Noname manuscript No.
(will be inserted by the editor)

Metaheuristics for Solving a Multimodal Home-Healthcare Scheduling Problem

Gerhard Hiermann · Matthias Prandtstetter · Andrea Rendl · Jakob Puchinger · Günther R. Raidl

the date of receipt and acceptance should be inserted later

Abstract We present a general framework for solving a real-world multimodal home-healthcare scheduling (MHS) problem from a major Austrian home-healthcare provider. The goal of MHS is to assign home-care staff to customers and determine efficient multimodal tours while considering staff and customer satisfaction. Our approach is designed to be as problem-independent as possible, such that the resulting methods can be easily adapted to MHS setups of other home-healthcare providers.

We chose a two-stage approach: in the first stage, we generate initial solutions either via constraint programming techniques or by a random procedure. During the second stage, the initial solutions are (iteratively) improved by applying one of four metaheuristics: variable neighborhood search, a memetic algorithm, scatter search and a simulated annealing hyper-heuristic. An extensive computational comparison shows that the approach is capable of solving real-world instances in reasonable time and produces valid solutions within only a few seconds.

This work is partially funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) within the strategic programme I2VSplus under grant 826153 (CareLog).

M. Prandtstetter, A. Rendl, J. Puchinger
AIT Austrian Institute of Technology GmbH
Mobility Department, Dynamic Transportation Systems
Giefinggasse 2, 1210 Vienna, Austria
E-mail: {matthias.prandtstetter | andrea.rendl | jakob.puchinger}@ait.ac.at

G. Hiermann, G. R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/186, 1040 Vienna, Austria
E-mail: raidl@ads.tuwien.ac.at

1 Introduction

We are facing an increasingly aging population which results in a rising demand for healthcare. Home-healthcare is a particularly growing industry, where elderly are nursed at home instead of at retirement homes, which is preferred by most clients. However, planning home-healthcare schedules is difficult, since nurses do not operate stationary but need to travel from patient to patient. Furthermore, many side constraints, such as patients' preferences, visiting time windows or travel times depending on the chosen mode of transportation, have to be taken into account. Thus, many home-healthcare providers struggle to find economic schedules that meet the increasing demand for home-healthcare and keep customers as well as employees satisfied.

So far, shifts and tours are planned mostly manually. However, manual planning is problematic for several reasons: first, the quality of a schedule strongly depends on the expertise of the planner. Second, large problem settings are often divided into independent subproblems that are solved separately, which leads to suboptimal schedules. Third, generating schedules is a very time-consuming task that consumes valuable manpower. Therefore, *automatically* solving large-scale home-healthcare planning problems in reasonable time is particularly important.

The goal of *multimodal home-healthcare scheduling* (MHS) is to determine and assign sequences of nursing tasks (at patient homes) to home-care staff ("nurses") respecting various side constraints, like (preferred) time-windows and employer, nurse, and patient satisfaction as well as the modality of routes: nurses use different modes of transportation (cars or public transportation), which inherently leads to travel times dependent on the chosen modality.

The MHS problem is a combination of the *vehicle routing problem with time windows* (VRPTW) [5,6] and the *nurse rostering problem* (NRP) [8]. Both are NP-hard problems and generally difficult to solve in practice; exact algorithms are only useful for very small instances. Since we aim at solving large-scale real-world instances of the MHS, we propose and compare different (hybrid) metaheuristic approaches: variable neighborhood search, a memetic algorithm, scatter search, and a simulated annealing hyper-heuristic. An important benefit of our approach is flexibility: our framework can easily be extended or adapted to various changes in the problem definition, in particular to new constraints.

This article is structured as follows: we start with a thorough discussion of related work in Section 2 and continue with a formal problem description (including a discussion of the dataset) in Section 3. In Section 4, our general solving approach, as well as methods for initial solution generation and simplification are presented. Sections 5 to 8 describe our four different metaheuristic approaches in detail. Computational results on large-scale real-world instances are presented and analyzed in Section 9. Finally, we conclude our work and give an outlook on future work.

2 Related Work

Research in the field of home-healthcare problems started around 1997/98. Begur et al. [3] describe a *decision support system* for tour and roster planning in a home-healthcare company in the USA using a model which does not consider time windows. Cheng and Rich [10] present a mixed integer programming formulation as well as a heuristic approach to tackle a home-healthcare problem. The 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 four nurses and ten 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 optimal solutions have been found for 17 out of the 40 test instances in less than a second compared to the exact approach with runtimes between 3 to 20 minutes.

Bertels and Fahle [4] use a combination of *linear programming* (LP), *constraint programming* (CP) and *simulated annealing* as well as *tabu search* based metaheuristics. They consider problem instances for a single day with 20 to 50 nurses and 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 that are then improved using metaheuristics and a solution pool. We extend these approaches by also including additional soft constraints and multimodality.

Eveborn et al. [12] 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 matching by local improvements and splitting until a predefined termination condition is reached. The problem definition contains similar constraints as in our work and they also consider different travel times based on the mode of transportation assigned to the nurse. However, the examined instances are rather small (up to 21 nurses and 123 jobs). Based on their results, Eveborn et al. reported that the travel time savings of their approach are about 20% compared to solutions made by a human counterpart. More recent results from their real-world project LAPS CARE can be found in [13]. In this work the benefits of their planning tool when used in practice are presented. Based on data from different Swedish home healthcare providers, the overall efficiency could be improved by 6–12% as well as the service quality.

Rasmussen et al. [23] also formulate the problem as a set partitioning problem and describe a branch-and-price approach. Their model incorporates connected visits (i.e. restrictions on the order in 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.

In the context of Austrian home-healthcare, Trautsamwieser et al. [30] provide a variable neighborhood search approach to solve real-world instances for one small urban (140 jobs, 13 nurses) and several rural regions (up to 512 jobs, 75 nurses). The investigated problem statement is similar to ours, but only unimodal transportation, i.e. just one mode of transportation equal to all nurses, is considered. However, their model covers additional constraints for breaks during shifts. The approach is able to find optimal solutions for small, randomly generated instances (20 jobs, 4 nurses). Furthermore, a comparison with an actual assignment used for a real-world instance is presented and potential savings of 45% in travel times are shown. Trautsamwieser et al. [29] also tackled the home-healthcare problem in case of flood disasters and their impact on the solution finding process using the approach mentioned before. Further research was done by Rest et al. [26], analyzing the potential influence of natural disasters on home-healthcare services and how they can be considered in planning algorithms.

Parallel to our work Matta et al. [18] analyzed the main processes and decisions of home-healthcare problems from an operations management perspective. They also present an *Integrated Definition for Function Modeling* (IDEF) activity-based model showing the internal and external influences on the general decision making process. Furthermore, they propose a hierarchical framework where major decisions from the operational (assignments, routing) to the strategic level (one to five years in the future) are analyzed based on the data obtained by nine home-healthcare service providers. Another very recent contribution was made by Rest and Hirsch [25], where a tabu search approach for solving a multimodal version of the problem described in [30] is presented. A major difference between our work is their use of time-dependent travel times whereas in our work we use stochastic travel time estimations.

First results of our work have been presented in Rendl et al. [24], focusing on the hybridization with a constraint programming initialization step. Furthermore, an investigation of the influence of accurate travel times on the solution process has been presented in Prandtstetter et al. [22]. The major contributions of this article are a detailed formulation of the problem including a formal definition, and an in-depth description of the solution approaches with justifications of design choices and parameters. Additionally, this work presents an extended evaluation of the results with more information on the structure and the applicability of the obtained solutions.

3 The Multimodal Home-Healthcare Scheduling Problem

The MHS problem deals with finding an assignment of nurses to patient requests ("jobs") and scheduling tours for visiting the patients while minimizing violations of side constraints and the total travel time. For the latter, the nurses' *mode of transportation* is of crucial importance, since in general, the mode choice has significant influence on expected (and actual) travel times [22].

3.1 Problem Definition

Jobs. We are given a set of customers $\mathcal{C} = \{1, \ldots, C\}$ who may book one or more jobs, summarized in the set of jobs $\mathcal{J} = \{1, \ldots, J\}$. The customer of a job $j \in \mathcal{J}$ is given by $cust(j) \in \mathcal{C}$. For each job j there is a start time window $[\mathrm{s}_j, \mathrm{e}_j]$ where s_j denotes the earliest and e_j the latest possible start time of j, a favored start time t_j^* where $\mathrm{t}_j^* \in [\mathrm{s}_j, \mathrm{e}_j]$ and a duration dur_j. We represent the time by a set of discrete points throughout a day, defined by $\mathcal{T} = \{0, \ldots, T\}$. The time points are given in τ-minute intervals, where $\tau = 5$ in our instances. Thus, $\mathrm{s}_j, \mathrm{e}_j, \mathrm{t}_j^* \in \mathcal{T}$ while $\mathrm{dur}_j \in \mathbb{N}$ where $\mathrm{dur}_j \cdot \tau$ gives the duration in minutes.

Nurses. All (available) nurses $\mathcal{N} = \{1, \ldots, N\}$ can only work within specific time windows, given in the set of time windows $\mathcal{W}_n$ for nurse n. Each time window $tw \in \mathcal{W}_n$ is a set of consecutive points of time. The global minimal and maximal numbers of working hours per day, are denoted $\mathrm{h}_{\min}$ and $\mathrm{h}_{\max}$.

Qualifications. The ordered set of qualifications $\mathcal{Q} = \{csw, vn, hn, ahn, mn\}$ represents the different kinds of nurses with respect to their qualifications: community service worker (*csw*), visiting nurse (*vn*), home-care nurse (*hn*), advanced home-care nurse (*ahn*) and medical nurse (*mn*). We write $\mathrm{q}_1 < \mathrm{q}_2$, with $\mathrm{q}_1, \mathrm{q}_2 \in \mathcal{Q}$, if qualification q_1 is weaker than qualification q_2 (and is thus contained in it). Note that $csw < vn < hn < ahn < mn$. Each job j requires a qualification q_j, and nurse n with qualification q_n may only perform j, if $\mathrm{q}_j \leq \mathrm{q}_n$, with $\mathrm{q}_j, \mathrm{q}_n \in \mathcal{Q}$.

Refusal Reasons. Patients and nurses may refuse one another; for instance, a nurse may refuse to attend a patient who owns a dog if she has a dog allergy, or a female patient might refuse a male nurse. In such cases, the nurse may not be assigned to the respective job in the roster. There are various reasons for refusing a nurse/customer, summarized as set $\mathcal{F} = \{dog, cat, smoker, male, female\}$. All refusal reasons stated by customer $c \in \mathcal{C}$ are given in set $\mathcal{A}_c \subseteq \mathcal{F}$. For instance, if customer c has refusal reason set $\mathcal{A}_c = \{cat, male\}$, then the customer has a cat and does not want to be treated by a male nurse. On the other hand, nurses specify their refusal reasons $\mathcal{A}_n \subseteq \mathcal{F}$, with $n \in \mathcal{N}$. For example, $\mathcal{A}_n = \{cat, male\}$ implies that nurse n has a cat allergy and is male. Therefore, whenever $\mathcal{A}_c \cap \mathcal{A}_n \neq \emptyset$, nurse n may not be assigned to customer c.

Multimodality. Each nurse $n \in \mathcal{N}$ states her preferred mode of transportation $\mathrm{p}_n \in \{car, publicTransport^1\}$. For every customer c and nurse n, we are given wgs84 coordinates (current standard for global positioning using longitude and latitude coordinates) of their home locations by $loc(c)$ and $loc(n)$, respectively. The travel time from locations $loc(i)$ to $loc(i')$ using mode of transportation p is retrieved from the distance matrix $\mathbf{tt}^{\mathrm{p}}_{ii'} \in \mathcal{T}$ where $i, i' \in \mathcal{C} \cup \mathcal{N}$. For instance, an entry $\mathbf{tt}^{\mathrm{car}}_{ii'} = 4$ states that driving from $loc(i)$ to $loc(i')$ takes 4 time units, thus, $4 \cdot \tau = 20$ minutes with $\tau = 5$. Note that the actual travel times are rounded up to the next discrete time unit. Travel time estimates are based on data from the Viennese public transportation system, and a large set

[1] Note that public transportation also includes walking in our case.

of historical data from Viennese floating car data [28]. Note that these travel time estimations depend on the day and the mode of transportation, not on the actual time of departure.

Pre-Allocated Jobs. In addition to nursing jobs, we also consider *pre-allocated jobs* $\mathcal{J}^{pre}$ (e.g., team meetings, administrative work, etc.) that are assigned to a fixed location, a fixed nurse and a fixed time. In our instances, these jobs make up about 5% of all jobs. For each job $j \in \mathcal{J}^{pre}$ we are given the pre-allocated nurse by $alloc(j) \in \mathcal{N}$.

Solution. A solution $\sigma = (\mathcal{R}, \mathcal{S})$ to the MHS is a roster consisting of a set of tours $\mathcal{R} \subseteq 2^{\mathcal{J}}$ and a mapping of jobs to actual starting times $\mathcal{S} : \mathcal{J} \to \mathcal{T}$. Every nurse $n \in \mathcal{N}$ is associated with exactly one tour R^n, hence $|\mathcal{R}| = |\mathcal{N}|$. Note that each tour R^n starts and ends at the nurse's home location $loc(n)$. Given assignments in $\mathcal{R}$ and $\mathcal{S}$, the actual sequence of visits is obtained by sorting each job $j \in R^n$ based on its starting time assignment in $\mathcal{S}$. Therefore, $S(R_i^n) \leq \ S(R_j^n), \forall R_i^n, R_j^n \in \mathcal{J}, i \leq j$ where R_k^n denotes the k-th job in the tour of nurse n. If $R^n = \emptyset$ then nurse n is not scheduled for the current day.

Constraints. The problem constraints are split into hard and soft constraints, where hard constraints have to be fulfilled, and soft constraints may be violated, though their violation should be as small as possible. Note, that the declaration of constraints as either hard or soft was coordinated with our home-healthcare partner and represents the provided specifications. The hard constraints are:

– all jobs must be assigned to a nurse, i.e. must be part of a tour

$$\forall j \in \mathcal{J} : \exists n \in \mathcal{N} : j \in R^n \tag{1}$$

– each job must be assigned to exactly one nurse

$$\forall n, m \in \mathcal{N} : n \neq m \Rightarrow R^n \cap R^m = \emptyset \tag{2}$$

– nurses must be qualified to perform the assigned jobs

$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathrm{q}_j \leq \mathrm{q}_n \tag{3}$$

– starting times of two consecutive jobs (of one nurse) must be chosen such that traveling between them (using the appropriate mode of transportation) is possible

$$\forall n \in \mathcal{N} : \forall 1 \leq i \leq |R^n| - 1 : t(R_i^n) + \mathrm{dur}_{R_i^n} + \mathbf{tt}^{\mathrm{p}_n}_{R_i^n R_{i+1}^n} \leq t(R_{i+1}^n) \tag{4}$$

– each nurse may only work within the given working time windows

$$\forall n \in \mathcal{N} : \forall j \in R^n : \exists tw \in \mathcal{W}_n : \{t(j), \ldots, t(j) + \mathrm{dur}_j\} \subseteq tw \tag{5}$$

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

$$\forall j \in \mathcal{J}^{pre} : n = alloc(j) \Rightarrow j \in R^n \tag{6}$$

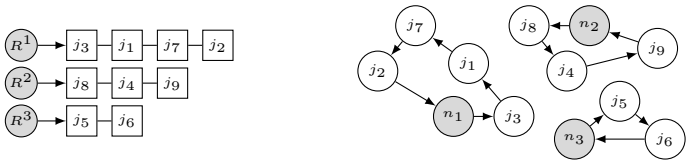
Soft constraints are:

– qualifications of nurses and (assigned) jobs should match

$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathrm{q}_j = \mathrm{q}_n \tag{7}$$

– the starting time of each job should lie within the specified time windows

$$\forall j \in \mathcal{J} : \mathrm{s}_j \leq t(j) \leq \mathrm{e}_j \tag{8}$$



(a) (b)

Fig. 1: Illustration of the representation and actual tours using an exemplary assignment.

– the actual starting time of each job should be the favored starting time stated by the customer

$$\forall j \in \mathcal{J} : t(j) = \mathrm{t}_j^* \tag{9}$$

– all refusal reasons should be considered in the roster

$$\forall n \in \mathcal{N} : \forall j \in R^n : \mathcal{A}_n \cap \mathcal{A}_{cust(j)} = \emptyset \tag{10}$$

Representation. We represent each tour R^n of a nurse $n \in \mathcal{N}$ by a list of jobs, where each job $j \in \mathcal{J}$ is assigned to at most one tour R^n and has a unique position i in the tour denoted R^n_i, with $i = 1, \ldots, |R^n|$; see Fig. 1.

3.2 Objective Function

Among all valid solutions, we want to find one that is most convenient for employer, nurses and patients at the same time. Our objective function $ob(\sigma)$ assigns a real number to a solution σ such that valid solutions evaluate to values between zero and one, and invalid solutions evaluate to values greater than one:

$$ob(\sigma) = \sum_{i=1}^{6} v_i + \sum_{i=7}^{10} v_i \cdot \gamma_i \cdot \phi_i + \sum_{i=11}^{13} v_i \cdot \gamma_i \cdot \phi_i \tag{11}$$

It includes 13 influencing terms that we divide into three parts: the first six are for *hard constraint violations* where the violation terms $v_1, \ldots, v_6$ denote the number of hard constraint violations in the solution. The second influencing terms account for *soft constraint violations* with violation terms $v_7, \ldots, v_{10}$ that are weighted with weights γ_i and normalized with factors ϕ_i. The third part represents *additional aspects* that influence the solution quality (such as working time and travel time) and is represented by the violation terms v_{11}, v_{12}, v_{13} and weighted and normalized with γ_i and ϕ_i, respectively. Weights γ_i are chosen such that $\sum_{i=7}^{13} \gamma_i = 1$ holds. The 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). Consequently, the worst valid solution always has a smaller objective value than the best invalid solution. A detailed description of all influencing terms, weights and normalization factors can be found in Table 1.

Table 1: Influencing terms in the objective function.

Term	γ_i	Meaning
v_1	—	contains the number of missing job assignments, i.e., the number of jobs not assigned to a nurse (1)
v_2	—	contains the number of assignments of more than one nurse to a job (2)
v_3	—	contains the number of invalid job-assignments due to insufficient qualification of the assigned nurse (3)
v_4	—	contains the number of travel time violations, i.e., the number of times a nurse cannot reach a customer before the service should start (4)
v_5	—	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 (5)
v_6	—	contains the number of violations concerning pre-allocated jobs that must not be shifted to other times or nurses (6)
v_7	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 $\lvert i - i' \rvert$ (7)
v_8	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 (8)
v_9	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 considered to be equally bad (9)
v_{10}	0.2	quantifies violations of refusal reasons stated by nurses and customers and is normalized over the number of jobs (10)
v_{11}	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_{12}	0.1	quantifies the working time of all nurses $n \in \mathcal{N}$ up to the daily maximal working time
v_{13}	0.1	quantifies the overall travel time of all nurses

3.3 Instance Data and Test Environment

Our project partner, Sozial Global AG, provided us with a set of real-world instances. We use this dataset throughout our work, i.e. for the intermediate evaluations as well as the final comparisons. In the dataset, all patients and nurses are located in an area of about around Vienna, where the jobs are roughly equally distributed. The exact number of jobs and available nurses per day is given in Table 2, however, note that not all available nurses are expected to work every day. Therefore, the job/nurse-ratio is rather low. In fact, in the final solutions (Sec. 9), only 35% of the nurses are actually scheduled. Unfortunately, we are not able to make the dataset publicly available due to data protection issues.

Table 2: Instance dimensions as used for the computational experiments.

	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
# jobs	711	700	679	682	708	699	717	679

However we extracted additional information about the features of the tested instances (e.g. time window settings, distributions of aspects) which can be found in the Appendix (`https://www.ads.tuwien.ac.at/w/Research/Problem_Instances`). Using these values we tested our approaches on additional, randomly generated instances. A description of the random instance generator and the results of our approaches can also be found in the Appendix.

All approaches are implemented within the same framework in Java (version 1.6); JaCoP [16] has been used as constraint programming solver. The preliminary tests were run for 10 minutes per run (except those for Scatter Search which were run 75 minutes), 10 runs per test, on a single core of an Intel(R) Core(TM) i7-2630QM CPU 2.0GHz with (up to) 4GB RAM assigned. All tests for final evaluation where run on a single core of a 8xDualcore AMD Opteron 870 2GHz with a maximum of 2GB RAM assigned.

4 Initial Solution Generation and Job Time Improvement

Our approach first generates an initial solution which is then further improved by a metaheuristic based algorithm. We consider two methods to generate an initial solution: a constraint programming (CP) based approach and a simple random construction procedure.

The CP based 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 VRPTW model from [27] and is described in [24]. The random construction algorithm generates a solution by traversing the list of available jobs in random order and assigning jobs to a nurse in a cyclic manner. It only ensures satisfaction of two constraints: each job is assigned to a nurse, cf. Eq. (1), and pre-assigned jobs are assigned to the proper nurse, cf. Eq. (6).

Job Start Time Improvement. After generating an initial solution (and during the subsequent optimization procedure), we internally improve the job start times in each tour by using a 2-step greedy algorithm illustrated in Fig. 2: In a first step, the earliest possible start time for each job R_i^n in the tour is computed, considering the travel time from the location of the previous job, $loc(R_{i-1}^n)$ (or the home location $loc(n)$, if $i = 1$), to the location of job R_i^n, as well as the starting time window of each job R_i^n and the nurses' working time windows TW_n. While the travel time constraints, cf. Eq. (4), are never violated, the time window constraints, cf. Eq. (8), are only satisfied where possible. This results in a schedule s where each job starts as early as possible.

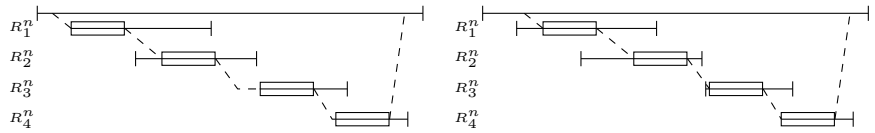


Fig. 2: An example for the 2-step greedy heuristic for determining job starting times for nurse $n \in \mathcal{N}$: first step (a), starting with the first job all jobs are set to the earliest possible start time considering the job durations as well as the travel times between the jobs (dashed lines);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

In a second step, all jobs $R_i^n \in R^n$ with $1 \leq i \leq |R^n| - 1$, are iteratively moved backwards as far as possible (respecting the time windows of the jobs) in decreasing order, starting with $R^n_{|R^n|-1}$. The resulting schedule s' is thus minimal with respect to working time (where the working time corresponds to the interval starting with the beginning of the first job and ending with the end of the last job). Since soft constraints, like meeting the desired start times at the customers, are not explicitly considered by this greedy algorithm, either schedule, s or s', can be better. Inherently, the better one is chosen as the actual start time assignment.

5 Variable Neighborhood Search

Variable neighborhood search (VNS) [15] and *variable neighborhood descent* (VND) [15] are two local search based metaheuristics that are somehow complementary to each other and therefore often used in combination, referred to as General VNS. While VND tries to locally improve given solutions by systematically exploring and switching between different neighborhood structures, VNS applies random moves in order to escape local optima. For the systematic exploration within VND we introduce three different move types that define the neighborhood structures:

shift job: shifts one job from one tour to another tour. The shifted job is placed at the best position in the new tour (according to the objective function).

reposition job: determines the best feasible position of one job within its current tour, without reordering the other jobs.

swap nurses: swaps two nurses with each other, such that the tour of the first nurse is assigned to the second nurse and vice versa.

As step function, we apply a first improvement strategy. The initial neighborhood order is shift jobs, swap nurses, finally followed by reposition job. We apply dynamic neighborhood reordering as described in [21], which frequently

leads to better results as revealed by preliminary tests. This reordering is done by first storing the improvement/examined-ratio of each neighborhood and then reordering them when an improved solution has been found.

As mentioned above, we embed VND as a local search phase in a General VNS scheme [15] to escape local optima. The idea of 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. In our case, we apply $k+1$ random shift jobs in the k-th consecutive VNS iteration without improvement as shaking using $k = 1, \ldots, 5$.

6 Simulated Annealing Hyper-Heuristic

Simulated annealing hyper-heuristic (SAHH) [1] incorporates two main concepts taken from *simulated annealing* (SA) [20]—a metaheuristic inspired by a heat treatment process in metallurgy—on the one hand and from the concept of a *hyper-heuristic* [9]—"a heuristic to select heuristics"—on the other hand. The main idea is to iteratively choose among a set of *low-level heuristics* (LLHs) to be applied within the next step. The probability for choosing the individual LLHs is adapted according to their performances during the last iterations, while the acceptance of a newly obtained solution is based on the classical SA acceptance criterion. Our approach is an adaption of the work by Bai et al. [1] to the MHS. An outline is given in Algorithm 1. In the following, we discuss the features of our SAHH in more detail.

Low-level Heuristics (LLHs). We use six simple LLHs that we obtain by combining each of the three move types described in Section 5 ("shift job", "reposition job" and "swap nurse") with two step functions: classic "first improvement" (returning the first neighbor that is better than the current solution) and "random-neighbor" (returning a random neighbor). If the first improvement strategy is started in a local optimum, it will not return a (new) solution. In this case we proceed as if the acceptance criterion was not reached (line 8 and 15 in Algorithm 1) to assure that the statistics are updated accordingly.

The probability of selecting a LLH is based on an accepted/tested-ratio of a previous learning period, or the newly-created/tested-ratio prior to the reheating phase. The selected LLH creates a new solution that is accepted if it is better than the current solution. Otherwise, it is accepted with probability $e^{-d/t}$, where d is the difference between the new and the current solutions' objective values and t is the current temperature.

Temperature Settings. When running the algorithm for K iterations, the temperature is updated every $nrep$ iterations by Lundy and Mees' nonlinear function [17]: $t \leftarrow t/(1 + \beta t)$ where $\beta = (t_{\text{start}} - t_{\text{end}}) \cdot nrep/(K \cdot t_{\text{start}} \cdot t_{\text{end}})$ and t_{start} and t_{end} are the temperatures at the beginning and the termination of the algorithm. Bai et al. suggest that around 10% of the solutions should be accepted at the beginning and only 0.5% at the end of the search. However, after some preliminary testing, the acceptance probability at the beginning

Algorithm 1: Simulated Annealing Hyper-Heuristic

```
   Input: Solution σ
   Output: Best solution found σ*
 1 i ← 1; σ* ← σ;
 2 t ← t_start; t_imp ← t_start;
 3 resetLLHStats();
 4 while i ≤ K do
 5 |   h ← selectLLH(H);
 6 |   σ' ← h(σ);
 7 |   h.tested ← h.tested + 1;
 8 |   if σ' = null then d ← +∞ else d ← ob(σ') − ob(σ)
 9 |   if d < 0 then
10 |   |   σ ← σ';
11 |   |   h.new ← h.new + 1; h.accept ← h.accept + 1; C ← C + 1;
12 |   |   if ob(σ') < ob(σ*) then σ* ← σ'
13 |   else
14 |   |   if e^{−d/t} < Random() then
15 |   |   |   σ ← σ';
16 |   |   |   h.accept ← h.accept + 1; C ← C + 1;
17 |   |   end
18 |   |   C ← C + 1;
19 |   end
20 |   if reheating = true then
21 |   |   t_imp ← t_imp/(1 − (beta() ∗ t_imp));
22 |   |   if t_imp > t_start then t_imp ← t_start t ← t_imp;
23 |   else if i % nrep = 0 then
24 |   |   t ← t/(1 + (beta() ∗ t));
25 |   end
26 |   if i % LP = 0 then
27 |   |   if C/LP < r_end then
28 |   |   |   reheating ← true; t_imp ← t_imp/(1 − (beta() ∗ t_imp));
29 |   |   |   t ← t_imp; σ ← σ*;
30 |   |   |   adjustLLH_New();
31 |   |   else adjustLLH_Accept() resetLLHStats();
32 |   end
33 |   i ← i + 1;
34 end
35 return σ*;
```

was changed to 5%, leading to better results. Thus the probabilities to accept a new solution are $e^{(-1.0/t_{\text{start}})} = 0.05$ and $e^{(-1.0/t_{\text{end}})} = 0.005$, respectively.

Preliminary Experiments. We performed preliminary experiments to derive the best parameter setting for the SAHH. More specifically, we tested different parameter values for d (the difference between the new and the current solutions' objective values) to obtain t_{start} and t_{end} in order to reflect the probabilities of selecting a worse solution. The experimental results using values of $0.5, 1.0$ and 2.0 for d did not clearly indicate a best setting, thus we have chosen a value of $d = 1.0$ to provide a balance between finding local optima and escaping them. The parameter setup as used in the final comparison (Sec. 9) is summarized in Table 3.

Table 3: Parameter settings for SAHH

SAHH Parameters	**Settings**
K (# iterations)	10000
acceptance prob. (start)	0.05
acceptance prob. (end)	0.005
learn period	$K/500$
nrep (#iter/temp)	6 (#neighborhoods)
min weight	0.1

7 Memetic Algorithm

Evolutionary algorithms (EAs) are population based search approaches that imitate evolutionary behavior by adapting concepts such as selection, recombination, and mutation to breed new and better solutions. Hybridizing EAs with local search procedures in form of *memetic algorithms* (MAs) [19] allows to overcome the limited exploitation capabilities of classical EAs. Burke et al. [2] presented promising results for the NRP using an MA, thus we expected a similarly good behavior and adapted it to our problem. In the following, we describe the essential components of our approach.

Initialization. The *random diverse constructor* remembers assignments from previously generated solutions in order to derive a new solution that is as diverse as possible. More specifically, for each job j, we select the nurse that has been least often assigned to j in already generated solutions and add j to the end of the nurse's tour. Ties are broken randomly. The initial solution from the CP-based or random approach is also part of the initial population.

Selection. We use binary tournament selection, where solutions are selected by repeatedly picking two solutions at random and including the better one into the set of selected solutions.

Recombination. Since each solution consists of an ordered list of tours (one tour for each nurse), we implement a special recombination operator: given two parent solutions (P_1, P_2), the offspring basically represents a copy of P_2 with a randomly chosen tour R^n being replaced by tour $R^n \in P_1$. For this purpose, the offspring is initially a copy of P_2 (Fig. 3a) where all elements being member in $R^n \in P_1$ are removed (Fig. 3b). Now, every remaining job in tour R^n in the offspring is greedily moved to another tour (Fig. 3c). Finally, the tour R^n from P_1 is copied to the offspring.

Mutation. To increase diversity, the offspring is mutated by randomly selecting a tour R^n 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.

Local Search. With 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. We apply two local search algorithms: variable neighborhood descent (VND) as described in

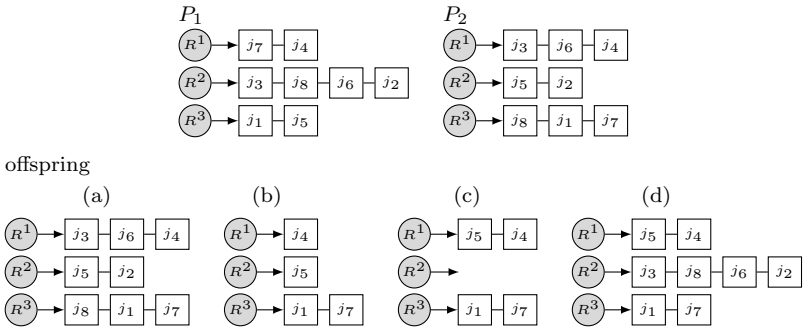


Fig. 3: An example for the MA's recombination operator where R^2 is selected: (a) the offspring is initialized to P_2, (b) all jobs of R^2 in P_1 are removed from the offspring, (c) all remaining jobs in R^2 in the offspring are moved, and (d) R^2 in the offspring is filled with the jobs of R^2 in P_1.

Section 5, and cyclic search of neighborhoods (CNS). CNS is similar to VND but always turns to the next neighborhood structure in a cyclic manner when either a local optimum in one neighborhood structure, or a time limit has been reached. A similar search technique was proposed by Di Gaspero et al. [11], the so-called *token-ring search*, for a timetabling problem.

Replacement. The population is replaced using a steady-state approach [31]: the offspring replaces the most similar solution that is worse than the offspring in the current population. The similarity of two solutions is determined by the number of equal job-to-nurse assignments in both solutions. If two solutions are equally similar, we remove one randomly.

Preliminary Experiments. In our preliminary experiments, we evaluated three different setups: first, a pure evolutionary algorithm (the MA without local search phase) followed by the MA with VND as local search procedure, and finally the MA with CNS. Table 4 summarizes the results.

First, we observe that the hybrid setups are more promising than the pure EA. Second, the embedding of CNS seems to be more reasonable than the application of VND. This results from the relatively tight time limit for VND/CNS: since the VND examines the first neighborhood until no further improvement can be found, it often does not reach subsequent neighborhoods. CNS, in contrast, achieves improvements for all defined neighborhood structures, and therefore better exploits the existing improvement potential.

Furthermore, the tests revealed that the population size of 100 is promising. For the recombination operator, we observed that it is capable of improving the solution since we move jobs to the best possible positions. We also tested a simple random shift-job-move as a second mutation operator, where a job is randomly selected and shifted to another nurse. Preliminary results showed that both operators perform almost equally well, but more in favor of the previously described mutation operator (clearing a tour by moving the jobs to other nurses). Table 5 shows the final setup of the proposed MA.

Table 4: Objective values obtained with no local search, 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

Table 5: Final parameter settings for MA.

MA Parameter	**Settings**
population size	100
selection size	2
operator probabilities	
- recombination	1.0
- mutation	1.0
- improvement	0.01
termination	time limit or 2000 iterations without an improvement

8 Scatter Search

The second population based approach considered here is Scatter Search [14]. It uses a small and diverse population of solutions, the *reference set* (*RefSet*), and creates subsets of small size from this pool using a *subset generation* method. These subsets are then combined using a *solution combination* method to create new solutions that are subsequently improved using VND with a time limit as described for the MA. Based on the *reference set update method*, these new solutions replace some of the current ones in the reference set. The algorithm terminates as soon as no new solutions have been included into the reference set after an update. In the following we discuss details about the subset generation, combination and improvement and update method.

Initialization. The initial population is created as it is in the MA.

Subset Generation. To generate subsets $\mathcal{U}$ out of the reference set, a classical subset generation approach is used (cf. [14]): Let the reference set $RefSet = \{\sigma_1, \sigma_2, \ldots, \sigma_{|RefSet|}\}$ be a sorted set where $ob(\sigma_1) \leq ob(\sigma_2) \leq \ldots \leq ob(\sigma_{|RefSet|})$ holds. We first create subsets of all pairs of solutions:

$$\mathcal{U}^1 = \bigcup(\sigma_i, \sigma_j), \quad \text{where } i = 1, \ldots, |RefSet| - 1, j > i. \tag{12}$$

Next, we create subsets consisting of 2-sets and the best solution not in the set (duplicate subsets are removed):

$$\begin{aligned} \mathcal{U}^2 = \bigcup(\sigma_i, \sigma_j, \sigma_k), \quad & \text{where } (\sigma_i, \sigma_j) \in \mathcal{U}^1, \\ & k = min\{k | i \neq k \neq j, \sigma_k \in RefSet\}. \end{aligned} \tag{13}$$

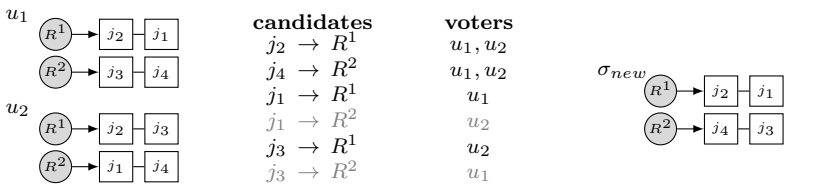


Fig. 4: Example for the construction by voting subset combination method: Using a list of candidates (middle) of two solutions (left) to create a new solution (right). Unused candidates are gray.

At last, we create subsets of size $k = 4$ to the size of the *RefSet* (in our case 5) that contain the k best solutions, which leads to the final set of subsets $\mathcal{U}$

$$\mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2 \cup (\sigma_1, \sigma_2, \sigma_3, \sigma_4) \cup (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5). \tag{14}$$

To avoid using the same subsets in successive iterations, we have to assure that each created subset contains at least one new solution.

Subset Combination 1. For the combination of solutions for each subset, we implement a "construction by voting" combination algorithm as suggested by Glover et al. [14] and described by Burke et al. [7]. This approach creates a new solution using *candidates* and *voters*. A candidate is an assignment of a job j to a tour R^n whereas voters are the solutions of a subset. Fig. 4 shows an example with two solutions. First, all possible candidates are created and the solutions from the subset with the same job-to-tour assignment are linked to the corresponding candidate. Second, the list of candidates are sorted in descending order of the number of solutions (voters) linked to it. Then, the construction starts at the beginning of the list.

If an assignment of a job j has two candidates with the same number of voters, a tie breaking mechanism is used to choose a candidate. First, the number of successful votes are counted (i.e., how many times a candidate of a solution was used to construct the new solution), and the candidate with the number of least used voters is chosen. Since this number may 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 search, the candidate is removed.

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 [14]. Path relinking generates solutions using the neighborhood 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 to the guiding solution. Each entry in π is a neighbor of the previous solution, starting with a neighbor of the initiating solution and ending with the last solution before reaching the guiding solution.

Algorithm 2: Subset Combination: Path-Relinking

Input: sorted set of solutions $U = \{u_1, \ldots, u_l\}$ of size l, where $ob(u_1) \geq ob(u_2) \geq \ldots \geq ob(u_l)$
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 $ob(N'_q) < ob(\sigma')$ **then** $\sigma' \leftarrow N'_q$
    end
    $\sigma \leftarrow \sigma'$;
  end
end
return σ;

In our approach, we work with subsets of solutions $U \in \mathcal{U}$ that can contain more than two solutions, thus our path-relinking algorithm was adapted accordingly. The outline of the algorithm is given in Alg. 2. To create a solution σ_{new} using a subset $U \in \mathcal{U}$ of size l we move iteratively from the worst solution u_1 to the best u_l. Then $\sigma_{new} = u'_l$, where

$$u'_i = u'_{i-1} \rightarrow_{PR} u_i, i = 2, ..., l, u'_1 = u_1 \tag{15}$$

and the path-relinking operator $A \rightarrow_{PR} B$ performs $difference(A, B)/l$ moves from A to B (the function *difference* returns the number of different job to tour assignments). To determine which moves are performed first, we calculate the change in the objective per move and perform the move with the best change (that can also be a worsening move). In order to prevent high runtimes due to a possible large set of moves to be tested, we restrict the number of calculated moves with parameter $s_{\max}$.

Reference Set Update. For the update method we used a straight forward strategy: Each solution created by the subset combination method will replace the worst solution in the current reference set, but only if it is not already in the set. Thus we destroy the diversity of the reference set, as proposed by Burke et al [7], in order to have more time to search in better regions (based on the objective value).

Preliminary Experiments. In our preliminary experiments, we evaluate our two subset combination methods: construction by voting (referred to SS1 in the following) and the modified path-relinking procedure (SS2). For SS1, we set the local search time limit to 30 seconds and restrict the reference set size to 4, so that fewer subsets and fewer solutions are created. SS2 uses 10 seconds and a reference set size of 5. Both parameter settings were obtained after initial experiments.

Table 6: Objective values obtained by 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 ±0.00291	0.04030 ±0.00383	0.03865 ±0.00397	0.08639 ±0.0087
SS2	0.03220 ±0.00062	0.03202 ±0.00091	0.03057 ±0.00120	0.03300 ±0.00104

Table 7: 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

The comparison in Table 6 reveals that SS1 is not suitable for our problem: SS1 creates inferior solutions, and thus decreases the performance of the local search procedure that requires more time to get into better regions first. To overcome this problem, the runtime per local search call would need to be increased, which would lead to fewer iterations for the overall procedure. Therefore, for our final setup we choose SS2; the final parameter settings are given in Table 7.

9 Computational Results

In our computational evaluation, we compare the proposed metaheuristic approaches with respect to solution quality and runtime performance. Furthermore, we analyze the impact of the initial solution on the optimization procedures. In the following, we first specify the experimental setup (our dataset and basic setup is described in Sec. 3.3) and continue discussing each metaheuristic, starting with the most successful one.

For each instance, metaheuristic and initial solution generation method, we performed 10 runs with a time limit of 75 minutes per run. Initialization was either done via CP techniques or via the random initialization method. While the former always results in *valid* solutions, the latter sacrifices validity in favor of speed (runtimes reduce from about 20 second to far below one second). We use the same initial solution for all of the ten runs and metaheuristic settings.

Table 8 shows the final objective values depending on the construction heuristics, averaged over 10 runs. Values labeled with ± are standard deviations. Note that instance day 3 includes a data-error (the corresponding nurse for a pre-assigned job is not available) resulting in invalid solutions. How-

Table 8: Final objective values for each metaheuristic, averaged over 10 runs. Note that instance 3 is unsolvable due to a data error concerning a pre-assigned job where the respective nurse is marked as unavailable.

	constr.	init.	VNS	MA	SAHH	SS
day 1	cp	0.0885	0.02833 ±0.00108	**0.02521 ±0.00077**	0.03098 ±0.00457	0.08564 ±0.00082
	rand.	162.1513	0.02870 ±1.70076	**0.02437** ±0.00100	0.03102 ±0.00350	4.04139 **±0.00086**
day 2	cp	0.0894	0.02945 ±0.00091	**0.02570** ±0.00097	0.03208 ±0.00287	0.08626 **±0.00000**
	rand.	152.1521	0.02917 ±0.00137	**0.02577 ±0.00067**	0.03137 ±0.00215	3.03994 ±0.00082
day 3	cp	1.0860	1.02743 ±0.90110	**1.02398 ±0.00096**	1.02991 ±0.00178	2.03854 ±0.00176
	rand.	148.1494	1.02740 ±0.00087	**1.02380** ±0.00070	1.02915 ±0.00141	2.03765 **±0.00030**
day 4	cp	0.0896	0.02936 **±0.00108**	**0.02612** ±0.00109	0.03211 ±0.00399	0.03947 ±0.00223
	rand.	155.1532	0.02958 ±0.00145	**0.02601 ±0.00052**	0.03287 ±0.00161	0.03907 ±0.00075
day 5	cp	0.0885	0.02919 ±0.00154	**0.02537 ±0.00110**	0.03193 ±0.00257	0.04190 ±0.00122
	rand.	168.1506	0.02957 ±0.00083	**0.02579 ±0.00046**	0.03206 ±0.00244	0.04033 ±0.00094
day 6	cp	0.0867	0.02798 ±0.00105	**0.02483 ±0.00089**	0.03096 ±0.00264	0.04107 ±0.00104
	rand.	145.1523	0.02811 ±0.00122	**0.02422 ±0.00085**	0.03246 ±0.00346	0.03944 ±0.00147
day 7	cp	0.0872	0.02694 ±0.00230	**0.02414** ±0.00087	0.02947 ±0.00154	0.08357 **±0.00030**
	rand.	158.1446	0.02737 ±0.00168	**0.02377 ±0.00049**	0.02812 ±0.00235	4.03999 ±0.00212
day 8	cp	0.0843	0.02773 ±0.00115	**0.02500 ±0.00062**	0.03129 ±0.00315	0.04005 ±0.00126
	rand.	165.1476	0.02851 **±0.00085**	**0.02482** ±0.00102	0.02993 ±0.90067	0.03811 ±0.00138

ever, the CP does not fail for this instance since it assumes that nurses of pre-assigned jobs are always available (which is generally true).

Memetic Algorithm. The results in Table 8 show that the MA finds the best results, independent of the initial solution generation approach. When using random initialization, the objective values are slightly better, probably since the random solution generation is much faster than the CP-based approach and thus leaves more time for the MA. We illustrate the development of the objective value and its components in Figure 5. The graphs depict that major objective function improvements are achieved when the number of nurses is adjusted. Interestingly, even an increase in the number of employed nurses can lead to a significant reduction in the objective value.

General VNS. The second best approach in terms of objective values is General VNS. First, we observe that VNS benefits from the CP construction method. Moreover, in some cases, VNS fails to find a valid solution for the

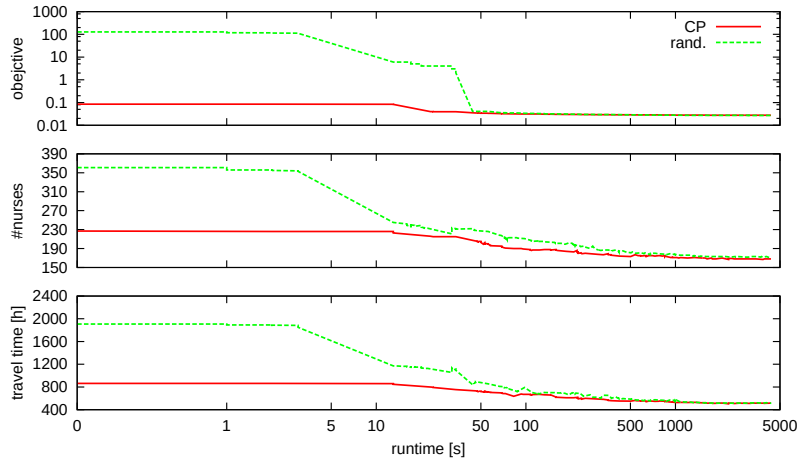


Fig. 5: Development of objective value, number of nurses, and total travel time over time obtained via the MA initialized either by the CP or the random procedure (one exemplary run of day 4).

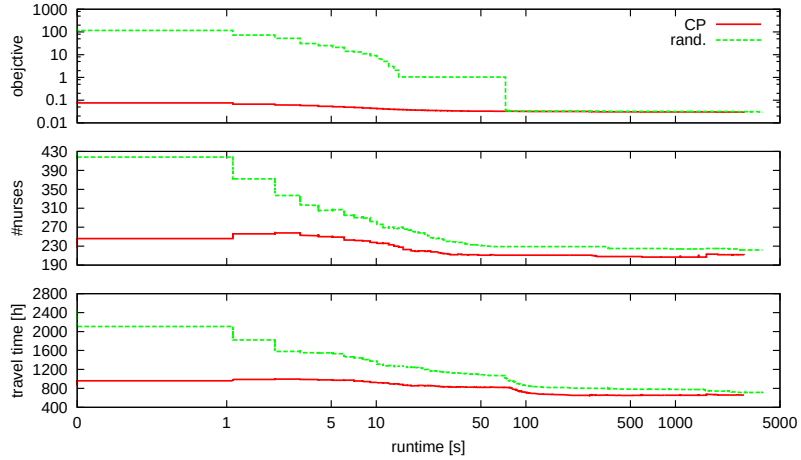


Fig. 6: Development of objective value, number of nurses, and total travel time over time obtained via the VNS initialized either by the CP or the random procedure (one exemplary run of day 4).

test runs with random initialization (cf. Fig. 6, where we show the objective value development for a representative sample run).

SAHH. The simulated annealing hyper-heuristic is the approach with the most permanent improvement behavior, as shown in Figure 7. The most interesting observation is that for the SAHH initialized with the CP, the improvements during the first phase are mainly caused by increasing the number of nurses.

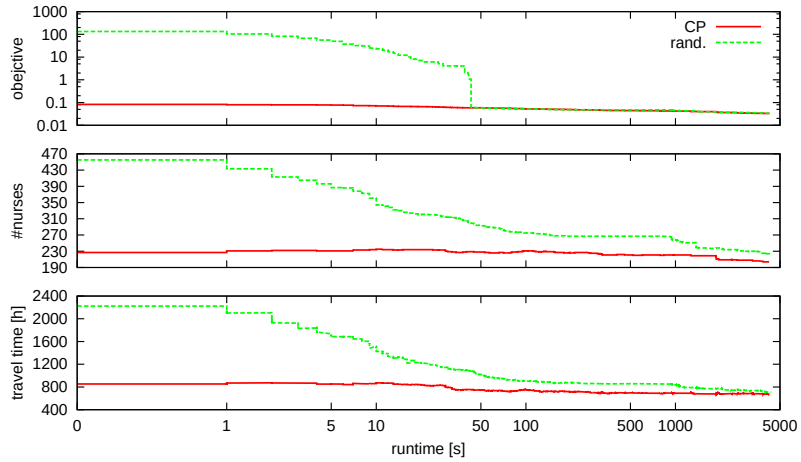


Fig. 7: Development of objective value, number of nurses, and total travel time over time obtained via the SAHH initialized either by the CP or the random procedure (one exemplary run of day 4).

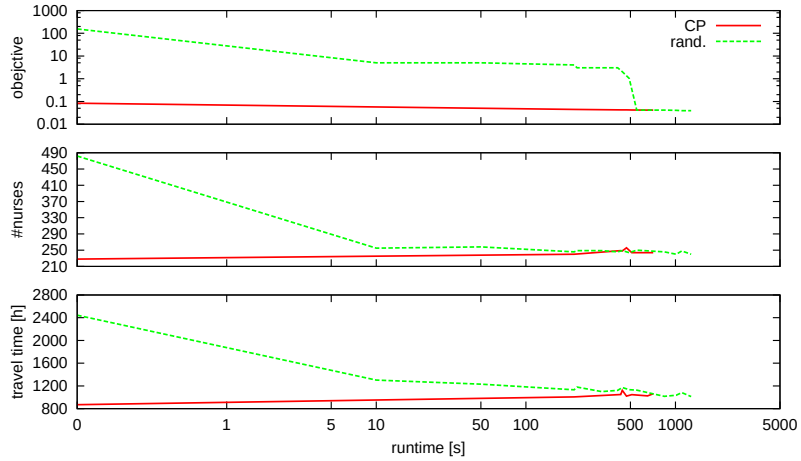


Fig. 8: Development of objective value, number of nurses, and total travel time over time obtained via the SS initialized either by the CP or the random procedure (one exemplary run of day 4).

Scatter Search. The scatter search approach benefits most from the usage of CP for constructing initial solutions—for random initialization this approach often does not reach a feasible solution. As the results show, this approach generates better solutions than the SAHH when started with an initial solution found by the CP. Moreover, for instances where the Scatter Search can find a valid solution (without hard constraint violations) it is in general better than a solution found by the SAHH. As Figure 8 illustrates, the main improvement

Table 9: Number of nurses finally scheduled for each day.

	day 1	day 2	day 3	day 4	day 5	day 6	day 7	day 8
available	509	491	504	482	496	518	505	495
VNS	218	216	217	212	217	217	214	214
MA	174	172	173	168	173	173	175	169
SAHH	217	216	215	213	212	213	213	210
SS	209	213	246	242	248	252	202	245

in the objective value is achieved rather early in the search with only small improvements later on.

General Observations. In summary, all methods produce substantial improvements within about 100 to 300 seconds, which is most satisfactory. The generated solutions share some similarities but are generally rather different from each other. For instance, we observe that all approaches reach solutions where the number of nurses is relatively low (see Tab. 9). More specifically, the MA produces solutions with the smallest number of nurses, while the VNS produces solutions with the highest number of nurses. It is quite interesting that the number of scheduled nurses may increase early in the search but decrease later on, cf. Fig. 7 or 8. This implies that good solutions incorporate only a small number of nurses, although the number of nurses should not be explicitly minimized during optimization, as otherwise the search process could get stuck in a local optimum.

To provide an in-depth view of the obtained solutions, we present the objective value components of one of the tested instances in Figure 9. The figure shows which parts of the objective value contribute most to the final solutions for the VNS, MA, SS and SAHH: the overall working time, the travel time and the violations of the desired starting times. As can be seen, the superior behavior of the MA approach results in noticeably reduced travel times and desired starting time violations among the examined metaheuristic approaches.

Table 10 shows the relative contributions to the objective function as well as the values for the CP solution. We can see that the latter has no time window violations, no overtime and each qualification matches with the required ones. The major impact on the solution quality comes from the violations of the desired starting times ($\frac{3}{4}$ of the objective value). To obtain better solutions, all approaches decrease these violations by allowing small time window violations and distances between qualifications.

Statistical Tests. To compare the approaches in a meaningful way, we performed a *Wilcoxon rank sum test* for each instance using the results of our 10 runs with an error probability of 5%. Based on the objective values, we can obtain the following domination order using the CP solution:

$$MA < VNS \not< SS < SAHH \tag{16}$$

Fig. 9: The impact of the soft constraints on the objective value for day 4.

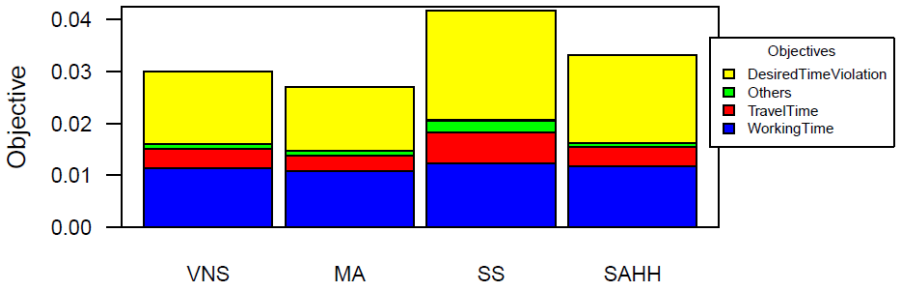


Table 10: Objective value contributions and number of nurses for day 4.

	init.	VNS	MA	SAHH	SS
# Nurses	227	212	168	208	242
Objective Value	0.08651	0.02994	0.02709	0.03324	0.04170
Aspects	<0.01%	<0.01%	<0.01%	<0.01%	<0.01%
Desired Time	76.15%	46.24%	45.92%	51.05%	50.37%
Time Window	0.00%	<0.01%	<0.01%	<0.01%	0.64%
Over Time	0.00%	0.00%	0.00%	0.00%	0.00%
Qualification Distance	0.00%	3.51%	3.16%	2.17%	5.04%
Travel Time	5.74%	12.30%	10.67%	11.40%	14.62%
Working Time	18.11%	37.95%	40.24%	35.36%	29.33%

This means that the MA does consistently obtain significantly lower objective values over every instance than the other approaches. The same holds for the VNS and the Scatter Search which dominate the SAHH approach. Between the VNS and the Scatter Search the results are not so clear and no approach dominates the other for every instance.

No strict ordering exists for random initialized solutions: the MA dominates the other three approaches, whereas the VNS is better than Scatter Search but not significantly better than the SAHH. However, the SAHH is not significantly better than Scatter Search. Based on these results, we observe that the MA outperforms the other approaches significantly comparing the objective value. Since the SAHH and SS are both outperformed based on the initialization, we also assume that the VNS approach generally performs slightly better than the SAHH and SS.

Conclusions and Future Work

In this work, we presented a novel two-step approach to tackle a large-scale real-world multimodal home-healthcare scheduling problem. First, we generate an initial solution that is subsequently improved by one of four metaheuristic approaches. One particular benefit of our approach is its flexibility: since all

problem-specific factors are only incorporated into the objective function, the approach can be easily adapted to other home-healthcare problem settings by simply adapting the objective function; the metaheuristic approaches need not be changed. In our computational evaluation, we see that all metaheuristics produce satisfactory results in reasonable time, where some approaches strongly benefit from *valid* initial solutions that we generate with a CP based approach.

For future work, we plan to extend our current work to generate solutions for multiple days. For those, we need to consider additional constraints, like shift and working hour constraints holding over a longer period of time. It will be interesting to see which of the presented metaheuristic approaches performs best in a multi-day setting.

Acknowledgments. This work is part of the project CareLog, partially funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the strategic programme I2VSplus under grant 826153. We thankfully acknowledge the CareLog project partners Verkehrsverbund Ost-Region GmbH (ITS Vienna Region), Sozial Global AG, and ilogs mobile software GmbH. We also thank our reviewers for their helpful comments.

References

1. Bai, R., Blazewicz, J., Burke, E.K., Kendall, G., Mccollum, B.: A simulated annealing hyper-heuristic methodology for flexible decision support. 4OR: A Quarterly Journal of Operations Research **10**(1), 43–66 (2012)
2. Bai, R., Burke, E.K., Kendall, G., Li, J., McCollum, B.: A hybrid evolutionary approach to the nurse rostering problem. IEEE Transactions on Evolutionary Computation **14**(4), 580–590 (2010)
3. Begur, S.V., Miller, D.M., Weaver, J.R.: An integrated spatial DSS for scheduling and routing home-health-care nurses. Interfaces **27**(4), 35–48 (1997)
4. Bertels, S., Fahle, T.: A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Computers & Operations Research **33**, 2866–2890 (2006)
5. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation Science **39**(1), 104–118 (2005)
6. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation Science **39**(1), 119–139 (2005)
7. Burke, E.K., Curtois, T., Qu, R., Berghe, G.V.: A scatter search approach to the nurse rostering problem. Journal of the Operational Research Society **61**(11), 1667–1679 (2010)
8. Burke, E.K., De Causmaecker, P., Berghe, G.V., Van Landeghem, H.: The state of the art of nurse rostering. Journal of Scheduling **7**(6), 441–499 (2004)
9. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Handbook of Metaheuristics, International Series in Operations Research & Management Science, chap. 16, pp. 457–474. Springer (2003)
10. Cheng, E., Rich, J.L.: A home health care routing and scheduling problem. Technical Report CAAM TR98-049, Rice University (1998)
11. Di Gaspero, L., Schaerf, A.: Multi-neighbourhood local search with application to course timetabling. In: E.K. Burke, P.D. Causmaecker (eds.) PATAT, *LNCS*, vol. 2740, pp. 262–275. Springer (2002)
12. Eveborn, P., Flisberg, P., Rönnqvist, M.: LAPS CARE – an operational system for staff planning. European Journal of Operational Research **171**, 962–976 (2006)

13. Eveborn, P., Rönnqvist, M., Einarsdóttir, H., Eklund, M., Lidén, K., Almroth, M.: Operations research improves quality and efficiency in home care. Interfaces **39**(1), 18–34 (2009)
14. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **29**(3), 653–684 (2000)
15. Hansen, P., Mladenović, N.: Variable neighborhood search. In: F.W. Glover, G.A. Kochenberger (eds.) Handbook of Metaheuristics, pp. 145–184. Kluwer Academic Publisher, New York (2003)
16. Krzysztof, K., Szymanek, R.: Jacop java constraint solver (2011). URL `http://www.jacop.eu`
17. Lundy, M., Mees, A.: Convergence of an annealing algorithm. Mathematical Programming **34**, 111–124 (1986)
18. Matta, A., Chahed, S., Sahin, E., Dallery, Y.: Modelling home care organisations from an operations management perspective. Flexible Services and Manufacturing Journal pp. 1–25 (2012)
19. Moscato, P.: Memetic algorithms: a short introduction, pp. 219–234. McGraw-Hill, Maidenhead, UK (1999)
20. Nikolaev, A.G., Jacobson, S.H.: Simulated annealing. In: M. Gendreau, J.Y. Potvin, F.S. Hillier (eds.) Handbook of Metaheuristics, *International Series in Operations Research & Management Science*, vol. 146, pp. 1–39. Springer (2010)
21. Prandtstetter, M., Raidl, G.R., Misar, T.: A hybrid algorithm for computing tours in a spare parts warehouse. In: C. Cotta, P. Cowling (eds.) Evolutionary Computation in Combinatorial Optimization - EvoCOP 2009, *LNCS*, vol. 5482, pp. 25–36. Springer (2009)
22. Prandtstetter, M., Rendl, A., Puchinger, J.: The influence of accurate travel times on a home health care scheduling problem. In: Proceedings of the 5th International Workshop on Freight Transportation and Logistics. Mykonos, Greece (2012)
23. Rasmussen, M.S., Justesen, T., Dohn, A., Larsen, J.: The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. Tech. Rep. 11-2010, DTU Management Engineering (2010)
24. Rendl, A., Prandtstetter, M., Hiermann, G., Puchinger, J., Raidl, G.: Hybrid heuristics for multimodal homecare scheduling. In: N. Beldiceanu, N. Jussien, E. Pinson (eds.) 9th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'12), *LNCS*, vol. 7298, pp. 339–355. Springer, Nantes, France (2012)
25. Rest, K.D., Hirsch, P.: A tabu search approach for daily scheduling of home health care services using multi-modal transport. pp. 373–377. Odysseus 2012 - 5th International Workshop on Freight Transportation and Logistics, Mykonos, GREECE
26. Rest, K.D., Trautsamwieser, A., Hirsch, P.: Trends and risks in home health care. Journal of Humanitarian Logistics and Supply Chain Management **2**, 34–53 (2012)
27. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier, New York, NY, USA (2006)
28. Toplak, W., Koller, H., Dragaschnig, M., Bauer, D., Asamer, J.: Novel road classifications for large scale traffic networks. In: Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems, pp. 1264–1270 (2010)
29. Trautsamwieser, A., Gronalt, M., Hirsch, P.: Securing home health care in times of natural disasters. OR Spectrum **33**, 787–813 (2011)
30. Trautsamwieser, A., Hirsch, P.: Optimization of daily scheduling for home health care services. Journal of Applied Operational Research **3**, 124–136 (2011)
31. Whitley, D., Kauth, J.: GENITOR: A different genetic algorithm. pp. 118–130 (1988)