Master thesis

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

Claes Arvidson and Emelie Karlsson

LiTH - MAT - EX - - 04 / 04 - - SE

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

Optimeringslära, Linköpings Universitet

Claes Arvidson and Emelie Karlsson

LiTH - MAT - EX - - 04 / 04 - - 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 an inhomogeneous work force at libraries and other service institutions is time consuming for manual shcedulers, but well suited for optimization software. Such a problem is studied. The problem concerns five different task types, two types of workers and around 100 tasks to be scheduled in total weekly. Also weekends are to be scheduled and worker satisfaction is taken into account. The objective of the scheduling is to create a ten week rotating schedule in which the stand-in staff members are evenly distributed.

A mathematical model is formulated for the problem, which is solved using the commercial CPLEX solver and by using two different LNS heuristic implementations. The first heuristic schedules week blocks to the workers, while the second 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 since it focuses on finding a good weekend distribution before creating the rest of the schedule.

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

URL for electronic version:

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

Acknowledgements

The authors of this thesis 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 uncertainty and who has been guiding us through the whole project. Thank you for the time and effort you have spent on our thesis. We would also like to thank 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 thesis topic and for answering all our questions many times.

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 supported us until the end.

Nomenclature

Most of the reoccurring terms and abbreviations are described here.

Optimization terms

Heuristic	An algorithm designed to find sufficiently good, but not necessarily feasible solutions.

Library terms

Fetch list	The library task of collecting books from shelves, to be delivered elsewhere.
Library on wheels	The task of driving a library bus with books to remote areas of town.

Abbreviations

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 all staff members. Poor work distribution among staff can cause problems and result in a shortage of staff at certain critical shifts. If a worker 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 libraries and other similar institutions 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 the Central 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 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 creates a scheduling challenge for the library as it requires 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 at hand 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 problem studied is to create a ten week schedule for all staff members and tasks at the library. Each staff member's ten week schedule should consist of two five-week schedules which should be as similar as possible, but not identical since some tasks are performed only even or odd weeks. The ten week schedule should be rotational.

The schedule should also 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, what tasks should be assigned when as well as the constraints for how these tasks should 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 a part of the every day library tasks as well. The problem of allocating 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. expidetionsdisken), 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 for which the worker is scheduled during a whole day, while the other tasks are scheduled for only one shift. The outer tasks can be performed by either librarians or 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)	Administring loans, library cards and the loaning machine	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, the Central Library of Norrköping also has responsibilities that fall outside of its normal daily activities. One such responsibility is staffing a smaller library subsidiary in Hageby, situated in the suburban area of Norrköping, during weekends. It is decided that only librarians are qualified for this task. While some librarians prefer to work only at HB, others prefer never to.

Working during a weekend means working both Saturday and Sunday. When not working at HB, the Friday evening shift is also part of the weekend work. The weekend work is compensated with free days, often placed at the week following upon weekend work, in this thesis referred to as rest week.

Since the number of visitors at the library varies throughout the day and also during different days of the week, 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.

Another library task 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 city with books as well as other library services. The Library on Wheels only operates a few times a week

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

and the schedule differs between odd and even weeks in accordance with 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 subset of the staff members. The library meetings are fixed in time and take place every fifth weeks on Monday mornings from 8:15 a.m. to 10:00 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 workers 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.

The Central Library of Norrköping currently has 39 library staff members, 23 of which are librarians and 16 of which are assistants. These have different availabilities for performing tasks, depending on their working hours and the amount of inner work they have. In the standard case, each worker 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 free days, placed according to the wishes of the worker.

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

All staff members have a five week schedule in the same manner as the sample worker. However, in order to meet the weekend demands, illustrated in Table 1.2, staff members have to be assigned for weekend work at different weekends. Thus, a staff memeber's schedule can be shifted in order to make sure that all weekend demands are 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 placed. Firstly, workers are only allowed to take at a maximum one task per day. This guarantees that workers also have time for inner tasks. Secondly, workers are only allowed to perform a maximum of four tasks per week, in order for them to have a completely free day. Thirdly, workers can only be assigned to PL at a most once a week. Also, there is a maximum limit of PL tasks in total over all ten weeks which varies between different staff members, but which is at the most four. Lastly, a worker should not have the same shift more than twice a week. This guarantees that there is fairness in the schedule so that no worker has to work too many times at a shift which is considered to be "bad". An example of a feasible weekly schedule for the sample worker is provided in Table 1.5.

The availability of all workers is provided by the library. Some workers work differently even and odd weeks while other never work evenings, only work weekends or never work weekends. This data was used in order to solve the scheduling problem.

Table 1.4: Availability schedule for a sample worker. Yellow signifies that the worker 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							

Table 1.5: Example of a feasible week for a worker.

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 schedule variation

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. When we studied their schedule, we observed that only one stand-in was present at the worst day. Thus, 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 day with the least number of stand-ins.

In this context, a stand-in is defined as a worker who is available for outer tasks during the first three shifts and who is not scheduled for any shift during

that day. Both librarians and assistants can be stand-ins, but only librarians are qualified to be assigned to any task. Since the regular hour library activities are 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 for 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 the Central Library of Norrköping can be solved using mathematical optimi zation methods. This 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 in the thesis work for solving the problem described above was to first formulate a mathematical model for the problem, which was then solved using two different methods. The first method involved modeling the problem in the AMPL programming language and solving it by the commercial optimization solver CPLEX. The second method involved creating an independent solver in C++ and solving it using a heuristic. Worker availability data was imported from Excel sheets to data files used to solve the problem through Visual Basic.

Two different heuristics were tested in the second approach. The first heuristic distributes week blocks to workers. 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 workers. In order to move through the solution space and find better solutions, both heuristics were based on a Large Neighbourhood Search (LNS) 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 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 for these implementations are given in Chapter 5. Lastly, conclusions from the work and suggestions for further development are presented in Chapter 6.

1.5 Contributions of the authors

The authors of this thesis have worked both individually and together. The mathematical model was constructed and implemented by both authors, while 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 own heuristic. Chapter 6 was written by both authors.

Chapter 2

Literature review

The scheduling problem is a mathematical optimization problem which has been studied since the 1950s and concerns creating a feasible and satisfactory schedule for workers or machines performing tasks. One of the most extensive overviews in the area is provided by Ernst et al. (2004). They state that, although the complexity of the scheduling problem has not increased in recent years, the mathematical models used to solve the scheduling problems have become more realistic and refined. Modern 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 modeling refinement as well as the development of more powerful computational methods, it is possible today to solve scheduling problems in a more satisfactory way than before.

In this chapter, the scheduling problem is classified into different subcategories which are areas related to the thesis work presented in this paper. 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) writes 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. Decisions will then have to be made regarding the amount of maintenance workers needed and which machine the workers are assigned to look after.

There are numerous variants to the PTSP. Studies on these have been made by Krishnamoorthy and Ernst (2001) who gives a list of attributes that com-

monly 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	Minimise assignment cost
	T	Worktime costs including overtime
	W	Minimise number of workers
	U	Minimise unallocated tasks

Many of the most basic and a few more complex problems 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 assigment 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 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). The 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, 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 the present 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 Minimisation Personnel Task Scheduling Problem

A close relative to the PTSP is the Shift Minimisation 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 in 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 minimised.". In doing so, it is possible to determine the lowest number of staff and the mix of staff a company need in order to be able to complete the tasks and still be operational. They also presumed that the pool of workers is unlimited for each skill group, which is not the case in the present 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 is needed to ensure 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 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 independent problems which can be solved independently. A solution to 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. Which constructive and improvement heuristic they considered can be seen in their article.

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 used described 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 the scheduling problem often mentioned in literature is provided by Thompson (1988). The problem studied in this PhD thesis concern 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 representative 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 other 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 economical 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 and is similar to our goal 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 differs 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 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 the 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 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 will be 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 spread time constraint forcing tasks to spread out with time gap 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 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 TFISP, where both types are restricted by the following: Each machine (worker) cannot handle more than one job at a time, can only handle a subset of the jobs and preemption of jobs is not allowed. The difference between them 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. 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 difference in 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 combing 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 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. An objective function is used where 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 Modeling 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 values such as worker satisfaction as an objective

function. Such values are usually considered when scheduling is done manually, but have been forgotten or set aside in mathematical modeling.

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. Increasing variance in the schedule is done 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 modeling 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 uses 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 optimal 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 in cooler 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), although the article compares these results with the results obtained from a branch-and-cut algorithm. Also Shahnazari-Shahrezaei et al. (2013) uses 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 modeling 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 between between them will be presentedi n 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 SMPTSP, the TSP, the FJSP, the TFISP and the OFISP. The first two are relevant to our problem as they concern task assignment and, in the case of TSP, 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 number of stand-in personnel is to be maximized. Such problems often arise in the maintenance job scheduling problem, as is discussed in Section 2.4.4. As a secondary objective, we are also interested in making the schedule varied and with a repeating pattern at an interval of five weeks. Modeling of such constraints is discussed in Section 2.5. The two sections are complementary to the previous sections, as they are less relevant in the overall problem formulation

but more relevant concerning the objective function.

Chapter 3

The mathematical model

In this chapter the mathematical model implemented to solve this problem will be presented. Prior to the objective function and constraints, the most significant sets and variables will be provided to give the reader some basic knowledge of the implemented model. 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 had 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 workers want a day free from outer tasks to be able to attend meetings or perform inner tasks. Another example is that some have two alternating schedules for odd and even weeks. These specific cases have to be modeled and result in a variety of sets and variable definitions. Hence, only the most important ones are listed below. A complete list of the definitions can be found in Appendix A.

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 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 day d
S_3	Set of first three shifts on a weekday
J_d	Set of task types available day d

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

Let,

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

$$H_{iwh} = \begin{cases} 1, & \text{if worker } i \text{ works weekend } h \ (= 1, 2) \text{ in week } w \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

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

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

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

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

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

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

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

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

$$m_{wdsD} = \begin{cases} 1, & \text{if a meeting is placed on week } w, \text{ day } d, \text{ shift } s \text{ at department } D \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

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

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

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

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

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

3.2 Objective function

As the problem consists of multiple objective functions it has been necessary to weigh them against each other using parameters. These are shown in Equation 3.16 below as M and N.

$$maximize \quad M \cdot s^{min} - N \cdot \sum_{i \in I} \sum_{w \in W_5} \sum_{d \in D_5} \sum_{s \in S_3} d_{iwds} \tag{3.16}$$

The two first objective functions represent the lowest amount of stand-in librarians and assistants found. The second objective function is a preference from the library that two weeks with a five-week interval should be as similar as possible (e.g. week 1 and 6, 2 and 7 etc.).

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, $M \gg N$ in our case.

3.3 Constraints

To model the present problem it has been of relevance to divide many of the constraints into weekend and weekday constraints. Several help constraints have also been added to avoid multiplication of two variables, making the problem non-linear. These help constraints have been left out of this chapter for simplicity reasons. Instead, these can be seen in Appendix A.

3.3.1 Demand and assignment constraints

The most crucial constraint is to ensure that the demand of workers is met each day. This can be modeled as:

$$\sum_{i \in I} x_{iwdsj} = demand_{wdsj}, \ \forall w \in W, d \in D, s \in S, j \in J_d \tag{3.17}$$

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

The following constraint says whether or not a worker is assigned a task during a weekday, where the evening shifts and task type *Library on Wheels* are excluded:

$$y_{iwds} = \sum_{j \in J_d \setminus \{B\}} x_{iwdsj}, \ \forall i \in I, w \in W, d \in D_5, s \in S_3 \tag{3.18}$$

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

To ensure that no worker is assigned more than one task the following constraint is implemented:

$$\sum_{s \in S} \sum_{j \in J_d} x_{iwdsj} \leq 1, \ \forall i \in I, w \in W, d \in D \tag{3.19}$$

However, if we allow a person to have two shifts at the Library on Wheels on a day, Equation 3.19 has to be slightly modified, which is left out of this chapter.

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

$$\sum_{s \in S_d} \sum_{d \in D} x_{iwdsP} \leq 1, \ \forall i \in I, w \in W \tag{3.20}$$

$$\sum_{w \in W} \sum_{s \in S_d} \sum_{d \in D} x_{iwdsP} \leq 3, \ \forall i \in I \tag{3.21}$$

The duration of a PL (in the equations named "P"), which is from 08:00-16:00, is the cause of these preferences; some workers are required to have some alloted time to perform their inner tasks.

Another preference is to have various start times of the assigned tasks each week, so that the more and less desired shifts are evenly distributed. The following equation is implemented to meet such requirements:

$$\sum_{d \in D_5} y_{iwds} \leq 2, \ \forall i \in I, w \in W, s \in S_3 \tag{3.22}$$

Equation 3.22 allows a worker to have at most two tasks starting at the same hour. Worth noting is that task type *Library on Wheels* is disregarded in the constraint as the variable y_{iwds} is used; see Equation 3.18 for the definition.

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

$$\sum_{d \in D_5} \sum_{s \in S_d} \sum_{j \in J_d} x_{iwdsj} \leq 4, \ \forall i \in I, w \in W \tag{3.23}$$

Equation 3.23 allows a worker at most four weekday shifts per week. The exception, which is one of the BokB workers, is left out of this equation for simplicity reasons. To model this completely a new subset of I needs to be created where that worker is left out.

3.3.2 Weekend and rotation constraints

To further model the problem weekends and week rotations have to be considered. The following three constraints are the most basic constraints regarding weekends H_{iwh} and week rotations r_{iw}:

$$\sum_{w \in W} r_{iw} = 1, \ \forall i \in I \tag{3.24}$$

$$\sum_{w \in W} H_{iwh} \leq 1, \ \forall i \in I, h = 1, 2 \tag{3.25}$$

$$r_{iw} \geq H_{iw1}, \ \forall i \in I, w \in W \tag{3.26}$$

Equation 3.24 provides all workers with a rotation of their schedule regardless if they are working weekends or not. Equation 3.25 allows a worker a maximum of two weekends ($h = 1 and h = 2$) per ten weeks. Lastly, Equation 3.26 in combination with Equation 3.24 ensures that the schedule rotation is aligned with the first weekend, if the worker is due for weekend work. In case the worker is not due for weekend work, the rotation of the schedule is free.

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

$$r_{i(mod_{10}(w+4)+1)} \geq H_{iw2},\ \forall i \in I, w \in W \tag{3.27}$$

Worth noting is that if a worker is assigned both weekends then Equation 3.27 provides the same information as Equation 3.26.

Every workers 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 workers are assigned tasks only when available. This is provided in the following equation:

$$x_{iwdsj} \leq \sum_{v \in V} r_{iv} * qualavail_{i(mod_{10}(w-v+10)+1)dsj},\ \forall i \in I, w \in W, d \in D, s \in S_d, j \in J_d \tag{3.28}$$

Table 3.1 gives a better understanding of how the modulus function in the parameter $qualavail_{iwdsj}$ is used. w represent the current week in the relative schedule and v represent *v-1* rotations to the right from the relative schedule. The resulting week after rotations can be seen in the cells.

Table 3.1: Resulting table of the function $mod_{10}(w - v + 10) + 1$

		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 relative schedule where everyone is said to work weekend the first week. Say that we look at a worker'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 represent the previous ninth week, $w = 9$ from the old schedule. The availability to look at in the old schedule is, therefore, week nine.

A worker is supposed to work with the same task both Saturday and Sunday when working a weekend. This can be modeled with the following constraints:

$$\sum_{j \in J_d} x_{iw61j} + \sum_{j \in J_d} x_{iw71j} = 2 * \sum_{h=1}^{2} H_{iwh},\ \forall i \in I, w \in W \tag{3.29}$$

$$x_{iw61j} = x_{iw71j},\ \forall i \in I, w \in W, j \in J_d \tag{3.30}$$

Friday evening is also included in extension to working Saturday and Sunday, unless the worker is assigned to HB. It is, however, not a necessity to perform the same task Friday evenings as during the weekend, thus Fridays are not included in Equation 3.30. Equation 3.31 below adds Fridays to the weekend:

$$\sum_{j \in J \setminus \{L\}} x_{iw54j} = f_{iw}, \ \forall i \in I, w \in W \tag{3.31}$$

"L" in the equation refers to the task Library on Wheels. The variable f_{iw} is, as mentioned in the variable declaration, equal to one if and only if a worker is working weekend as well as not being assigned to HB.

It is of interest to implement a constraint to prevent workers from being assigned to HB more than once every ten weeks to avoid unfairness. The constraint can be modeled as:

$$\sum_{w \in W} \sum_{d=6}^{7} x_{iwd1B} \leq 2, \ \forall i \in I \tag{3.32}$$

"B" in this equation refers to HB. The combination of Equations 3.30 and 3.32 ensure that once a worker is assigned to HB it will be for two consecutive weekend days, which is the amount of days the last mentioned equation allow a worker every ten weeks. Why this is preferable is described in Section 1.1.1.

3.3.3 Objective function constraints

To calculate the variable s^{min} used in the objective function, 3.16, the following equation is implemented:

$$s^{min} \leq L \cdot \sum_{i \in I_{lib}} l_{iwd} + A \cdot \sum_{i \in I_{ass}} a_{iwd}, \ \forall w \in W, d \in D_5 \tag{3.33}$$

l_{iwd} and a_{iwd} are binary variables stating whether a worker is a stand-in or not on a specific day. s^{min} is being maximized in the objective function; therefore, it will assume the lowest value of the sum of stand-in librarians and assistants for any day during the ten weeks. Just as with previous equation, help constraints have been left out for simplicity reasons. In this case regarding how l_{iwd} and a_{iwd} are determined.

If $L < A$ 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 task types. Therefore, it is desired to set $L \geq A$.

As stated in Section 3.2, two weeks with a five-week interval should be as similar as possible. Equation 3.34 in combination with the objective function equation, 3.16, provides this preference.

$$d_{iwds} = |y_{iwds} - y_{i(w+5)ds}|, \ \forall i \in I, w \in W_5, d \in D_5, s \in S_3 \tag{3.34}$$

The decision variable d_{iwds} in this equation states if there is a difference in assignment between two tasks at the same hour and day for two weeks, w and $w+5$. The differences occur if, say, an Exp task is assigned on Monday week one at 08:00-10:00 and no task is assigned the same shift Monday week six. Thus, a variable minimized in the objective function.

3.3.4 Meeting constraints

At the library there are both library meetings as well as department meetings, where both occur with a five-week interval. Library meetings are set to take place from 08:00-10:00 on Mondays, whereas department meetings are more freely distributed. A few workers are not assigned library meetings, since they are needed at the stations to keep the library running. The set I_{big} in the equation below consists of all workers who are to be assigned library meetings. The constraints modeling library meetings are as follows:

$$\sum_{w \in W_5} M_{w11} = 1 \tag{3.35}$$

$$M_{(w+5)11} = M_{w11},\ \forall w \in W_5 \tag{3.36}$$

$$\sum_{s=1}^{3} \sum_{j \in J_1 \setminus \{L\}} x_{iw1sj} \leq 1 - M_{w11},\ \forall i \in I \setminus I_{big}, w \in W \tag{3.37}$$

Equation 3.35 and 3.36 assign two library meetings with a five-week interval during the ten-week scheduling period. Equation 3.37 makes sure that the workers that do not attend library meetings are not assigned any other task the day of the meeting.

The constraints added to model department 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 Equation 3.38 and 3.39.

$$\sum_{w \in W_5} \sum_{d \in D_5} \sum_{s \in S_3} m_{wdsD} = 1 \tag{3.38}$$

$$m_{(w+5)dsD} = m_{wdsD},\ \forall D = 1, \ldots, 3, w \in W_5, d \in D_5, s \in S_3 \tag{3.39}$$

$$m_{wdsD} + x_{iwdsj} \leq 1,\ \forall D = 1, \ldots, 3, i \in I_D, w \in W, d \in D_5, s \in S_3, j \in J_d \tag{3.40}$$

$$m_{wdsD} \leq \sum_{v \in V} r_{iv} * qualavail_{i(mod_{10}(w-v+10)+1)dsE},\ \forall D = 1, \ldots, 3, i \in I_D, w \in W, d \in D_5, s \in S_3 \tag{3.41}$$

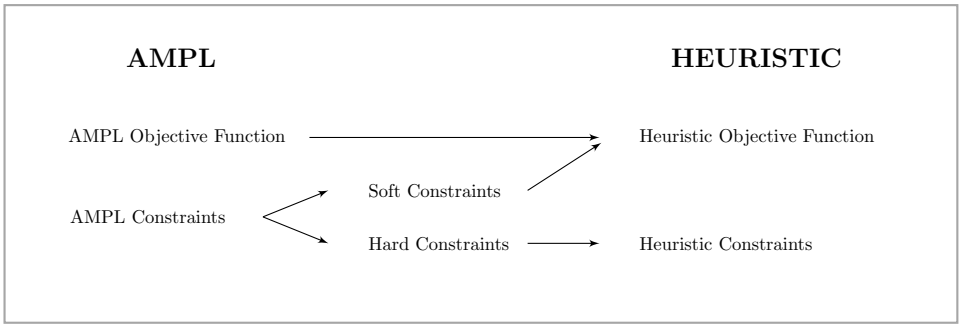
Equation 3.40 prohibits a worker from multitasking, i.e. attend a meeting as well as work with a task. Lastly, Equation 3.41 enables department meetings only when everyone in that department is available to be assigned a task; the rotation is taken into the account. "E" in Equation 3.41 stands for Exp and is used due to it is the only task type everyone is available for and *qualavail* is dependent on the variable *j*.

Chapter 4

Two heuristics

In the AMPL implementation, the model is identical to the one described in Chapter 3. However, in order to implement the heuristics, the model's constraints had to be relaxed. Figure 4.1 illustrates this process as some constraints considered hard in the original model are softened in the heuristics. The alterations of the model for the two heuristics is described in Tables 4.1 and 4.2. The reason for relaxing the model is so as to create a larger neighbourhood for the heuristic to search through. This allows it to move more freely between solutions and increases the chance of finding the optimal solution.

Figure 4.1: Illustration of the difference between the model used in AMPL and the model for a heuristic



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

Table 4.1: Week block scheduling approach; soft and relaxed constraints

Soft Constraints	
Affecting constraints	Constraint description
3.17	The amount of workers needed for every shift and task type in the library.
3.21	Number of PL per ten weeks restriction.
Relaxed Constraints	
Affecting constraints	Constraint alteration
Several. $W = W_5$ in all constraints.	Five week scheduling instead of ten week scheduling.
3.34	Any two weeks w and $w+5$ shall be as similar as possible.
BokB-constraints	BokB placed according to constraints, but the schedule is fixed.
Availability data	Even and odd week workers have availability at each shift according to the stricter of the two sets.
3.35 - 3.41	Meetings are not implemented.

Table 4.2: Task distribution approach; soft and relaxed constraints

Soft Constraints	
Affecting constraints	Constraint description
3.19	One task per day restriction.
3.23	Four tasks per week restriction.
3.20	One PL per week restriction.
3.21	Number of PL per ten weeks restriction.
3.22	Not more than two tasks at the same shift in a week restriction.
Relaxed Constraints	
Affecting constraints	Constraint alteration
Several. $W = W_5$ in all constraints.	Five week scheduling instead of ten week scheduling.
3.34	Any two weeks w and $w+5$ shall be as similar as possible.
BokB-constraints	BokB placed according to constraints, but the schedule is fixed.
Availability data	Even and odd week workers have availability at each shift according to the stricter of the two sets.
3.35 - 3.41	Meetings are not implemented.

4.1 Week block scheduling approach

The implemented algorithm is based on Large Neighborhood Search (LNS) where small parts of the current solution is destroyed and repaired. In Appendix B, Figure B.1, is a flow chart of the implemented heuristic presented.

Every workers' information such as availability and qualification is inserted

into an Excel table. It is then written to a text file using Visual Basic code, which in turn is read and used by the heuristic.

4.1.1 Block creation

A big part of this heuristic is to create the big pool of unique week appearances. These are then filtered for each of the worker based on its availability. The workers availability are generalized into three categories: *Weekend week*, *weekrest week* and *weekday week*, where 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 up to three task types. Table 4.3 below is a representation of a general week block with all possible tasks for each day d and shift s. I in the table represents "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 generalized weekblock with all existing 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 more than one shift. The duration of a PL is three shift and I refers to the entire day. Hence, 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:

1. At most two tasks can be assigned the same shift and week.
2. No more than two evenings are allowed each week, one of which is required to be a Friday.
3. At most one PL is allowed in a weekblock.
4. Saturday and Sunday shall always contain the same task type.
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 without tasks.

Too see the growth of the problem when more tasks are added, consider the following example: If item 1, 2, 3, 5 and 6 are disregarded there exists $6^5 * 3 = 23,328$ unique weekblocks. In contrast, if Exp and Info were to be considered separately, instead of the combination of the two, the possible combinations would be $10^5 * 4 = 400,000$. By applying all conditions the total amount of unique block appearances for this implementation are 4,175.

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						

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

This block contains five tasks; two of them are weekend tasks and three are weekday tasks. Which week this block can be assigned to is dependent on the worker's rotation. Since Hageby is assigned to the weekblock one can conclude that only a librarian can have this block assigned to itself. Due to this fact, the Desk tasks can imply either Exp or Info desk work as librarians are qualified for both.

4.1.2 Block percolation

After creating all existing week combinations they are percolated to each of the workers based on their availability in each of the three categories mentioned in Section 4.1.1. Table C.1 in Appendix C shows the results after this percolation has been made for all workers.

All of the values in the table are a subset of the total amount of 4,175 blocks. Knowing the structure of the problem, one can deduce that the amount of available weekrest blocks are always less than or equal to the available weekday blocks, as it should be. The only difference between the two mentioned block categories is when a worker is free from work due to its weekrest. Therefore, using this information one can interpret that they are equal when the worker is never working weekends.

Looking at a generalized worker'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, are higher for weekend blocks. This is not the case as all blocks without weekend tasks are removed in the percolation. This means that all combinations with "No task" on weekends are removed, as well as the case that is shown in Table 4.6. The case in Table 4.6 occurs when a worker is assigned weekend Desk tasks and therefore can not be assigned any other task that Friday.

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

4.1.3 Rotation assignment

There are 35 weekend workers available of which 21 are librarians and 14 are assistants. The demand of weekend workers each week is seven, i.e. the demand for five weeks is exactly $7 * 5 = 35$ workers. Another requirement in excess to seven workers each weekend is that at least four of them have to be librarians due to three librarians are needed in Information desks and one in Hageby. Therefore, it deems reasonable to swap rotations between workers in the destroy so that it always remains feasible.

Table 4.5: Typical availability for a generalized worker. Yellow signifies that the worker is available. In parenthesis, the weekend shift hours.

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							
Weekrest 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		

A random generator is used in the assignment of rotations that always makes sure that the two mentioned requirements are met. Furthermore, all of the BokB-workers have fixed weekends and hence are not given new rotations in the destroy/repair loop and is more thoroughly described in Section 4.1.4 below.

Table 4.7 shows a destroy/repair iteration regarding rotation assignments. The amount of workers being destroyed in each iteration is three.

Yellow indicates that a worker, either librarian or assistant, has been destroyed that week and green indicates that a worker 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 amount of qualified workers remains the same after a repair. To understand this, imagine that

Table 4.7: An iteration in the destroy/repair loop showing a swap of weekends when three workers 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

two librarians with different rotations are destroyed, then two cases can occur: Either they are assigned the same rotation as before or they swap weekends.

4.1.4 Assignment of Library on Wheels

In order to avoid creating enormous amounts of block combinations, BokB tasks are assigned manually. To assign BokB manually also lead to a fix weekend rotation for the five BokB-workers. This slightly reduces the degrees of freedom of the problem as the BokB-workers' rotation will remain unchanged. However, as there are only a couple different set of feasible BokB assignments it shall not be the deciding factor for the quality of the solution.

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

4.1.5 Initial solution

The initial solution is created in a similar fashion as the repair function in this heuristic. Based on a greedy heuristic the best weekblock for a random worker and week is found and inserted. This is done until every worker have one weekend block, one weekrest block and three weekday blocks assigned to them.

To find the best weekblock several costs have been introduced to measure whether a block is good or bad to assign a worker. Say, if the library demands two librarians at the Information desk Monday at 08:00 and currently there are one assigned, then it would be good to assign another one. Such an assignment will, therefore, be rewarded using a cost. Good assignments will provide negative costs and bad assignments will provide positive costs to the objective function value.

Table 4.8 together with Figure 4.2 shows an increment where different cost parameters have to be considered when evaluating a block before assigning it to a worker. In order to calculate demand costs for the PL in the library, assume that this block which is being evaluated is to be inserted at week three.

Table 4.8: A block example to be evaluated using costs

	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

	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 many of the implemented costs must be carefully chosen. Presently, there are 16 existing costs where some of them are correlated with each other. The solution behaves differently depending on the relation between the costs changes.

Six of the costs are presented in Figure 4.2. They have all been assigned unique values so that, for example, positive assistant demand cost differs from negative assistant demand cost in absolute value. To explain why, one can imagine a case where four workers are demanded of which two have to be librarians, see Table 4.9. The first case, where one assistant and three librarians have been assigned, is more desirable than the case with three assistants and one librarian. In the first case it is still a feasible solution as the qualification of assistants are a subset of the librarians. The second case is feasible, however, not optimal due to the exceeding use of librarians. The optimal solution is shown in green as Case 3.

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

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

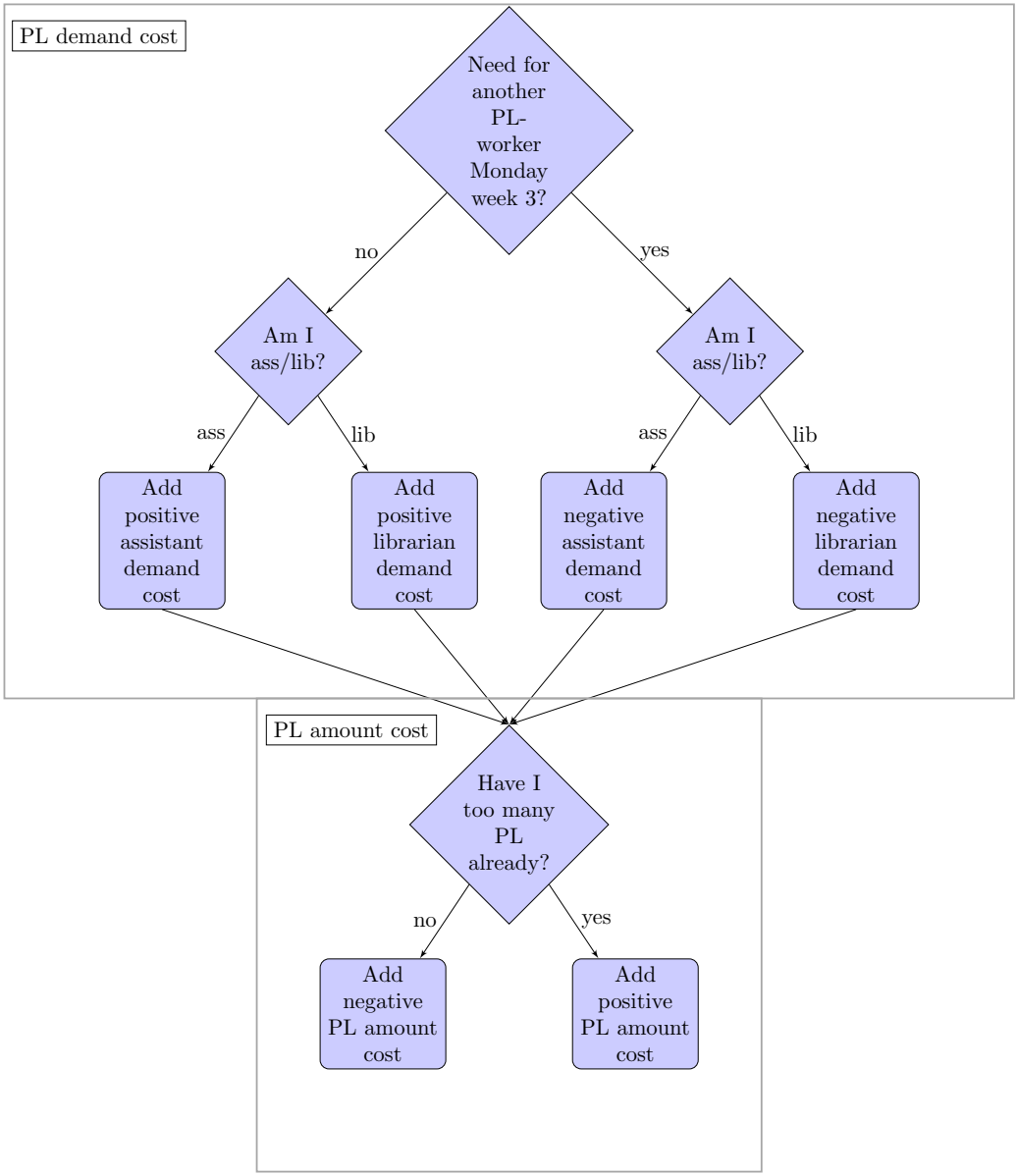
The complete list of costs with description can be seen in Table 4.10.

Whenever a new week block is to be inserted in a repair all available blocks for that worker has to be given a cost. This cost will be based on the current amount of workers assigned the tasks in the library. The block with the lowest cost will be the inserted block in the repair. An illustrative example can be seen in Table 4.11.

4.1.7 Destroy

The destroy consists of two contiguous steps: Choosing three workers to destroy and then destroying their assigned blocks. The BokB tasks remain unchanged even if a worker assigned those tasks is destroyed.

Figure 4.2: A flow chart of appearing costs when a PL is assigned a block on a Monday, third week relative to the library schedule.



4.1.8 Repair

Initially in the repair function the destroyed workers are assigned new rotations, see Table 4.7, followed by five new week blocks. Which order the blocks are inserted in is randomly generated. For each iteration is one of the three workers randomly chosen followed by one block that has not been reassigned until all 15 destroyed weeks have been repaired. Which week block that is chosen is based on the costs listed in Table 4.10 and shown as an illustrative example in Table

Table 4.10: List of all costs with description.

Demand costs	
Cost name	Description
Demand_few_ass	In need of more assistants to fill quota.
Demand_few_lib	In need of more librarians to fill quota.
Demand_many_ass	More assistants assigned than needed (redundancy).
Demand_many_lib	More librarians assigned than reference value.
Demand_few_total	Incorrect amount of workers assigned a task.
Demand_many_total	Incorrect amount of workers assigned a task.
Demand_evening_cost	Incorrect amount of workers assigned an evening task (more crucial).
Demand_PL_good_ass	Assigning a PL to an assistant when empty before assignment.
Demand_PL_good_lib	Assigning a PL to a librarian when empty before assignment.
Demand_PL_bad_ass	Assigning a PL to an assistant when already assigned by another.
Demand_PL_bad_lib	Assigning a PL to a librarian when already assigned by another.
PL amount costs	
Cost name	Description
PL_good_amount	Assigning PL when in need of more for feasibility.
PL_violate_amount	Assigning more PL than allowed to that worker.
Weekend costs	
Cost name	Description
HB_amount	No or more than one HB workers 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 assignment.

Table 4.11: Cost evaluation of two blocks. The lower the total cost is, the more desired to insert in repair (the values do not coincide with the implemented ones).

	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							

$$Totalcost = -Demand\_PL\_good\_lib + PL\_violate\_amount + Demand\_many\_lib - Demand\_few\_lib = 5,050$$

	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			

$$Totalcost = -Demand\_few\_lib - Demand\_evening\_cost + Demand\_many\_lib = -17,700$$

4.11.

4.1.9 Evaluation of solution

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

All costs are regarded in the evaluation function. If there are any demand differences in the library or any differences in amount of PL assigned to a worker these are given the respective costs. Furthermore, the amount of stand-ins is evaluated here provided with a negative cost showing that these are desired.

If there are no infeasibilities in the solution the evaluation function will return the value zero, unless there are stand-ins. In case there are stand-ins a negative value can be returned.

4.1.10 Final phase

At the end of a run there are mostly a few infeasibilities left that do not get solved. To fix this, a final phase was implemented. Whenever a run is "relatively close" to a solution and it is "theoretically possible" to find a solution the final phase is initiated. "Relatively close" means in this case that the only infeasibility is the lack of workers at a handful of weekday shifts. "Theoretically possible" means that there are more stand-ins available all days where the infeasibility occurs.

The final phase destroys one random week for a random worker and immediately repairs it, resulting in the same or better solution until it is feasible.

Due to well chosen cost parameter values the run will always eventually get relatively close and reach the final phase. If there are fewer stand-ins available a day where workers are lacking the solution will be discarded before entering the final phase.

4.2 Task distribution approach

The second heuristic approach tested distributes individual tasks to workers 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 distributed and in the second phase the rest of the weekday tasks are distributed.

The primary method used in this approach is a Large Neighbourhood Search (LNS) together with a Simulated Annealing (SA) accept function. Destroying and repairing the solution, as is customary in LNS, helps leading the solution out of local optima or plateaus. Similarly, SA is used in order to allow the solution to move in a less favorable directions to move out of local optima regions.

4.2.1 Costs

The search algorithm is guided using costs, both for worker schedules and the whole library schedule. Each worker has a number of costs, as displayed in Table 4.12. Each of these costs represents a badly placed 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 worker costs.

Table 4.12: Individual worker costs and cost weights.

Stand-in cost		
Cost	Weight	Cost description
C_{SI}	W_{w_SI}	Cost from having a task on a day where the worker can be a stand-in.
Infeasibility costs		
Cost	Weight	Cost description
C_{Task_D}	$W_{w\_Task\_D}$	Cost from all tasks exceeding the daily limit.
C_{Task_W}	$W_{w\_Task\_W}$	Cost from all tasks exceeding the weekly limit.
C_{PL_D}	$W_{w\_PL\_W}$	Cost from all PL exceeding the weekly limit.
C_{PL_Tot}	$W_{w\_PL\_Tot}$	Cost from all PL exceeding the total limit.
C_{SShift_W}	$W_{w\_SShift\_W}$	Cost from the number of task performed at the same shift in a week exceeding the weekly limit.

The weekend objective function, as referred to in the table, is associated with the weekend distribution phase of the problem. There are three types of costs in this objective function, each measuring a certain aspect of a good weekend schedule. Since we want to increase the number of stand-ins at the most critical days, the worst shift or day of 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 solutions for which the worst shift or day is identical. All costs in this objective function 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 a greater number of tasks. The weekend objective function becomes the sum of the weighted costs displayed in Table 4.13.

After a certain number of iteration the weekend phase is finished and the weekday distribution phase is entered. Here, the worker objective function is used to identify infeasibilities in the worker schedules and is simply a sum of the worker costs for all workers. 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 and is to be maximized.

4.2.2 Weekend phase

The weekend phase is the first phase the algorithm enters and it's flow is described also in Appendix B Figure B.2. The LNS component is part 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.

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 workers.
C_{D_m}	W_{D_m}	Cost of day with minimum number of workers.
C_{SI_a}	W_{SI_a}	Average number of stand-ins at a day.
C_{S_a}	W_{S_a}	Average number of workers at a shift.
C_{D_a}	W_{D_a}	Average number of workers at a day.
Worker Objective Function		
Cost	Weight	Cost description
C_{Task_D}	W_{Task_D}	Cost from all tasks exceeding the daily limit for all workers.
C_{Task_W}	W_{Task_W}	Cost from all tasks exceeding the weekly limit for all workers.
C_{PL_W}	W_{PL_W}	Cost from all PL exceeding the weekly limit for all workers.
C_{PL_Tot}	W_{PL_Tot}	Cost from all PL exceeding the total limit for all workers.
C_{SShift_W}	W_{SShift_W}	Cost from the number of task performed at the same shift in a week exceeding the weekly limit for all workers.
Weekday Objective Function		
Cost	Weight	Cost description
C_{SI_m}	-	Cost of day with minimum number of stand-ins.

The reason for implementing the weekend phase separate from the distribution of other tasks is because of the fact that the number of stand-ins in the final schedule depends to a large extent on the weekend-worker constellation. In particular, it is the week rest following upon weekend work which reduces the number of workers at a particular shift or 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 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:

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

where

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

and

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

T is the temperature of the SA accept function, which has an initial value of T_0 and which cools down with the iteration count 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 produced solution is infeasible, that is if the difference between the number of available workers at each shift and the number of required workers is negative. Such schedules are discarded and a new destroy and repair is performed. The number of available workers is calculated per shift and is the sum of all workers who are available at that shift and who do not have any tasks scheduled at that day.

The algorithms for distributing, 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 placing the whole schedule. Since the weekend objective function is only a measurement of the current number of stand-ins and available workers, this number will be greatly reduced when placing the weekday tasks.

Table 4.14: Worker availability placing only weekends. Intensity of red indicates number of workers.

	Num 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
	Num 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
	Num 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

In order to get a better measurement of the number of available workers during this phase, evening tasks and BokB tasks are distributed after each weekend repair. The evening tasks are distributed 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 distribution is visible in Tables 4.14 and 4.15. High intensity of red in a cell indicates a higher number of workers. It can be seen that the first table is generally redder and thus there are fewer available workers in the second table. This state reflects better the real number of available workers for a certain weekend constellation.

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

	Num 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

	Num 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

	Num 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

4.2.3 Weekday phase

When entering the weekday phase, all weekends, evenings and BokB tasks are already placed. In this phase, the rest of the tasks, referred to as "weekday" tasks are to be placed. The demand for a week at this point is illustrated in Table 4.16 and is identical for all weeks.The algorithm for distributing weekday tasks is found in Appendix B, Figure B.3, and greatly resembles the algorithm for distributing weekends.

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

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

When destroying and repairing tasks, the tasks 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 workers and the demand at each task. This makes sure that the most critical tasks are placed first.

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

Table 4.16: Worker 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

placed on the task.

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

Chapter 5

Results and discussion

5.1 Results

In the following sections, results from testing 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

Implementing the model in AMPL and running the model in CPLEX 12.5 gives the resulting values displayed in Table 5.1. The weights in equations 3.16 and 3.33 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 running the Mathematical Model in CPLEX.

	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 3.16, is zero.

5.1.2 Week block scheduling approach

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

Worth noting is that these values are most certainly not optimal as they have mostly been assigned using intuition. However, a few relations were determined by either running some tests or using reason. An example is either if $PL\_good\_cost > Demand\_PL\_good\_ass$ or $PL\_good\_cost > Demand\_PL\_good\_lib$. If so, all workers that can be assigned another PL will be, leaving the library

Table 5.2: List of all costs used in 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 amount 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

overstaffed with PL workers. The reason is because the net of $-PL\_good\_cost + Demand\_PL\_good\_ass/lib$ will be negative meaning it is a preferable insertion even though the library will be overstaffed.

Table 5.3 shows the amount of successful iterations on a run. This run is using the same cost parameters shown in Table 5.2.

Table 5.3: Amount of successful iterations

Successful iterations	420
Iterations (total)	638
%	∼66

Based on the same data as in Table 5.3 the average stand-in value per iteration can be calculated. It is seen in Table 5.4. A librarian is multiplied by a

Table 5.4: Average stand-in value per iteration

Stand-in value	105
Number of iterations	420
Average stand-in value	0.25

factor two and an assistant is multiplied by a factor one.

Having an average stand-in value of 0.25 means that most of the times no stand-ins are found. Table 5.5 shows statistics of how many stand-ins were found after 420 successful iterations.

Table 5.5: Amount of stand-ins found after 420 successful iterations.

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 solution is found. Every time the value skyrockets a solution has been discarded and a new iteration starts. This is done until a solution is found.

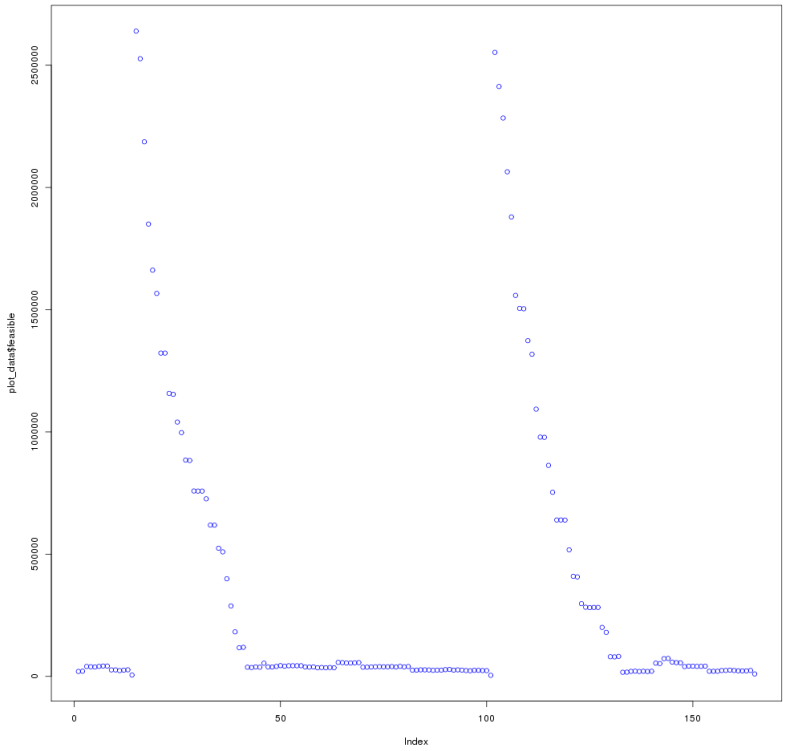


Figure 5.1: Plot of the objective function value after every new destroy and repair iteration.

5.1.3 Task distribution approach

In order to find a good value for the weights described in Section 4.2.1 in the previous chapter, different tuning tests were performed. These tests resulted in

the weights presented in Table 5.6. These weights are used in all test in this section unless stated otherwise.

Table 5.6: 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
Worker/Worker 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

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, while the latter three are simple 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 worker cost and the worker objective function costs measuring the same aspect of the schedule have been given the same weights, as is illustrated in Table 5.6. This is due to the fact that the 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 5.7 and 5.8. The results for weight tuning are displayed in Table 5.9.

Table 5.7: Results from the task distribution 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 5.8: Results from the task distribution 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 5.9: Results from the task distribution heuristic when varying weights. $It_{wend} = 1000$ 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 weekends 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 (Table 5.1).

Studying Table 5.7, 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 thus the result does not only depend on the number of iterations but also on chance.

Similarly, in Table 5.8, 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 stochastic 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 5.9 shows 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 bad 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 5.2. The effects of the SA accept function are visible in the plot as smaller or larger dives in the objective function value. In the measuring, 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 5.7.

In Figure 5.3 and Figure 5.4, the three different cost components from 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,

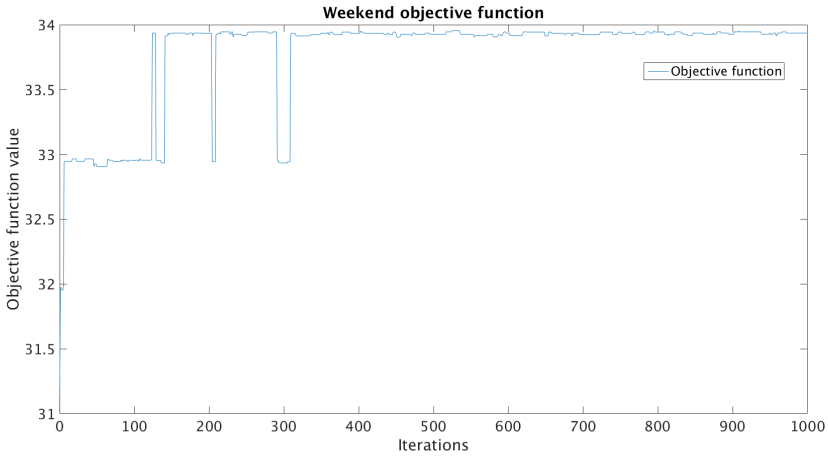


Figure 5.2: The weekend objective function for 1000 iterations

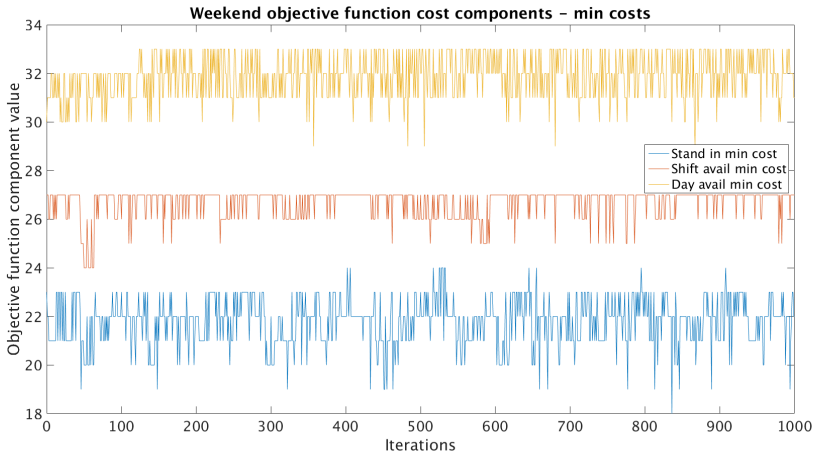


Figure 5.3: The weekend objective function minimum cost components for 1000 iterations

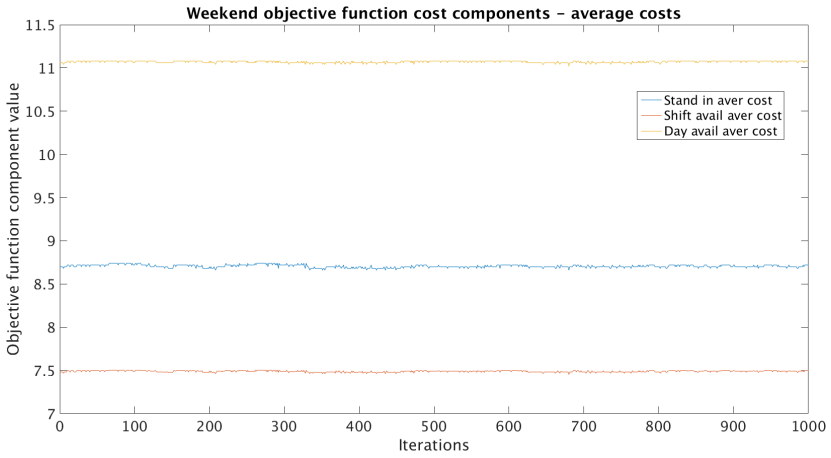


Figure 5.4: The weekend objective function average costs components for 1000 iterations

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 workers barely changes when shifting their weeks.

Attempts were made to correlating the min components of the weekend objective function value. However, the results were fluctuating between different tests so no clear correlations could be identified. This suggests that they are measuring three independent aspects.

5.2 Discussion

In this section, the different 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 5.10 lists pros and cons with the implemented week block scheduling approach. Some pros and cons were considered before the heuristic was chosen. Mainly the con concerning the exponential growth was taken into consideration as an estimation of the upper limit of the problem size was done. Also, 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.

Weekends, solution time and costs are no major issues as they can be avoided by a few smarter implementations. For instance, weekends can be improved by assigning values when each worker is available on a day and from those numbers create an even distribution of possible stand-ins and therefore increase the lowest value of stand-ins through the days. This shall implicitly decrease the solution

time as less iterations will be required. However, to always be able to create a pool of week appearances regardless of problem size can easily become a major issue. Just by adding meetings and the assignment of Library on Wheels tasks to the problem makes it grow considerably.

Table 5.10: Pros and cons with the implemented week block scheduling approach

Pros	Cons
The same amount of week block appearances will exist for five and ten weeks.	Weekends needs to be assigned in a more systematically way in order to achieve reasonable results regarding lowest amount of stand-ins through the days.
Quick iterations when destroying and repairing.	The amount of unique block appearances grows exponentially in case more task types are added, such as meetings.
	The solution time can vary considerably as several random generators have been used.
	A great deal of costs are needed (some correlated), where each of them affects the solution procedure.

5.2.2 Task distribution approach

The results from the previous section point to the fact that finding a good weekend distribution is essential in order to get a good final schedule. When It_{wend} is increased the results improve, up to a limit of around 500 iterations. The results are comparable to those of the AMPL implementation. This suggests that the weekend objective function used gives an indication of a good schedule. However, it only gives an indication, not an optimal solution, as the results seem to converge to a value slightly lower than the AMPL result.

Table 5.9 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 5.11.

Table 5.11: Pros and cons with the implemented task distribution approach

Pros	**Cons**
In most cases, the method finds the 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.	
The method can be used to find a good weekend schedule. The rest of the schedule can then be placed using another method such as CPLEX.	

Chapter 6

Concluding remarks

In this thesis work, a mathematical model for the given staff scheduling problem at the Library of Norrköping was developed. The model was run on the commercial optimization solver CPLEX, which generated an optimal schedule with three librarian stand-ins and zero assistant stand-ins at the worst day. This is better than the current schedule at the library, where only one staff member was 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 in the final schedule. The heuristic which focused on weekend scheduling separate from the rest of the problem thus performed better.

Although good results were obtained both by using the AMPL model and the second heuristic, there is room for further development in the modeling of the problem. One of the things which could be done in a more robust way is the modeling of exceptions and preferences among the staff members. In the current model, these are simply modeled as hard constraints. If these could be modeled in a more general way or in separation from the core problem, the model would not be so sensitive to changes in exceptions or staff preferences.

Several components of the model could be simplified. For example, the library on wheels could be modeled separately from the rest of the problem. Another suggestion is to simplify every other week staff memebers' schedules so that the person gets two five week schedules instead. Also, a more general way of describing meetings would simplify their distribution. One way of doing this could be to produce schedules which are not entirely feasible according to the problem description, but which can be used as a basis for manual scheduling.

Regarding the implementations, there are several components missing in order for the heuristics to fully correspond to the mathematical model. Although these features are not crucial, they could be implemented in order to make the heuristics correspond to the full problem. Furthermore, a more systematic framework for testing and evaluating the heuristics could be developed in order to be able to measure results which are more comparable.

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(1-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.

S. Pisinger, D. Ropke. *Large Neighborhood Search*, pages 399–419. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1665-5.

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
D	Set of days in a week
D_5	Set of all five weekdays
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_LOW	Set of librarians available to work in library on wheels
I_free_day	Set of workers that shall be assigned a free weekday per week
I_odd_even	Set of all workers with odd or even weeks
I_weekend_avail	Set of workers available for weekend work
I_big_meeting	Set of all workers that attend big meetings
I_no_PL	Set of workers not working on PL
I_weekend_avail	Set of workers working 3-4 times PL in 10 weeks
Dep	Set of departments
I_dep{Dep}	Set of workers in departments
V	Set of possible week rotations (shift the week by 1-10 steps)

A.2 Variables

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

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

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

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

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

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

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

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

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

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

$$m_{wdsD} = \begin{cases} 1, & \text{if a meeting is placed on week } w, \text{ day } d, \text{ shift } s \text{ at department } D \\ 0, & \text{otherwise} \end{cases} \tag{A.11}$$

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

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

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

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

A.3 Parameters

$$N1l = \text{a value to prioritize the amount of stand-in librarians} \tag{A.16}$$

$$N1a = \text{a value to prioritize the amount of stand-in assistants} \tag{A.17}$$

$$N1 = \text{a value to prioritize total number of stand ins} \tag{A.18}$$

$$N2 = \text{a value to prioritize similar weeks} \tag{A.19}$$

$$avail\_day_{iwd} = \begin{cases} 1, & \text{if worker } i \text{ is available for work week } w, \text{ day } d \\ 0, & \text{otherwise} \end{cases} \tag{A.20}$$

$$demand_{dsj} = \text{number of workers required day } d, \text{ shift } s \text{ for task type } j \tag{A.21}$$

$$qualavail_{iwdsj} = \begin{cases} 1, & \text{if worker } i \text{ is qualified and available week } w, \text{ day } d, \text{ shift } s \text{ for task type } j \\ 0, & \text{otherwise} \end{cases} \tag{A.22}$$

$$LOW\_demand_{wds} = \text{number of workers required day } d, \text{ shift } s \text{ at the library on wheels} \tag{A.23}$$

A.4 Objective function and constraints

$$maximize \quad M \cdot s^{min} - N \cdot \sum_{i \in I} \sum_{w \in 1..5} \sum_{d \in 1..5} \sum_{s \in 1..3} d_{iwds} \tag{A.24}$$

$$\sum_{i \in I} x_{iwdsj} = (1 - M_{wds}) demand_{dsj}, \ \forall w \in W, d \in 1..5, s \in S_d, j \in J_d \backslash \{L\} \tag{A.25}$$

$$\sum_{i \in I} x_{iwdsj} = demand_{dsj}, \ \forall w \in W, d \in 6..7, s \in S_d, j \in J_d \tag{A.26}$$

$$\sum_{i \in I} x_{iwdsL} = LOW\_demand_{wds}, \ \forall w \in W, d \in 1..5, s \in S_d \tag{A.27}$$

$$\sum_{d \in 1..5} \sum_{s \in S_d} \sum_{j in J_d} x_{iwdsj} \leq 4, \ \forall i \in I \backslash \{36\}, w \in W \tag{A.28}$$

$$\sum_{w \in 1..5} M_{w11} = 1 \tag{A.29}$$

$$M_{(w+5)11} = M_{w11} \tag{A.30}$$

$$\sum_{w \in W} \sum_{d \in 1..5} \sum_{s \in S_d} M_{wds} = 2 \tag{A.31}$$

$$\sum_{s\in 2..3}\sum_{j\in J_d\backslash\{L\}} x_{iw1sj} \leq 1 - M_{w11},\ \forall i \in I\backslash I\_big, w \in W \tag{A.32}$$

$$\sum_{w\in 1..5}\sum_{d\in 1..5}\sum_{s\in 1..3} m_{wdsD} = 1,\ \forall D \in Dep \tag{A.33}$$

$$m_{(w+5)dsD} = m_{wdsD},\ \forall D \in Dep, w \in 1..5, d \in 1..5, s \in 1..3 \tag{A.34}$$

$$m_{wdsD}+x_{iwdsj} \leq 1,\ \forall D \in Dep, i \in I\_dep\{Dep\}, w \in W, d \in 1..5, s \in 1..3, j \in J_d \tag{A.35}$$

$$m_{wd2D}+m_{wd3D}+x_{iwd1P} \leq 1,\ \forall D \in Dep, i \in I\_dep\{Dep\}, w \in W, d \in 1..5, s \in 1..3 \tag{A.36}$$

$$m_{wdsD} \leq \sum_{v\in V} r_{iv}\cdot qualavail_{i(mod_{10}(w-v+10)+1)dsE},\ \forall D \in Dep, i \in I\_dep\{Dep\}\backslash\{39\}, w \in W, d \in 1..5, s \in \tag{A.37}$$

$$M_{w11} + \sum_{d\in 1..5}\sum_{s\in 1..3} m_{wdsD} \leq 1,\ \forall D \in Dep, i \in I\_big \cup I\_dep\{Dep\}, w \in W \tag{A.38}$$

$$\sum_{j\in J_d} x_{iwdsj} \leq 1,\ \forall i \in I, w \in W, d \in D, s \in S_d \tag{A.39}$$

$$\sum_{s\in S_d}\sum_{j\in J_d\backslash\{L\}} x_{iwdsj} \leq 1 - x_{iwds_2L},\ \forall i \in I, w \in W, d \in 1..4, s_2 \in S_d \tag{A.40}$$

$$\sum_{s\in S_5}\sum_{j\in J_d\backslash\{L\}} x_{iw5sj} \leq 1,\ \forall i \in I, w \in W \tag{A.41}$$

$$\sum_{j\in J_d\backslash\{L\}} x_{iw5sj} \leq 1 - x_{iw51L},\ \forall i \in I, w \in W, s \in 1..3 \tag{A.42}$$

$$\sum_{s\in S_d}\sum_{j\in J_d} x_{iwdsj} \leq 1,\ \forall i \in I, w \in W, d \in 6..7 \tag{A.43}$$

$$\sum_{d\in 1..4}\sum_{j\in J_d} x_{iwd4j} \leq 1,\ \forall i \in I\backslash\{36\}, w \in W \tag{A.44}$$

Appendix B

Flow charts

B.1 Weekly scheduling approach

B.2 Task distribution approach

Figure B.1: A flow chart of the implemented heuristic with weekly scheduling

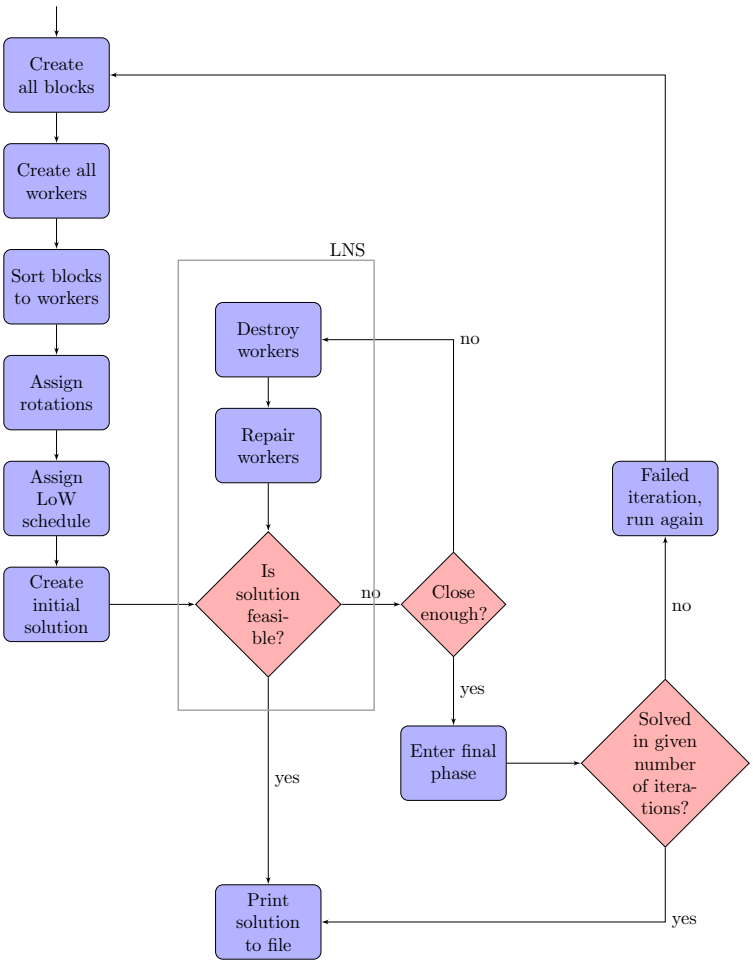


Figure B.2: Algorithm for distributing weekends.

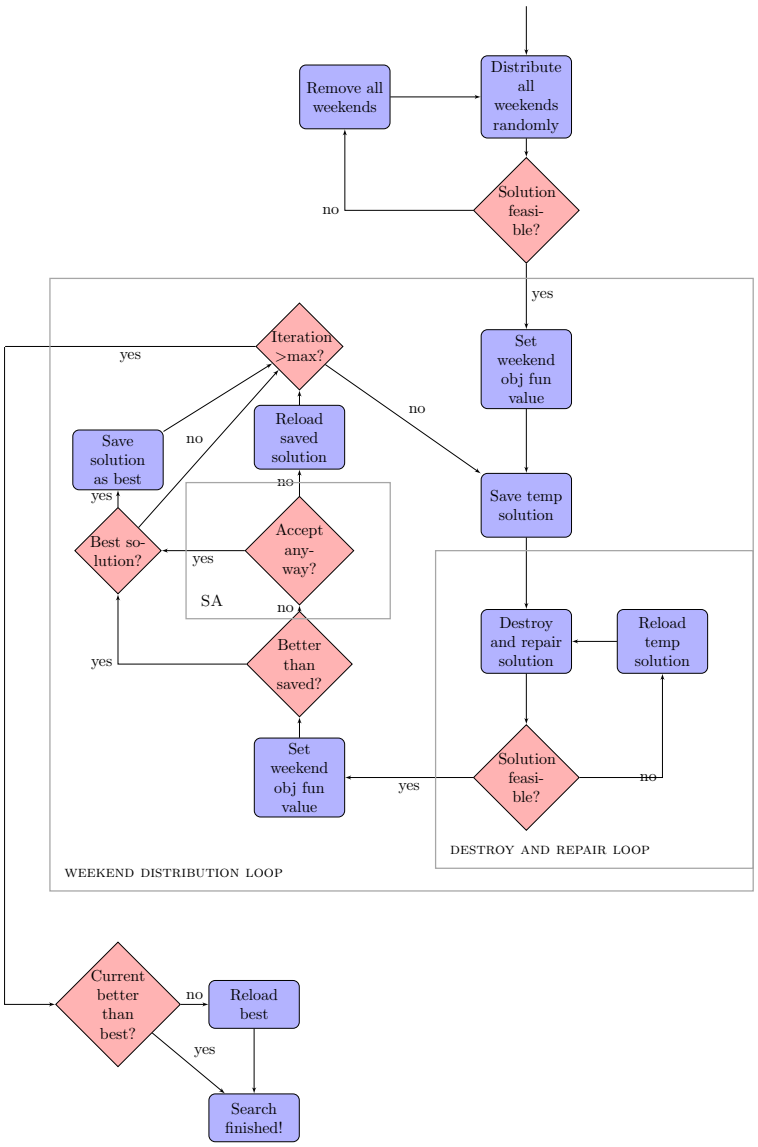
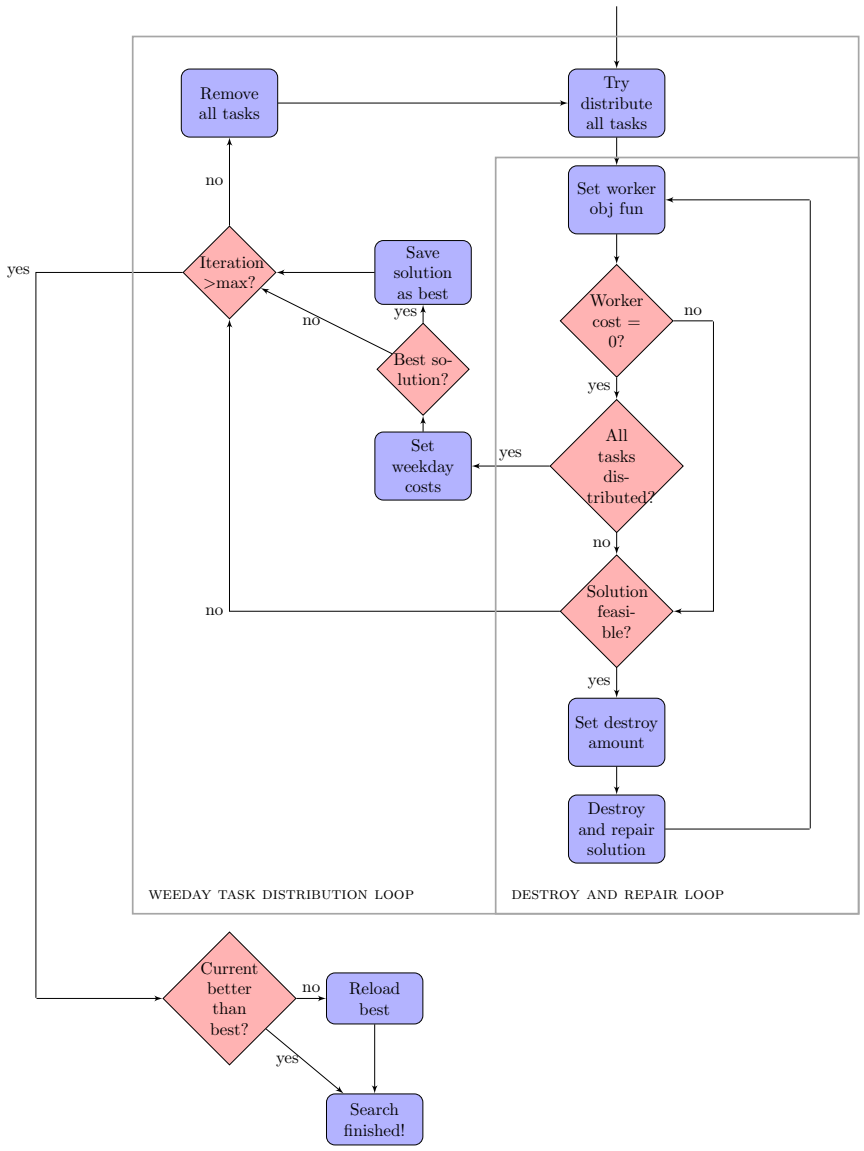


Figure B.3: Algorithm for distributing weekday tasks.



Appendix C

Weekblock table

Table C.1: Number of assignable unique blocks for the workers based on their availability and qualification.

Worker	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