FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Metaheuristics for a Multimodal Home-Health Care Scheduling Problem

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Gerhard Hiermann
Matrikelnummer 0525205

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: o.Univ.-Prof. Dipl.-Ing. Dr. Günther Raidl
Mitwirkung: Dipl.-Ing. Dr. Jakob Puchinger

Wien, 10.05.2012

(Unterschrift Verfasser) (Unterschrift Betreuung)

Erklärung zur Verfassung der Arbeit

Gerhard Hiermann
Ketzergasse 376-382/3/10, 1230 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to thank my advisors, Prof. Günther Raidl from the Vienna University of Technology for his constructive feedback and support and Jakob Puchinger of the Austrian Institute of Technology (AIT) for entrusting me with this project and his continuous support throughout the whole thesis.

I also want to thank the AIT, especially Andrea Rendl and Matthias Prandtstätter for their technical support, feedback and their overall work on the framework this thesis is built on.

I would like to thank my parents for their never ending support. Without them this would not have been possible. Thanks to Max for his helpful comments on the thesis as well as the distraction from it.

And to Anna, for keeping me on track and her support, encouragement and much more throughout these last couple of years.

Abstract

The multimodal home healthcare scheduling (MHS) problem tackled in this thesis is modelled based on the operational process of a Viennese home healthcare provider. The problem is to find an assignment of jobs to nurses as well as the order in which these jobs are performed by each nurse under consideration of contractual and legal constraints and preferences as well as travel time based on the selected mode of transport.

The existing approach by the *Austrian Institute of Technology* is part of the project *Carelog*, an automatic solving architecture to create reasonably good schedules based on constraint programming and variable neighbourhood search. The focus of this thesis was to implement several different metaheuristic approaches into the existing framework to solve one day real world instances and compare the results with the existing approach.

Three metaheuristics have been selected and implemented based on experiences with them on related problems, the vehicle routing problem with time windows and the nurse rostering problem. First a simulated annealing hyper heuristic, a general optimization approach using a set of so-called low-level heuristics, was implemented. This approach yielded similarly good results compared to the existing approach in most of the tested instances. The second heuristic is a so-called memetic algorithm. This population-based approach known for its good performance when tackling related problems achieved the best results in all test instances within a relatively short amount of time and provided an in-depth view into the structure of good solutions. The third approach, the scatter search, is also a population-based approach using more deterministic techniques in contrast to memetic algorithm. Results show that this metaheuristic needs high runtimes to achieve a comparable solution quality of solutions and indicate that this approach is not the best choice for this kind of problem.

In various tests the impact of parameter and design decisions on the performance of the heuristics were observed and documented. Additionally a comparison with the existing approach is provided using real world instances. The results show that the memetic algorithm performs best. Also in terms of their applicability in praxis, the MA provided the most satisfying solutions.

Kurzfassung

Das *multimodal home healthcare scheduling* (MHS) Problem beschäftigt sich mit der Zuweisung von HeimhelferInnen zu PatientInnen unter Berücksichtigung von Präferenzen sowie gesetzlicher und vertraglicher Bestimmungen als auch mit der Erstellung von Touren anhand der zuvor festgelegten Zuteilung. Im Zuge des Projekts *CareLog* des *Austrian Institute of Technology* (AIT) wurde bereits ein Framework zur automatisierten Lösung eines solchen Problems anhand der Fallstudie der Organisation Sozial Global entwickelt. Untersucht wurde das Lösen von Ein-Tages-Problemen anhand einer vom AIT entwickelten Zielfunktion. In dieser Arbeit werden die weiteren entwickelten Lösungsmethoden, die in das bestehende Framework implementieren wurden, beschrieben und mit dem vorhandenen Lösungsansatz verglichen.

Da das MHS mit den gut untersuchten *vehicle routing problem with time windows* und dem *nurse rostering problem* verwandt ist, wurden drei Lösungsansätze aus diesem Bereich ausgewählt und an das MHS angepasst. Die erste Metaheuristik ist eine *simulated annealing hyper heuristic*, welche mittels einer Menge an so genannten *low-level heuristics* eine gute Lösung sucht. Bereits dieser Ansatz zeigte vergleichbar gute Ergebnisse mit dem existierenden Ansatz. Der zweite Lösungsansatz ist ein Populations-basierter, ein so genannter *memetic algorithm*. Dieser erzielte die besten Ergebnisse bereits nach kurzer Zeit. Der dritte Ansatz ist ebenfalls ein Populations-basierter, ein so genannter *scatter search*, der deterministischere Techniken anwendet als der zuvor genannte *memetic algorithm*. Die Resultate zeigen jedoch, dass dieser Ansatz sehr hohe Laufzeiten benötigt, um vergleichbar gute Ergebnisse zu erzielen.

In einer Vielzahl an Tests wurden Entscheidungen bezüglich Parameter und Methoden auf ihre Auswirkung auf die Leistung der Ansätze überprüft sowie der bereits existierende Ansatz mit den in dieser Arbeit Beschriebenen anhand von Instanzen aus der Praxis verglichen. Die Ergebnisse zeigen, dass vor allem der *memetic algorithm* sehr gute Ergebnisse erzielt. Auch im Bezug auf die praktische Verwertbarkeit liefert der *memetic algorithm* die besten Lösungen.

Contents

CHAPTER 1

Introduction

The aim of this thesis is to adapt and evaluate metaheuristic approaches of related problems for a multimodal home-healthcare scheduling (MHS) problem. As the average age is increasing and patients prefer to be nursed at home it is of great significance to solve big real world instances of this problem in an adequate amount of time. The MHS problem consist of finding an assignment of home-care staff (nurses) to customers (patients) as well as an ordered tour based on the assignment. In addition to time windows for services provided by a nurse, contractual and legal issues have to be considered which add an additional degree of complexity to the problem. Also multimodality is considered, where nurses have different travel times caused by their mode of transport (e.g. car, public transport). The focus of this work lies on solving and optimizing one day instances of the MHS problem.

The implementation is part of the project CareLog[1] of the Austrian Institute of Technology (AIT) in cooperation with Sozial Global, ITS Vienna Regions and ilogs. The problem description is based on the working procedures of Sozial Global, a Vienna based Home-Healthcare company. However, a focus of this project is to provide a flexible solving architecture which can be used by other Home-Healthcare companies with a minimal grade of adaptation needed.

The current scheduling task is performed by a human scheduler using the planning tools provided by the company ilogs. As these schedules rely on the huge expertise of the planning staff, an automatic scheduling approach should help them by providing reasonable good schedules for further adaptation. Additionally the consideration of travel times and multimodality should help to improve the schedules, as the current (human) procedures do only consider static travel times.

Research in the field of *home health care* (HHC) problems started around 1997/98 with the work of Begur et al. [4] and Cheng and Rich [22]. Begur et al. describe a *decision support system* for a home-healthcare company in the USA using a model which does not consider time windows.

Cheng and Rich present a *mixed integer programming* (MIP) formulation as well as a heuristic approach to tackle an HHC problem. In their problem definition they distinguish between

[1]The project CareLog is partially funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the strategic programme I2VSplus

full-time and half-time nurses and consider lunch-breaks. Their approach consists of a randomized greedy algorithm to construct a first solution. In a second step the algorithm removes assignments of two nurses with at least one of those nurses working overtime while fixing all other patient to nurse assignments. The greedy algorithm is then started again on this partial solution. The approach is evaluated using a set of randomly generated instances with 4 nurses and 10 patients where optimal solutions are known and a set of three larger instances (up to 300 nurses and 900 patients). The authors were able to generate solutions for two out of the three large data sets using their method. For the random instances they found the optimum for 17 of the 40 test instances in less then a second compared to the exact approach with runtimes between 3 to 20 minutes.

Bertels and Fahle [8] use a combination of *linear programming* (LP), *constraint programming* (CP) and *simulated annealing* as well as *tabu search* based metaheuristics. Considering problem instances for a single day with 20 to 50 nurses and from 111 to 326 jobs, the focus of their work is to provide reasonable solutions in relatively short time (600 to 840 seconds per instance). They use a combination of LP and CP to create initial solutions, which are then improved using metaheuristics and a solution pool. The main differences to our problem formulation are the missing additional soft constraints (only preferred time windows are considered) and the missing multimodality aspect.

Eveborn et al. [28] use a set-partitioning formulation to provide a flexible architecture and solve the problem using *repeated matching*. Starting from an initial matching, this approach iteratively creates a new perfect matching by local improvements and splitting until a predefined termination condition is reached. Their problem definition also considers multimodality and contains many constraints of this work but do not include preferred starting times or preferences. However, the instances they use are rather small (up to 21 nurses and 123 jobs). Based on their results, Eveborn et al. reported that the travelling time savings of their approach are about 20% (on a moderate guess) compared to solutions made by the human counterpart.

Rasmussen et al. [46] formulate the problem also as a set partitioning problem and describe a branch-and-price approach. Their model incorporates connected visits (dependencies on the order of the tour) but allows some jobs to be left unassigned using a priority system. Their tests use real-world instances with up to 15 nurses and 150 jobs.

Based on the work in this thesis a journal article has been submitted:
Gerhard Hiermann, Matthias Prandtstetter, Andrea Rendl, Jakob Puchinger, and Günther R. Raidl. Metaheuristics for solving a Multimodal Home-Healthcare Scheduling Problem, submitted 2012.

Outline of this thesis

The MHS is strongly related to two other problems, the *vehicle routing problem with time windows* (VRPTW) and the *nurse rostering problem* (NRP), so chapter 2 will give a description of these problems, followed by a more detailed problem definition of the MHS. The next chapter will provide an introduction to the applied heuristic approaches and gives an overview of related work done for the VRPTW and NRP. In chapter 4 the existing approach including the objective function, the solution generation as well as the already implemented metaheuristic will be discussed in detail. Chapter 5 will then describe the implemented metaheuristics this thesis is

focused on and provides an in-depth view into these algorithms. The experiments, including the preliminary parameter tests as well as the final comparison are provided in chapter 6. In the final chapter 7 the thesis is closed with concluding remarks.

CHAPTER 2

Description of the Problem

An MHS problem can be described as a combination of a *vehicle routing problem with time windows* (VRPTW) and the *nurse rostering problem* (NRP) as the problem consists of finding an assignment of nurses to patients with respect to a large number of side constraints (e.g. legal and contractual issues) as well as the corresponding tours of the nurses.

This chapter will first describe these related problems and then give the definition of the MHC problem tackled in this thesis.

2.1 Vehicle Routing Problem with Time Windows (VRPTW)

The *vehicle routing problem with time windows* (VRPTW) is a generalization of the classic *vehicle routing problem* (VRP) which can be defined as follows based on the formulation of Laporte et al. [38]:

Let $G = (V, A)$ be a graph where $V = \{0, \ldots, m+1\}$ is a set of vertices representing customers with the depot located in vertex $v_0 = v_{m+1}$ and A is a set of arcs. Every arc $(i, j), i \neq j$ has a non-negative value assigned to it representing the *travel time* (or *travel cost*) with the values being symmetrical, hence arc $(i, j) = (j, i)$. Additionally assume that there are up to n vehicles available, each having a capacity of D. The VRP then consists of designing a set of tours (routes) minimizing the overall travel time in such a way that each city in $V \setminus \{0\}$ is visited exactly once and all vehicles start from v_0 and end in v_{m+1}.

The VRPTW extends the VRP by introducing a *time window* $[s_j, e_j]$ for every node $v_j \in V$ [24]. For the depot nodes v_0 and v_{m+1} a time window is also provided, i.e., $[s_0, e_0] = [s_{m+1}, e_{m+1}] = [E, L]$, where E and L are the earliest and latest possible departure/arrival times from/to the depot. The new objective extends the VRP by additionally assuring that the tours satisfy all time windows, i.e., no vehicle arrives in node $j \in V \setminus \{0\}$ later then e_j or has to wait until $max(s_j, s_i + dur_i + tt_{ij}), i \in V, i \neq j$ to start the service, where tt_{ij} represents the travel time from node i to j and dur_i the service duration (for the depot $dur_0 = dur_{m+1} = 0$).

The problem now is not only to find proper tours (and minimize their cost), but to ensure that the time windows are not violated. Some formulations relax the time window constraint by

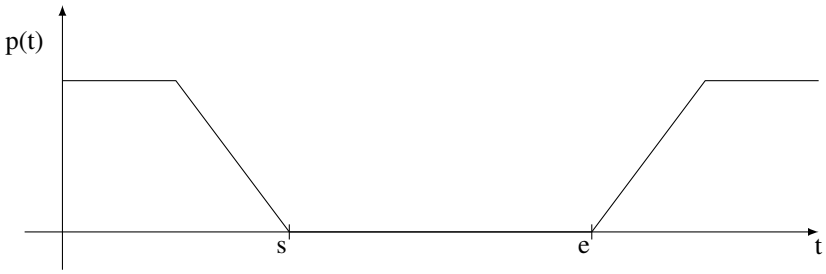


Figure 2.1: An example for time windows and a possible penalty function $p(t)$

measuring the violation and incorporate it into the objective function. An example can be seen in Figure 2.1 where a penalty of $p(t)$ is added to the objective if the service starts earlier than s or ends later than e.

Earlier research in the field of VRPTW was more focused on exact methods, like in Desrosiers et al. [26] and Cordeau et al. [24]. A survey of the heuristic approaches are provided in Bräysy et al. [10, 11].

2.2 Nurse Rostering Problem (NRP)

The *nurse rostering problem* (NRP) consists of finding a periodic duty roster for nurses in a hospital considering a large number of hard and soft constraints.

Figure 2.2 shows an example of a roster also described in [21] where a variable x_{ij} with values of the domain of possible shifts assigned to nurses n_i on day d is used. The hard constraints, which have to be met by the produced schedule, are usually the coverage of staff demands per day per shift and the skill requirements. To measure the quality of a schedule the number of soft constraint violations are used. These constraints include personnel policies (e.g. maximum number of consecutive working days), preferences and day-offs.

A bibliographic survey is provided by Cheang et al. [21] where a wide variety of formulations is presented due to hospital-specific requirements. Burke et al. [15] later provided a more detailed survey of the research done in the field of nurse rostering as well as an overview of the approaches. Based on the classification of Burke et al., Causmaecker et al. [20] only recently presented an enhanced classification of the different NRP formulations based on an $\alpha|\beta|\gamma$ classification approach.

2.3 Multimodal Home-Healthcare Scheduling Problem

For the *multimodal home-healthcare scheduling* (MHS) problem the goal is to find an assignment of nurses to patient requests (jobs) as well as the corresponding tours for the visits while minimizing the total travel time and the violations of side constraints. As the problem allows different *modes of transport* (MOT) for the nurses (i.e. by car of public transport), the travel time depends on the selection.

	d_1	d_2	d_3	d_4	d_5	d_6	d_7
n_1	E	E	L	N	-	E	L
n_2	E	L	E	-	N	N	-
n_3	L	-	-	E	E	L	N
n_4	N	N	-	L	L	-	E

Figure 2.2: Example of a roster for a week with 3 different shifts (E ... early shift, L ... late shift, N ... night shift, - ... day off)

An instance of an MHC problem consists of following sets and variables:

Nurses $\mathcal{N}$ This set contains all N nurses available to be scheduled. Each nurse can only work within specified time windows. These time windows are given for each nurse $n \in \mathcal{N}$ in the set TW_n. Not every nurse has to be part of the schedule, but the scheduled ones have a general minimal and maximal number of working hours per day (h_{min}, h_{max}).

Time $\mathcal{T}$ Time is represented by a set of discrete points in time $\mathcal{T}$ dividing the day into T time units of τ minutes. For the instances used in this work, $\tau = 5$.

Jobs $\mathcal{J}$ Contains each of the J jobs which have to be serviced by a nurse in $\mathcal{N}$. For each job $j \in \mathcal{J}$ where there exists an associated start time window $[s_j, e_j]$ where s_j is the earliest starting time and e_j the latest starting time, a favoured start time t_j with $s_j \leq t_j \leq e_j$ and a duration dur_j. The time variables s_j, e_j, t_j are of $\mathcal{T}$and $dur_j \cdot \tau$ gives the duration in minutes, where $dur_j \in \mathbb{N}$.
Set $\mathcal{J}$ consists of two types of jobs: pre-allocated jobs $\mathcal{J}^{pre}$ (around 5% of all jobs), e.g. team meetings and administrative work, which are assigned to be done at fixed locations and fixed times and by a specific nurse ($alloc(j)$ is used to retrive nurses n pre-allocated to job $j \in \mathcal{J}^{pre}$); and non-preallocated (reassignable) jobs $\mathcal{J}^{\overline{pre}}$ which are placed by a specific customer $cust(j)$ where the function $cust(j)$ returns the customer of a set $\mathcal{C}$ associated with job j.

Qualifications $\mathcal{Q}$ Every job in $\mathcal{J}$ and every nurse in $\mathcal{N}$ has an attribute qualification with one of the values in the sorted set $\mathcal{Q}$ assigned. For this problem setup the set is given as $\mathcal{Q} = \{csw, vn, hn, ahn, mn\}$ representing their qualification: community service worker (*csw*), visiting nurse (*vn*), homecare nurse (*hn*), advanced homecare nurse (*ahn*) and medical nurse (*mn*). As $\mathcal{Q}$ is a sorted set, following equation holds: $csw < vn < hn < ahn < mn$. These qualifications represent the skill of a nurse $n \in \mathcal{N}$ and also the minimum qualification required to perform a job $j \in \mathcal{J}$ with respect to the previously defined order. This means that a nurse n can perform a job j only if the qualification needed for job j is lower or equal than the qualification of nurse n.

Refusal Reasons $\mathcal{F}$ Customers and nurses may have contradicting preferences leading to a refusal of the nurse/customer. The various reasons are summarized in the set $\mathcal{F} = \{dog, cat, smoker, male, female\}$. All refusal reasons of a customer $c \in \mathcal{C}$ are given by

set $\mathcal{A}_c$ and for a nurse $n \in \mathcal{N}$ by $\mathcal{A}_n$. This means that if a customer c has the refusal set $\mathcal{A}_c = \{cat, male\}$, then the customer has a cat and does not want to be serviced by a male, whereas the same refusal set of a nurse n ($\mathcal{A}_n = \{cat, male\}$) indicates that nurse n has a cat and is a male. This means that whenever $\mathcal{A}_c \cap \mathcal{A}_n \neq \emptyset, c \in \mathcal{C}, n \in \mathcal{N}$ nurse n may not be assigned to customer c.

Preferred Mode of Transport $\mathcal{P}$ As the problem also considers multimodality (i.e. different modes of transport), each nurse $n \in \mathcal{N}$ states her preferred mode of transport p_n where $\mathrm{p}_n \in \mathcal{P}$ and $\mathcal{P} = \{car, publicTransport\}$. The home location of each customer $c \in \mathcal{C}$ and nurse $n \in \mathcal{N}$ is retrieved by $loc(c)$ and $loc(n)$, respectively. The travel time from location $loc(a)$ to $loc(b)$ using transport mode p is retrieved from the distance matrix $\mathrm{tt}^{\mathrm{p}}_{ab} \in \mathcal{T}$ where $a, b \in \mathcal{C} \cup \mathcal{N}$. This means, for instance, that an entry $\mathrm{tt}^{\mathrm{car}}_{ab} = 3.5$ states that driving from $loc(a)$ to $loc(b)$ takes 3.5 time units, thus $3.5 \cdot \tau = 17.5$ minutes. Note that the actual travel times rounded up to the next time unit, e.g. 17.5 minutes travel time are rounded up to 20 minutes. Travel time estimates are based on data from the Viennese public transport system, and a large set of historical data from Viennese floating car data [53].

Given these sets of instance data, a solution for the problem is denoted as $\sigma = (\mathcal{R}, \mathcal{S})$ consisting of a set of tours $\mathcal{R} \subseteq 2^{\mathcal{J}}$ and a mapping of jobs to starting times $\mathcal{S} : J \to \mathcal{T}$ representing the underlying roster. Each nurse $n \in \mathcal{N}$ is associated with exactly one tour, i.e., $R^n \in \mathcal{R}$. Therefore the number of routes is equal to the number of nurses $|\mathcal{R}| = |\mathcal{N}|$. Each tour $R^n \in \mathcal{R}$ associated to nurse n starts and ends at the nurse's home location $loc(n)$. Although a sequence to the jobs along a tour is introduced by set $\mathcal{S}$ only, let us, for simplicity, denote by R^n_k the k-th job in the tour of nurse $n \in \mathcal{N}$. If $R^n = \emptyset$ holds, nurse $n \in \mathcal{N}$ is not scheduled for the current day.

The constraints in the problem setup of this thesis are split into hard constraints, these constraints have to be fulfilled for a feasible solution σ, and soft constraints, for which the violations are tried to be minimized. In the following listing of constraints, (2.1) - (2.6) are considered as hard constraints and (2.7) - (2.10) as soft constraints.

- all jobs must be serviced by a nurse, i.e. must be part of a tour
$$\forall j \in \mathcal{J} : \exists n \in \mathcal{N} : j \in R^n \tag{2.1}$$

- each job may only be serviced by one nurse
$$\forall n, m \in \mathcal{N} : n \neq m \Rightarrow R^n \cap R^m = \emptyset \tag{2.2}$$

- nurses must be qualified to perform the assigned jobs
$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathrm{q}_j \leq \mathrm{q}_n \tag{2.3}$$

- the starting times of two consecutive jobs (of one nurse) must be chosen such that traveling between them (using the appropriate mode of transport) is possible
$$\forall n \in \mathcal{N} : \forall 1 \leq i \leq |R^n| - 1 : t(R^n_i) + \mathrm{dur}_{R^n_i} + \mathrm{tt}^{\mathrm{p}_n}_{R^n_i R^n_{i+1}} \leq t(R^n_{i+1}) \tag{2.4}$$

- the nurse may only work within the given working time windows

$$\forall n \in \mathcal{N} : \forall j \in R^n : \exists tw \in \mathrm{TW}_n : \{t(j), \ldots, t(j) + \mathrm{dur}_j\} \subseteq tw \tag{2.5}$$

- pre-allocated jobs may only be assigned to the proper nurse

$$\forall j \in \mathcal{J}^{pre} : n = alloc(j) \Rightarrow j \in R^n \tag{2.6}$$

- qualifications of nurses and (assigned) jobs should match

$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathrm{q}_j = \mathrm{q}_n \tag{2.7}$$

- the starting time of each job must lie within the specified time windows

$$\forall j \in \mathcal{J} : \mathrm{s}_j \leq t(j) \leq \mathrm{e}_j \tag{2.8}$$

- the actual starting time of each job may be the favoured starting time stated by the customer

$$\forall j \in \mathcal{J} : t(j) = \mathrm{t}_j \tag{2.9}$$

- all refusal reasons have to be considered in the roster

$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathcal{A}_n \cap \mathcal{A}_{cust(j)} = \emptyset \tag{2.10}$$

CHAPTER 3

Methods

This chapter will provide an introduction to the metaheuristic approaches described in this thesis as well as an overview of related work done in the field of VRPTW and NRP using these kinds of heuristics.

First the variable neighbourhood search will be introduced as this approach is part of the existing framework and is also used as embedded local search procedure in some of the implemented metaheuristics. Then a general description of the three metaheuristics adapted in this thesis will be provided. The selection of these methods was not only based on their performance tackling the related VRPTW and NRP, but also based on their novelty on HHC and MHS problems in general. Further motivation for their selection are described in the respective sections.

3.1 Variable Neighbourhood Search

Variable neighbourhood search (VNS) [33] procedures try to improve a given solution by searching in their neighbourhoods. A neighbourhood $N_i(x)$ is a set of solutions reachable by a single change, a so-called move. An example of a move is a single bit-flip of a binary string from 0 to 1 or vice versa. The neighbourhood defined by this move would contain all binary strings with one single bit different from the actual string.

A *variable neighbourhood descent* (VND) [33] uses an ordered list of neighbourhood structures $N_1, N_2, \ldots, N_l$. These are searched systematically by starting from the first structure N_1 and switching to the next one N_2 only if no improvements could be found. If an improvement was found at any time in the search, the algorithm starts with the first neighbourhood N_1 again. If no improvement could be found in all neighbourhood structures, the algorithm terminates.

Determining which solution of $N_i(x)$ is selected depends on the strategy, also called step-function. Some of these are listed below:

- *Random*: one of the neighbours is selected at random
- *Next-Improvement*: the first neighbour better than the actual solution is selected

- *Best-Improvement*: the best neighbour of all possible neighbours is selected
- *Best-Of-Improvement*: the best neighbour of a subset of neighbours is selected

As a VND may get stuck in a rather poor local optimum, this procedure is often embedded in a general VNS scheme [33]. While VND tries to locally improve given solutions by systematically exploring and switching between different neighbourhood structures, VNS applies random moves in order to escape local optima. The idea of a VNS is to alter a solution by performing a *shaking*, i.e., a perturbation move, each time the embedded local search procedure is unable to find an improvement for the current best solution. A possible implementation of shaking also used in this thesis is to apply $k + 1$ random shift moves in the k-th consecutive iteration without improvement.

Related work

Burke et al. [16] proposed a VNS to improve schedules of hospitals in Belgium with 20 nurses. Their approach uses constraint specific as well as large neighbourhood structures in a VND procedure embedded in a VNS. However, not every neighbourhood is searched until no improvements can be found but uses a limited tabu search. They stated that this approach produces good results but is very problem specific.

In the field of the VRPTW Rousseau et al. [49] presented a constraint-based approach using constraint programming to obtain a first solution and a VND for the improvement. Their results on the benchmark of Solomon [51] were promising as they achieved good results but with higher runtime. Bräysy et al. [12] proposed a reactive VNS approach with an embedded VND. Their approach constantly uses route-elimination procedures followed by the VNS for improvements which lead to 4 new best solutions for the Solomon benchmark [51].

Motivation

The VNS approach excels with its fast and efficient way to be adapted to any optimization problem. As there are only a minimal number of parameters to be optimized, the performance depends on the designed neighbourhood structures and its search order. The MHS problem has several straight forward neighbourhood definitions, like a shift of a job or the swap of whole tours, which are derived from the related VRPTW. As these neighbourhoods can also be used by other heuristics, the implementation of a VND as a optimization procedure was obvious.

3.2 Simulated Annealing Hyper-Heuristic

Simulated annealing (SA) [42] is a local search algorithm based on a heat treatment process in metallurgy. Here a controlled heating and cooling process is used so that the atoms find new positions to finally yield in a state of a closely minimal internal energy.

To mimic this behaviour the SA algorithm uses a probabilistic decision function based on a slowly decreasing temperature variable and the current solution's objective value which also allows worse solutions to be accepted. Therefore this algorithm is able to escape local optima in constrast to a classical local search algorithm. The algorithm starts with a high temperature,

where many worse solutions may be accepted. During the run of the algorithm the temperature decreases and only few worse solutions may be used leading to a converging state. Different implementations also introduced a re-heating procedure to improve the overall performance. This procedure increases the temperature to the point where the last best improvement could be found.

A *hyper heuristic* is a "heuristic to select heuristics" [18] by maintaining a set of so-called *low-level heuristics* and decides at each iteration which heuristic to apply next. A Simulated Annealing Hyper Heuristic as proposed by Bai et.al. [3] selects from the set of heuristics in a probabilistic way based on an tested/accepted ratio of a previous learning period or the newly-created/tested ratio prior to the reheating phase. Each iteration the selected low-level heuristic creates a new solution and is accepted if it is either better then the current solution or with an probability of $e^{-d/t}$, where d is the difference between the new and the current solution and t is the current temperature. When running the algorithm K iterations the temperature is updated every $nrep$ iteration by Lundy and Mees' nonlinear function [39]

$$t = t/(1 + \beta t), \quad \text{where } \beta = ((t_{start} - t_{end}) \cdot nrep/K \cdot t_{start} \cdot t_{end}), \tag{3.1}$$

and t_{start} and t_{end} are the temperatures at the beginning and the termination of the algorithm and $nrep$ is the number of iterations until the temperature will be changed again.

In their work, Bai et.al. [3] tested this algorithm for 3 different problems, i.e. NRP, *timetabeling* and *bin packing*. Parameter-wise they only changed the total number of iterations K and the used low-level heuristics for each problem. The results where very promising, as for the NRP this algorithm could find better solutions than the best algorithm at this time (2003).

Related work

The previously described approach is based on the work of Cowling et al. [25]. They presented a hyper-heuristic especially tailored for the NRP. In their approach they used choice-function to rank the low-level heuristics during the search. These choice functions return information regarding the individual, the joint performance of pairs and the time since the last call of the low-level heuristics. Compared to the results of other approaches by Aickelin et al. [1] and Dowsland [27] their approach proved to be more robust as it could find a valid solution each test run.

Based on this work, Burke et al. [19] formulated a more general hyper-heuristic approach and presented a *tabu-search hyper-heuristic*. Like the SAHH described before (which was developed shortly after), this approach was not only tested for the NRP but for the timetabling and bin packing problem and produced similar good results as the previous hyper-heuristic by Cowling et al. [25].

In the field of VRPTW Pisinger et al. [43] presented a general heuristic to solve different VRP problems, including the VRPTW using an *adaptive large neighbourhood search* (ALNS) approach. According to Burke et al. [17] this approach is a so-called permutation hyper-heuristic. The ALNS chooses and applies two neighbourhoods each iteration: a destroy neighbourhood and a repair neighbourhood. The selection of the neighbourhoods is controlled stochastically by an adaptive layer based on their past performance. Using this approach they were able to improve the best known solutions of over one third of the tested instances [51].

Motivation

A simulated annealing heuristic was considered in the early stages of this thesis. After the initial research in the literature the extension to a hyper heuristic using more than a single neighbourhood structure for the search was selected. This approach has only been used for scheduling and bin packing problems but another hyper heuristic – the ALNS described before – performed very well for the VRPTW. Thus using this method seems very promising for a highly constrained routing problem like the MHS.

3.3 Memetic Algorithm

Algorithms of the family of *evolutionary algorithms* (EA) [35] simulate the natural evolution. Every step the algorithms try to use the given set of solutions, the so-called *population*, to gain a better solution than before and every individual in the population competes to get into the reproduction process. To compare the individuals an objective (evaluation) function is applied to assign an objective (fitness) value to each one.

In this thesis a *genetic algorithm* (GA) was implemented and it proceeds in general as follows. First the population will be initialized randomly or by using a seeding algorithm. Then these initial solutions will be evaluated. Now the evolution starts, until a predefined terminating condition has been reached. Usually the GA terminates when a certain number of generations has been created or a predefined good solution has been found.

The evolution process starts with a *selection procedure* where a number of individuals (solutions) are selected to take part in the reproduction process. Then the selected individuals will be *recombined* and/or *mutated*. Recombination combines features of parents, while mutation typically performs small random alterations. After this step, the created new individuals will be evaluated and the original generation will be replaced by the newly created ones using a *replacement strategy*. During this replacement, some individuals from the original generation may stay in the newly created generation.

To benefit from the population-based search of the GA and local search, both are combined to a hybrid, a so-called *memetic algorithm* (MA). The idea is to simulate cultural evolution through propagation and adaptation of information from one generation to the following one [41]. Typically the local search is used before the replacement takes place. This approach is sometimes also referenced as a *hybrid EA* [32].

Related work

Aickelin et al. [1] proposed a GA to tackle an NRP defined by Dowsland [27] for a UK hospital's requirements. The problem was to create a weekly schedule for approximately 30 nurses. Dowsland's model uses a binary representation where all possible shift patterns are enumerated and each nurse is allocated to exactly one of them. Constraints are handled using penalty values for each nurse to pattern assignment. The Aickelin et al. [1] approach works with so-called *co-operating sub-populations*. In this approach the solution string is sorted by the qualifications of the nurses and sub-populations, containing all solutions of the population but using its own objective function, are combined using individuals of complementary populations. Compared to

previous results using tabu search [27] the approach was fast and robust but failed to find better solutions.

Later on Aickelin et al. [2] presented another GA approach using indirect representations of the problem. A solution is encoded as a permutation of the nurses and is decoded using one of 3 different greedy construction algorithms based on (1) the cover, (2) the contribution (i.e. the preferences of the nurses) and (3) a mixture of both of them. Using a combination of both greedy strategies for the GA, Aickelin et al. [2] were able to find new best solutions for their test instances also used in [1, 27].

Burke et al. [13] presented an MA for automated creation of schedules for hospitals in Belgium. Their approach uses a tabu search as local search procedure and produces better results than their previous work using a hybrid tabu search [19]. In 2007, Burke et al. proposed another hybrid EA using the model of Dowsland [27]. Their approach embedded an SAHH as described in the previous section 3.2, but not in the traditional way. Instead of calling the SAHH to independently search for improvements for each individual, this approach stores information of the search globally, i.e., the parameters are changed throughout every call of the SAHH. With this hybrid approach they achieved even better results not only in terms of finding the best known results, but also in terms of robustness compared to other approaches using the same definition and test data [1–3, 25, 27].

For the VRPTW Blanton et al. [9] proposed a first hybrid EA using a greedy algorithm to construct feasible solutions based on the permutation encoded order of customers in the individuals. As summarized in the survey of Bräysy et al. [11] many hybridizations with GAs created good results using construction heuristics [7, 9], local searches [36, 44, 52] and other metaheuristics [6, 34].

Motivation

The use of a population based approach like the GA to tackle this problem was motivated by the overall good performance on related problems in the literature. Using the hybrid variant was introduced early in the development process, as first results suggested that the pure GA would not perform very well. This corresponds with the observations made for related problems where a hybridisation improved the performance of the search. Another motivation to maintain a population was the possible benefit when trying to solve multi day solutions, as a large pool of solutions might be used for refinement.

3.4 Scatter Search

Scatter Search is another population based approach proposed by Glover et al. [31]. It presents a more deterministic approach to population based search methods. The population of solutions, the so-called *reference set*, is initialized to be diverse explicitly by selecting a diverse subset of solutions from a construction method. To create new solutions, a *subset generation* method creates subsets of solutions which are later combined by a *subset combination* method, improved and then, based on the *update* strategy, included into the population.

In their ongoing work they also presented techniques for the subset generation and combination [30, 40]. For the subset generation an iterative approach was suggested. The first set of subsets $\mathcal{U}^1$ (type 1) consists of all pairs of the solutions

$$\mathcal{U}^1 = \bigcup(\sigma_i, \sigma_j), \quad \text{where } i = 1, \ldots, |RefSet| - 1, j > i. \tag{3.2}$$

The following type 2 set $\mathcal{U}^2$ holds all subsets by creating a subset for each type 1 subset and adding the best solution not already in the subset to it.

$$\mathcal{U}^2 = \bigcup(\sigma_i, \sigma_j, \sigma_k), \quad \text{where } (\sigma_i, \sigma_j) \in \mathcal{U}^1, k = min\{k | i \neq k \neq j, \sigma_k \in RefSet\}. \tag{3.3}$$

The same procedure is applied for type 3 subsets $\mathcal{U}^3$ using $\mathcal{U}^2$. As this might create duplicate subsets (i.e. subsets containing the same solutions) only one of the duplicates is generated. If the number of solutions in the reference set is larger than 4 then subsets of size $i = 5, ..., |RefSet|$ containing the i best solutions are generated, where $|RefSet|$ is the size of the reference set, i.e.,

$$\mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2 \cup \mathcal{U}^3 \cup (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) \cup \ldots \cup (\sigma_1, \sigma_2, \ldots, \sigma_{|RefSet|}). \tag{3.4}$$

As combination method they presented *path relinking*. Path relinking generates solutions using the neighbourhood space. Given two solutions, the *initiating solution* and the *guiding solution*, the method identifies the intermediate solutions by calculating a path π of moves from the initiating solution and the guiding solution. Each entry in π is a neighbour of the previous solution, starting with a neighbour of the initiating solution and ending with the last solution before reaching the guiding solution.

After creating new solutions these are improved using for example local search heuristics and afterwards some of these are put into the reference set. Glover et.al. discussed this process and introduced a *tier*-based approach. In this approach, the reference set is split into 2 or 3 tiers. In tier 1 the best solutions are stored, starting with the best solutions to the worst. This part will be updated by replacing worse solutions with the best new solutions created. Tier 2 holds a diverse set of solutions where solutions are stored and replaced based on their contribution to the overall diversity of the set. Tier 3 was proposed by Laguna and Marti [37] where so-called *best generators* are stored. These generators are solutions contributing best when used in the combination method.

Related work

Based on their research of the NRP for hospitals in Belgium, Burke et al. [13] used a scatter search to tackle the problem more efficiently. They used a constructive solution combination technique (see also section 5.3) and tested a hill-climbing and a VND as local search procedure. Compared with the MA approach in [13] the scatter search underperforms when using hill-climbing and produces better results when using VND. However, when VND was used, the computation time increased significantly.

Russel et al. [50] presented a scatter search for the VRPTW using an arc subset generation approach and a global route combination approach where a set partitioning algorithm is used to create new solutions. As local search procedure a previously proposed reactive tabu search [23]

was used. Their approach was able to find 6 best known solutions out of 52 instances [51]. Other research using scatter search was done only for related problems, like the VRPTW with split deliveries by Belfiore [5]

Motivation

As a more deterministic approach, the scatter search seemed promising to find reasonable good solutions using more thorough search techniques then the other selected metaheuristics. The possible high runtimes needed to produce competitive solutions as for related problems was also considered, but the possible gain in terms of solution quality favoured the use of this approach. Additionally the scatter searches' pool might also be used to refine multi day solutions, although the small size might not help as much as the population created by the MA.

CHAPTER 4

Existing Approach

The metaheuristics implemented for this thesis are built into the already existing solving architecture created at the AIT. This framework includes several vital procedures like loading problem instances, creation of the travel time matrix and logging as well as the representation and evaluation of the solution.

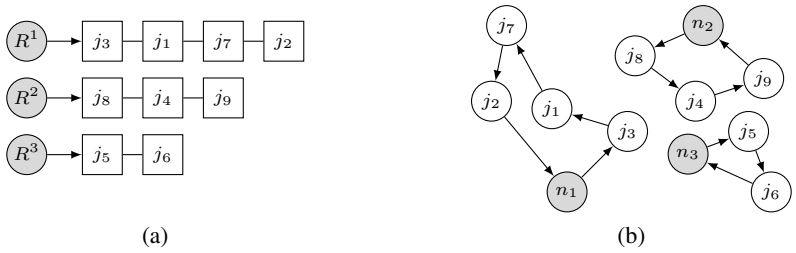
This existing architecture also contains a constraint programming (CP) approach to generate initial solutions in addition to a random generation method and a VND-VNS improvement heuristic with several neighbourhood structures. As the metaheuristics implemented in this thesis use this framework and will be compared to the existing approach, this chapter will provide details of parts of the framework.

4.1 Solution Representation

In the existing approach a solution $\sigma = (\mathcal{R}, \mathcal{S})$ is created by assigning jobs $j \in \mathcal{J}$ to routes $R \in \mathcal{R}$. Each of the job $j \in \mathcal{J}$ is assigned to at most one tour R^n and each job $j \in R^n$ is assigned to a specific position in tour R^n, i.e., for each job $j \in \mathcal{J}$ assigned to tour R^n has a unique position i in the tour represented by R^n_i, where $i = 1, \ldots, |R^n|$. An example is shown in Fig. 4.1:

The set of starting times $\mathcal{S}$ is then constructed using a 2-step greedy algorithm illustrated in Fig. 4.2: In a first step, the earliest possible start times for each job R^n_i for nurse $n \in \mathcal{N}$, where $i = 1, \ldots, |R^n|$, are computed under consideration of the travel time from the location of the previous job $loc(R^n_{i-1})$ (or the the home location $loc(n)$, if $i = 1$) to the location job of R^n_i as well as all time windows, i.e., starting time window of job R^n_i and the nurses' working time windows TW_n. While the travel times are never violated, the time windows are considered as soft constraints (see Sec. 2.3), thus violations of these windows are possible and allowed.

In a second step, all jobs $R^n_i \in R^n$ except the last are moved backwards as far as possible. After each of the two steps, either version can be better due to particular soft constraints, like meeting the desired start times of the customers. Therefore, the assignment is selected



(a) (b)

Figure 4.1: Illustration of the representation (a) and actual tours (b) using an exemplary assignment

(a) (b)

Figure 4.2: An example for the 2-step greedy heuristic for determining job starting times for nurse $n \in \mathcal{N}$: first step (a), starting with the first job, all jobs are set to the earliest possible start time considering the job durations as well as the travel times between the jobs (dashed lines); second step (b), starting from the second to last job, the jobs are moved as far as possible to the end (with respect to their durations and travel times between them).

that results in a better objective value which will be described next. This improvement step is performed whenever a solution is constructed or has been modified.

4.2 Objective Function

As described in the MHS formulation in section 2.3 the problem consists of several constraints representing the interests of employer, employees and customers and a solution should be optimal for all parties. Thus, the problem is defined as a minimization problem with an objective function $ob(\sigma)$ which assigns each solution σ a real number such that valid solutions evaluate to values between 0 and 1 (including), and invalid solutions evaluate to values greater than 1. For this purpose the objective function is split into three parts: the first one computes the number of hard constraint violations ($v_1, \ldots, v_4$), i.e., violations making a solution invalid, the second one is a weighted sum of soft constraint violations ($v_5, \ldots, v_8$), i.e., tolerated but undesired constraint violations, and the third one introduces additional contributions to the objective like working time or overall travel time ($v_9, v_{10}v_{11}$):

$$ob(\sigma) = \sum_{i=1}^{4} v_i + \sum_{i=5}^{8} v_i \cdot \gamma_i \cdot \phi_i + \sum_{i=9}^{11} v_i \cdot \gamma_i \cdot \phi_i \tag{4.1}$$

As can be seen, the soft constraints and additional contributions to the objective function are weighted using factors γ_i and ϕ_i, with $5 \leq i \leq 11$. The values for γ_i are chosen such that

$\sum_{i=5}^{11} \gamma_i = 1$ holds, see table 4.1. The factors ϕ_i are normalization factors. I.e., for each term i a normalization factor ϕ_i is computed such that the highest possible value maps to 1 (e.g. the working time is normalized with respect to the maximal working time, which is ten hours in our case due to legal restrictions). In combination with the weighting factors γ_i it is therefore assured that the worst valid solution is better than the best invalid solution. Note that constraint 4 (travel time constraint) is always fulfilled by the construction of a solution (see previous section 4.1). To enable evaluations of schedules and rosters as finally coarried out in every day's business, this term is nevertheless improtant since unforseen incidents may occur leading to violations of this contraint.

In table 4.1 a brief summary of the terms in the objective function is presented including the settings of the weights γ_i as later used for final computational experiments. These weights were found reasonable to qualify good solutions by favouring solutions with a low number of jobs performed by overqualified nurses, a low derivation from the starting time windows as well as a low number of preference violations.

4.3 Initial Solution Construction

To obtain a first (initial) solution two approaches are implemented in the existing approach: a *constraint programming* (CP) approach and a simple random construction procedure. The CP approach uses decompositions (qualification-wise and spatial clustering) to reduce the problem size and thus improve the runtime performance. The constraint model is an extension of the classical vehicle routing problem with time windows (VRPTW) model from [48] and is described in detail in [47]. The random construction algorithm constructs a solution by traversing the list of available jobs in random order and assigning a job to a nurse in the list of nurses in a cyclic manner. It only ensures that every job is assigned to a nurse (pre-assigned jobs are kept by the originally intended nurse). Note, that the initial solutions produced from the CP approach are valid, while those from the random construction heuristic are (most likely) invalid.

4.4 VND-VNS

The improvement heuristic used in the existing approach is a VNS with an embedded VND as described in section 3.1. The implemented VND uses the following neighbourhood structures (Figure 4.3):

shift job: This move shifts one job $j \in \mathcal{J}^{\overline{pre}}$ from one tour R^{i_1} to another tour $R^{i_2}, i_1 \neq i_2$. In the new tour the best position (according to the objective function) is searched for the shifted job.

reposition job: A job $R^n_{i_1} \in \mathcal{J}^{\overline{pre}}$ of a nurse $n \in \mathcal{N}$ is moved to another position i_2, where $i_2 = 1, \ldots, |R^n|$ and $i_2 \neq i_1$, without reordering the other jobs.

swap nurses: By this move, two nurses $n_1 \in \mathcal{N}$ and $n_2 \in \mathcal{N}$ are swapped with each other, i.e., the tour of the first nurse is then handled by the second nurse and vice versa. Excluded from the swap are nurses with at least one job $j \in \mathcal{J}^{pre}$ assigned to them.

Table 4.1: Influencing terms in the objective function

	γ_i	
v_1	—	contains the number of invalid job-assignments, i.e. either a missing assignment of a job to a nurse or an insufficient qualification of a nurse (Constraint 1 - 3)
v_2	—	counts the number of travel time violations, i.e., the number of times a nurse cannot reach a customer before the service should start. (Constraint 4)
v_3	—	contains the number of violations of nurse availabilities, i.e. the number of jobs that are assigned to nurses outside of the nurses' time window. (Constraint 5)
v_4	—	contains the number of violations concerning pre-allocated jobs that must not be shifted to other times or nurses. (Constraint 6)
v_5	0.2	quantifies the distances from the required qualification q_j of all jobs $j \in \mathcal{J}$ to the qualification q_n of the assigned nurse $n \in \mathcal{N}$. The distance between two qualifications q_i and $\mathrm{q}_{i'}$ is defined as $\lvert i - i' \rvert$. (Constraint 7)
v_6	0.2	quantifies the deviation from the start time windows $[\mathrm{s}_j, \mathrm{e}_j]$ of all jobs $j \in \mathcal{J}$. Violations of these time windows are penalized using a quadratic function except that deviations of three hours and above are considered equally bad. (Constraint 8)
v_7	0.1	quantifies the deviation from the desired start time t_j of all jobs $j \in \mathcal{J}$. Deviations from the start time are linearly penalized, except that similarly to time window violations, deviations of one hour and above are assumed to be equally bad. (Constraint 9)
v_8	0.2	quantifies violations of preferences stated by nurses and customers and is normalized over the number of jobs. (Constraint 10)
v_9	0.1	quantifies the working time outside the daily maximal working time (additional hours of work are counted as overtime and are therefore higher paid, resulting in higher costs for the employer)
v_{10}	0.1	quantifies the working time of all nurses $n \in \mathcal{N}$ up to the daily maximal working time
v_{11}	0.1	quantifies the overall travel time of all nurses.

As step function a next improvement strategy is applied and the initial neighbourhood order is *shift jobs*, *swap nurses*, followed by *reposition job*.

Although this initial neighbourhood order leads to rapid improvements during the starting phase of VND, preliminary tests revealed that dynamic neighbourhood reordering as applied in [45] frequently leads to better results. This reordering is done by first storing the improve-

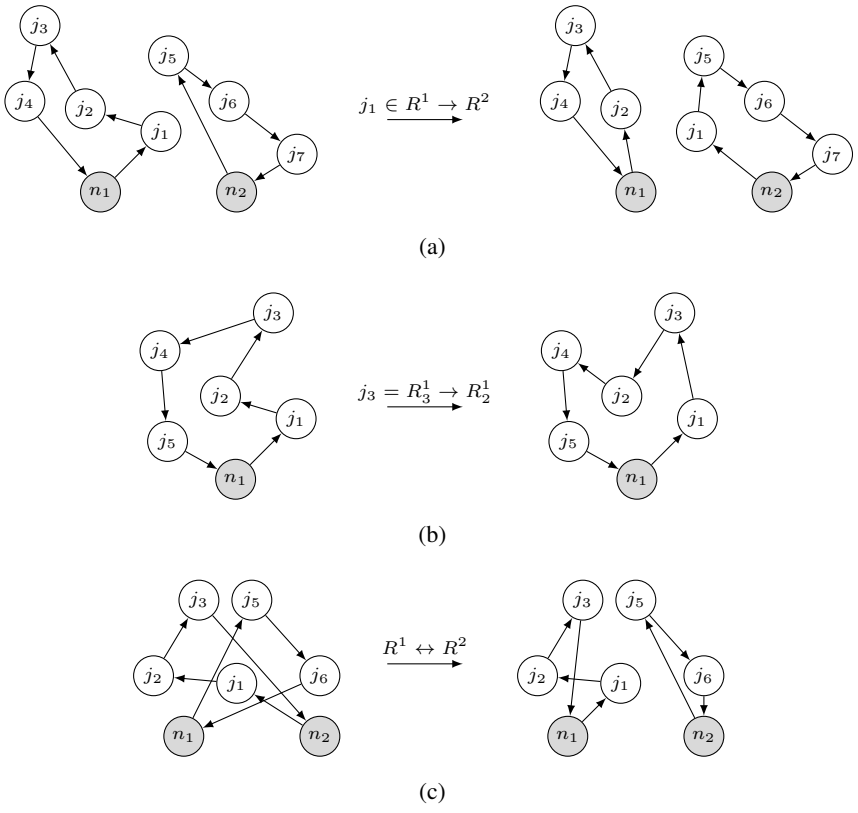


Figure 4.3: Examples for the 3 neighbourhood structures: (a) shift job, (b) reposition job and (c) swap nurse.

ment/examined ratio of each neighbourhood and then reordering them when an improved solution has been found. The outline of the VND approach can be seen in algorithm 4.1.

As the algorithm may get stuck in a rather poor local optimum using VND only, it is embedded as a local search phase in a general VNS scheme as described in section 3.1 using a random shift job move as shaking procedure and $k = 1, \ldots, 5$ (see Algorithm 4.2).

Input: Solution σ, Neighborhood ordering N
Output: Best solution σ^* found

$l \leftarrow 1$; $l_{\max} \leftarrow |N|$; $\sigma^* \leftarrow \sigma$;
while $l \leq l_{\max}$ **do**
  $\sigma \leftarrow N_i(\sigma^*)$;
  $N_l.examined \leftarrow N_l.examined + 1$;
  if $ob(\sigma) < ob(\sigma^*)$ **then**
    $\sigma^* \leftarrow \sigma$;
    $l \leftarrow 1$;
    $N_l.improved \leftarrow N_l.improved + 1$;
    $reorder(N)$;
  else
    $l \leftarrow l + 1$;
  end
end
return σ^*;

Algorithm 4.1: VND

Input: Solution σ, Integer $k_{\max}$
Output: Best solution σ^* found

$k \leftarrow 1$; $\sigma^* \leftarrow \sigma$;
while $k \leq k_{\max}$ **do**
  $\sigma \leftarrow$ *shaking*(k, σ^*); `// perform` k `random shift moves on` σ^*
  $\sigma \leftarrow$ VND(σ); `// do local search (in our case the VND)`
  if $ob(\sigma) < ob(\sigma^*)$ **then**
    $\sigma^* \leftarrow \sigma$;
    $k \leftarrow 1$;
  else
    $k \leftarrow k + 1$;
  end
end
return σ^*;

Algorithm 4.2: VNS

CHAPTER 5

Metaheuristics

Based on the methods described so far, this chapter will present the three metaheuristics implemented into the existing framework to tackle the MHS problem. An important feature of the solving architecture developed at the AIT is the ability to be adapted to the requirements of different providers easily, thus the use of constraint dependent solving methods have been omitted.

Additional information about the parameters and operators used is also provided in the respective sections of the metaheuristics. Further preliminary experiments are described in the next chapter.

5.1 Simulated Annealing Hyper-Heuristic

The developed approach differs from Bai et al.'s SAHH (see section 3.2) as the low-level heuristics and parameters have been adapted to solve the MHS problem. The outline of the algorithm is shown in algorithm 5.1.

The low-level heuristics for the NRP proposed in the work of Bai et al. uses low-level heuristics creating solutions based on specific constraints. As this thesis focus on a constraint-independent approach, more general low-level heuristics have been implemented. These are variants of simple local search procedures, which are created by combining every move type described in section 4.4 with two improvement strategies, i.e. the classic next improvement and a random-improvement strategy, leading to a total number of six low-level heuristics.

The use of random improvement is obvious as the SAHH should be able to accept also worse solutions in order to escape local optima. Using next improvement in favour of best improvement provides the higher chance to escape a local optimum after a worse solution of the same neighbourhood structure has been accepted. Another feature of next improvement is the lower search time needed in average as not the whole neighbourhood has to be searched. The same arguments can be used when comparing with the best of improvement strategy.

Bai et al. suggest that around 10% of the solutions should be accepted at the beginning (r_{start}) and only 0.5% at the end of the search (r_{end}). As described in section 3.2 the probabilistic

Input: Solution σ
Output: Best solution found σ^*

$i \leftarrow 1$; $\sigma^* \leftarrow \sigma$;
$t \leftarrow t_{start}$; $t_{imp} \leftarrow t_{start}$;
$resetLLHStats()$;
while $i \leq K$ **do**
  $h \leftarrow selectLLH(H)$;
  $\sigma' \leftarrow h(\sigma)$;
  $h.tested \leftarrow h.tested + 1$;
  if $\sigma' = null$ **then** $d \leftarrow +\infty$;
  else $d \leftarrow ob(\sigma') - ob(\sigma)$;
  if $d < 0$ **then**
    $\sigma \leftarrow \sigma'$;
    $h.new \leftarrow h.new + 1$; $h.accept \leftarrow h.accept + 1$; $C \leftarrow C + 1$;
    if $ob(\sigma') < ob(\sigma^*)$ **then** $\sigma^* \leftarrow \sigma'$;
  else
    if $e^{-d/t} > Random()$ **then**
      $\sigma \leftarrow \sigma'$;
      $h.accept \leftarrow h.accept + 1$; $C \leftarrow C + 1$;
    end
    $C \leftarrow C + 1$;
  end
  if $reheating = true$ **then**
    $t_{imp} \leftarrow t_{imp}/(1 - (beta() * t_{imp}))$;
    if $t_{imp} > t_{start}$ **then** $t_{imp} \leftarrow t_{start}$;
    $t \leftarrow t_{imp}$;
  else if $i \,\%\, nrep = 0$ **then**
    $t \leftarrow t/(1 + (beta() * t))$;
  end
  if $i \,\%\, LP = 0$ **then**
    if $C/LP < r_{end}$ **then**
      $reheating \leftarrow true$; $t_{imp} \leftarrow t_{imp}/(1 - (beta() * t_{imp}))$;
      $t \leftarrow t_{imp}$; $\sigma \leftarrow \sigma^*$;
      $adjustLLH_{New}()$;
    else
      $adjustLLH_{Accept}()$;
    end
    $resetLLHStats()$;
  end
  $i \leftarrow i + 1$;
end
return σ^*;

Algorithm 5.1: Simulated Annealing Hyper-Heuristic procedure

Table 5.1: Parameter settings for SAHH

SAHH Parameters	**Settings**
K (# iterations)	10000
acceptance prob. (start)	0.10
acceptance prob. (end)	0.005
learn period	$K/500$
nrep (#iter/temp)	6 (#neighbourhoods)
min weight	0.1
change of temperature t	Lundy and Mees' nonlinear function: $t \leftarrow t/(1+\beta t)$, $\beta = ((t_{\text{start}} - t_{\text{end}}) \cdot nrep/K \cdot t_{\text{start}} \cdot t_{\text{end}})$

decision whether to accept an inferior solution is based on the formula $e^{-d/t}$. To calculate the start and end temperature a proper value for the term d in the following equations has to be found: $e^{(-d/t_{start})} = 0.1$ and $e^{(-d/t_{end})} = 0.005$.

After some preliminary testing $d = 1.0$ was selected, thus 10% of the solutions with an objective value of 1.0 worse than the current one are selected at the beginning and 0.5% at the end. The final settings are summarized in table 5.1.

5.2 Memetic Algorithm

As described in section 3.3 an MA might be encoded using a alternative form of representation to benefit from special recombination operators or heuristic decoding algorithms. In an early stage of development, a binary as well as an integer based encoding has been considered to represent an assignment of a job to a nurse of vice versa. The both binary approaches as well as the integer representation of the nurses assigned to a job would need a decoding algorithm to determine the order in which the jobs are served. This would need additional computational effort or perhaps another heuristic which would decrease the overall performance of this approach. The other integer encoding (job to nurse) is the existing representation when each job only occurs once and the number of jobs per nurse is known.

Another design choice made early was the number of offsprings created in each iteration. The final choice of a single offspring per iteration was made after first test runs indicated that the creation of a larger number of offsprings with the additional local search calls lead to a slow converge of the population to good solution regions.

In the following, the initialization, selection and replacement as well as the special operators implemented for this representation and problem are described in detail. A summary of the final operators used can be found in table 5.2.

Initialization. To avoid duplicates as well as to provide an initial diversity of the population a random diverse constructor was implemented to create the initial solutions. These are constructed using a memory of already used assignments in previously generated solutions. To construct a new solution most different from the already constructed ones, for each job j, a

Table 5.2: Operator setup for the Memetic Algorithm (MA)

MA Operator	Settings
selection	binary tournament
recombination	tour replace crossover
mutation	move all unfixed missions to best other nurse
replacement	replace the worst similar solution in the population pool
improvement	cyclic search of neighborhoods
	- reorder mission neighborhood (best of improvement)
	- shift mission neighborhood (best of improvement)
	- swap nurse neighborhood (best of improvement)

nurse that has been least often assigned to j is selected and job j is added to the end of the tour of the nurse. Possible ties are broken randomly. The constructor memory is initialized with the starting solution (CP or random) by increasing the count of the corresponding job to nurse assignments. This solution is also always part of the initial population.

Selection. As selection strategy a binary tournament selection is used where solutions are selected by repeatedly picking two solutions at random and including the better one into the set of selected solutions. Note that this approach does not check for duplicates, i.e., solutions may appear multiple times in the set of selected ones.

Recombination Operator. As solutions are lists of tours (one for each nurse), a special recombination operator was implemented (Figure 5.1): given two parent solutions (P_1, P_2), the offspring is initially set to have the same tours as P_2. Then one nurse of P_1 is selected at random (n). Each non-preallocated job of the tour R^n assigned in P_1 is removed in the offspring solution. Then every remaining job of R^n in the offspring solution is moved to the best other nurses' tours R^m, where $n \neq m, m \in \mathcal{N}$. In the final step, the jobs of R^n in P_1 are assigned to R^n in the offspring solution.

This recombination operator was implemented early in the thesis and performed very well due to the fact that the changes made to the parent solution are rather small – only the jobs of two tour are moved – thus preserving much of the solution structure. By selecting a nurses' tour at random, the operator need very little computational time which allows the local search procedure more of the overall runtime. An additional design choice was made by allowing not only the parent so be identical, but also to allow the random selection of a nurses' tour which is identical in both parents. By allowing parents to become offsprings, a possible local search call will improve this parent further, thus this intentional creation of duplicates enhance the ability of the MA to exploit the search space.

Mutation. To provide higher diversity during the search of the MA, the offspring is mutated by selecting a tour R^n at random and reassigning all unfixed jobs of this tour to other tours R^m, where $n \neq m$ and $n, m \in \mathcal{N}$, minimizing the objective value.

This mutation procedure was considered due to the structure of the MHS problem. The idea is to support the search to remove nurses from the schedule by clearing their tours, i.e.,

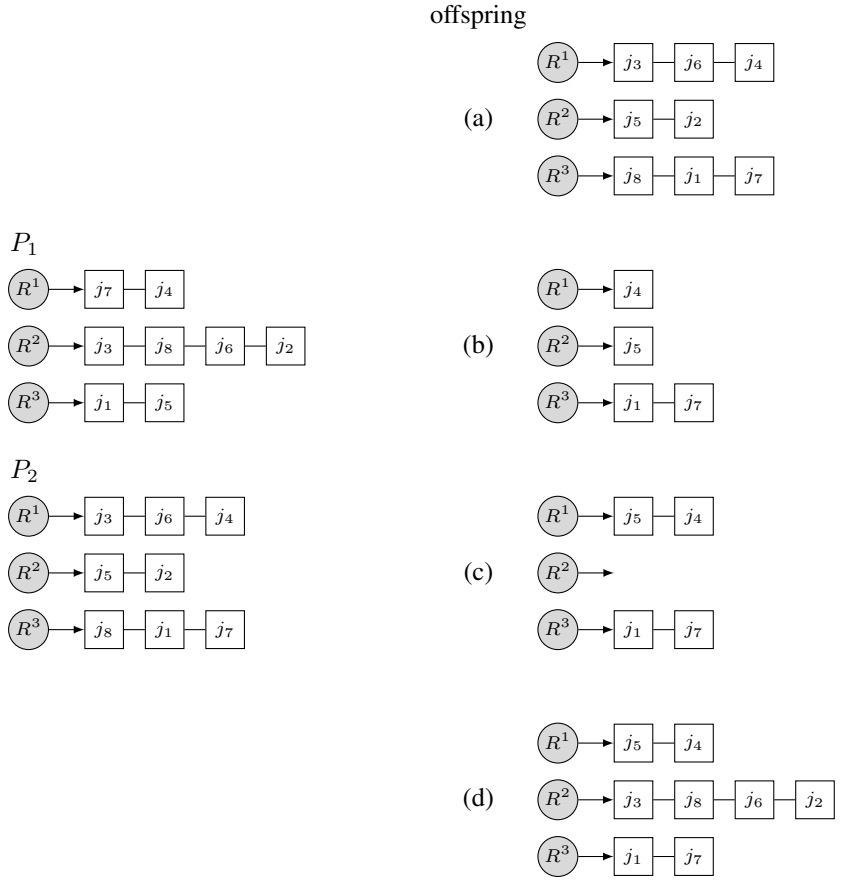


Figure 5.1: An example for the recombination operator (where R^2 is selected) step by step: (a) offspring is set to be as P_2, (b) all jobs of R^2 in P_1 are removed from the offspring, (c) all remaining jobs in R^2 in the offspring are moved, (d) R^2 in the offspring is filled with the jobs of R^2 in P_1

performing chained shift job moves which may not be considered by the local search procedure. Another mutation operator tested was a single job shift move but preliminary results showed that the other operator performs better.

Local Search. For a given probability, a local search heuristic tries to further improve the offspring. To achieve a controlled balance between exploration and exploitation, the heuristic aborts after a certain time limit. Two local search algorithms are applied: a VND, as described in section 4.4 and a cyclic search of neighbourhoods (CNS). The CNS is similar to the VND but always turns to the next neighbourhood structure in a cyclic manner when a local optimum in one neighbourhood structure or a time limit has been reached.

A similar approach is the so-called *token-ring search* proposed by Gaspero et al. [29]. Preliminary results showed that the CNS yields better results than the VND.

Replacement. The population is replaced using a slightly changed steady-state approach [35], where an offspring always replaces the most similar solution in the current population that is

Table 5.3: Parameter settings for SS

SS Parameters	**Settings**
RefSet size	5
combination operator	path-relinking
s_{max}	100
time-limit LS	10sec
termination	time limit or no improvement after an iteration

worse than the offspring. To calculate the similarity of two solutions, the number of same job to nurse assignments in the compared solutions are counted.

This straight forward approach ensures a minimal grade of diversity by omitting duplicates. In preliminary tests a alternative replacement strategy was tested. This alternative replaced the most similar worse solution in the population with the newly created one. This lead to a slow converge of the average quality of the solutions in the population and did not yield considerable better results.

5.3 Scatter Search

Like for the MA a scatter search metaheuristic also consists of several operators like the subset combination and the improvement operator. For the subset generation the classic approach as described in section 3.4 is used with a minor modification: instead of creating all three types of subsets, only type 1 and 2 are created and k was set to four. Thus only one subset containing the best four solutions is created and added to $\mathcal{U}$.

This has been modified after preliminary results showed that the metaheuristic need very high runtimes to find reasonable solutions, thus the number of solutions generated and improved per iteration was reduced.

For the improvement the same local search procedures as described in the previous section are used. The initialization is also done using a random diverse constructor. In the remainder of this section the implemented subset combination methods will be described in detail as well as the update method. The final settings are summarized in table 5.3.

Subset Combination 1. For the combination of solutions for each subset a 'construction by voting' combination algorithm as suggested by Glover et al. [31] and described by Burke et al. [14] was implemented. This approach creates a new solution using candidates. A candidate for our problem is an assignment of a job j to a tour R^n. Figure 5.2 shows an example using two solutions as voters and procedure itself is outlined in algorithm 5.2.

First, all possible candidates are created and the solutions from the subset with the same assignment are linked to the corresponding candidate. Then the list of candidates are sorted descending by the number of solutions (voters) linked to it. After this step, the construction starts at the beginning of the list.

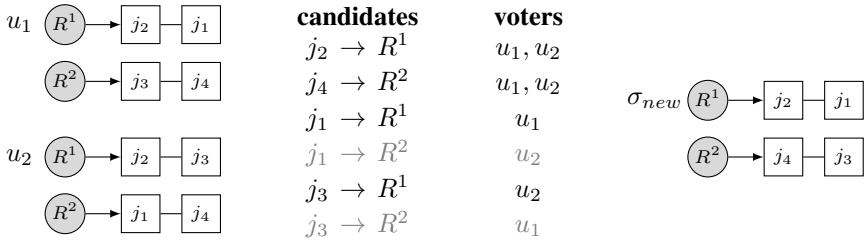


Figure 5.2: Example for the construction by voting subset combination method: using a list of candidates (middle) of two solutions (left) to create a new solution (right). Unused candidates are grey.

Input: set of solutions U
Output: constructed solution σ
1 $\sigma \leftarrow createEmptySolution()$;
2 $assignFixedMissions(\sigma)$;
3 **foreach** $j \in \mathcal{J}^{\overline{pre}}$ **do**
4 **foreach** $u \in U$ **do**
5 **if** $\exists c_{j,R} \in C$ **then** $Voters[c_{j,R}] \leftarrow Voters[c_{j,R}] \cup u$);
6 **else**
7 $Voters[c_{j,R}] \leftarrow \{u\}$;
8 $C \leftarrow C \cup c_{j,R}$;
9 **end**
10 **end**
11 **end**
12 $sortByVoters(C)$; // $c_{j1,R1} > c_{j2,R2}$ if $|Voters[c_{j1,R}]| > |Voters[c_{j2,R2}]|$
13 **foreach** $c_{j,R} \in C$ **do**
14 $C' \leftarrow \bigcup c_{j,i} \in C$, where $\forall i \in \mathcal{R}$;
15 $c_{j,R} \leftarrow selectFrom(C')$;
16 $apply(c_{j,R}, \sigma)$;
17 $C \leftarrow C \setminus \{c_{j,i}\}, \forall i \in \mathcal{R}$;
18 **end**
19 **return** σ;

Algorithm 5.2: Subset Combination: "construction by voting"

If an assignment of a job j has two candidates with the same number of voters, a tie breaking mechanism is used to decide, which candidate will be applied. First the number of successful votes of the voters are counted (i.e. how many times a candidate of a solution was used to construct the new solution), and the candidate with the least used voters is used. As this count could also be equal, a second tie breaker compares the sum of objective values of the voters and takes the candidate with the lowest sum.

If a candidate with an already assigned job is encountered during the search, the candidate is removed.

Input: sorted set of solutions $U = \{u_1, \ldots, u_l\}$ of size l, where
$obj(u_1) \geq obj(u_2) \geq \cdots \geq obj(u_l)$
Output: constructed solution σ

$\sigma \leftarrow u_1$;
for $i \leftarrow 2, \ldots, l$ **do**
  $d \leftarrow \lfloor difference(\sigma, u_i)/l \rfloor$;
  for $z \leftarrow 1, \ldots, d$ **do**
    $N' \leftarrow \{\sigma^* | \sigma^* \in N^{shift}(\sigma)$ where job assignment in σ was different to $u_i\}$;
    $\sigma' \leftarrow N'_1$;
    for $q \leftarrow 2, \ldots, max(|N'|, s_{max})$ **do**
      if $obj(N'_q) < obj(\sigma')$ **then** $\sigma' \leftarrow N'_q$;
    end
    $\sigma \leftarrow \sigma'$;
  end
end
return σ;

Algorithm 5.3: Subset Combination: Path-Relinking

Subset Combination 2. After some preliminary tests indicating that the previously described approach does not perform well, a second approach was designed and implemented using a path-relinking algorithm (see section 3.4). Instead of using the proposed approach when using multiple guiding solutions in Glover et al. [31] – where the initial solution moves to a solution created by combining all parent solutions – a different procedure was designed: To create a solution σ_{new} out of a sorted subset $U = \{u_1, \ldots, u_l\}$ of size l, where $ob(u_1) \geq \ldots \geq ob(u_l)$, the algorithm moves iteratively from the worst solution u_1 to the best u_l. Then $\sigma_{new} = u'_l$, where

$$u'_i = u'_{i-1} \rightarrow_{PR} u_i, i = 2, ..., l, u'_1 = u_1 \tag{5.1}$$

and the path-relinking operator $A \rightarrow_{PR} B$ moves A $difference(A, B)/l$ steps to B (the function $difference$ returns the number of different job to tour assignments). To determine which moves are performed first, the change in the objective per move is calculated and the move with the best change (which can also be worse) is performed. In order to prevent high runtimes due to a possible large set of moves to be tested, the number of moves to be calculated is restricted with a parameter s_{max}. The outline of this algorithm is shown in alg. 5.3.

This approach is built upon the idea that a good solution for further improvement is situated between solutions in the reference set. Thus by performing a fixed number of steps toward these solutions and trying to improve the first solution as much and deteriorate as least as possible, a solution not only of good quality in terms of the objective function should be generated but also a solution with an rather unexamined neighbourhood space.

Reference Set Update. After each combination of a subset, the generated solution will be improved and replaces the worst solution in the current reference set, but only if it is not already

in the set. The diversity control, as considered in the original scatter search proposed by Glover et al. [31] will be not be maintained. Burke et al. [14] recommend this step in order to have more computational time to search in better regions (based on the objective value). Preliminary results suggested that the use of a diverse set does not improve the overall performance.

CHAPTER 6

Results

This section describes the experimental results obtained. This includes the comparison of all described approaches with long runtimes as well as parameter tests with shorter but reasonable runtimes.

6.1 Test Environment

The tests used 8 instances with the following dimensions:

	day 1	day 2	day 3	day 4	day 5	day 6	day 7	day 8
# nurses	509	491	504	482	496	518	500	505
# missions	711	700	679	682	708	699	717	679

The real-world instances tested were taken from a random selection of 2011 instances provided by Sozial Global. All techniques have been implemented within the same framework in Java 1.6.

The tests for the final comparison where run on a single core of a Intel(R) Core(TM)2 Quad CPU Q9300 2.50GHz with (at most) 4GB RAM assigned. The parameter tests were run on a Intel(R) Core(TM) i7-2630QM CPU 2.0GHz with (at most) 16GB RAM assigned.

6.2 Parameter Testing

To justify the selection of final parameter setting this subsection will describe the tested parameters as well as the impact of algorithm design changes made. The tests used 4 instances (day 4-7) and were run 10 times per instance with a maximum runtime of 10 minutes per run. For the most interesting cases, tables with the average best solutions are provided as well as the standard deviation below.

Table 6.1: SAHH parameter tests with different d-values in $e^{-d/t} = r$

	day 4	day 5	day 6	day 7
init.	0.08962	0.08854	0.08667	0.09038
$d = 0.5$	0.03493 ±0.00285	0.03396 ±0.00158	0.03273 ±0.00066	0.03614 ±0.00294
$d = 1.0$	0.03425 ±0.00201	0.03424 ±0.00183	0.03291 ±0.00119	0.03502 ±0.00247
$d = 2.0$	0.03427 ±0.00205	0.03393 ±0.00261	0.03233 ±0.00168	0.03537 ±0.00202

Simulated Annealing Hyper-Heuristic

Two influential parameters for the SAHH are the number of iterations (and therefore the annealing duration) as well as the starting and ending acceptance ratio. Another factor tested was the acceptance of random moves increasing the number of hard constraint violations.

Number of iterations K. Starting from a rather low number of iterations (1000) the number was steadily increased as the search ended after a short amount of time. It was also observed that the number of iterations performed in a time limit of 10 minutes differs between the runs. Thus the call of the algorithm was built into a loop, so that the algorithm will restart with the best solution found so far if the iteration limit was reached before the time limit was exceeded.

Calculation of the inital temperature t_{start} | t_{end}. As described in section 5.1 the start and end temperature were calculated by defining the value d in the formula $e^{-d/t} = r$, where r is the desired probability at the beginning or at the end. As the selected d has an high impact for the calculation of the temperature, different assignments have been tested. As shown in table 6.1, no value seems to outperform the other. Thus 1.0 was selected to provide a balance between exploration and exploitation for larger runtimes.

Acceptance of invalid solutions. The general approach for a simulated annealing procedure is to also accept far worse solutions (but with a very low probability). As part of the preliminary tests, an additional acceptance criterion was introduced. This only allows solutions with at most the same number of violated hard constraints as the current solution and therefore only accepting more 'valid' solutions. As the results show (Table 6.2), this additional criterion does not lead to a considerable improvement in this short amount of time used in the tests. As such a restriction may lead to a high probability to get stuck in a local optima it was removed in the final algorithm.

Memetic Algorithm

For the memetic algorithm different local search procedures were tested, i.e. VND and CNS. Further another mutation operator was considered and the influence of the population size was observed.

Table 6.2: SAHH parameter tests comparing the performance of accepting only solutions at most the same number of hard constraint violations (only valid) and the classic variant where all could be accepted (all)

	day 4	day 5	day 6	day 7
init.	0.08962	0.08854	0.08667	0.09038
all	0.03425 ±0.00201	0.03424 ±0.00183	0.03291 ±0.00119	0.03502 ±0.00247
only valid	0.03470 ±0.00256	0.03437 ±0.00185	0.03279 ±0.00166	0.03607 ±0.00229

Table 6.3: Objective values obtained using no local search (pure EA), VND and CNS

	day 4	day 5	day 6	day 7
init.	0.08962	0.08854	0.08667	0.09038
pure EA	0.03853 ±0.00108	0.03849 ±0.00065	0.03692 ±0.00137	0.03902 ±0.00112
MA with VND	0.03097 ±0.00089	0.03093 ±0.00066	0.02928 ±0.00094	0.03154 ±0.00101
MA with CNS	0.02940 ±0.00065	0.02958 ±0.00065	0.02808 ±0.00059	0.03039 ±0.00107

Local search procedure. Table 6.3 presents results for three different setups: a pure evolutionary algorithm (EA), i.e. the MA without local search phase, the VND and the CNS. At a first glance, it can be observed that the hybrid setups are more promising than the pure EA. Furthermore, the embedding of CNS seems to be more reasonable than the application of VND. This can be explained by the fact, that the time limit for VND/CNS is relatively tight such that VND, which examines the first neighbourhood until no further improvement can be found therein, often does not reach the second or even third neighbourhood. CNS, in contrast, achieves improvements for all defined neighbourhood structures and therefore better exploits the existing improvement potential. The results show that the use of a local search procedure is an important addition but also that the recombination operator performs good on its own.

Mutation operator. A second mutation operator was also tested: a simple random shift job move where a job is selected at random and is shifted to another nurse. Preliminary results showed that both operators perform almost equally well, but more in favour of the mutation operator described in section 5.2 (clearing a tour by moving the jobs to other nurses).

Population size. The influence of the population size also affects the progress of the search, i.e. the smaller the size, the sooner the population converges to a local optimum and the larger the population, the longer it searches in less promising regions. After preliminary tests a population size of 100 was selected.

Table 6.4: Objective values obtained SS1 and SS2 (after 75 minutes)

	day 4	day 5	day 6	day 7
init.	0.08962	0.08854	0.08667	0.09038
SS1	0.03897 ±0.00062	0.04030 ±0.00091	0.03865 ±0.00120	0.08639 ±0.00104
SS2	0.03220 ±0.00062	0.03202 ±0.00091	0.03057 ±0.00120	0.03300 ±0.00104

Scatter Search.

As for the MA, the influence of both local search procedures were tested as well as the effect of the reference size on the performance of this approach. Another test was performed to compare the two subset combination technique described in Section 5.3.

Subset combination technique. As shown in table 6.4, the path-relinking method outperforms the 'construction by voting' approach significantly. The reason for this lies in the low objective value of the constructed solutions in the second approach, and thus the inefficient local search call. Note that these experiments were run for 75 minutes to obtain detailed information on these time consuming approaches.

Local search procedure. Both local search procedures, the VND and CNS, were tested as improvement method in the final scatter search approach. These tests indicated that both procedures perform equally well. Thus VND was selected for the final comparison as it provides a more deterministic and thorough performance.

Reference set size. Changes in the size of the reference set were also tested and it was observed that a size of 5 provides a reasonable good balance between the computational effort of each iteration using larger sizes and to avoid to converge to local optima early due to a small set size.

6.3 Empirical Comparison

For these 8 days, each algorithm was run 10 times per initial construction method and 75 minutes per run. Table 6.5 shows the results of these test runs. Depending on the construction method (CP or Random Construction), each improvement algorithm uses the same initial solution. The value in the 3. column shows the objective value of the initial solution. For each algorithm (starting from column 4) the average best solution is provided as well as the standard derivation (below). The VND results have been obtained by using the objective value after the first VND call in the VNS. Note: instance 3 (day 3) has an error in the data which makes it impossible to create a valid solution.

As table 6.5 shows, the MA approach obtained the best results over all tested instances, whether using the CP or random initialized solution. Also in terms of robustness the MA performs best compared to the other approaches as it produces very similar results in terms of the objective value.

Table 6.5: Final objective values for each metaheuristics, averaged over 10 runs

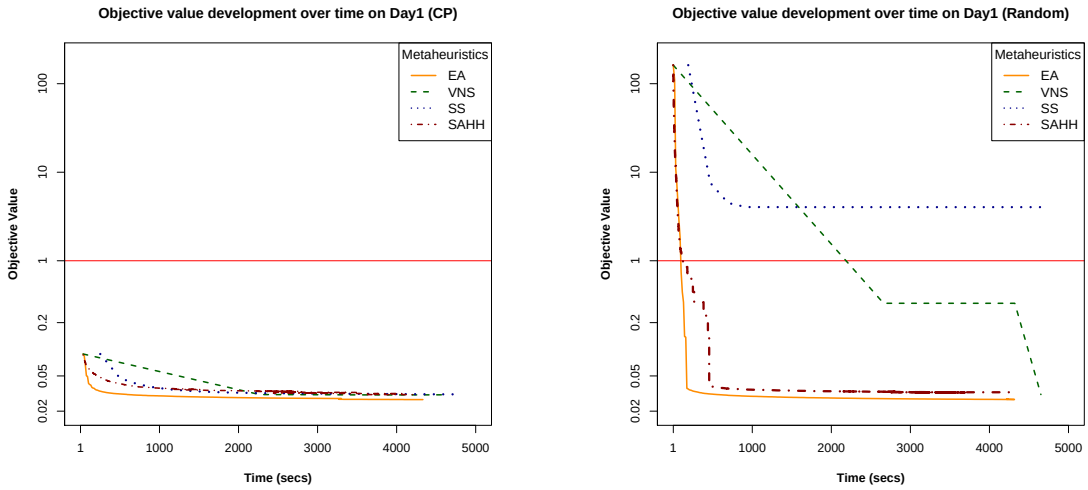
	constr.	init.	VND	VNS	EA	SAHH	SS
day 1	cp	0.0885	0.03080 ±0.00067	0.03038 ±0.00034	0.02744 ±0.00057	0.03260 ±0.00090	0.03097 ±0.00034
	rand.	162.1513	0.33083 ±0.48339	0.33064 ±0.48340	0.02707 ±0.00031	0.03257 ±0.00058	4.03197 ±0.00065
day 2	cp	0.0894	0.03090 ±0.00039	0.03038 ±0.00028	0.02794 ±0.00037	0.03381 ±0.00081	0.03120 ±0.00050
	rand.	152.1521	0.53162 ±0.52725	0.43109 ±0.51647	0.02767 ±0.00037	0.03321 ±0.00082	3.03098 ±0.00055
day 3	cp	1.0860	1.02879 ±0.00055	1.02837 ±0.00044	1.02600 ±0.00051	1.03063 ±0.00096	1.02931 ±0.00051
	rand.	148.1494	1.02895 ±0.00043	1.02851 ±0.00040	1.02553 ±0.00036	1.03023 ±0.00070	2.02875 ±0.00048
day 4	cp	0.0896	0.03132 ±0.00044	0.03083 ±0.00048	0.02874 ±0.00042	0.03302 ±0.00073	0.03220 ±0.00051
	rand.	155.1532	0.03232 ±0.00083	0.03188 ±0.00087	0.02784 ±0.00056	0.03470 ±0.00113	0.03069 ±0.00040
day 5	cp	0.0885	0.03117 ±0.00052	0.03076 ±0.00050	0.02850 ±0.00044	0.03283 ±0.00083	0.03202 ±0.00077
	rand.	168.1506	0.03167 ±0.00057	0.03131 ±0.00068	0.02805 ±0.00047	0.03286 ±0.00061	0.03106 ±0.00048
day 6	cp	0.0867	0.03028 ±0.00055	0.02964 ±0.00060	0.02665 ±0.00048	0.03200 ±0.00080	0.03057 ±0.00068
	rand.	145.1523	0.03040 ±0.00051	0.02992 ±0.00056	0.02646 ±0.00025	0.03228 ±0.00076	0.02947 ±0.00043
day 7	cp	0.0904	0.03209 ±0.00049	0.03165 ±0.00054	0.02947 ±0.00035	0.03377 ±0.00098	0.03300 ±0.00050
	rand.	149.1516	0.03270 ±0.00085	0.03218 ±0.00080	0.02900 ±0.00042	0.03318 ±0.00062	1.03271 ±0.00049
day 8	cp	0.0872	0.02914 ±0.00071	0.02846 ±0.00056	0.02576 ±0.00038	0.02994 ±0.00061	0.02995 ±0.00046
	rand.	158.1446	0.22922 ±0.63228	0.22893 ±0.63227	0.02520 ±0.00041	0.02969 ±0.00064	4.46026 ±0.53455

The VNS approach implemented by the AIT got the second best scores but only if started with a valid solution, i.e., a solution constructed using CP. For day 1, 2 and 8 the VNS failed to find valid solutions using a randomly initialized solution for at least one run.

These instances seem to be difficult to solve using only a randomly generated solution as for them and the day 7 instance the scatter search as proposed in this thesis could not find a valid solution on any run. The results of the scatter search are similar to those of the VNS except for the mentioned instances using random starting solutions.

From the results of the SAHH it can be observed that this approach may not find solutions as good as the other approaches, but it is independent of the starting solution and could find a solution for the previously mentioned difficult instances every run.

More information on the behaviour of the metaheuristics can be obtained by comparing the changes of the objective value with the changes of the average travel time and the number of nurses during the search. These values have been taken from intermediate solutions of the search procedures using the best run of the day 1 instance with either the CP (6.1a,6.2a,6.2c) or the random generated solution (6.1b,6.2b,6.2d). Note: The VNS values shown in the figures are obtained every iteration of the VNS, i.e., not every step of the embedded VND is plotted.



(a) (b)

Figure 6.1: Changes in the objective value over runtime for day1 for CP (a) and random (b) initialization.

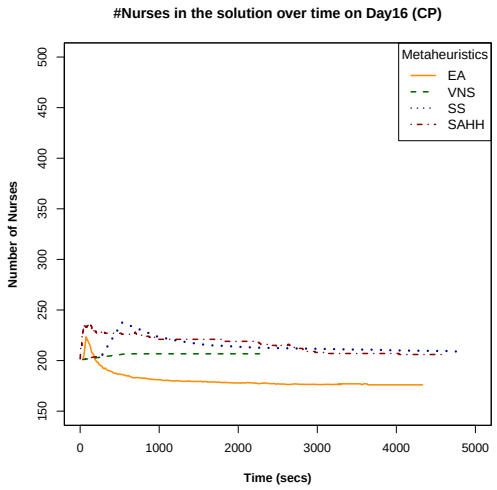
The changes of the objective value (6.1a,6.1b) show that the MA finds a solutions better than obtained by the other approaches early in the search and is able to improve it further. As it can be seen in figure 6.2a the number of nurses increases slightly while the average tour length (Figure 6.2c) decreases at the same time. Therefore it can be argued that first improvements of the initial solution generated by the CP can be obtained by increasing the number of nurses and decreasing the travel time. More interesting are the further improvements made by the MA. By decreasing the number of nurses below the number of the initial value while still having travel times also below the initial one the MA can outperform the other approaches.

A similar observation can be made when looking at figures 6.2b and 6.2d for the randomly initialized solution. Here the first improvements are made by optimizing the travel times and reducing the number of nurses at the same time (as, by construction, all nurses have at least one job assigned). After the initial decrease of the average travel time it increases again while the number of nurses still decreases.

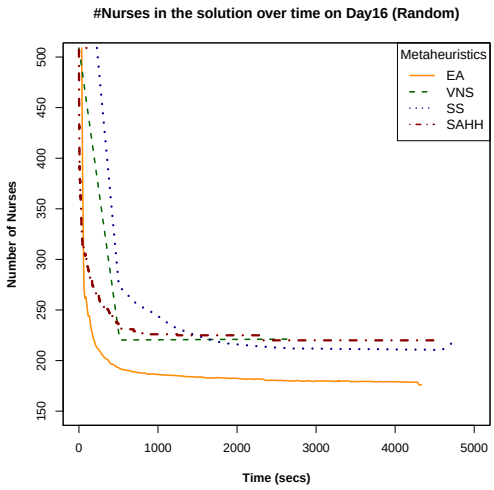
Both observations suggest, that a good solution based on the objective function used in this thesis is more characterised by a low number of nurses used than on a low average travel time per nurse.

The changes in the objective value of the approaches for the other instances in this comparison are shown in figures 6.3, 6.4, 6.5 and 6.6. These show a similar behaviour as the discussed results of the day 1 instance. The MA and the SAHH converges early to the region of very good solutions but only the MA is able to further decrease the objective value thus leading to superior results.

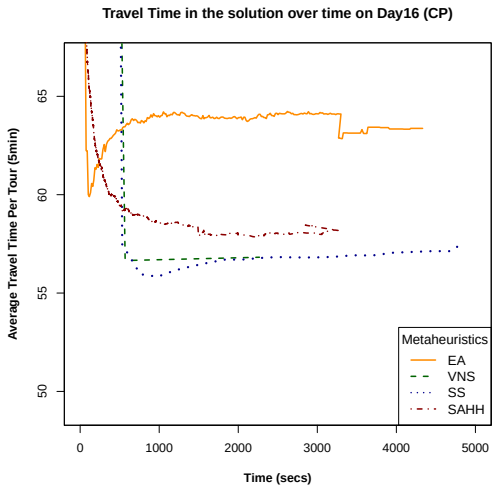
With respect to the number of nurses used within the solution, it can be observed that all approaches reach solutions where the final number of nurses employed is relatively low (see table 6.6). Such solutions are preferred as future work will consider solving multi day instances



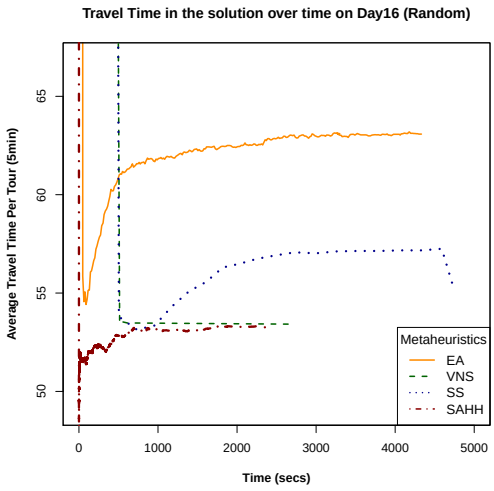
(a)



(b)



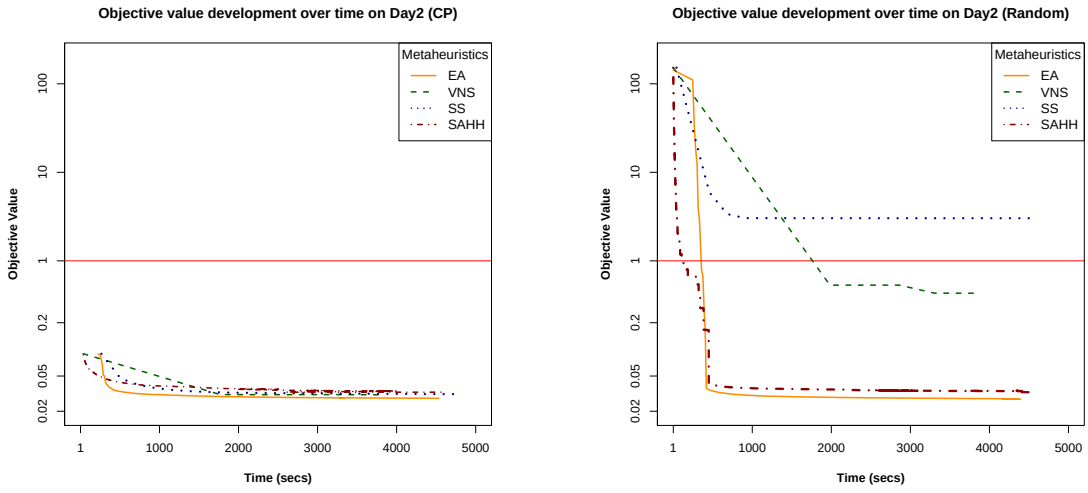
(c)



(d)

Figure 6.2: Key indicator performance over runtime for day1: number of used nurses for CP (a) and random initialization (b); average travel time per nurse for CP (c) and random initialization (d).

which have additional constraints for employed working days. Thus a low number of nurses will provide an additional grade of flexibility when refining a day's schedule.



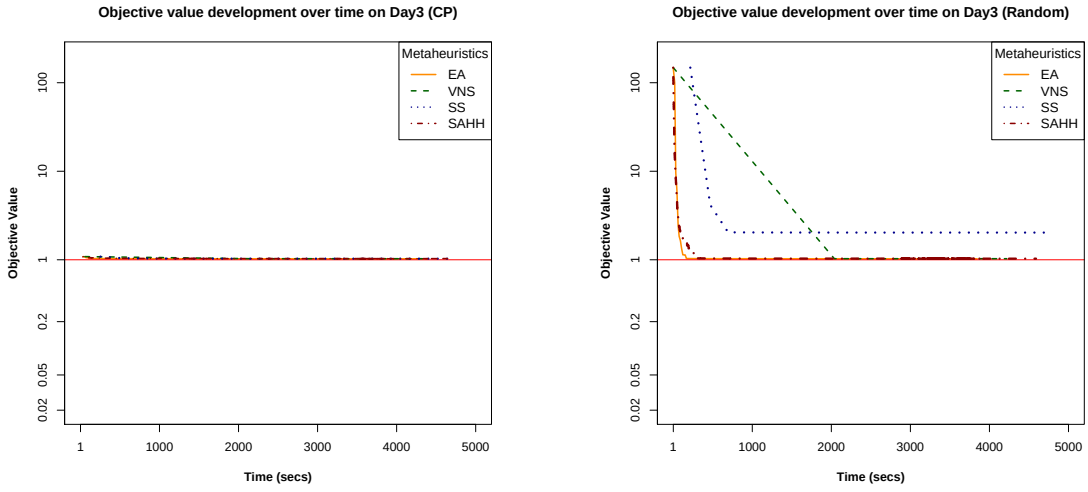
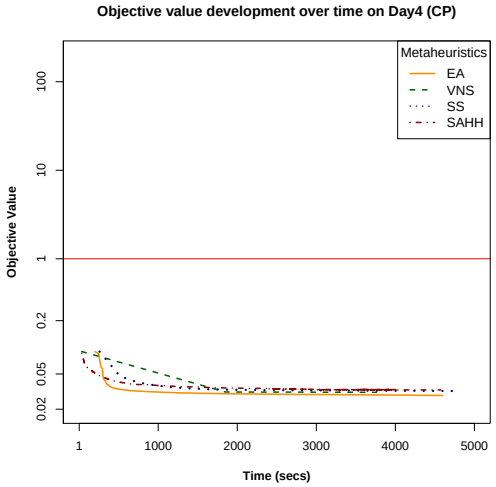
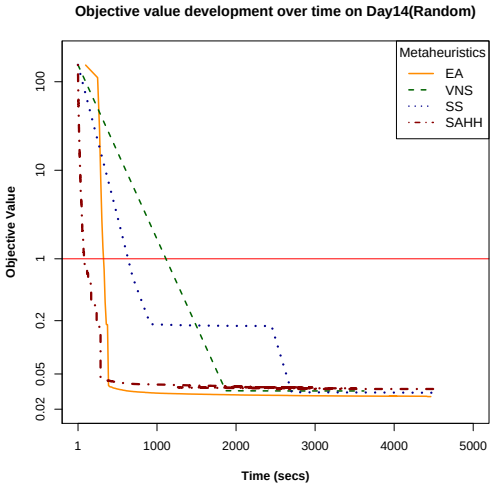


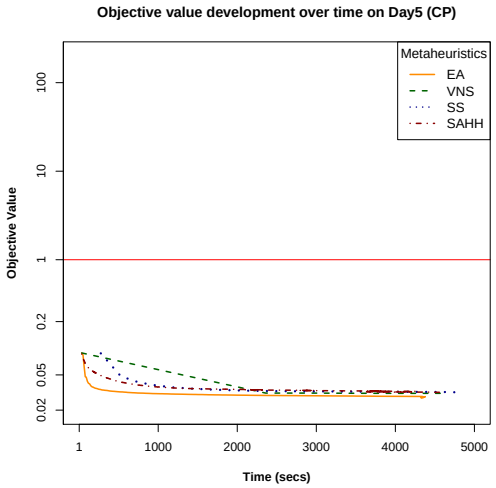
Figure 6.3: Changes in the objective value for day 2 with CP (a) and random initialization (b); and day 3 with CP (c) and random initialization (d). Note that day 3 has an error in the instance data making it impossible to create a valid solution.



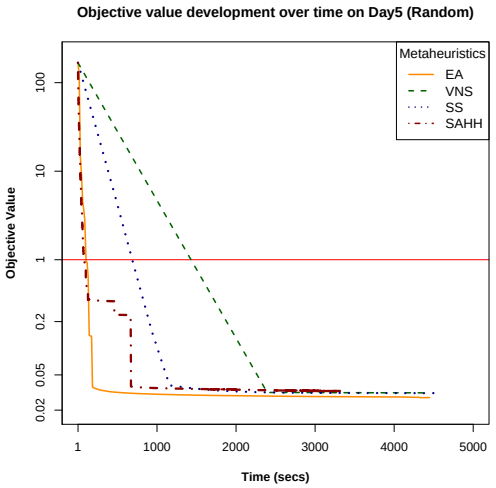
(a)



(b)

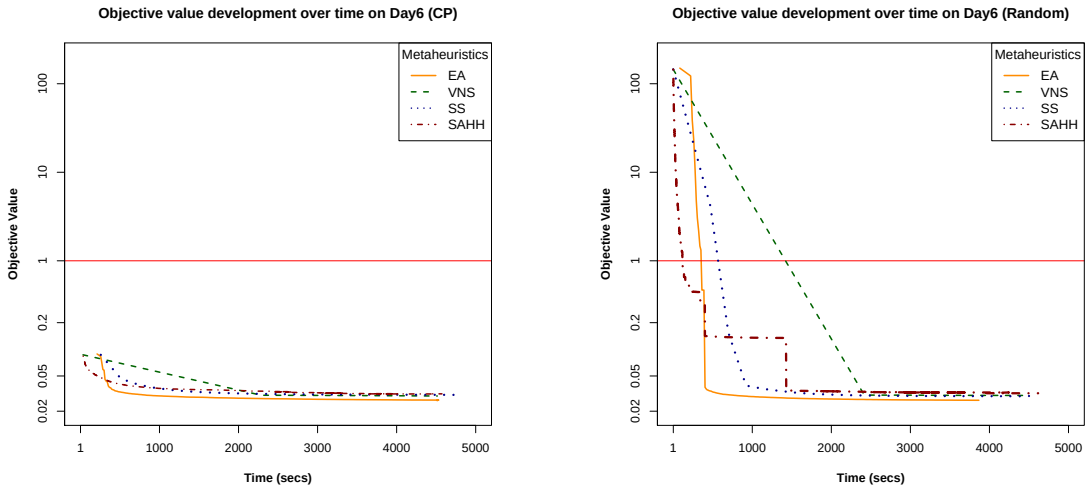


(c)

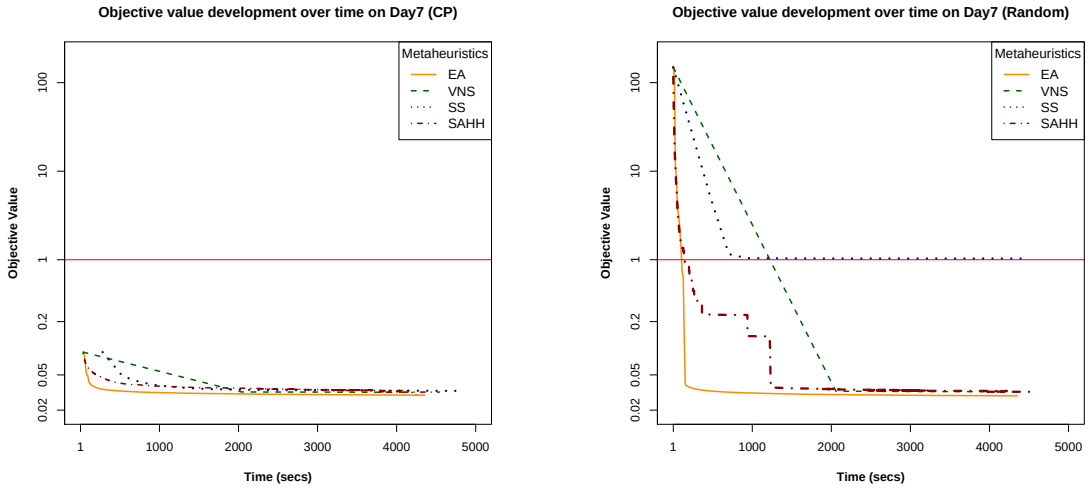


(d)

Figure 6.4: Changes in the objective value for day 4 with CP (a) and random initialization (b); and day 5 with CP (c) and random initialization (d).

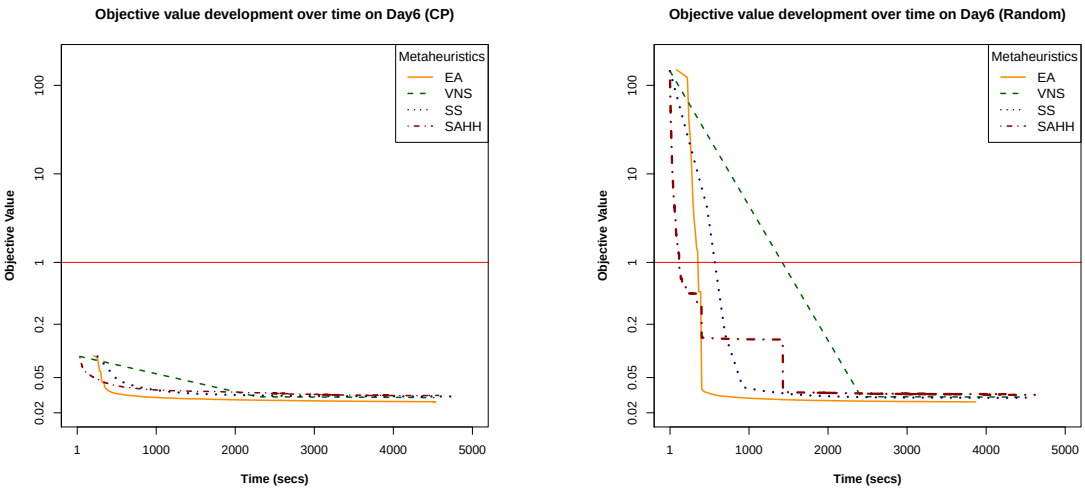


(a) (b)



(c) (d)

Figure 6.5: Changes in the objective value for day 6 with CP (a) and random initialization (b); and day 7 with CP (c) and random initialization (d).



(a) (b)

Figure 6.6: Changes in the objective value for day 8 with CP (a) and random initialization (b).

Table 6.6: Number of nurses finally scheduled for each day.

	day 1	day 2	day 3	day 4	day 5	day 6	day 7	day 8
VNS	200	205	212	208	204	202	216	211
MA	175	178	175	178	178	174	183	180
SAHH	202	206	213	202	203	203	216	204
SS	201	206	213	212	210	212	216	208

CHAPTER 7

Conclusion

In this thesis three metaheuristics were implemented tackling a multimodal home healthcare scheduling problem based on the working procedures of a Viennese home healthcare company. A detailed description of the problem and solution as well as the solving architecture for the single-day instance was provided. The algorithms were selected for their performance on related problems, i.e., the vehicle routing problem with time windows and the nurse rostering problem, and adapted to the MHS.

First a simulated annealing hyper heuristic was adapted by using the already existing neighbourhoods as low-level heuristics and experimenting with the parameters and temperature values. Also a constrained variant for the random improvement strategy was proposed but discarded after preliminary results indicated no improvement compared to the unconstrained variant. The results show the good performance in comparison with the existing VND and VNS approaches.

As a second approach a memetic algorithm was implemented which was redesigned early to a steady-state approach using a tailored recombination operator. Another improvement was obtained using a cyclic search of the neighbourhoods as embedded local search procedure. As the results of the final comparison showed, this approach performs best compared to the other proposed metaheursitics.

The last approach implemented was a scatter search. After first tests, the *construction by voting* approach was observed to not be suitable to tackle the problem, thus a path-relinking like algorithm was proposed which improved the performance considerably. However, even with this improvement and after experiments with different parameters and local search procedures, this search performed worst of all compared approaches when using a randomly initialized solution, as the search was not able to find valid solutions on each run of 4 out of the tested 8 instances.

Future work could consist of using these metaheuristics to solve multi-day solutions. This could be realised using the proposed algorithms in a repeating 2-phase approach where solutions are created per day and refined by adapting the objective function. Of all presented approaches, the memetic algorithm might perform very well considering its ability to find good results early in the search with a low number of nurses used. Furthermore the population maintained by

the MA could be used to refine a multi-day solution without the need to start a new search procedure.

Bibliography

[1] Uwe Aickelin and Kathryn A. Dowsland. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3:139–153, 2000.

[2] Uwe Aickelin and Kathryn A. Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761–778, April 2004.

[3] Ruibin Bai, Jacek Blazewicz, Edmund K. Burke, Graham Kendall, and Barry Mccollum. A simulated annealing hyper-heuristic methodology for flexible decision support. Technical report, School of Computer Science, University of Nottingham, England, 2006.

[4] Sachidanand V. Begur, David M. Miller, and Jerry R. Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35–48, 1997.

[5] Patrícia Belfiore and Hugo Tsugunobu Yoshida Yoshizaki. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil. *European Journal of Operational Research*, 199(3):750–758, 2009.

[6] Jean Berger and Mohamed Barkaoui. A hybrid genetic algorithm for the capacitated vehicle routing problem. In *GECCO*, pages 646–656, 2003.

[7] Jean Berger, Martin Salois, and Regent Begin. A hybrid genetic algorithm for the vehicle routing problem with time windows. In *Canadian Conference on AI*, pages 114–127, 1998.

[8] Stefan Bertels and Torsten Fahle. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33:2866–2890, October 2006.

[9] Joe L. Blanton, Jr. and Roger L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 452–459, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[10] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[11] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[12] Olli Bräysy. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, December 2003.

[13] Edmund K. Burke, Peter I. Cowling, Patrick De Causmaecker, and Greet Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001.

[14] Edmund K. Burke, Timothy Curtois, Rong Qu, and Greet Vanden Berghe. A scatter search approach to the nurse rostering problem. *Journal of the Operational Research Society*, 61(11):1667 – 1679, 2010.

[15] Edmund K. Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, November 2004.

[16] Edmund K. Burke, Patrick De Causmaecker, Sanja Petrovic, and Greet Vanden Berghe. Variable neighborhood search for nurse rostering problems. In Mauricio G. C. Resende, Jorge Pinho de Sousa, and Ana Viana, editors, *Metaheuristics: Computer Decision-Making*, chapter 7, pages 153–172. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[17] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R. Woodward. A classification of hyper-heuristics approaches. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 15, pages 449–468. Springer, 2nd edition, 2010.

[18] Edmund K. Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, chapter 16, pages 457–474. Springer, 2003.

[19] Edmund K. Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.

[20] Patrick Causmaecker and Greet Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14(1):3–16, February 2011.

[21] B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems – a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, December 2003.

[22] Eddie Cheng and Jennifer Lynn Rich. A home health care routing and scheduling problem. Technical Report CAAM TR98-049, Rice University, 1998.

[23] Wen-Chyuan Chiang and Robert A. Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9(4):417–430, 1997.

[24] Jean-François Cordeau, Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, and François Soumis. VRP with time windows, 2002.

[25] Peter I. Cowling, Graham Kendall, and Eric Soubeiga. Hyperheuristics: A robust optimisation method applied to nurse scheduling. In *PPSN*, pages 851–860, 2002.

[26] Jacques Desrosiers, Yvan Dumas, Marius. M. Solomon, and François Soumis. Time Constrained Routing and Scheduling. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 8 of *Network Routing*, chapter 2. Elsevier, 1995.

[27] Kathryn A. Dowsland. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research*, 106(2-3):393–407, 1998.

[28] Patrik Eveborn, Patrik Flisberg, and Mikael Rrönnqvist. Laps care - an operational system for staff planning. *European Journal of Operational Research*, 171:962–976, 2006.

[29] Luca Di Gaspero and Andrea Schaerf. Multi-neighbourhood local search with application to course timetabling. In *PATAT*, pages 262–275, 2002.

[30] Fred Glover. A template for scatter search and path relinking. In *Artificial Evolution*, pages 1–51, 1997.

[31] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684, 2000.

[32] Crina Grosan, Ajith Abraham, and Hisao Ishibuchi. *Hybrid Evolutionary Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[33] Pierre Hansen and Nenad Mladenović. Variable neighborhood search. In Fred W. Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publisher, New York, 2003.

[34] Wee-Kit Ho, Juay Chin Ang, and Andrew Lim. A hybrid search algorithm for the vehicle routing problem with time windows. *International Journal on Artificial Intelligence Tools*, 10(3):431–449, 2001.

[35] John H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975.

[36] Soonchul Jung and Byung Ro Moon. A hybrid genetic algorithm for the vehicle routing problem with time windows. In *GECCO*, pages 1309–1316, 2002.

[37] Manuel Laguna and Rafael Martí. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2):235–255, October 2005.

[38] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358, 1992.

[39] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34:111–124, 1986. 10.1007/BF01582166.

[40] Rafael Martí, Manuel Laguna, and Fred Glover. Principles of scatter search. *European Journal of Operational Research*, 169(2):359–372, 2006.

[41] Pablo Moscato. *Memetic algorithms: a short introduction*, pages 219–234. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[42] Alexander G. Nikolaev and Sheldon H. Jacobson. Simulated annealing. In Michel Gendreau, Jean-Yves Potvin, and Frederick S. Hillier, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 1–39. Springer, 2010.

[43] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[44] Jean-Yves Potvin and Samy Bengio. The vehicle routing problem with time windows part ii: Genetic search. *INFORMS Journal on Computing*, 8(2):165–172, 1996.

[45] Matthias Prandtstetter, Günther R. Raidl, and Thomas Misar. A hybrid algorithm for computing tours in a spare parts warehouse. In Carlos Cotta and Peter Cowling, editors, *Evolutionary Computation in Combinatorial Optimization - EvoCOP 2009*, volume 5482 of *LNCS*, pages 25–36. Springer, 2009.

[46] Matias Sevel Rasmussen, Tor Justesen, Anders Dohn, and Jesper Larsen. The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. Technical Report 11-2010, DTU Management Engineering, May 2010.

[47] Andrea Rendl, Matthias Prandtstetter, Gerhard Hiermann, Jakob Puchinger, and Günther Raidl. Hybrid heuristics for multimodal homecare scheduling. In Nicolas Beldiceanu, Narendra Jussien, and Éric Pinson, editors, *9th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'12)*, volume 7298 of *Lectures Notes in Computer Science*, pages 339–355, Nantes, France, June 2012. Springer Verlag.

[48] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[49] Louis-Martin Rousseau, Michel Gendreau, and Gilles Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8(1):43–58, 2002.

[50] Robert A. Russell and Wen-Chyuan Chiang. Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, 169(2):606–622, 2006.

[51] SINTEF. Vrptw benchmark instances. `http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/`, 2011. [Online; accessed 25-Apr-2012].

[52] Sam R. Thangiah, Kendall E. Nygard, and Paul L. Juell. Gideon: a genetic algorithm system for vehicle routing with time windows. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, volume i, pages 322 –328, feb 1991.

[53] Werner Toplak, Hannes Koller, Melitta Dragaschnig, Dietmar Bauer, and Johannes Asamer. Novel road classifications for large scale traffic networks. In *Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems*, pages 1264–1270.