DEGREE PROJECT, IN MEDICAL ENGINEERING , SECOND LEVEL
STOCKHOLM, SWEDEN 2015

Algorithm Construction for Efficient Scheduling of Advanced Health Care at Home

TONIMA AFROZE, MOA ROSÉN GARDELL

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF TECHNOLOGY AND HEALTH

Abstract

Providing advanced health care at home rather than in a hospital creates a greater quality of life for patients and their families. It also lowers the risk of hospital-acquired infections and accelerates recovery. The overall cost of care per patient is decreased. Manual scheduling of patient visits by health care professionals (HCPs) has become a bottleneck for increased patient capacity at SABH, a ward providing advanced pediatric health care at home (“Sjukhusansluten Avancerad Barnsjukvård i Hemmet” in Swedish), since many parameters need to be taken into account during scheduling. This thesis aims to increase the efficiency of SABH’s daily scheduling of personnel and resources by designing an automated scheduler that constructs a daily schedule and incorporates changes in it when needed in order to remove scheduling as a limitation for increased patient capacity. Requirements on a feasible schedule are identified in cooperation with SABH and literature is investigated about similar areas where the scheduling process has been automated. The scheduling is formulated as a computerized problem and investigated from the perspective of theoretical computer science. We show that the scheduling problem is NP-hard and can therefore not be expected to be solved optimally. The algorithm for scheduling the visits minimizes violations of time windows and travel times, and maximizes person continuity and workload balancing. The algorithm constructs an initial solution that fulfills time constraints using a greedy approach and then uses local search, simulated annealing, and tabu search to iteratively improve the solution. We present an exact rescheduling algorithm that incorporates additional visits after the original schedule has been set. The scheduling algorithm was implemented and tested on real data from SABH. Although we found the algorithm to be efficient, automatic transfer of data from the patient journal system is an imperative for the scheduler to be adopted.

Keywords: advanced health care at home, home assistance, vehicle routing problem, time windows, scheduling, rescheduling, routing, optimization

Sammanfattning

Barn som får avancerad sjukvård hemma istället för på sjukhus tillfrisknar ofta snabbare och risken för vårdrelaterade infektioner minskar. Barnen och deras familjer blir mer välmående av att få vistas i sin hemmiljö. På Astrid Lingrens barnsjukhus i Stockholm erbjuds avancerad hemsjukvård av avdelningen Sjukhusansluten Avancerad Barnsjukvård i Hemmet (SABH). För att schemalägga när patienterna ska besökas av sjukvårdspersonalen behöver många olika faktorer beaktas, detta sker idag helt manuellt. Den manuella schemaläggningen utgör en naturlig begränsning av SABHs patientkapacitet. Denna uppsats syftar till att effektivisera schemaläggningsprocessen hos SABH genom att föreslå en automatiserad lösning som hanterar koordinering av personal och resurser och dem förändringar som behöver göras i schemat under dagen, för att få bort schemaläggningsprocessen som ett hinder mot ökad patientkapacitet. Krav på schemaläggningen identifieras i diskussion med SABH och genom att studera litteratur kring liknande områden där schemaläggning lösts automatiserat. Vi formulerar schemaläggningen som ett datologiskt problem och analyserar det med utgångspunkt i teoretisk datalogi. Vi visar att problemet är NP-svårt och därför inte kan förväntas lösas optimalt inom rimlig tid. Vår lösning approximerar istället fram ett rimligt svar, där fokus hos algoritmen är att patienterna ska besökas de tider de behöver, personalens restider ska vara så korta som möjligt samtidigt som arbetsbördan hos personalen ska vara så lika fördelad som möjligt och patienterna ska, i den mån det är möjligt, få vård av samma personal. Med en girig algoritm konstrueras ett initialt schema som uppfyller de grundläggande kraven, detta schema förbättras med lokalsökning, simulated annealing och tabusökning. En exakt lösning framställs för uppdatering av schemat. Algoritmen för att lägga ett dagligt schema (utan uppdateringar) implementerades och testades med riktigt data från SABH. Vår algoritm visade sig vara effektiv, men för att kunna göra hela schemaläggningsprocessen effektiv behöver den integreras med journalsystemet.

Nyckelord: avancerad sjukvård i hemmet, hemtjänst, schemaläggning, omplanering, ruttoptimering, optimering

Acknowledgments

We would like to express our sincere thanks to SABH at Karolinska University Hospital for making this thesis work possible, letting us visit you and giving indispensable insight into your work and commenting on our thesis work.

We offer our special thanks to Prof. Viggo Kann for your continuous encouragement and valuable suggestions.

We thank Prof. Kristina Groth and Per Englund at the Innovation Center of Karolinska University Hospital for introducing this project to us and for supervision during the process.

We express our appreciation to Softronic for support during the implementation phase and especially thank Kristian Samuelsson for constructive advice and guidance.

Thanks to Dr. Zarina Nahar Kabir, Prof. Mattias Gardell and Anthony Fiscella for all the time you have spent proofreading our thesis and for your valuable suggestions.

I, Tonima, thank my family for all your love and support and for putting up with the thesis work being my (almost) sole topic of conversation for the past months.

I, Moa, would like to give my sincere thanks to my family, particular my mother Yvonne, without you this thesis had not been possible and a special thanks to the light in my life, my children Yassin and Amir, for your endless source of strength, joy and love.

Tonima Afroze
Moa Rosén Gardell

Contents

1 Introduction **1**
- 1.1 Aim and Objectives . . . 1
- 1.2 Description of the Ward . . . 2
- 1.3 Automated Scheduler Components . . . 3
- 1.4 Problem Formulation . . . 4
  - 1.4.1 Daily Scheduling Problem . . . 5
  - 1.4.2 Rescheduling Problem . . . 6
- 1.5 Limitations . . . 6
- 1.6 Input Parameters . . . 7

2 Theory **9**
- 2.1 Complexity Theory . . . 9
- 2.2 Approximative Solutions . . . 11
  - 2.2.1 Heuristics . . . 12
- 2.3 Scheduling Problems for Advanced Health Care at Home . . . 12

3 Method **15**
- 3.1 Testing . . . 15

4 Results **17**
- 4.1 Proof of NP-completeness . . . 17
  - 4.1.1 Daily Scheduling Problem $\in$ NP . . . 17
  - 4.1.2 Daily Scheduling Problem is NP-hard . . . 18
  - 4.1.3 Daily Scheduling Problem is NP-complete . . . 18
  - 4.1.4 Rescheduling Problem is NP-complete . . . 19
- 4.2 Modeling Data . . . 19
- 4.3 Algorithm . . . 21
  - 4.3.1 Initial Solution of the Daily Scheduling Problem . . . 21
  - 4.3.2 Mathematical Functions . . . 22
  - 4.3.3 Local Search Algorithm of the Daily Scheduling Problem . . . 24
  - 4.3.4 Rescheduling Algorithm . . . 26
- 4.4 Evaluation of Algorithms . . . 28
  - 4.4.1 Complexity . . . 28
  - 4.4.2 Solution's Distance from the Optimal Solution . . . 29
- 4.5 Testing . . . 29
- 4.6 Presentation at the Ward . . . 30

5 Discussion **33**
- 5.1 Improving the Algorithms . . . 33
- 5.2 Testing Method . . . 34
- 5.3 Confidentiality . . . 34
- 5.4 Usefulness of the Automated Scheduler . . . 35

5.5 Risks and Vulnerabilities 36

6 Conclusion **39**

A Database Schema **43**

List of Figures

1 Components of a fully functional automated scheduler. 4
2 The structure of complexity classes. 9
3 The idea behind Karp reduction. 11
4 Illustration of a synchronization relation 20
5 Illustration of a precedence relation 20
6 Sequence of algorithms 21

List of Tables

1 Input parameters to the SABH scheduling problem 7
2 Constraints addressed in HHCSP papers. 13
3 Optimization parameters included in HHCSP papers. 14
4 Notation used for modeling DSP and RSP. 20
5 Database Schema 43

List of Algorithms

1 Reduction 19
2 Initial Solution 22
3 Local Search 25
4 Rescheduling 27

Abbreviations

DSP Daily Scheduling Problem.

HCP Health Care Professional.

HHCSP Home Health Care Scheduling Problem.

NP Nondeterministic Polynomial time.

P Polynomial time.

RSP Rescheduling Problem.

SABH Hospital managed advanced pediatric home care ward ("Sjukhusansluten Avancerad Barnsjukvård i Hemmet in Swedish).

VRPTW Vehicle Routing Problem with Time Windows.

Glossary

Complexity class Set of problems with related complexity.

Coordinator The person who is responsible for the manual scheduling currently and will manage the automated scheduler.

County An administrative division of Sweden.

DSP The problem of creating a schedule for a set of given visits with additional constraints.

Efficent algorithm An algorithm whose complexity is polynomial in time or space with respect to the input size to the algorithm.

Node A treatment that is needed for a patient. A node requires the participation of exactly one HCP. A visit can therefore be represented by a combination of multiple nodes.

NP A class of problems for which a solution can be verified in polynomial time.

NP-completeness A class of problems that belong to NP and are NP-hard.

NP-hard A class of problems that are least as difficult to solve as any problem in NP.

P A class of problems that can be solved in polynomial time.

Person continuity The number of HCPs that each patient meets.

Precedence relation A temporal relation between two nodes where one node must be performed a certain period of time before another.

SABH Ward at the hospital for which the scheduler is being designed.

RSP The problem of incorporating new visits into an existing schedule.

Space An amount of time within the HCP's working hours when no visit is scheduled/Unscheduled time, where unanticipated visits can be placed.

Synchronization relation a relation between two nodes where the two have to performed at the exact same time.

Time duration Time required to perform a visit.

Time window Specifies the earliest and latest time that a visit can start.

Visit A treatment that is needed for a patient. A visit can require the participation of multiple HCPs.

1 Introduction

Treating patients at home creates a social value for the patient and its family. A home environment provides comfort through audiovisual and olfactory familiarity. It can provide a sense of security and deflect the focus from the illness, and can accelerate recovery. The patient can participate in everyday family life and family members can take part in both household activities and tend to the patient. Avoiding hospitals can prevent the acquisition of infections. Manual scheduling of patient visits by health care professionals (HCPs) has become a bottleneck for increased patient capacity at SABH, a ward providing advanced pediatric health care at home ("Sjukhusansluten Avancerad Barnsjukvård i Hemmet" in Swedish) in Stockholm, Sweden. In this thesis we describe an automated scheduler that is designed to increase efficiency of daily scheduling of personnel and resources.

We continue this section by describing the ward, the components required for an automated scheduler to be ready for adoption and define the SABH scheduling problem in detail, which we separate into two problems: daily scheduling (DSP) and rescheduling (RSP). In section 2 we proceed to explain the required concepts of theoretical computer science needed and describe previous research on scheduling problems for advanced health care at home. Our project methodology and methods of testing the algorithm are described in section 3. In section 4 we analyze the complexity class of DSP and RSP. We describe our algorithms and evaluate them theoretically, through testing on real data, and via a presentation at the ward. In section 5 we discuss how the automated scheduler affects patient safety, confidentiality and employees' work environment, and mention extensions of our work. Finally, in section 6 we make concluding remarks about the impact of the scheduler.

1.1 Aim and Objectives

The aim of this thesis is to increase the efficiency of SABH's daily scheduling of personnel and resources by designing an automated scheduler. This will allow an increase in patient capacity which will lead to an improved quality of life for more patients and their relatives. This thesis will address the following objectives:

i) identify the SABH scheduling problem,

ii) prove which complexity class[1] the problem belongs to,

iii) propose an algorithm that solves the problem,

iv) implement a prototype based on the algorithm in collaboration with SABH,

v) evaluate the usefulness of the prototype in the intended environment of use.

[1]This concept is described in section 2.1.

1.2 Description of the Ward

SABH was founded 1996 as a project and has been shown to be cost-effective. The care performed by SABH is at least 30 percent lower in cost than the same care performed at Astrid Lindgren's Children's Hospital, the estimated savings per year is ten million Swedish crowns [1]. The cost for the patient's families as well as the total cost of the society are decreased, as the parents do not have to live at the hospitals and can tend to their other daily activities. SABH plays an important role in relieving the beds within the hospital [2].

SABH offers health care in the home by sending health care personnel to the patient's home. Mobile care teams consisting of a variety of health care professionals (HCPs) such as nurses, physicians and child carers, visit the patients regularly, and are available 24 hours a day for questions or emergency visits. There is a central coordinator that coordinates the patient visits, communicates with patients and their families as well as wards at the hospital, and makes decisions regarding new patients and care of the patients [3].

Patients admitted to SABH are critically ill. The patients range in age from infant to 19 years and patient categories include, among others, newborn children that need light therapy to treat jaundice, premature children that need nutritional support or with congenital malformations or diseases, children with acute diseases such as pneumonia, gastro-intestinal diseases, burns, rehabilitation at home after surgical procedures and palliative care. Patients will only be admitted to SABH if it is medically safe to receive care at home and if the patient would otherwise be admitted to the hospital. Other criteria are that the parents have chosen this type of care and can manage taking care of the patient at home, and that the travel time between the home and the hospital is reasonable [3].

The mobile care teams can perform multiple types of monitoring and treatments, based on the individual patient's needs. They offer services in monitoring patients' condition, recovery, and growth process. Injections and infusions are administered. Lactation and nutrition support, light and oxygen therapy and pain relief are performed. In addition to taking care of the patient's physical health, SABH also considers the patient's mental health and psychosocial situation by offering support and advice to the parents. SABH offers help in processing the care situation for both the patients and their relatives. They put effort in building up the parents' trust and making them comfortable with the home care. Addressing the psychosocial factors improves the mental health of the patients, which positively impacts physical health [2]. Some types of therapy cannot be performed at the patients' homes, and the patients need to visit the remitting ward or SABH's HCPs at the hospital.

SABH accounts for ten percent of Astrid Lindgren children's hospital's inpatient capacity. Although the patients are treated in their homes, they are technically admitted to the hospital and hence the requirements for the quality of care are equivalent to those at the hospital. Patient can be admitted at the ward for a few hours up to a couple of months. The mean length of stay is 6–7 days. Currently, the team of personnel at SABH includes 4 physicians, 14 nurses and 10 assistant nurses, and treats 25–30 patients per day [3]. During a conversation at SABH (Rylander E 2015, secton head at SABH, oral communication, 10th February) we were told that creating a daily schedule and updating it to incorporate necessary changes is easily done for 15 patients, but it is considered difficult and stressful to do for 25 patients. Due to the benefits of advanced health care at home, an increase in patient capacity at SABH is desired, but manual scheduling has been identified as a bottleneck. Therefore, an automated scheduler would facilitate daily planning.

1.3 Automated Scheduler Components

In order to obtain a fully functional automated scheduler, all of the components in figure 1 are necessary. These components relate to patients' needs, resource availability and presentation of the schedule. The program that runs the scheduling algorithm is a core component of the system. Its output is suggested schedules to the information storage system, based on received input. Patient information such as name, address, and care plan is gathered from the common county journal system and the ward's own database used for saving additional information. The staff schedules are obtained from the personnel work schedule database and travel times between the patients' homes are acquired from an online routing service. To rearrange the schedule of the staff who are already performing visits, their geographic positions are gathered using GPS. A user interface for the coordinator is needed for exchanging data.

An interface that informs staff members of their daily schedules, the schedule for the whole ward and changes in the schedule would further improve the system. To secure patient safety, it is important to ensure that messages about the changes are received by the staff and a confirmation of receipt can be communicated. Another possible improvement is to include a user interface for patients which informs them of the time period during which they will be visited and by whom.

The algorithm, information storage, and an integration with a travel routes calculator were implemented.

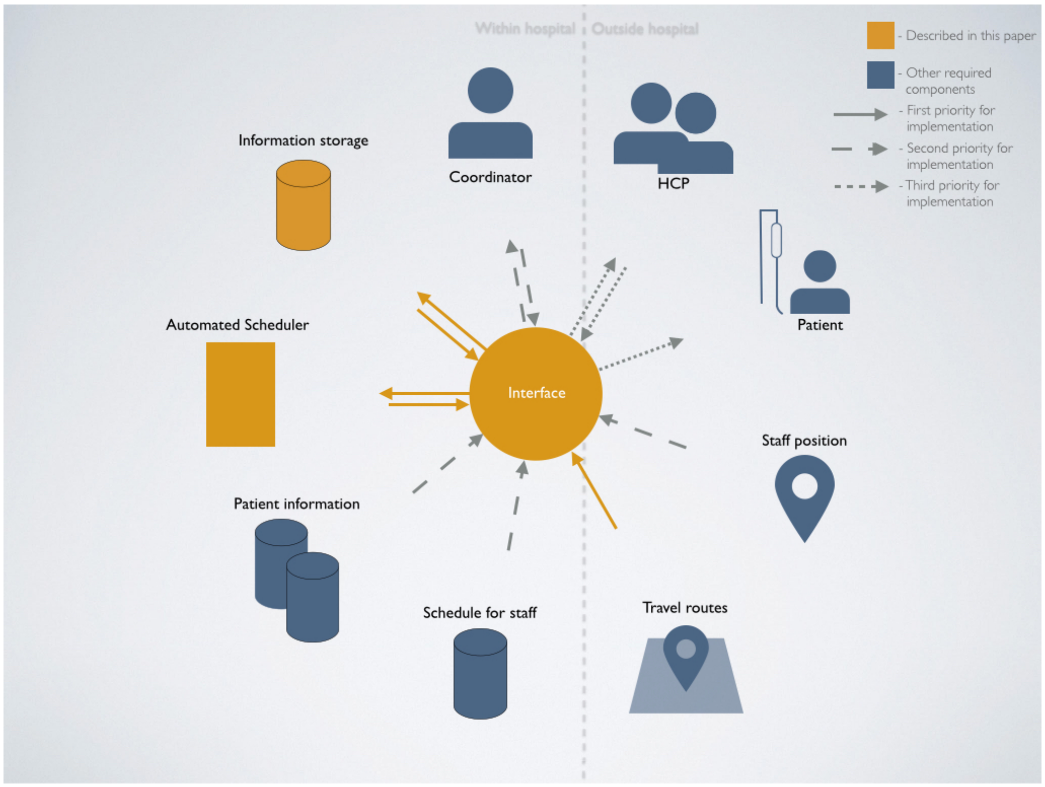

Figure 1: Components of a fully functional automated scheduler. The dashed line separates system components within and outside the hospital network. Confidential data inside the hospital are controlled by established safety routines, while the confidentiality outside of the hospital needs to be ensured.

1.4 Problem Formulation

Many different variables need to be considered in order to guarantee patient safety while delivering medical treatment and care in patients' homes [2] instead of at the hospital.

The coordinating nurse plans the distribution of personnel, medical equipment and cars to match the available resources with the resources needed for the patients' treatments. This combination is noted on a large whiteboard located in the ward. The process is manually administered: the coordinator gets information from the two patient information systems together with "non-digital" lists of available resources and previously documented information about the patients. The travel times are estimated based on experience of road conditions and traffic patterns.

[2]spread over Stockholm County (an area of 6526 km^2 [4])

We divide the scheduling problem into two distinct problems, these will henceforth be referred to as the daily scheduling problem (DSP) and the rescheduling problem (RSP).

Our algorithms fulfill the constraints listed in section 1.4.1 and 1.4.2. They will

- schedule up to 200 patients per day,
- use input parameters and output a set of tasks for each HCP to do during his/her shift, where start time for each visit is specified
- not specify how personnel should use open spaces in the schedule

1.4.1 Daily Scheduling Problem

The constraints that need to be satisfied in DSP are that

- all visits are assigned to the required number of HCPs,
- visits with temporal precedence relations are performed within the required time span from each other,
- patients are visited within their given time windows[3],
- assigned HCPs have the right competence for providing the required treatment needed for the visit,
- there is enough[4] space [5] for adjustments.

In addition to the constraints the objectives are that

- person continuity is maximized[6],
- the total travel time of the HCPs is minimized,
- workload of the HCPs is balanced.

Although HPCs try to schedule visits with their patients during their working hours, for regular visits with rigid time windows, it is not always possible to assure person continuity in this way – other HCPs may need to perform the visit. The personnel work in three shifts and the coordinator is available only during the day shift. The scheduling is updated continuously during the day – the coordinator sets a rough schedule at the end of the shift and rearranges it the following day incorporating new information that might have come during the night. When the coordinator is not available, one of the nurses is in charge of the phone and handling unplanned events.

[3]Time windows specify the earliest and latest time that a visit can start.

[4]predefined amount

[5]An amount of time within the HCP's working hours when no visit is scheduled

[6]The number of HCPs that each patient meets.

1.4.2 Rescheduling Problem

Input to RSP is the output from DSP, a list for each HCP with visits to perform at certain times. Assigned to each route/HCP are also a vehicle and a list of medical equipment. It is known for which visit they are needed, which visits that have been performed, if they are being performed at the moment as well as the location of the HCPs. There is also new information that needs to be incorporated in the existing schedule where minimizing the rearrangements is an additional optimization goal to the daily schedule.

There are many reasons why the schedule needs to be continuously updated during the day. For example; staff calls in sick, a patient needs an unplanned visit, a planned treatment requires more personnel than originally allocated or a new patient is admitted. New visits may need to be incorporated into the schedule which may result in the need of a rearrangement.

1.5 Limitations

The scope of this project has been limited in the following ways. Due to time constraints, the focus of this project was on designing an effective algorithm and creating a prototype for it – not a fully functional application. Since implementation is time consuming, only the scheduling (not the rescheduling) algorithm was implemented.

The algorithms are designed to schedule visits for one ward. Interaction with SABH for understanding the problem, designing and testing was limited by the availability of personnel at the ward. The implementation was done as a proof of concept for the intended use of the algorithm, which strongly limits the generalizability of the implemented version of the algorithm. Some of these limitations are discussed in section 5.

Algorithms to solve DSP and RSP were designed. The algorithm for DSP was implemented.

1.6 Input Parameters

The input parameters to the algorithm are listed in Table 1.

Parameter	Description
$H = \{1, ..., m\}$	Set of HCPs
w_h	Working day for HCP h ($h \in H$)
$prof_h$	Profession of h ($h \in H$)
$\mathbb{P}$	Set of patients
V_p	Set of visits for patient p ($p \in \mathbb{P}$)
$tw^e_{v_p}$	Earliest time v_p can be started ($v_p \in V_p$)
$tw^l_{v_p}$	Latest time v_p has to be started ($v_p \in V_p$)
td_{v_p}	Time duration for v_p ($v_p \in V_p$)
ME_{v_p}	Set of medical equipment needed for v_p ($v_p \in V_p$)
$PROF_{v_p,y}$	Set of HCPs needed for v_p,including their profession respectively ($v_p \in V_p$)
$V_{p,prec}$	Visits V_{prec} have precedence relations to v_p ($V_{prec}, v_p \in V_p$)
VE	Set of vehicles
ME	Set of medical equipment
$D = d(a, b)$	Set of travel times between each patients a and b
$CONT = continuity_{p,h}$	Set of relations between each p and h

Table 1: Input parameters to the SABH scheduling problem

2 Theory

In this chapter we explore concepts of theoretical computer science including complexity theory explaining NP-complete problems, approximative solutions to solve them and current research on home health care scheduling problems. We describe similarities and differences between them and the scheduling problem.

2.1 Complexity Theory

There are several kinds of computerized problems, where it is common that a problem can be considered as a function, that takes inputs (the problem *instances*) and returns an output (the *solution* of the problem) [5, p. 1]. The *complexity* of an algorithm is a measure of how much resources in terms of time and space the algorithm needs to calculate the output, in relation to the input size given to the algorithm [6, p. 29] The complexity can be stated with an asymptotic upper bound (O notation) which states the largest amount of time or space that the algorithm may allocate [6, p. 36–37].

There has been a lot of research about different types of problems, finding algorithms to solve them and trying to decrease the complexity of those algorithms [5, p. 10–11]. According to these research the general complexity needed to solve problems has been discovered and *complexity classes* have been defined to categorize them, see figure 2. The limits between the classes are not strict – some of them are vague and it is uncertain whether there is a distinction between some classes. Classes with lower complexity can (but do not have to) be subclasses of those with higher complexity [5, p. 10–11]. When an algorithm is to be designed to solve a problem, it can be suitable to define which complexity class the problem belongs to. If the complexity class of a problem is known, one can know what kind of complexity one is expected to get from the algorithm. In many applications algorithms with low complexity are preferred even though they give an approximative solution instead of the optimal [6, p. 452].

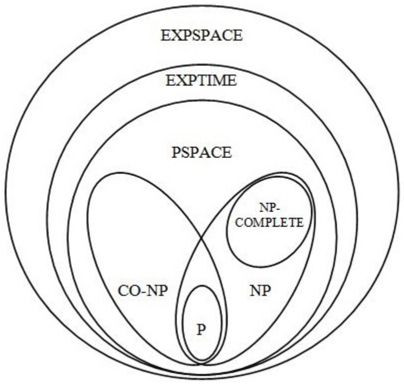

Figure 2: The structure of complexity classes.

The complexity classes P, NP, NP-hard and NP-complete are now to be described. One definition of an efficient algorithm is that the complexity of the algorithm is in polynomial time or space according to the input size to the algorithm [6, p. 33], this definition will be used further on. One complexity class is polynomial time (P). It means that there are algorithms that solve problems within this class that have polynomial complexity, and according to our definition of efficiency it means that the problem can be solved efficiently [6, p. 465]. Many problems are formulated as optimization problems, where the objective is to find the minimum or maximum of some entity according to some criterion. This optimization problem can be reformulated as a decision problem, where the question is if it is possible to get a specific number of that entity fulfilling some criteria. The answer to the problem is either yes or no [6, p. 454]. If there is a polynomial way to verify that the answer is yes, given an instance to a problem and a solution that are said to fulfill the criteria, then the problem belongs to the class nondeterministic polynomial time (NP). P is suspected to be a subclass of NP. It is not proven that the classes are distinct. They might be identical. However all research indicates that there is a distinction [6, p. 464–465].

A problem is defined to be NP-hard if it is at least as difficult to solve as any problem in NP [5, p. 30]. If a problem is both NP-hard and belongs to NP the problem is said to be NP-complete [5, p. 21]. By definition it follows that all NP-complete problems are connected in the sense that if there is a way to efficiently solve one of them, it implies that all of them can be solved efficiently. There might exist such an efficient algorithm (the contrary has not been proven), but regardless of all efforts that have been put into finding one, no such solution has been found [6, p. 451–452].

To prove that a problem, X, is NP-hard, one makes a polynomial Karp reduction of another already proven NP-hard problem, Y, to X. The steps of the reduction are illustrated in figure 3. A black-box, or an oracle, is introduced that, given correct input, returns a solution to X. The input to Y is rearranged to the input needed for X, so the answer that is given from the oracle corresponds to the answer of Y, for every correct instance of Y. This shows that if it is possible to solve X efficiently, it is also possible to solve Y efficiently, and since Y is believed to be hard to solve then X is at least as hard to solve as Y. The rearrangement of the input from Y to the input to X needs to be done in polynomial time, otherwise the reduction itself will be inefficient and the proof falls [6, p. 453–454].

To prove that the reduction is correct one needs to consider the complexity of the reduction (it needs to be polynomial) and prove that the answer to X is always identical to the answer of Y, for all instances that go through the reduction (i.e. it is not needed to consider instances of problem X that never will be input from the reduction of problem Y) [5, p. 17–18].

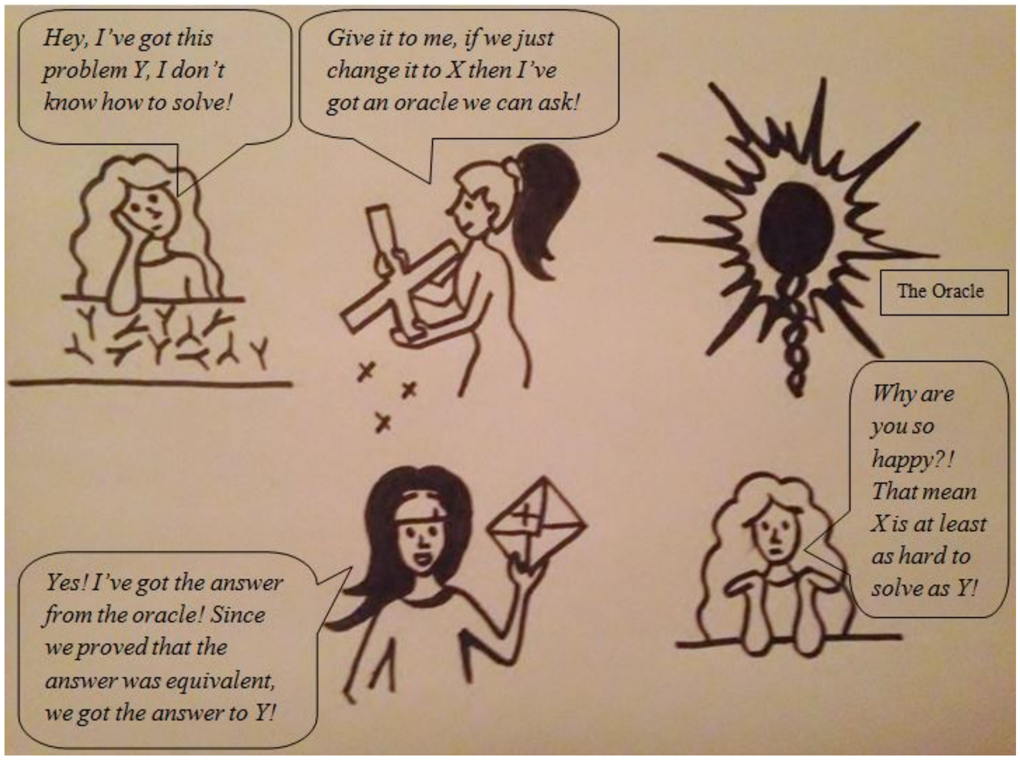
Hey, I've got this problem Y, I don't know how to solve!
Give it to me, if we just change it to X then I've got an oracle we can ask!
The Oracle
Why are you so happy?! That mean X is at least as hard to solve as Y!
Yes! I've got the answer from the oracle! Since we proved that the answer was equivalent, we got the answer to Y!

Figure 3: The idea behind Karp reduction.

2.2 Approximative Solutions

In the real world, NP-complete problems exist, and it is not sufficient to prove that they are NP-complete and then give up on a solution in the belief that an optimal solution cannot be found in polynomial time (if P is distinct from NP). Even those problems need to be solved and can be with approximative solutions. An approximative solution of an NP-hard optimization problem is a solution that fulfills the criteria of the problem and returns an answer, even though the answer is not necessarily optimal. The result from the algorithm is evaluated efficiently and, if possible, compared with the optimal result which gives a measurement of how good the solution is. These algorithms are called *approximation algorithms* [5, p. 39]. For some algorithms that receive an approximative solution, it is not possible to state how good the solution is compared to the optimal – these algorithms are called heuristics and are useful because of fast evaluation and often high quality of solutions on many instances, even though the quality cannot be guaranteed [5, p. 78–79].

2.2.1 Heuristics

There are numerous heuristics that can yield good solutions to NP-complete problems. Starting with an initial solution, local search can be used to find better solutions within the solution space, where the solution is tweaked and the new solution is analyzed in order to be accepted or rejected [7]. The behavior of a local search algorithm on a certain problem instance depends on three aspects: the quality of the initial solution, the neighborhood function, and the strategy to select new solutions. The neighborhood function defines which solutions that can be reached by tweaking the current solution. There is a trade-off between narrow and broad neighborhoods. Narrow neighborhoods allow faster traversing through the neighborhood while broad ones cover a larger part of the solution space in each iteration. The selection strategy of new solutions define how and when to analyze new solutions [5].

One method of tweaking solutions is using k-opt, in which k edges between nodes in a route are removed and reconnected in a different way [7].

Simulated annealing can be used to search for better solutions within the solution space. If the tweaked solution is better than the original, the former is accepted and this solution is tweaked in the next iteration. This way, we hope to move towards better solutions. In order to escape local minima, there is always a possibility to proceed with worse solutions, so as to find better ones in subsequent iterations [8, p. 25]. Worse solutions are accepted with some probability, which in the beginning of the search is large, resulting in the algorithm accepting almost any solution. The probability of accepting a solution is defined in equation 1 for maximization problems [8, p. 25], where $Quality(S)$ and $Quality(R)$ are the solution scores of the non-tweaked and tweaked score respectively. t is called a temperature, and determines how large the probability will be. It is set to a large number at first, so as to allow many solutions to be accepted, and is decreased over time.

$$P(t, R, S) = e^{(Quality(R)-Quality(S))/t} \tag{1}$$

Tabu search is a way of exploring the solution space by avoiding returning to previously tried solutions. Each new solution is stored in a tabu list and subsequent solutions are compared to them. Solutions may not be reused for l following iterations, where l is a predefined value [8, p. 26].

2.3 Scheduling Problems for Advanced Health Care at Home

A well-studied similar problem to the scheduling problem is the Vehicle routing problem with Time Windows (VRPTW). In VRPTW the aim is to find routes that minimize the violation of time windows, travel time, and the number of vehicles performing the routes. This problem is NP-hard (see for example [9]). Even deciding if a feasible solution exists is NP-hard [10]. This problem occurs in widespread areas and has been thoroughly studied. There are exact algorithms that solve problem instances with up to 100 stops, but these algorithms require more time than is reasonable in most applications [9]. In order to make the algorithms usable for health care applications additional constraints need to be considered. Although several research studies have been performed on the Home Health Care Scheduling problem (HHCSP), to the best of our knowledge, none of them cover the same combination of constraints as our approach, see tables 2 and 3.

Articles	Daily scheduling	Cover all visits	Competence match	Staff availability	Precedence of visits	Synchronization of visits
[11]	✓			✓	✓	✓
[12]		✓	✓	✓	✓	✓
[13]		✓	✓	✓		✓
[14]	✓		✓			
[15]	✓	✓	✓		✓	
[16]		✓	✓		✓	
[17]	✓	✓	✓	✓		
[18]		✓	✓	✓		✓
[19]	✓		✓		✓	
[20]	✓	✓		✓		✓
DSP	✓	✓	✓	✓	✓	✓

Table 2: Constraints addressed in HHCSP papers.

There are more parameters, assumptions and approaches that distinguish those projects than the tables show. The tables were designed to emphasize the relevant similarities and differences from our project. Furthermore, intended use of those solutions are in home assistance, while our intended use is in advanced health care at home.

Home assistance is designated to elderly home care. The patients are not admitted to a hospital. Home assistance is used for long-term care where the same type of care is needed on a regular basis. Home assistance is a part of primary care whose purpose is to provide basic (medical and non-medical) care that does not require medical or technical competences from the hospital [21]. In advanced health care at home however, as described in the introduction 1, the patients are admitted to a hospital, can be admitted during a varying time span, and the need of care is not regular.

The distinctions between home assistance and advanced health care at home have implications for the implementation of the algorithm. For visits that recur over months or years, it is feasible for the user to enter a large amount of patient information into the software system for each type of visit, as is common for home assistance scheduling tools. It is not so when visits are needed for a very limited period of time and changes in care plan are frequent. In advanced health care at home, where changes are more likely than unlikely, the usefulness of a scheduling tool is augmented if it can solve updates (addressed in [14]).

At a study visit at Tieto Sweden Healthcare & Welfare AB (Jörgen L 2015, Senior Sales Manager, oral communication, 19th February) we were told that most home assistance clinics in Sweden have been using some kind of scheduling tool since the last decade. In the introduction stage it was used to increase the efficiency in the work flow in order to reduce the number of staff. During the last decade the number of patients per work hour has increased and scheduling tools have played an important role in that process. We were

Articles	Minimize travel time	Maximize person continuity	Minimize overwork	Balance workload
[11]	✓	✓		✓
[12]	✓			✓
[13]	✓			
[14]	✓	✓	✓	
[15]	✓	✓		
[16]	✓	✓		✓
[17]			✓	✓
[18]		✓		✓
[19]	✓	✓		✓
[20]	✓	✓		
DSP	✓	✓	✓	✓

Table 3: Optimization parameters included in HHCSP papers.

also told that their implementation of Eveborn's [22] solution on HHCSP has been trialed in advanced health care, but it was clear that it was not transferrable without adjustments.

Solomon [23] suggests an insertion heuristic for obtaining an initial solution to VRPTW, which was proven to give optimal or near optimal results in a reasonable amount of time. In his approach the number of vehicles (corresponding to HCPs in our problem) are unlimited. The algorithm is greedy; in each step it chooses the best move in order to optimize the result. A seed customer (corresponding to a visit) is selected and inserted in a route. Then the routes are filled successively with customers until the capacity (corresponding to work hours) of the vehicle is filled. The seed customer is the first customer in the route, and is selected among the unplanned customers by some criterion, or a combination of criteria (for example farthest customer from the depot (corresponding to the hospital)). The idea is to use the seeds to get a good clustering of visits for each vehicle, then the following customers are selected by a minimization function. When the capacity of one vehicle is filled the procedure starts over with a new vehicle, until all customers are served. The result is widely accepted and used in subsequent research (e.g. [24, 25]) as an initial solution to VRPTW or extensions of it. Potvin and Rousseau [25] show that parallel construction of the routes improve the result with random distributed instances, compared with sequential construction that Solomon performs.

3 Method

This chapter describes the project methodology, what software development tools that have been used, and testing methods.

As software development method, we used the agile scrum method. We divided the project in small parts, sprints, of one to two weeks. After each sprint we evaluated the process and planned the following sprint in the project management system JIRA. We had as close cooperation with the ward during the entire process as possible.

The problem was defined based on information from informal interviews and observations at SABH as well as field studies performed by the Innovation Center at Karolinska University Hospital. The problem was inspected from a theoretical computer science perspective; the complexity class of the problem was identified and proven (see section 4.1). Solutions of problems within the same complexity class were studied in order to develop an algorithm that solves the problem (see section 2.3). A prototype of the algorithm was implemented in C#. A database was built using SQL Server Express to store the input and output from the algorithm. The distance matrix was retrieved from Nokia's REST-API HERE.

We concluded the project with a presentation to the ward to get feedback on our interpretation of the problem, understand if we have included all important parameters and learn about what they would like to be done in the future for the system to be usable. We also presented a questionnaire to gather more information about these issues.

3.1 Testing

Multiple testing methods were employed to verify that the code performed as intended and to validate that the algorithm fulfilled the ward's functional requirements. Specifically, the testing objectives were to

i) detect code bugs (and handle them),

ii) understand how the parameters in the objective function should be weighted,

iii) understand which of the suggested swaps in the local search improve the schedule,

iv) obtain an estimate of the runtime of the program,

v) find the number of iterations that we can run (and if they differ between shifts), and

vi) investigate how a fully functional implementation would affect the ward's way of working.

Testing was based on real data from the ward and the number of visits to be scheduled could easily be controlled by choosing to include only the day, evening or night shift (containing 2–30 visits and 2–8 HCPs). In accordance with the Swedish Patient Data Act all patient data was anonymized by substituting names and social security numbers with fictitious identification numbers and addresses with nearby addresses. Furthermore the data was entered manually into our program, due to bureaucracy at the hospital that, in order to respect the Swedish Patient Data Act, had a robust process for access to the journal system.

For verification, we ran our program on data from the ward, and compared our schedule with the ward's. This allowed us to detect and handle code bugs, adjust the weights of different optimization parameters (travel time, time window exceeding, person continuity and amount of free time between visits) and estimate runtime. We determined the number of iterations to run for each shift by setting a threshold for an acceptable schedule score.

We investigated how the algorithm was affected by the weights by changing one of them and keeping the others constant.

4 Results

In this section we begin by proving the NP-completeness of DSP and RSP. We then proceed to describe how the input parameters are reformulated and present our algorithms for DSP and RSP. Further, we describe the findings from empirical testing of the DSP algorithm at the ward and the optimizations made based on them.

4.1 Proof of NP-completeness

We are now going to prove that DSP and RSP are NP-complete.

4.1.1 Daily Scheduling Problem $\in$ NP

To show that the corresponding decision problem to DSP belongs to NP, we first formulate the decision problem and then propose a solution and show that it is verifiable in polynomial time.

DSP is formulated as a decision problem by introducing two goals (G_1, G_2) and the question is: given the instance of the problem described in Table 1, is there a schedule such that each task has been performed, all constraints have been fulfilled, the total traveling times for the HCPs are less than or equal to G_1, and the person continuity is less than or equal to G_2?

To show that the SABH problem belongs to NP, a solution is proposed and shown to be verifiable in polynomial time. Assuming a solution consists of a list of patients, and for each patient, the HCP(s) for his/her visit(s) is given, together with the time for the visit(s) and which car and medical equipment to take. We now need to verify that

- all patients have been assigned as many HCPs as they need,
- all times for the visit(s) are within given time windows,
- the visits assigned to each HCP is within his/her working hours, and
- there are no collisions in the schedule of an HCP.

This can be done as follows.

- For all parameters in the solution, verify that they are included in the input instance.
- For each of the patient's visits, verify that the total number of HCPs is the same as required, that they have right profession and that the visit starts within the time window.
- For each visit in the solution, verify that every involved HCP is available; for every involved HCP, the sum of the start time and time duration of the visit and travel time from the previously visited patient's home and back to the hospital is within the working day of the individual.

- Verify that there are no collisions in the schedule of an HCP; for each pair of subsequent visits, the time difference between them must be at least the sum of the time duration of the first visit and the travel time between them.

4.1.2 Daily Scheduling Problem is NP-hard

To show that DSP is NP-hard we show that an algorithm that can solve it can also solve the Minimum Metric Traveling k-Salesperson (MMTkS) problem, and therefore is as least as hard to solve as MMTkS.

MMTkS is a well-known NP-hard problem [5, p. 409]. The problem instance consists of a set of cities C, an initial city s, and the distances d between all cities. The goal is to find a set of k subtours such that each contains the initial city s, all other cities $\in C$ are visited in at least one of the subtours, and the total length of all subtours is as small as possible [5, p. 409]. This optimization problem can be formulated as a decision problem by introducing a goal M, and the question is: is there a set of k subtours such that each contains s, all cities $\in C$ are visited in at least one subtour, and the total length of the tours is less than or equal to M?

Algorithm 1 shows a Karp reduction from MMTkS to SABH. The reduction has the complexity $O(|C|k)$, that is, in polynomial time.

The correctness of the reduction is now to be proven by showing that an answer from one of the problems will always imply the same answer in the other problem. If the answer to SABH is *yes* it means that it has been possible to perform all the tasks with the available resources and the total amount of travel distances is less than or equal to G. This implies that the answer to MMTkS is also *yes*, because in the reduction the only limitation of resources is the number of HCPs and vehicles which is set to k, there is just one task per patient, and the number of patients is equal to the number of cities. Each task requires a nurse, and all staff members are set to be nurses. The tasks do not take any time and do not require any other resources than a nurse and a car. The distances between the patients are the same as the distances between the cities, and the initial city s is set to the location of the hospital, which is the starting point for the nurses and always part of a tour. On the other hand, if the answer to MMTkS is *yes* it means that all cities have been visited in k tours, which implies that with this reduction all tasks have been performed with the available resources. This shows that it is at least as hard to solve the SABH problem as it is to solve MMTkS. Therefore, the SABH problem is NP-hard.

4.1.3 Daily Scheduling Problem is NP-complete

DSP is NP-complete because it is in NP and is NP-hard. Unless $P = NP$, it is preferable to approximate a solution instead of searching for the optimal one. This proof is for asymptotic instances, assuming that the instance size increases towards infinity. In the SABH problem the instance sizes will not increase limitlessly; the ward strives to double their current patient capacity of 30 patients. After consultation with them we have limited the maximum patient capacity that the automated scheduler will handle to 200 patients per day. An increased

Algorithm 1 Reduction

```
MMTkS $(C, s, d, k, M)$
$G_1 \leftarrow M$, $G_2 \leftarrow \infty$, $ME \leftarrow 1$, $H \leftarrow \{1, ..., k\}$, $\mathbb{P} \leftarrow \{1, ..., |C|\}$, $VE \leftarrow k$, $hospital \leftarrow s$,
$D \leftarrow d$
for all i in range($k$) do
    $w_h \leftarrow \infty$
    $prof_h \leftarrow nurse$
for all $p \in \mathbb{P}$ do
    $V_p \leftarrow v_p$
    $td_{v_p} \leftarrow 0$
    $tw^e_{v_p} \leftarrow -\infty$
    $tw^l_{v_p} \leftarrow \infty$
    $ME_{v_p} \leftarrow 0$
    $PROF_{v_p,y} \leftarrow nurse$
    $V_{p,prec} \leftarrow \emptyset$
    $\{continutiy_p\} \leftarrow 0$ for all HCPs in $H$
SABH$(H, P, VE, ME, D, G)$
```

number of patients will also increase the number of employees, cars, visits per day, and medical equipment. Although an instance of 200 patients is a small number compared to infinity, for an NP-complete problem the resources needed to compute the optimal solution increases drastically even for small instances, so there is no reason to believe that it is reasonable to search for the optimal solution even for such small instances.

4.1.4 Rescheduling Problem is NP-complete

Spliet and colleagues [26] showed that the vehicle rescheduling problem (VRSP) is NP-hard by reduction to VRP. They defined the problem as VRP with an additional constraint: to minimize the deviation from the original schedule, each deviation was given a penalty. In the special case when the penalty is set to zero VRSP is equivalent to VRP. We have already proven that the SABH problem is NP-complete. The same reduction from the rescheduling problem to the SABH problem can be made. No additional verification is needed to confirm the feasibility of a solution to the problem, so it is in NP. Hence the rescheduling problem is NP-complete.

4.2 Modeling Data

The input data to the problem is described in Table 1, the additional notations needed for modeling the data are described in Table 4. We represent the problem as a set of nodes that need to be ordered in distinct routes. Every route is assigned to an HCP with a given competence. The nodes represent visits that need to be performed. There is a distinct node for each individual HCP that needs to be involved in a visit. We formulate a synchronization relation between nodes corresponding to the same visit: represented as left and right pointers from the nodes (see figure 4). Temporal precedence relations are represented as next and previous pointers from the nodes (see figure 5). We assume that synchronization and precedence relations only occur between nodes that represent visits for the same patient.

Each node u has predefined attributes, given by the corresponding visit in the input, such as which competence, time duration (td_{v_u}), and medical equipment (ME_{v_u}) that are needed to perform the visit, within which time window ($tw^e_{v_u}, tw^l_{v_u}$), and at which location it needs to be performed. Each node will have additional attributes start time and space. The HCPs' breaks and meetings are also represented as nodes. The location is set to the hospital and it is predefined which HCP is allowed to be assigned to it.

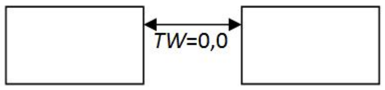

Figure 4: Illustration of a synchronization relation. The two nodes belong to the same visit.

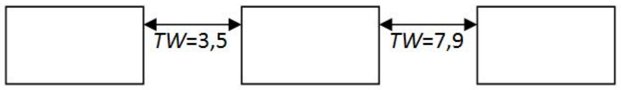

Figure 5: Illustration of a precedence relation. The first node needs to be followed by a node 4 ± 1 hours later which in turn needs to be followed by a node 8 ± 1 hours later.

Notation	Description
R	Set of routes
s	A complete schedule
u	A node
ST_u	Start time for u
$ST_{u,x}$	Start time for u adapted to ST_x
WL	Workload on a route
$space_{u,prev}$	Space between u and the previous node in route

Table 4: Notation used for modeling DSP and RSP.

4.3 Algorithm

In this section we describe the algorithms for solving DSP and RSP. DSP is divided into two distinct algorithms; the first gives an initial feasible schedule and the second improves it with local search. The algorithms are described sequentially and the relevant mathematical functions used in the algorithms are described separately. The sequence of algorithms is illustrated in figure 6.

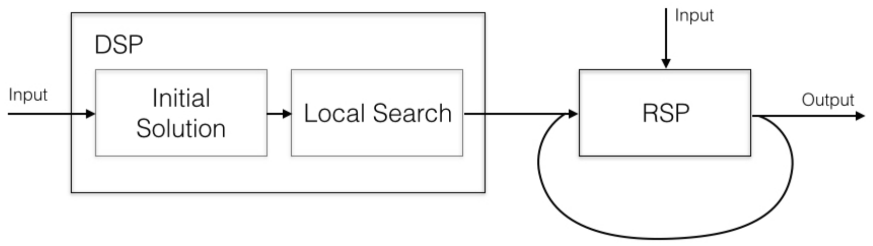

Figure 6: Sequence of algorithms.

We have proven that both DSP and RSP are NP-hard, so the algorithms that solve them need to make approximations. The first approximation is common for both of them: allowing an extension of a visit's time window or an HCP's working hours. The amount of extensions are added as a minimization factor, the importance of respecting the time windows for each visit and the working hour for each HCP can be weighted. Furthermore the HCPs begin and end their shift at the hospital. In DSP we assume that the number of medical equipment and vehicles are unlimited and that there exists a solution to the problem instance. RSP does not make this assumption, but is simplified by disregarding the optimization goals common in DSP and just finds the minimum number of changes from to the input schedule.

4.3.1 Initial Solution of the Daily Scheduling Problem

To produce an initial solution we modified Solomon's [23] insertion heuristic to conform to our constraints. The routes are constructed in parallel: a route is initialized for each HCP by inserting a *seed* between two *dummy nodes* (start and end at the hospital). The criterion used to select the seeds was narrowest or broadest time window. Until every node has been inserted into a route, the cost of placing each unplanned node u at each possible position is calculated (if it has not already been calculated). The node with the least cost is inserted in the position with least cost. The algorithm is described with pseudocode in algorithm 2.

Algorithm 2 Initial Solution

```
$R \leftarrow []$  ▷ Select a seed and insert it in a route, with the hospital as start and end points
for all $h \in H$ do
    $route_h \leftarrow 0, seed, 0$
    add $route_h$ to $R$
                                   ▷ Calculate the best position to put each unvisited node
minHeap $\leftarrow$ new $Heap(|nodes|^2)$
for all $u \notin R$ do
    for all positions $i \in R$ do
        minHeap.put($c_2(i, u, i+1)$)
                                   ▷ Insert the node with least cost in its best position
while there are unvisited nodes do
    best $\leftarrow$ minHeap.get()
    insert $best$
    update all nodes' $ST$ and $space$ affected by the insertion
    for all $u \notin R$ do
        minHeap.put($c_2(best-1, u, best)$)
        minHeap.put($c_2(best, u, best+1)$)
return $R$
```

4.3.2 Mathematical Functions

To calculate the cost of placing an unvisited node u between two nodes i and $i+1$ in a route in the initial solution equation 2 is used. If u needs to be synchronized with other already placed node(s), the best start time for all of them is calculated and then equation 2 is calculated using that start time. If it is decided that u is to be placed at that position, ST_u is set to the start time used when the cost was calculated, for u and all synchronized nodes, and updates are made in all routes affected by it. Time windows for nodes with precedence relations to the inserted node are adjusted to respect the timespans. This will just have effect on unvisited nodes in the current iteration but, since the penalty is set to a high value, a node that is already placed in a route and has a precedence relation to the mentioned node will be selected as the worst node in the next iteration. α_{1-4} represent the weights for travel time, shifting the following visit, time windows, and space respectively.

$$c_1(i,u,i+1) = \alpha_1 c_{11}(i,u,i+1) + \alpha_2 c_{12}(i,u,i+1) + \alpha_3 c_{13}(i,u,i+1) + \alpha_4 c_{14}(i,u,i+1) \quad (2)$$

c_{11}, expressed in equation 3 calculates the extra travel time. c_{12}, expressed in equation 4 calculates how much the visits before and after u need to be shifted if u is inserted. c_{13}, expressed in equation 5 sums the penalties of exceeding each node's time window and/or a route's time window (corresponding to the assigned HCP's working time). Equation 6 expresses c_{14} is a measure of how urgent it is to reach a patient, since it is better to plan with margin to be prepared to adjust to unforeseen events.

Solomon [23] used equation 3, 4 and the inverse of 6, since in VRPTW space between visits is undesirable.

$$c_{11}(i,u,i+1) = d(i,u) + d(u,i+1) - d(i,i+1) \tag{3}$$

$$c_{12}(i,u,i+1) = |ST_{i+1,u} - ST_{i+1,i}| \tag{4}$$

$$c_{13}(i,u,i+1) = \gamma_i p_i + \gamma_u p_u + \gamma_{i+1} p_{i+1} + \gamma_R p_R \quad \left| \begin{array}{l} \gamma_k = \begin{cases} lateness_k & \text{if } lateness_k > 0 \\ 0 & \text{otherwise} \end{cases} \\ lateness_k = ST_k - tw^l_{v_k} \\ k = i, u, i+1, R \end{array} \right. \tag{5}$$

$$c_{14}(i,u,i+1) = |ST_u - tw^l_{v_u}| \tag{6}$$

Equation 2 expresses the cost of inserting a node between two nodes in a route, irrespective of which HCP the route belongs to. Equation 7 adds this cost to the workload of the route, and the value of person continuity between the patient and the HCP (lower value corresponds to preferred match).

$$c_2(i,u,i+1) = c_1(i,u,i+1) + continuity_{u,h} + WL \tag{7}$$

To calculate the cost of placing a node u between two nodes i and $i+1$ in a route in the local search algorithm, equation 9 is used, which uses equation 8. c_1 is the same as c_3 without taking into account extra time and urgency. These measurements are necessary when constructing the initial solution, but not when improving it. For improvement, the focus is on the optimization goals. This also reduces the calculation time.

$$c_3(i,u,i+1) = \alpha_1 c_{11}(i,u,i+1) + \alpha_3 c_{13}(i,u,i+1) \tag{8}$$

$$c_4(i,u,i+1) = c_3(i,u,i+1) + continuity_{u,h} + WL \tag{9}$$

The local search algorithm calculates the relief of removing a node from its route. The calculation is described in equation 10, which sums the difference in travel time (equation 11) and time window violation without and with u (multiplied by their respective weights). The worst node will have the largest value for c_5.

$$c_5(i,u,i+1) = \alpha_1(c_{13}(i,\emptyset,i+1) - c_{13}(i,u,i+1)) + \alpha_3 c_{51} \tag{10}$$

$$c_{51}(i,u,i+1) = d(i,i+1) - d(i,u) - d(u,i+1) \tag{11}$$

4.3.3 Local Search Algorithm of the Daily Scheduling Problem

After an initial solution is found local search is performed to find better solutions. Starting with the initial solution we perform 2-OPT to obtain new solutions. Simulated annealing and tabu search is used to avoid getting stuck in the search space.

We now describe the local search algorithm, see algorithm 3. We set a certain initial temperature t (to a large value) and a tabu list length l. For a set number of iterations, we tweak the solution, running both simulated annealing and tabu search to decide whether to accept the new solution or not. Since we have defined our problem as a minimization problem, we change the subtraction in the probability function described in equation 1, producing the probability in equation 12.

We define a move to be tabu if the node has been placed in the same route, with the same preceding and succeeding node, at a time close to an entry in the list (we define a time span). After a fixed set of iterations has passed, we state the move as no longer tabu.

$$P(t,R,S) = e^{(Quality(S)-Quality(R))/t} \tag{12}$$

After a certain number of iterations, the best solution found so far, s^*, is returned as the final schedule.

The modification of a solution using the changeSchedule function proceeds as follows. We remove the two edges from the node that has the greatest negative effect on its route's score. The negative effect is calculated by comparing the cost of exceeding time windows and the travel time if each node is included or excluded (equation 10 describes the calculation). The node yielding the largest difference in costs is removed from its position and reinserted in the best feasible position. If the new solution is not accepted, we try placing the node in its next best place. This process is repeated a maximum of a times (we decide a) until a solution is accepted. Thereby we have defined the size of the *neighborhood* to a solution as $a \cdot n$, where n is the number of nodes.

In order to explore more of the search space, every jth iteration we choose a node randomly instead of choosing the worst node. The subsequent steps are the same as for the worst node.

Algorithm 3 Local Search

```
Input: Initial solution $s$
$t \leftarrow$ initial temperature
$l \leftarrow$ tabu list length
$i \leftarrow$ number of iterations
$a \leftarrow$ number of times to try to find a position for node
$j \leftarrow$ frequency of removing random nodes
$s^* \leftarrow s$                                        ▷ Best solution yet
$iteration \leftarrow 0$                                  ▷ Current iteration number
repeat
    new $\leftarrow$ CHANGESCHEDULE($s$, $iteration$)
    for $ii \leftarrow 0$ to $k$ do
        if $Score(new) < Score(s^*)$ then
            $s \leftarrow new$
            $s^* \leftarrow new$
        else if random number between 0 and $1 < e^{(Score(new)-Score(s))/t}$ then
            $s \leftarrow new$
        else if $ii \neq a$ then
            new $\leftarrow$ CHANGESCHEDULE($s$, $iteration$)
    Decrease $t$
    $iteration++$
until repeated $i$ times
return $s^*$

function CHANGESCHEDULE($s$, $iteration$)
    if $iteration \mod j = 0$ then
        $u \leftarrow$ random node
    else
        $u \leftarrow$ worst node
        while $u$ is the latest added node in $tabulist$ do
            $u \leftarrow$ new worst node
        Remove $u$ from $s$
    $bestPosition \leftarrow$ best position to place $u$

    while $bestPosition$ is tabu do
        $bestPosition \leftarrow$ new best position

    Insert $u$ in $bestPosition$ in $s$
    $tabulist \leftarrow u, bestPosition, iteration$
    Update affected routes in $s$
    return $s$
```

4.3.4 Rescheduling Algorithm

We are first going to suggest a solution if just one new node needs to be incorporated in the schedule. According to the simplification of the problem the algorithm first finds the schedules that cause minimum changes. They are scored according to the other minimization goals and the best one of them is returned. We assume that a number i is given that tells us how many changes in the schedule are acceptable. If the algorithm has changed i nodes the resulting schedule will be displayed and the coordinator can use it to handle the rest of the necessary changes manually.

We suggest a best-first search to find the schedule that costs least changes in the original schedule. The first step in the algorithm is to insert the new node in its calculated best position. For each feasible position in s it calculates the best start time for u_{new} (in that position) by recursively calling equation 4 in both directions. Based on these calculations we can choose the start time that affects the least number of nodes. u_{new} is preliminarily placed in the position that causes least impact in the schedule and following updates are preliminarily made. If there are unacceptable extensions of time windows caused by the insertion, we calculate the best node to move to avoid the violations – considering both the relief on the route it is currently placed in and the impact it will have to put it in another route. These calculations are stored and from the calculated best move, the algorithm calculates which subsequent move that is best. In this way the algorithm always chooses to start calculations from the best move so far and hence the preliminary updates done so far to get to that solution. This procedure is repeated until a solution is found or the algorithm has reached maximum number of changes i.

For a position to be $feasible$ for a node it is required that:

i) it fulfills all requirements given in previous algorithms,

ii) there is an available vehicle for the HCP to perform the visit,

iii) nodes that are already performed cannot be rescheduled,

iv) HCPs that have begun a visit are unavailable until the time duration of the node has passed, and

v) the required medical equipment are available in the route, which is true if:

   a) it is already being used in the planned route,
   b) it is not used in a route, or
   c) there is enough time for the HCP that is performing the route to pick it up from an existing route and redeliver it when it is needed.

If more than one node needs to be inserted the problem gets more complicated despite our simplification. We could reuse the algorithm for insertion of one node, but the problem of determining in what order we should insert the nodes arises. Comparing the result of inserting them in the order of all permutations can only be a solution if there is a very small

Algorithm 4 Rescheduling

```
Input: Current schedule $s$, new node $u_{new}$, max changes $i$, number of nodes $n$

Struct $Move$ contains
    priority
    node
    position

minHeap ← new $Heap(i \cdot n^3)$
minHeap.put($u_{new}$)
repeat
    $u_{new}$ ← minHeap.get()
    Preliminarily insert $u_{new}$ in its best position
    Preliminarily update affected routes $r$
    if There are violations in $r$ then
        for all Nodes $u$ in routes affected by $u_{new}$ do
            $gain$ ← the relief if $u$ was removed from route
            $Move_{node}$ ← $u$
            for all $feasible$ positions for $u \in s$ do
                $cost_{possibleRoute}$ ← the cost of insert $u$ in the position
                $Move_{firstPriority}$ ← $cost_{possibleRoute} - gain$
                $Move_{position}$ ← position
                minHeap.put($Move$)
    else
        The schedule with least changes was found
        Optional:
        Perform the search for each move that requires the the same amount of changes
        Score each of these schedules with $c_2$
        return the best schedule
until $i$ nodes have been removed in all feasible combinations
return the best schedules so far
```

number of new nodes, since the number of permutations is the factorial of the number of new nodes. We need to decide in which order to insert the new nodes and hope that this yields a good result. Since nodes with narrow time windows are more likely to cause impact in the schedule the nodes with wider time windows are inserted first. When calculating the best position a slight modification is made, namely that the changes that are caused by a newly inserted nodes are not counted as violation, but the impact that they cause in the reinsertion is calculated.

4.4 Evaluation of Algorithms

In this section we theoretically evaluate the efficiency of our algorithms by calculating the complexity of each and discussing the distance from the optimal solution.

4.4.1 Complexity

The algorithm for constructing an initial solution has complexity $O(n^2 \log n)$, where n is the number of nodes corresponding to the total number of visits times the number of HCPs needed in each visit. Each iteration within the algorithm inserts one visit. The costs of placing all unvisited nodes before and after that node are calculated and put into the min heap. Putting and getting objects from a heap runs in $O(\log\ n)$. Each iteration the number of unvisited nodes decreases by one.

The local search runs i iterations and the changeSchedule function runs once per iteration. For removing the worst node within changeSchedule, we must go through all nodes within each of the m routes (equivalent to the number of HCPs). The maximum number of nodes per route is n. Thus the complexity for removing the worst node is $O(m{\cdot}n)$. To reinsert the node, we go through all nodes within the routes. We can insert a node a times. This yields a cost of $O(a{\cdot}n)$. Thus, the cost of the whole local search algorithm is $O(i{\cdot}(m{\cdot}n{+}a{\cdot}n)) = O(i{\cdot}m{\cdot}n)$.

The complexity of inserting one or multiple nodes is evaluated separately below. Even though the complexity is high, it is evaluated from the worst case scenario, but in a real word application the feasibility of positions will strongly limit the number of positions that need to be considered, as well as the number of nodes that are affected by an insertion.

i) *Inserting one new node*
The complexity of inserting one new node is $O(i \cdot n^3)$. To preliminarily insert one new node in the schedule the best feasible position needs to be calculated. In the worst case all positions are feasible, that is $O(n^2)$. Insertion will be repeated until i changes in the schedule have been made, unless the optimal result has occurred earlier. In the worst case, all moves result in an equal amount of changes, that is, all preliminary inserts need to be performed for all nodes. This complexity is for both space and time, since each calculated $Move$ needs to be stored. To score the schedules and return the best of them is done in O(n).

ii) *Inserting multiple new nodes*
The complexity of inserting multiple new nodes is the complexity of inserting one new

node multiplied by the number of nodes, O($|nodes_{new}| \cdot i \cdot (n + |nodes_{new}|)^3$)

4.4.2 Solution's Distance from the Optimal Solution

We developed heuristics instead of approximation algorithms. Therefore, it is not possible to theoretically evaluate how far the solution is from the optimal. How well it performed on the tested problem instances is described in the Testing section.

The correctness of the rescheduling problem is discussed below, for insertion of one node and multiple nodes separately.

i) *Inserting one new node*
If there exists a rescheduling with at most i changes, the algorithm will find the optimum. This will be proven by induction. The base case is that the number of changes is 1, that is if insertion of u_{new} does not require additional changes, it is trivial to see that the algorithm will find that solution. We assume that the algorithm has performed $|S_1|$ number of changes and that is the minimum amount of changes to move from s to solution S_1. Next we make the smallest feasible move from S_1 to next solution S_2. This gives us the smallest number of changes from S_1 to S_2, and because S_1 was the smallest number of changes from s to S_1, S_2 will be the smallest number of changes from S_1 to S_2. That proves that the least number of changes will be found.

The algorithm has not taken into account any of the other constraints given in RSP, except the ones that decide whether the move is feasible or not and minimize the number of changes. The optional portion of the algorithm takes the other optimizations goals in DSP and RSP into account. That part of the algorithm scores all solutions that require the least amount of changes based on the optimization goals. The solution with the lowest score is returned.

The simplification of RSP implies that the algorithm does not provide an optimal solution to DSP, or RSP if the simplification is ignored.

ii) *Inserting multiple new nodes*
The suggested algorithm for rescheduling multiple new nodes is, like the daily scheduling algorithm, a heuristic and it is not possible to theoretically evaluate how far the solution is from the optimal.

4.5 Testing

Our findings based on testing the algorithm implementation are described in this section.

On real data from the ward the initial solution managed to find a feasible schedule that had no or very small extensions of the time windows, almost independent of which of the seeds we used. When we unintentionally manipulated the data so that a feasible schedule did not exist, the quality of the result depended on which seed we had used, the overall extensions of time windows were much smaller when the seeds were based on broadest time windows compared to seeds with narrowest time windows.

The local search improved the initial schedule. We found that an increase of the neighborhood to $n \cdot k$ instead of n further improved the results. This was done by allowing the removed node to try to insert itself in k positions (selected in increasing cost order) instead of only the least costly position. Furthermore we found that the calculation of the worst node needed to take into consideration, except the extensions of time windows the node forced its neighbors to, the extension the node itself had, caused by another node. This allowed the algorithm to, if the worst node did not have a better place to be inserted in, move its neighbors.

The algorithm preferred to place the lunches at the beginning or end of an HCP's route (to decrease travel time). When we increased the penalty for exceeding these time windows to the highest possible value, the algorithm was forced to let the HCPs leave the hospital and perform visits and then return for lunch at the correct time.

Inserting synchronized visits often shifted other visits to outside their time windows.

We adjusted parameter weights by first equating the different parameters, so that the cost of a minute of travel time and a minute of time window extension affected the solution score equally. The workload balance and person continuity parameters were also adjusted to affect the score with the same amount. We then altered the weights and found that the algorithm improved the result according to the weights. The value of the temperature and the temperature decreasing function used to configure the probability of accepting a solution in the local search were strongly dependent on the magnitude of the score of the schedule. Without the tabu list the local search algorithm went into loops where it removed and reinserted the same node, or a small collection of nodes. The tabu list length was adjusted to avoid these loops according to our problem instances.

4.6 Presentation at the Ward

The automated scheduler was presented to the HCPs at SABH. The overall response towards the automated scheduler was that the HCPs welcomed the program and were keen on using it, although there was a worry of the risk of double documentation, increasing the workload of the HCPs. Their comments are presented below.

- The HCPs confirmed that we had modeled the problem correctly, taking into consideration the aspects that are important to them.
- One HCP mentioned that a lot of time is put into creating a schedule, and was open to changing the routines because there is a lot of room for improvement.
- One HCP inquired about how the positioning of HCPs will be done (used in rescheduling). We mentioned that existing services require "checking in" and "checking out" at each patient's home. The discussion that followed indicated that this would be an acceptable additional task if it were simple to perform, for example using a mobile application.
- It is important not to completely replace personal communication between HCPs by

using a mobile application for communication of updates in the schedule. Rather, there should be some consultation with the HCP who is involved.

- The HCPs were interested in using a minimal format of the scheduler for faster adoption of it. This shows an interest in the product, even before integration of all components shown in figure 1. The most critical components to implement for the scheduler to be useful include the integration with the patient journal information system and personnel work schedule database. Without these components, it would take too much time and effort of the HCPs to manually enter data.
- At present, the HCPs find their visits in the overall schedule. It would be helpful to see individual schedules separately, to get a better overview of their working day.
- A crucial component to make the complex system manageable is a well-designed user interface.
- The ward's own database was designed by an HCP and it will be difficult for the ward to maintain the database when he leaves. They are therefore looking for a new system to replace it and see the automated scheduler as an alternative.
- Changes of input data (such as visits and personnel work schedule) need to be simple to enter and the program needs to be able to update those changes efficiently. Currently, some information is only conveyed verbally, causing errors in scheduling. Therefore, good work routines need to be introduced in conjunction with the automated scheduler.
- It was considered positive that an automated scheduler would make it impossible to double book a person – something that occurs currently, posing a threat to patient safety.
- Traffic is currently a large stress factor for HCPs, so there is a demand for a scheduler that optimizes routes based on traffic patterns.

5 Discussion

In this section we discuss the automated scheduler with respect to computer security, patient safety, confidentiality, and employees' work environment. While this thesis focused on the optimization algorithm for the daily scheduling, many opportunities for extending the scope of the thesis remain. There are multiple directions that an extension could take.

5.1 Improving the Algorithms

The criteria to select seeds had a small impact on the solution if the problem instances was correct, when they were not it had huge effect on the result. The best criteria to select the seeds on different problem instances could be investigated to ensure good results on each instance. Examples of criteria that could be investigated are; person continuity, travel time, randomized, or a combination of them. Then the algorithm could start with identifying which kind of problem instance the input belongs to and running it on right seed. It could also be good to run the algorithm on multiple seeds, in parallel if the capacity of the computers allowed for it, and then select the best initial solution that the local search could improve.

The neighborhood in the local search was defined in order to make each iteration fast. When we increased the neighborhood by allowing the worst node to try to be inserted in more than one position the result was improved. The node to swap could have been selected by calculating the cost of each node in the current route and the cost of the node in the route with its lowest cost and swapping the node that yields the best improvement on the overall solution. This would have increased the running time of each iteration, and limited the total amount of iterations that could be run. Thereby the total search space that the algorithm traversed would have decreased. On the other hand the solution in each iteration would have improved and the probability of bypassing an optimum would have decreased.

The issue of nodes that negatively impact their synchronized neighbors' routes can be addressed in the local search algorithm by, after each such occurrence, swapping unsynchronized nodes within the route, to find better fitted positions. This search will be faster than the regular iteration in the search, because the search space is increased to the unsynchronized nodes time window in the node it is placed in, in worst case the length of the route.

The RSP algorithm needs to be implemented and tested. The order in which multiple new nodes should be inserted needs to be verified. Furthermore, the interface to handle manual arrangements needs to be developed. Our suggestion is that if the coordinator clicks on a node he/she will get a list of requirements that needs to be fulfilled. A view with suggestions of where he/she can put the node could be supported, that tells him/her the impact that move has on the new route, the current route, the person continuity if the node were moved there, as well as travel time and space.

The complexity of the RSP algorithm is high, which could impact the usefulness of the algorithm according to how large problem instances that it should solve and how fast the ward needs the result. We choose to propose an exact solution to the problem when just one node needs to be incorporated, since we did not have the resources needed to test an implementation of the algorithm. Just as for the DSP algorithms, approximations could be made to decrease complexity, which might give a good result even though its quality could not be evaluated theoretically.

5.2 Testing Method

Although using real data has strengthened the validity of the algorithm, the quantity of data was small and specific for one ward. Although the algorithm is to work for up to 200 patients, real instances of such a large volume of patients were not available to us. It is therefore difficult to draw general conclusions about the quality of the algorithm. Entering data manually into the program as opposed to obtaining it from a currently used information system increased the time taken for testing and considerably decreased the number of tests that could be performed. Furthermore, it introduced human errors in the test data and extensive troubleshooting had to be done to identify them. Testers of similar algorithms should consider simulating data to obtain more test instances. More extensive code testing and optimization as well as testing combinations of swaps and seeds could have been done given more time, which would have yielded a more effective implementation of the algorithm.

For better verification of the quality of the algorithm, it would be interesting to run Solomon's benchmark instances for VRPTW [23], which provides a measure of the quality of the travel time minimization aspect. This would also make our algorithm comparable with other HHSCP algorithms that have used the benchmarks.

Since the actual addresses were replaced by nearby addresses, this will slightly alter the resulting routes. However, this difference will not alter the routes by more than a few minutes and is therefore negligible in determining if the algorithm is functional.

5.3 Confidentiality

The database schema can be found in appendix A. Although it is possible to model patient and HCPs in one table, we have decided to store the patient and HCP related data separately, since the Swedish Patient Data Act requires rigorous logging of accessing patient data, which is of importance if the program is to be taken into use. The database schema reveals that each visit has a lot of attributes belonging specifically to it.

The issue of patient data confidentiality needs to be considered when we receive the distance matrix. We suggest three alternative solutions. The distance matrix could be received from a REST-API, if so the security level must be investigated, both that no information from the queries are stored and that the communication remains confidential. Alternatively, a map with predefined points may be used, and a local algorithm finds the closest point from the patient's address, and this point is used in the query. A third alternative is to implement a local algorithm that calculates the best path.

5.4 Usefulness of the Automated Scheduler

As mentioned in section 1.3 more implementation parts remain in order to get a fully functional application.

The possibility to lock some planned visits in the schedule and reschedule the rest, or manually swap visits would be useful.

The runtime for the DSP algorithm to get the initial solution is 1 minute for 30 nodes and each iteration in the local search took 1 second on an average computer. 30 nodes correspond to the average number of visits for 30 patients admitted to the ward. A twofold increase of the patient capacity would yield runtimes of 4 minutes and 4 seconds respectively (assuming that the visits needed for each patient would increase linearly with the number of patients and that the constants in the runtimes would stay constant with an increase of the input). An increase of the patient capacity by 7 times (to 210 patients) would increase the runtimes to 49 minutes and 49 seconds respectively. If the ward invested in a more powerful computer the runtime would decrease. Even if they do not, the runtime is acceptable for obtaining a schedule.

The response from the ward was overall positive and it was clear that an automated scheduler would facilitate the work of the coordinator. It was also clear that they want to have a role in further development of the system. They were positive to having an application for each HCP as it would facilitate the work of each individual by being able to view his/her personal schedule and all the resources needed during the day. However, they were skeptical to an automated function that would update their schedule without their approval. This can be solved in different ways. For example, the RSP could find a schedule suggestion that is sent to the HCP through the application, and the HCP could choose to accept or reject the suggestion. The suggestion can also be viewed by the coordinator so that he/she can call the HCPs and thereby retain personal communication and the sense of control for the HCPs.

It is important to investigate and implement the user interface for the coordinator, HCPs, and patients/parents. We suggest that further research weigh the positive effect of notifying the patient/parent of the preliminary arrival time and the assigned HCP (to increase the sense of security and participation, and decrease telephone communication from concerned parents), with the negative effect of planned visits being rescheduled from the notified time. It should be considered whether it is possible to integrate the application with the patient journal system or the local patient database, and if it is not, whether it is positive or negative to log information of the visits through the application. Logging might be positive because documentation can be done in near real time (from the patient's home during a visit) but negative because of double documentation and hence double work that can lead to misunderstanding and introduce more human errors in documentation. Relevant laws must be investigated [7] to identify which information that can be communicated through the application.

We believe that machine learning would be useful in further research. If all data from the daily schedule, the patient information, the first planned schedule and all rearrangements

[7] In Sweden the Swedish Patient Data Act.

during the day and possibly even evaluation parameters are stored, this can be used as input to a learning algorithm. Identification of visit types in a broad sense could be done with machine learning where the interest is to identify the parameters of a visit such as time duration, time window, temporal dependencies, and needed professions. Furthermore it could be used to identify good combinations of input, what kind of input to the scheduling algorithm that yields a good schedule (such as efficient but not stressful, high person continuity, or minimal number of rearrangements). The result could be used to develop an algorithm for a weekly schedule that decides which day visits should be planned. This gives the input to the daily scheduling algorithm. Additionally, machine learning could be used to identify when rescheduling will be needed. Small adjustments during the day can be incorporated without affecting the entire schedule but many of them can lead to a need of a full rescheduling. It would be useful to have a measurement of when these small changes will require a full rescheduling, in order to avoid last minute changes and might create a smoother transition between the original and rearranged schedule.

We have decided to use addresses to specify patients' locations. If the program were to be used in the future for scheduling and routing visits within, for example, a hospital building, geographic coordinates would be a more suitable identifier. Incorporating the change would require a minor adjustment, therefore in this first version of the program, we have used addresses to simplify interaction with the program. Returning to the possibility of extending the scheduler to cover a hospital, it may be desired that the program schedule several wards (containing HCPs belonging to different wards and patients requiring service from multiple wards).

Many organizations have scheduling problems similar to the SABH problem. Our algorithm can be adopted by other wards providing advanced health care at home and the learning feature of the algorithm facilitates adaptation of the scheduler to the users' activities. The scheduler can be extended to other fields within healthcare where visits need to be planned during the day and new tasks may appear after an initial schedule has been set. The algorithm is useful even for other industries where a similar number of tasks need to be scheduled (the suggested integration with patient journal systems will be unnecessary). Each patient can be generalized to correspond to a customer and HCP to a service provider of another sort. The algorithm is implemented in such a way that it is simple to change the source of the data to other data systems or manually input data. The lack of confidentiality requirements on patient data can facilitate the implementation by being able to enter real addresses in the distance matrix.

5.5 Risks and Vulnerabilities

Our automated scheduler builds on adequate input to the system: for each visit, it reserves a certain amount of time (the time required to complete the visit and time to reach the patient from the HCP's previous position). The rest of the schedule is planned by assuming that the HCP is free for the next task when the time duration has passed. On the other hand, the current scheduling system at SABH has incorporated flexibility in order to ensure patient safety by being able to handle unforeseen events. It is a risk that an automated system might reduce the flexibility in the scheduling with stronger dependencies on timekeeping in each task on the schedule, which exposes a vulnerability in the functionality of the ward.

The value of efficiency is ambiguous. If the efficiency is increased by increasing the workload on the staff, decreasing the time allocated to each visit or reducing the space between visits it can compromise patient safety, quality of the care, and employees' work environment. An introduction of an automated schedule could lead to such a development both as a direct effect of the optimization in the algorithm, and by the fact that each task during the day is more precisely measured than at present. Therefore a careless use of this measurement can lead to an increase of patient capacity without the necessary increase of resources that is needed to maintain patient safety, quality of the care and employees' work environment. This concern was lifted by the HCPs during our presentation and is legitimate. This problem has already been observed in the home assistance sector. On the other hand, if work flow is made more efficient by reducing the travel time the time can be put to better use. Furthermore, the main goal is to increase the efficiency in the scheduling process. Our strong advice is to be careful with how the measurements in the automated system are handled, as well as to respect the constants given by the coordinator such as time duration needed for each visit and space needed between visits.

6 Conclusion

Providing health care at home has many advantages for individuals and society. Scheduling such care requires taking many requirements into account. Manual planning limits the possibility of wards providing such care to increase their patient capacity. We have identified the requirements that need to be considered in scheduling, proven that the daily scheduling and rescheduling problems are NP-complete, and proposed an automated scheduler that can facilitate the planning for large numbers of patients. The automated scheduler is based on our designed algorithm which produces an approximative solution within a feasible amount of time in relation to SABH's needs. For daily scheduling we first construct an initial feasible solution using a greedy approach. We then improve the solution using local search, simulated annealing, and tabu search. Our rescheduling algorithm incorporates new visits into a set schedule using a best-first search. The solution fulfills all fundamental requirements – visits are assigned to HCPs with the required competence, all visits are scheduled, and, to a large extent, time windows are maintained. Additionally, the total travel time, continuity and distribution of workload among HCPs are held at satisfactory levels. Having implemented the algorithm and tested the prototype using real data from SABH, we have seen that an automated scheduler does optimize the scheduling. However, automatic transfer of data from the patient journal system is an imperative for the scheduler to be adopted. If this is done, the automated scheduler should improve the scheduling process and thereby remove scheduling as a bottleneck for increasing patient capacity.

References

[1] VASCO;. [Online; accessed 25-January-2015]. Available from: \url{http://www.sabh.nu/pdf/SABH_VASCO_rpt.pdf}.

[2] SABH; 2000. [Online; accessed 25-January-2015]. Available from: \url{http://sabh.nu/pdf/slutprojrapp.pdf}.

[3] SABH; 2015. [Online; accessed 26-January-2015]. Available from: \url{http://sabh.nu}.

[4] Land- och vattenarealer den 1 januari 2012. Statistiska centralbyrån; 2012.

[5] Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M. Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer; 2003.

[6] Kleinberg J, Tardos E. Algorithm Design. Pearson; 2006.

[7] Johnson D, McGeoch L. The Traveling Salesman Problem: A Case Study in Local Optimization. Local Search in Combinatorial Optimization. 2014;p. 215–310.

[8] Luke S. Essentials of Metaheuristics. George Mason University; 2014.

[9] Salvendy G. Handbook of Industrial Engineering: Technology and Operations Management. John Wiley Sons; 2001.

[10] Savelsbergh MWP. Local search in routing problems with time windows. Annals of Operations Research. 1985;4(1):285–305.

[11] Rasmussen MS, Justesen T, Dohn A, Larsen J. The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies. European Journal of Operational Research. 2012;219(3):598 – 610. Feature Clusters.

[12] Jemai J, Chaieb M, Mellouli K. The Home Care Scheduling Problem: A modeling and solving issue. In: Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on; 2013. p. 1–6.

[13] Mutingi M, Mbohwa C. Multi-objective homecare worker scheduling: A fuzzy simulated evolution algorithm approach. IIE Transactions on Healthcare Systems Engineering. 2014;4(4):209–216.

[14] Nickel S, Schröder M, Steeg J. Mid-term and short-term planning support for home health care services. European Journal of Operational Research. 2012;219(3):574 – 587. Feature Clusters.

[15] Liu R, Xie X, Augusto V, Rodriguez C. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. European Journal of Operational Research. 2013;230(3):475 – 486.

[16] Duque PAM, Castro M, Sörensen K, Goos P. Home care service planning. The case of Landelijke Thuiszorg. European Journal of Operational Research. 2015;243(1):292 – 301. Available from: http://www.sciencedirect.com/science/article/pii/S0377221714009126.

[17] Bachouch RB, Guinet A, Hajri-Gabouj S. An optimization model for task assignment in home health care. In: Health Care Management (WHCM), 2010 IEEE Workshop on; 2010. p. 1–6.

[18] Borsani V, Matta A, Beschi G, Sommaruga F. A Home Care Scheduling Model For Human Resources. In: Service Systems and Service Management, 2006 International Conference on. vol. 1; 2006. p. 449–454.

[19] Elbenani B, Ferland JA, Gascon V. Mathematical programming approach for routing home care nurses. In: Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on; 2008. p. 107–111.

[20] Redjem R, Kharraja S, Xie X, Marcon E. Coordinated multi-criteria scheduling of caregivers in home health Care Services. In: Automation Science and Engineering (CASE), 2011 IEEE Conference on; 2011. p. 519–524.

[21] Socialstyrelsen. Hemsjukvård i förändring;. Available from: www.socialstyrelsen.se.

[22] Eveborn P, Flisberg P, Rönnqvist M. Laps Carean operational system for staff planning of home care. European Journal of Operational Research. 2006;171(3):962 – 976.

[23] Solomon MM. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research. 1987;35(2):pp. 254–265.

[24] Müller J. Approximative solutions to the bicriterion Vehicle Routing Problem with Time Windows. European Journal of Operational Research. 2010;202(1):223 – 231.

[25] Potvin JY, Rousseau JM. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research. 1993;66(3):331 – 340.

[26] Spliet R, Gabor AF, Dekker R. The vehicle rescheduling problem. Computers Operations Research. 2014;43(0):129 – 136.

A Database Schema

The database used for the implementation of the automated scheduler is described in table 5. The underlined columns compose the key of each table.

Table Name	Columns
Visits	***ID***, ***PatientID, TimeDuration, TimeWindowStart, TimeWindowEnd, TWViolationPenalty, Profession, Workload, LeftID, RightID, NextID PrevID, NextStart, NextEnd, ShiftType, Description***
Patients	***ID***, ***Name, Location, Workload, Coordinate_ID***
Coordinates	***ID***, ***Latitude, Longitude***
Distances	***Latitude***, ***Longitude***, ***Time***, ***Distance***
HCPs	***ID***, ***Name, Prof, MaxWorkload***
Shifts	***ID***, ***StartTime, EndTime, ShiftType***
METype	***ID***, ***Name***
Medical Equipment	***ID***, ***Name, StartTime, EndTime***
Car	***ID***, ***RegistrationNr, StartTime, EndTime***
Route	***ID***, ***Workload, OverallScore, TravelTimeScore, TWViolationScore, ContinuityScore, WorkloadScore***
Solution	***ID***, ***Score***
MustHave	***VisitID***, ***HCPID***, ***Weight***
Shift_HCP	***ShiftID***, ***HCPID***
Route_HCP	***RouteID***, ***HCPID***, ***Name***
Route_Visit	***RouteID***, ***VisitID***, ***StartTime***
Route_ME	***RouteID***, ***MEID***, ***StartTime***, ***EndTime***
Route_Car	***RouteID***, ***CarID***, ***StartTime***, ***EndTime***
METype	***ID***, ***Name***

Table 5: Database Schema

TRITA 2015:079