six weights, belonging to the weekend objective function, were most experimented with and will be discussed further in this section. The first three, the min weights, are scaled relatively to each other and decide how much weight is placed on maximizing the number of stand-ins during the worst day, maximizing the number of staff members at the worst day and maximizing the number of staff members at the worst shift, respectively. Only maximizing the number of stand-ins at the worst day during the weekend phase would not give a complete estimation of the number of stand-ins in the final schedule, since this

Table 5.5: Weights used in the implementation.

Weight	Weight value
Weekend Objective Function Weights	
W_{SI_m}	0.1
W_{S_m}	0.1
W_{D_m}	10
W_{SI_a}	0.01
W_{S_a}	0.01
W_{D_a}	1
Staff Member/Staff Member Objective Function Weights	
W_{w_SI}	2 if librarian 1 if assistant
W_{Task_D} / $W_{w\_Task\_D}$	100
W_{Task_W} / $W_{w\_Task\_W}$	10
W_{PL_W} / $W_{w\_PL\_W}$	5
W_{PL_Tot} / $W_{w\_PL\_Tot}$	5
W_{SShift_W} / $W_{w\_SShift\_W}$	4

number also depends on how many workers are present in total during that day and during the shifts of the days. Thus, all three aspects were measured.

The next three weights, refered to as the average weights, belong to the three costs which give the average of each of the three aspects described above. They are simply calculated as one tenth of their equivalent min weights. The weights in Equation 4.1, W_{lib} and W_{ass} were set to 2 and 1, respectively, since it can be argued that one librarian can perform twice as many tasks as one assistant.

The individual staff member cost and the staff member objective function costs, which measure the same aspects of the schedule, have been given the same weights, as is illustrated in Table 5.5. This is due to the fact that these weights, relative to each other, decide what constraints are most strict. It is, for example more severe to break the rule of one task per day, than to break the rule of having too many shifts at the same time in a week, which is reflected in the weights. The stand-in weight W_{w_SI} is twice as high for librarians as for assistant.

The two phases of the implementation are run a specified number of iterations, referred to as It_{wend} and It_{wday}. These parameters were also trimmed and the results from different iteration values are displayed in Tables 5.6 and 5.7. The results for weight tuning are displayed in Table 5.8.

Table 5.6: Results from the task allocation heuristic when varying It_{wend}

It_{wend}/It_{wday}	**Heuristic cost**	**AMPL cost**
10/10	4.40	4.78
50/10	5.00	5.34
100/10	5.26	5.52
500/10	5.66	5.78
1000/10	5.54	5.66

The tables display two costs for each test, the heuristic cost and the AMPL cost. These are calculated as the average result of 100 runs. Each run takes

Table 5.7: Results from the task allocation heuristic when varying It_{wday}

It_{wend}/It_{wday}	**Heuristic cost**	**AMPL cost**
500/5	5.47	5.74
500/10	5.66	5.78
500/15	5.6	5.74
500/20	5.57	5.74

Table 5.8: Results from the task allocation heuristic when varying weights. Here, $It_{wend} = 1000$ and $It_{wday} = 20$ in all runs.

$W_{SI\_m}/W_{S\_m}/W_{D\_m}$	**Heuristic cost**	**AMPL cost**
0/0/0	2.85	3.66
1/1/1	5.43	5.54
10/0.1/0.1	5.11	5.36
0.1/10/0.1	5.43	5.60
0.1/0.1/10	5.57	5.80

between a few seconds and a few minutes to perform. The heuristic cost is the actual result of the heuristic, measuring how well the heuristic solves the problem. The AMPL cost is the result obtained when solving the problem to optimality in CPLEX, using the weekend work pattern found in the heuristic. Thus, this value measures how good the weekend phase works. Both figures should be compared to the optimal stand-in value for the problem, which is 6 (see Table 5.1).

Studying Table 5.6, it can be seen that the AMPL cost increases with more weekend iterations. However, after 500 iterations, the cost stops to improve and it is actually deteriorating slightly. This is probably indicating that the weekend objective function value can only give a rough measure of a good schedule, and the result does not only depend on the number of iterations but also on chance.

Similarly, in Table 5.7, the heuristic cost increases until a certain threshold, when the solutions are not improving any more. This might indicate that the second phase is not random enough, so the same schedules are found multiple times. This might be solved through implementing varying weights or introducing more randomness in task placement.

Table 5.8 shows the results from using a few extreme weight combinations. It is is clear from the table that, although the performance does not vary to any larger extent when shifting the focus in the weights, using a completely random search with no weights produces poor results. The last combination of weights seems to be performing slightly better than the others, and thus these weights were chosen in the implementation.

The weekend objective function value during a typical run is shown in Figure 5.3. The effects of the SA accept function are visible in the plot as smaller or larger dives in the objective function value. In the runs, the SA parameters $T_0 = 0.4$ and $\alpha = 0.985$. $It_{wend} = 1000$ were used. The graph suggests that a high weekend objective function value is found already within the 200 first iterations, which is consistent with the results in Table 5.6.

In Figure 5.4 and Figure 5.5, the three different cost components of the weekend objective function for the same run are displayed. In the first plot, the minimum costs are displayed. These seem to move between a few discrete values,

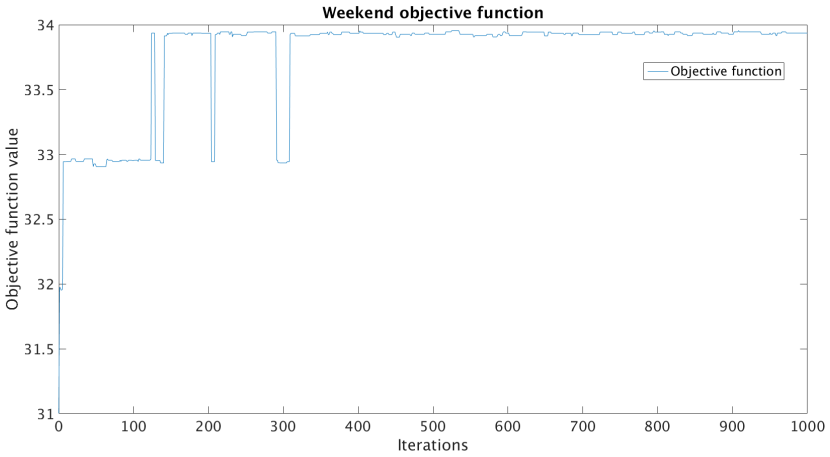


Figure 5.3: The weekend objective function for 1000 iterations

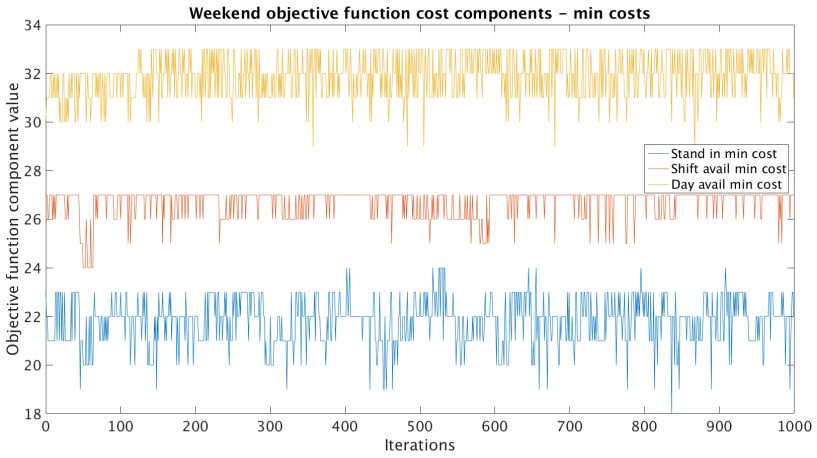


Figure 5.4: The weekend objective function minimum cost components for 1000 iterations

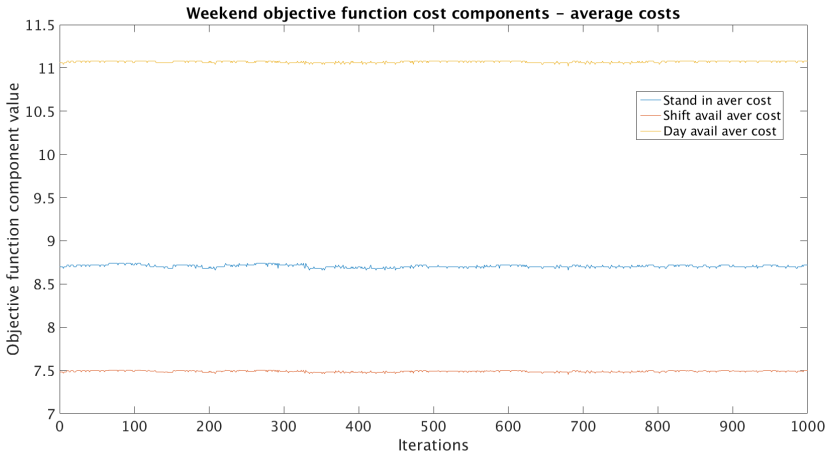


Figure 5.5: The weekend objective function average costs components for 1000 iterations

although a small improvement over the iterations can be seen, at least in the day avail cost component. The average values are almost constant throughout the iterations, as can be seen in the second plot. This is probably due to the fact that the total number of available staff members barely changes when shifting their weeks.

Attempts were made to correlate the min component values of the weekend objective function to each other. However, the results were fluctuating between different runs so no clear correlations could be identified. This suggests that they are measuring three independent aspects.

5.2 Discussion

In this section, the heuristic results, presented in Section 5.1, will be discussed. Pros and cons for both methods are given, so that they can be more easily compared.

5.2.1 Week block scheduling approach

Table 5.9 lists pros and cons with the implemented week block scheduling approach. Some pros and cons were considered before this heuristic was chosen. Mainly, the con concerning the exponential growth of week blocks was taken into consideration, and an estimate of the upper limit of the problem size was done.

The pro regarding the same amount of week blocks was taken into consideration, as the heuristic was thought of having firstly, a block construction phase and secondly, an assignment phase. However, the most significant pro of this approach is that many of the constraints are already met, implicitly, when the blocks are created. Constraints like maximum of one task per day, one evening per week, one PL per week, two shifts at the same hour per week and more are already taken care of.

Cons like rotations, solution time and costs are no major issues, as they can be avoided by a few smarter implementations. For instance, rotations can be improved by assigning a chosen value whenever a staff member is available as stand-in on a day. From the sum of those values an even distribution of possible stand-ins can be acquired by, for instance, always letting the difference between the lowest and the highest sum of stand-ins, through the days, be as small as possible. This improvement should be done instead of randomly generating the new rotations in the destroy and repair iteration.

The improvement mentioned above should also implicitly decrease the solution time, as less iterations would be required to find a feasible solution.

In contrast to the cons mentioned above, a major issue is to always be able to create a pool of possible week block appearances, regardless of the problem size. Just by adding meetings and the assignment of the library on wheels tasks would make the number of possible week block appearances grow considerably.

Table 5.9: Pros and cons with the implemented week block scheduling approach.

Pros	**Cons**
Many constraints are already met, implicitly, when the blocks are created.	Rotations need to be assigned in a systematic way in order to achieve reasonable results, regarding lowest number of stand-ins through the days.
The same number of week block appearances will exist for five and ten weeks.	The number of unique week block appearances grows exponentially in case more task types are added, such as meetings.
Quick iterations when destroying and repairing.	The solution time can vary considerably, as several random number generators have been used.
	More cost parameters are needed for this heuristic, where every one of them affect the solution procedure.

5.2.2 Task allocation approach

The results from the previous section point to the fact that finding a good weekend allocation is essential in order to produce a good final schedule. Thus, when It_{wend} is increased, the results improve, up to a limit of around 500 iterations. The results are in general comparable to those of the AMPL implementation. This suggests that the weekend objective function used gives a weekend schedule which, with a high probability, can provide a good basis for creating the complete schedule. However, as the results seem to converge to a value slightly lower than the AMPL result the weekend objective function must be seen only as a fairly good an optimal weekend schedule.

Table 5.8 suggests that the weights used in the weekend objective function are important for how well the implementation works, but only to a certain

degree. More testing could be done on the weights in order to see if the result can be further improved. However, since the objective function is an incomplete measure of a good schedule the results will never be completely optimal. This is listed as the main drawback in the cons column in Table 5.10.

Table 5.10: Pros and cons with the implemented task allocation approach

Pros	**Cons**
In most cases, the method finds an optimal solution.	The weekend objective function is not an exact measurement of a good weekend schedule.
The method is very fast compared to the CPLEX solver. Only a few iterations in each phase gives good results.	There are many weights and parameters to take into consideration in the implementaion.
The method can be used to find a good weekend schedule. The rest of the schedule can then be created using another method such as CPLEX, in a short time.	

Chapter 6

Concluding remarks

In this thesis work, a mathematical model for the staff scheduling problem at the Library of Norrköping was developed according to the demands given by the library. The model was solved by the commercial optimization solver CPLEX, which generated an optimal schedule, where three librarian stand-ins and zero assistant stand-ins was found on the worst day (or days). This is better than the current schedule at the library, where only one librarian and zero assistants were assigned as stand-in on the worst day.

When developing the two heuristics for solving the problem, the conclusion was reached that the weekend distribution is essential for the stand-in distribution. The heuristic that focused on making weekend scheduling separately from the rest of the problem, thus performed better. Thus, the first heuristic would probably work better if it took weekend allocation into consideration.

Although good results were obtained both by using the AMPL model and in one of the heuristics, there is room for improvement in the modeling of the problem. One thing which could be done in a more robust way, is the modelling of exceptions and preferences among the staff members. In the current model, these are simply modelled as hard constraints. If these could be modelled in a more general way or separately from the rest of the problem, the model would not be so sensitive to changes in exceptions or in the staff preferences.

Several components of the model could be simplified. For example, the library on wheels could be modelled separately from the rest of the problem. Furthermore, staff members schedules which differ between odd and even weeks also add complexity to the problem. One way of handling this could be to produce schedules which are not entirely feasible according to the problem description, but which can be used as basic schedules for which manual adjustment is needed.

In the heuristics, there are several components missing in order for them to fully correspond to the mathematical model. Although these features are not crucial, they could be implemented in order to make the heuristics more accurate. Furthermore, a more systematic framework for testing and evaluating the heuristics could be developed, in order to be able to measure the results more fairly. If one heuristic would be chosen for further development it would be the second one, since, in the first method, a very large number of blocks would have to be created if more task types were added.

The conclusion is that the mathematical model provides the best results,

followed by the second heuristic. Both meetings and a 10 week schedule are included in the mathematical model, which still are left out of both heuristics, due to lack of time. Also, the first heuristic needs a better rotation distribution in order to provide better results.

Bibliography

A. T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1-4):21–144, 2004.

M. Krishnamoorthy and A. T. Ernst. The personnel task scheduling problem. In *Optimization Methods and Applications*, pages 343–368. Springer, 2001.

M. Krishnamoorthy, A. T. Ernst, and D. Baatar. Algorithms for large scale shift minimisation personnel task scheduling problems. *European Journal of Operational Research*, 219(1):34–48, 2012.

L. G. Kroon, M. Salomon, and L. N. Van Wassenhove. Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research*, 82(1):190–205, 1995.

P. Smet and G. Vanden Berghe. A matheuristic approach to the shift minimisation personnel task scheduling problem. *Practice and Theory of Automated Timetabling*, pages 145–151, 2012.

P. Smet, T. Wauters, M. Mihaylov, and G. Vanden Berghe. The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega*, 46:64–73, 2014.

J. S. Loucks and F. R. Jacobs. Tour scheduling and task assignment of a heterogeneous work force: A heuristic approach. *Decision Sciences*, 22(4):719–738, 1991.

G. M. Thompson. *A Comparison of Techniques for Scheduling Non-Homogeneous Employees in a Service Environment Subject to Non-Cyclical Demand Volume I, Chapters l-7.* PhD thesis, The Florida State University, 1988.

K. Choi, J. Hwang, and M. Park. Scheduling restaurant workers to minimize labor cost and meet service standards. *Cornell Hospitality Quarterly*, 50(2): 155–167, 2009.

A. Rong. Monthly tour scheduling models with mixed skills considering weekend off requirements. *Computers & Industrial Engineering*, 59(2):334–343, 2010.

M. Hojati and A. S. Patil. An integer linear programming-based heuristic for scheduling heterogeneous, part-time service employees. *European Journal of Operational Research*, 209(1):37–50, 2011.

I. Gertsbakh and H. I. Stern. Minimal resources for fixed and variable job schedules. *Operations Research*, 26(1):68–85, 1978.

M. Fischetti, S. Martello, and P. Toth. Approximation algorithms for fixed job schedule problems. *Operations Research*, 40(Supplement 1):S96–S108, 1992.

S. O. Duffuaa and K. S. Al-Sultan. A stochastic programming model for scheduling maintenance personnel. *Applied Mathematical Modelling*, 23(5):385–397, 1999.

S. M. Roberts and L. F. Escudero. Scheduling of plant maintenance personnel. *Journal of Optimization Theory and Applications*, 39(3):323–343, 1983.

M. Akbari, M. Zandieh, and B. Dorri. Scheduling part-time and mixed-skilled workers to maximize employee satisfaction. *The International Journal of Advanced Manufacturing Technology*, 64(5-8):1017–1027, 2013.

S. Mohan. Scheduling part-time personnel with availability restrictions and preferences to maximize employee satisfaction. *Mathematical and Computer Modelling*, 48(11):1806–1813, 2008.

H. A. Eiselt and V. Marianov. Employee positioning and workload allocation. *Computers & Operations Research*, 35(2):513–524, 2008.

P. Shahnazari-Shahrezaei, R. Tavakkoli-Moghaddam, and H. Kazemipoor. Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. *Applied Mathematical Modelling*, 37(7):5424–5443, 2013.

R. J. Li. *Multiple Objective Decision Making in a Fuzzy Environment.* PhD thesis, Kansas State University, 1990.

D. Pisinger and S. Ropke. Large neighborhood search. In *Handbook of Metaheuristics*, pages 399–419. Springer, 2010.

Appendix A

Problem definitions

A.1 Sets

I	Set of workers
I_lib	Set of librarians (I_lib $\subseteq$ I)
I_ass	Set of assistants (I_ass $\subseteq$ I)
W	Set of weeks
W_5	Set of first five weeks
W_9	Set of first nine weeks
D	Set of days in a week
D_4	Set of first four weekdays
D_5	Set of all five weekdays
D_w	Set of weekend days, Saturday and Sunday
S_d	Set of shifts day d
S_3	Set of first three shifts on a weekday
J_d	Set of task types day d
I_{free}	Set of workers that shall be assigned a free weekday per week
$I_{oddEven}$	Set of all workers with odd or even weeks
$I_{weekendavail}$	Set of workers available for weekend work
I_{big}	Set of all workers that attend big meetings
I_{noP}	Set of workers not working on PL
I_{manyP}	Set of workers working 3-4 times with PL in 10 weeks
G	Set of groups
$I_G\{g\}$	Set of workers in group g
V	Set of possible week rotations (shifts the week by 0-9 steps forwards)

A.2 Variables

$$x_{iwdsj} = \begin{cases} 1, & \text{if worker } i \text{ is assigned in week } w\text{, day } d\text{, shift } s \text{ to a task } j \\ 0, & \text{otherwise} \end{cases} \tag{A.1}$$

$$H_{iwh} = \begin{cases} 1, & \text{if worker } i \text{ works weekend } h \ (= 1, 2) \text{ in week } w \\ 0, & \text{otherwise} \end{cases} \tag{A.2}$$

$$r_{iw} = \begin{cases} 1, & \text{if worker } i \text{ has its schedule rotated } w\text{-}1 \text{ steps} \\ 0, & \text{otherwise} \end{cases} \tag{A.3}$$

$$l_{iwd} = \begin{cases} 1, & \text{if librarian } i \text{ is a stand-in week } w\text{, day } d \\ 0, & \text{otherwise} \end{cases} \tag{A.4}$$

$$a_{iwd} = \begin{cases} 1, & \text{if assistant } i \text{ is a stand-in week } w\text{, day } d \\ 0, & \text{otherwise} \end{cases} \tag{A.5}$$

$$y_{iwds} = \begin{cases} 1, & \text{if worker } i \text{ works week } w\text{, day } d\text{, shift } s \text{ at task type E, I or P} \\ 0, & \text{otherwise} \end{cases} \tag{A.6}$$

$$W_{iwd} = \begin{cases} 1, & \text{if a worker } i \text{ is working a shift week } w\text{, day } d \\ 0, & \text{otherwise} \end{cases} \tag{A.7}$$

$$b_{iw} = \begin{cases} 1, & \text{if worker } i \text{ works at HB week } w \\ 0, & \text{otherwise} \end{cases} \tag{A.8}$$

$$f_{iw} = \begin{cases} 1, & \text{if worker } i \text{ is assigned to work friday evening week } w \\ 0, & \text{otherwise} \end{cases} \tag{A.9}$$

$$M_{wds} = \begin{cases} 1, & \text{if a big meeting is placed on week } w\text{, day } d\text{, shift } s \\ 0, & \text{otherwise} \end{cases} \tag{A.10}$$

$$m_{wdsg} = \begin{cases} 1, & \text{if a meeting is placed on week } w\text{, day } d\text{, shift } s \text{ for group } g \\ 0, & \text{otherwise} \end{cases} \tag{A.11}$$

$$d_{iwds} = \begin{cases} 1, & \text{if there is a difference in assignment of tasks at a shift} \\ & \quad s\text{, for a worker } i\text{, day } d \text{ between week } w \text{ and } w\text{+}5 \\ 0, & \text{otherwise} \end{cases} \tag{A.12}$$

$$l^{min} = \text{lowest number of stand-in librarians found (integer)} \tag{A.13}$$

$$a^{min} = \text{lowest number of stand-in assistants found (integer)} \tag{A.14}$$

$$s^{min} = \text{weighted sum with number of stand-in librarians and assistants} \tag{A.15}$$

$$\delta = \text{total number of shifts that differ for all staff members (integer)} \tag{A.16}$$

A.3 Parameters

$$N1l = \text{a value to prioritize the amount of stand-in librarians} \tag{A.17}$$

$$N1a = \text{a value to prioritize the amount of stand-in assistants} \tag{A.18}$$

$$N1 = \text{a value to prioritize total number of stand ins} \tag{A.19}$$

$$N2 = \text{a value to prioritize similar weeks} \tag{A.20}$$

$$avail\_day_{iwd} = \begin{cases} 1, & \text{if worker } i \text{ is available for work week } w, \text{ day } d \\ 0, & \text{otherwise} \end{cases} \tag{A.21}$$

$$demand_{dsj} = \text{number of workers required day } d, \text{ shift } s \text{ for task type } j \tag{A.22}$$

$$qualavail_{iwdsj} = \begin{cases} 1, & \text{if worker } i \text{ is qualified and available week } w, \text{ day } d, \text{ shift } s \text{ for task type } j \\ 0, & \text{otherwise} \end{cases} \tag{A.23}$$

$$LOW\_demand_{wds} = \text{number of workers required day } d, \text{ shift } s \text{ at the library on wheels} \tag{A.24}$$

A.4 Objective function and constraints

$$\text{maximize} \quad M \cdot s^{min} - N \cdot \delta \tag{A.25}$$

$$\sum_{i \in I} x_{iwdsj} = (1 - M_{wds}) demand_{dsj},\ w \in W, d \in D_5, s \in S_d, j \in J_d \backslash \{L\} \tag{A.26}$$

$$\sum_{i \in I} x_{iwdsj} = demand_{dsj},\ w \in W, d \in D_w, s \in S_d, j \in J_d \tag{A.27}$$

$$\sum_{i \in I} x_{iwdsL} = LOW\_demand_{wds},\ w \in W, d \in D_5, s \in S_d \tag{A.28}$$

$$\sum_{d \in D_5} \sum_{s \in S_d} \sum_{j in J_d} x_{iwdsj} \leq 4,\ i \in I \backslash \{36\}, w \in W \tag{A.29}$$

$$\sum_{w \in W_5} M_{w11} = 1 \tag{A.30}$$

$$M_{(w+5)11} = M_{w11} \tag{A.31}$$

$$\sum_{w \in W} \sum_{d \in D_5} \sum_{s \in S_d} M_{wds} = 2 \tag{A.32}$$

$$\sum_{s \in 2..3} \sum_{j \in J_d \backslash \{L\}} x_{iw1sj} \leq 1 - M_{w11},\ i \in I \backslash I_{big}, w \in W \tag{A.33}$$

$$\sum_{w\in W_5}\sum_{d\in D_5}\sum_{s\in S_3} m_{wdsg} = 1,\ g \in G \tag{A.34}$$

$$m_{(w+5)dsg} = m_{wdsg},\ g \in G, w \in W_5, d \in D_5, s \in S_3 \tag{A.35}$$

$$m_{wdsg} + x_{iwdsj} \leq 1,\ g \in G, i \in I_G\{g\}, w \in W, d \in D_5, s \in S_3, j \in J_d \tag{A.36}$$

$$m_{wd2g} + m_{wd3g} + x_{iwd1P} \leq 1,\ g \in G, i \in I_G\{g\}, w \in W, d \in D_5, s \in S_3 \tag{A.37}$$

$$m_{wdsg} \leq \sum_{v\in V} r_{iv}\cdot qualavail_{i\omega(v,w)dsE},\ g \in G, i \in I_G\{g\}\backslash\{39\}, w \in W, d \in D_5, s \in S_3 \tag{A.38}$$

$$M_{w11} + \sum_{d\in D_5}\sum_{s\in S_3} m_{wdsg} \leq 1,\ g \in G, i \in I\_big \cup I_G\{g\}, w \in W \tag{A.39}$$

$$\sum_{j\in J_d} x_{iwdsj} \leq 1,\ i \in I, w \in W, d \in D, s \in S_d \tag{A.40}$$

$$\sum_{s\in S_d}\sum_{j\in J_d\backslash\{L\}} x_{iwdsj} \leq 1 - x_{iwds_2L},\ i \in I, w \in W, d \in D_4, s_2 \in S_d \tag{A.41}$$

$$\sum_{s\in S_5}\sum_{j\in J_d\backslash\{L\}} x_{iw5sj} \leq 1,\ i \in I, w \in W \tag{A.42}$$

$$\sum_{j\in J_d\backslash\{L\}} x_{iw5sj} \leq 1 - x_{iw51L},\ i \in I, w \in W, s \in S_3 \tag{A.43}$$

$$\sum_{s\in S_d}\sum_{j\in J_d} x_{iwdsj} \leq 1,\ i \in I, w \in W, d \in D_w \tag{A.44}$$

$$\sum_{d\in D_4}\sum_{j\in J_d} x_{iwd4j} \leq 1,\ i \in I\backslash\{36\}, w \in W \tag{A.45}$$

$$\sum_{d\in D_5} x_{iwd1P} \leq 1,\ i \in I, w \in W \tag{A.46}$$

$$x_{iwdsP} = 0,\ i \in I_{noP}, w \in W, d \in D_5, s \in S_d \tag{A.47}$$

$$\sum_{w\in W}\sum_{d\in D_5} x_{iwd1P} \leq 4,\ i \in I_{manyP} \tag{A.48}$$

$$\sum_{w\in W}\sum_{d\in D_5} x_{iwd1P} \leq 3,\ i \in I\backslash\{I_{noP} \cup I_{manyP}\} \tag{A.49}$$

$$\sum_{w \in W} r_{iw} = 1,\ i \in I \tag{A.50}$$

$$\sum_{w \in W} H_{iwh} \leq 1,\ i \in I_{weekendavail}, h \in \{1,2\} \tag{A.51}$$

$$r_{iw} \geq H_{iw1},\ i \in I, w \in W \tag{A.52}$$

$$r_{i(2 \cdot w)} = 0,\ i \in I_{OddEven}, w \in W_5 \tag{A.53}$$

$$r_{i(mod_{10}(w+4)+1)} \geq H_{iw2},\ i \in I, w \in W \tag{A.54}$$

$$\sum_{s \in S_6} \sum_{j \in J_6} x_{iw6sj} + \sum_{s \in S_7} \sum_{j \in J_7} x_{iw7sj} = 2 \cdot \sum_{h \in \{1,2\}} H_{iwh},\ i \in I_{weekendavail}, w \in W \tag{A.55}$$

$$\sum_{s \in S_6} x_{iw6sj} = \sum_{s \in S_7} x_{iw7sj},\ i \in I_{weekendavail}, w \in W, j \in J_7 \tag{A.56}$$

$$\sum_{j \in J_d \backslash \{L\}} x_{iw54j} = f_{iw},\ i \in I_{weekendavail}, w \in W \tag{A.57}$$

$$b_{iw} = x_{iw61B},\ i \in I_{weekendavail}, w \in W \tag{A.58}$$

$$f_{iw} \geq H_{iwh} + (1 - b_{iw}) - 1,\ i \in I_{weekendavail}, w \in W, h \in \{1,2\} \tag{A.59}$$

$$f_{iw} \leq \sum_{h\{1,2\}} H_{iwh},\ i \in I_{weekendavail}, w \in W \tag{A.60}$$

$$f_{iw} \leq 1 - b_{iw},\ i \in I_{weekendavail}, w \in W \tag{A.61}$$

$$\sum_{w \in W} \sum_{d \in \{6,7\}} x_{iwd1B} \leq 2,\ i \in I_{lib} \backslash \{23\} \tag{A.62}$$

$$\sum_{d \in \{6,7\}} \sum_{j \in \{E,I\}} x_{23wd1j} = 0,\ w \in W \tag{A.63}$$

$$s^{min} \leq N1l \cdot \sum_{i \in I_{lib}} l_{iwd} + N1a \cdot \sum_{i \in I_{ass}} a_{iwd},\ w \in W, d \in D_5 \tag{A.64}$$

$$W_{iwd} \geq \sum_{s \in S_3} \sum_{j \in J_d} x_{iwdsj},\ i \in I, w \in W, d \in D_5 \tag{A.65}$$

$$W_{iwd} \geq \sum_{j \in J_d} x_{iwd4j},\ i \in I, w \in W, d \in D_5 \tag{A.66}$$

$$l_{iwd} \geq \sum_{v \in V} (r_{iv} \cdot avail\_day_{i\omega(v,w)d}) + (1 - W_{iwd}) - 1,\ i \in I_{lib}, w \in W, d \in D_5 \tag{A.67}$$

$$l_{iwd} \leq \sum_{v \in V} (r_{iv} \cdot avail\_day_{i\omega(v,w)d}),\ i \in I_{lib}, w \in W, d \in D_5 \tag{A.68}$$

$$l_{iwd} \leq 1 - W_{iwd},\ i \in I_{lib}, w \in W, d \in D_5 \tag{A.69}$$

$$a_{iwd} \geq \sum_{v \in V} (r_{iv} \cdot avail\_day_{i\omega(v,w)d}) + (1 - W_{iwd}) - 1,\ i \in I_{ass}, w \in W, d \in D_5 \tag{A.70}$$

$$a_{iwd} \leq \sum_{v \in V} (r_{iv} \cdot avail\_day_{i\omega(v,w)d}),\ i \in I_{ass}, w \in W, d \in D_5 \tag{A.71}$$

$$a_{iwd} \leq 1 - W_{iwd},\ i \in I_{ass}, w \in W, d \in D_5 \tag{A.72}$$

$$d_{iwds} \geq (y_{iwds} - y_{i(w+5)ds}),\ i \in I, w \in W_5, d \in D_5, s \in S_3 \tag{A.73}$$

$$d_{iwds} \geq -(y_{iwds} - y_{i(w+5)ds}),\ i \in I, w \in W_5, d \in D_5, s \in S_3 \tag{A.74}$$

$$\delta = \sum_{i \in I} \sum_{w \in W_5} \sum_{d \in D_5} \sum_{s \in S_3} d_{iwds} \tag{A.75}$$

$$x_{iwdsj} \leq \sum_{v \in V} (r_{iv} \cdot qualavail_{i\omega(v,w)dsj}),\ i \in I, w \in W, d \in D, s \in S_d, j \in J_d \tag{A.76}$$

$$y_{iwds} = \sum_{j \in J_d \backslash \{L\}} x_{iwdsj},\ i \in I, w \in W, d \in D_5, s \in S_3 \tag{A.77}$$

$$\sum_{d \in D_5} y_{iwds} \leq 2,\ i \in I, w \in W, s \in S_3 \tag{A.78}$$

$$\sum_{w \in W} x_{25w44L} = 8 \tag{A.79}$$

$$x_{36w14L} = 1 - x_{25w11L},\ w \in W \tag{A.80}$$

$$\sum_{w \in W} x_{37w34L} = 5 \tag{A.81}$$

$$x_{37w34L} = 1 - x_{37(w+1)34L},\ w \in W_9 \tag{A.82}$$

$$x_{36wd4j} = 0,\ w \in W, d \in D_4, j \in J_d \backslash \{L\} \tag{A.83}$$

Appendix B

Week block table

Table B.1: Number of assignable unique blocks for the staff members based on their availability and qualification.

Staff member	Weekend	Weekrest	Weekday
1	532	347	1580
2	1580	1580	1580
3	1063	347	1580
4	557	165	779
5	261	130	531
6	532	130	1580
7	261	130	531
8	261	29	531
9	115	12	247
10	532	130	1580
11	9	8	8
12	1063	347	1580
13	771	92	1190
14	265	29	489
15	51	18	120
16	495	130	843
17	237	69	267
18	532	279	1580
19	495	47	843
20	532	130	1580
21	227	227	227
22	2	1	1
23	3	2	2
24	11	5	47
25	1063	279	1580
26	5	4	4
27	213	106	425
28	2	2	2
29	426	106	1281
30	127	126	455
31	495	130	843
32	261	29	531
33	72	45	306
34	425	425	425
35	91	20	221
36	55	27	101
37	1063	347	1580
38	3	1	1
39	2	1	1