Hybrid Heuristics for Multimodal Homecare Scheduling

Andrea Rendl[1], Matthias Prandtstetter[1] Gerhard Hiermann[2], Jakob Puchinger[1], and Günther Raidl[2]

[1] AIT Austrian Institute of Technology
Mobility Department, Dynamic Transportation Systems
[2] Vienna University of Technology, Austria

Abstract. We focus on hybrid solution methods for a large-scale real-world multimodal homecare scheduling (MHS) problem, where the objective is to find an optimal roster for nurses who travel in tours from patient to patient, using different modes of transport. In a first step, we generate a valid initial solution using Constraint Programming (CP). In a second step, we improve the solution using one of the following metaheuristic approaches: (1) variable neighbourhood descent, (2) variable neighborhood search, (3) an evolutionary algorithm, (4) scatter search and (5) a simulated annealing hyper heuristic. Our evaluation, based on computational experiments, demonstrates how hybrid approaches are particularly strong in finding promising solutions for large real-world MHS problem instances.

1 Introduction

The demand for care of the elderly is constantly increasing in today's western world, thus efficient healthcare services are of great significance. The idea behind home healthcare is to nurse patients at home instead of at retirement homes: patients can book different kinds of jobs (e.g. cleaning, cooking, medical services) and nurses of adequate qualification visit patients in a tour, using a specific transport mode. The goal is to find a nurse roster where all jobs are assigned to adequate nurses and employer, nurse and customer satisfaction is maximized.

However, finding a good nurse roster is challenging, since the problem is a combination of two NP-hard problems: vehicle routing with time windows (VRPTW) [4, 5] and nurse rostering (NRP) [7]. Moreover, we consider a *large-scale real-world* setup from a Viennese public healthcare company. Therefore, we apply a *heuristic* approach since exact approaches are typically too expensive for large instances. The idea is to start with an initial valid solution that is then systematically improved by different metaheuristical approaches that we evaluate in this work. Another aim is to construct a *flexible framework* that can be applied in other homecare companies with similar side constraints. Therefore, our approach is as generic as possible, where side constraints can be easily altered. We first present the problem (Section 2), initial solution generation (Section 3), the metaheuristics (Section 4), our computational results (Section 5) and conclude our work in Section 6.

2 The Multimodal Homecare Scheduling Problem

The Multimodal Homecare Scheduling (MHS) problem deals with finding a roster for nurses who perform nursing services at patients' homes. Therefore, the problem contains a *rostering* component (assigning nurses to jobs, wrt various shift/working hours restrictions), as well as a *vehicle routing* component (nurses travel in tours from patient to patient, starting at their home). The multimodal aspect arises from the transport mode each nurse chooses for travelling.

Basically, we have a set of customers, the *patients*, who book one or more jobs, daywise summarised in the set of jobs $\mathcal{J} = \{1, \ldots, J\}$, with $1 \leq J$. Each job has an associated *location*, given as GPS coordinate, a *start time window* in which the job should start, and a *duration*. Furthermore, each job requires a minimal *qualification* $q \in \mathcal{Q}$ of whoever performs the job (e.g. taking blood samples requires a medical qualification). In our problem setup, $|\mathcal{Q}| = 5$, i.e. there are 5 different qualifications.

Each job has to be assigned to one of the N nurses defined in set $\mathcal{N} = \{1, \ldots, N\}$, with $1 \leq N$. Since nurses travel to patients starting from their home, we store the *home location* of each nurse as GPS coordinate. Each nurse has a *qualification* $q \in \mathcal{Q}$ and may only perform jobs that require a qualification that is less or equal the nurse's maximal qualification. Hence, nurses of higher qualification may perform lower-qualified jobs, but not vice versa. Nurses can only work within specified *time windows* that vary from day to day, and have a minimal and maximal *number of working hours* per day.

In addition to nursing jobs, we also consider *pre-allocated jobs* that nurses perform in addition to nursing (e.g. team meetings). These jobs are assigned to a fixed location and nurse and make up at most 5% of all jobs. Furthermore, patients and nurses can state *negative preferences* towards another, for instance, a nurse might refuse to work at a patient who owns a dog if she has a dog allergy, or a female patient might refuse a male nurse. In this case, the nurse may not be assigned to the respective job in the roster.

A novel aspect in this work is the consideration of *multimodality*: each nurse states her preferred *mode of transport* (public transport or car) that we consider in creating the roster. To obtain meaningful results, it is crucial to have accurate travel time estimates for each transport mode. Therefore, we obtain travel times according to the GPS coordinates of nurses and jobs based on data from (1) the Viennese public transport system, for public transport and (2) a large set of historical data from Viennese floating car data for transport by car.

2.1 Objective Function

In this work, we want to find a valid MHS roster, that is also (closely) *optimal* for employer, employee and customer and therefore

1. minimizes the **costs for the employer** (e.g. minimizes travel times),
2. maximizes **customer satisfaction** (e.g. by minimizes deviations from customer preferences, such as time window violations or preferences for nurses)

3. maximizes **nurse satisfaction** (e.g. minimizes deviations from contractual constraints and other nurse preferences)

The quality of a solution is assessed by an *objective function* that assigns a real number, the *objective value ob*, to a solution. We formulate the problem as a *minimization* problem, and design our objective function as a weighted sum of terms ob_i, each representing a type of constraint violation i:

$$ob = \gamma_j{*}ob_j + \gamma_{tw}{*}ob_{tw} + \gamma_q{*}ob_q + \gamma_p{*}ob_p + \gamma_d{*}ob_{dt} + \gamma_n{*}ob_n + \gamma_{tt}{*}ob_{tt}$$

We want to immediately determine the number of hard constraint violations from the objective value and therefore weight each term ob_i with a normalization factor γ_i. Thus, for each ob_i, we set γ_i such that the objective value of a valid solution lies between 0 and 1. Furthermore, we define each ob_i such that every hard constraint violation increases the objective value by 1. For instance, an objective value of '17.34' states that the respective solution contains 17 hard constraint violations. The remaining '0.34' represent the costs for the 'valid part' of the solution (e.g. travel time, working time, etc). We briefly summarize the types of (hard and soft) constraint violations we consider:

1. ob_j states how many **jobs are unassigned** in the solution, i.e. $ob_j = \sum_{j \in J} \cdot \chi(j)$ where $\chi(j)$ equals 0 if job $j \in J$ is assigned to a nurse, and 1 otherwise. γ_j equals the number of jobs J.
2. The square of the **deviation from the job time windows** in minutes (up a to a maximal deviation MAX), is added to the objective function in term ob_{tw} that is normalized by $\gamma_{tw} = MAX$.
3. The number of **qualification violations** is contained in term ob_q that is normalized by the number of jobs J.
4. ob_p counts the number of **violations of patient and nurse preferences** and γ_p equals the number of stated preferences.
5. ob_d summarizes the **deviation of the desired job start time** up to 60 minutes, which is also the normalization factor γ_d.
6. ob_n reflects the number of **working time violations** over all nurses according to the nurse's contract and shift regulations. The normalization factor γ_n corresponds to the maximal possible working time, 16 hours.
7. The **travel time** of each nurse to and from her home is represented by ob_{tt} that is normalized by the maximal travel time of 180 minutes.

2.2 Related Work

The home healthcare problem has been considered using different approaches: Eveborn et al. [8] choose a set-partitioning formulation for the problem, and primarily focus on the application of *repeated matching*. Bertels and Fahle [3] combine *linear programming* (LP) for computing optimal start times, *constraint programming* (CP) for generating initial solutions, and *simulated annealing* (SA) and *tabu search* (TS) based heuristic for subsequent solution improvement. Steeg and Schröder [15] use CP to construct an initial solution that is improved by

a combination of *Adaptive Large Neighborhood Search*(ALNS) and SA. Rasmussen et al [13] use a set partitioning formulation of the problem and apply a specialized branch-and-price solution algorithm.

The main difference of our work to other approaches lies within (1) the novel consideration of multimodality, (2) our objective to provide a *flexible* framework to tackle real-world HHC problems with varying side constraints (3) a novel, broad selection of metaheuristics for the MHS problem and (4) the excessively large size of instances that we tackle.

3 Initial Solutions with Constraint Programming

First, we require a *valid* initial solution, i.e. a solution where all jobs are assigned and none of the side constraints presented in Section 2 are violated. This is particularly challenging for three reasons: first, we cannot apply a simple construction heuristic, such as a random job-nurse assignment, since many side constraints have to be considered. Second, the instances are particularly large (about 700 jobs/500 available nurses). Third, we may only invest little time into the initial solution generation, since this step is just the first of many steps in our optimization process.

Constraint Programming(CP) is a particularly attractive candidate for initial solution generation: first, CP is strong in solving discrete decision problems (i.e. finding *first* solutions) [9, 14]. Second, CP search strategies – if set up correctly — can yield fairly good initial solutions (e.g. favoring low values for job time variables during search produces tight schedules since job times are set as early as possible). Third, the CP model can be easily altered to similar problem setups (of other healthcare companies) by simply editing side constraints, and hence satisfies our requirement of a flexible, extendable system.

The instances we consider are far too large to be solved within a single CP model, therefore we simplify the problem by solving each instance qualification-wise: we create a subinstance I_q for qualification q from instance I, where the jobs in I_q are those that require qualification q and the nurses are those whose highest qualification is q. This means we first start solving the instance for the highest qualification, 'medical nursing', that includes all medical nurses and all jobs that require a medical nurse. Then we continue with the second-highest qualification, and so on. Note, that in case we cannot solve a subinstance of qualification q, we iteratively add nurses of higher qualification to the instance.

3.1 The CP Model and CP Search Setup

We formulate the MHS problem as an extension of the Vehicle Routing Problem with Time Windows (VRPTW) model from [14] that contains three kinds of variables: (1) predecessor and successor variables, $pred_i$ and $succ_i$ of visit i that capture the sequence of tours, (2) nurse variables n_i, stating which nurse performs visit i, and (3) variables for the start time of each visit i, t_i. We extend the model with binary variables x_{ij} (x_{ij} is 1 if nurse i performs job j) for channeling

and add constraints to represent multi-modality, nurse and patient preferences, shift length as well as redundant constraints to increase propagation.

We apply a static search heuristic, where we first set half of the successor and predecessor variables (i.e. we start with fixing the job sequence), then set half of the nurse variables (nurse to job assignments), followed by the remaining half of predecessor/successor variables and remaining nurses. Finally, we search for job start times. We order nurses by their start time (nurses available in the morning come first), and order jobs by their desired start time. This way, search is extremely effective with an ascending value selection (smallest value first): typically no more than 10% of the decisions made during search are wrong.

3.2 Iterative Clustering

Decomposing the problem qualification-wise does not yield equally-sized subproblems. For instance, jobs of qualification 'basic homecare' typically make up 80% of all jobs, yielding again a far too large subproblem. We therefore introduce another decomposition step that is triggered for particular qualifications or if a time-limit is reached: decomposition by area.

A cluster consists of a set of jobs and nurses from the overall problem instance. Finding a *feasible* cluster, where the underlying instance is solvable (in reasonable time) is quite challenging, since both *vicinity* and *temporal availabilty* are crucial. In other words, for all jobs within a cluster, we need to find a set of nurses who can reach the jobs within time (vicinity) and who are available in the jobs' time slots (temporal availability). In our approach, we create and solve clusters one by one, and *iteratively* alter those clusters that we cannot solve straight away.

Creating Clusters using a Quadtree We reflect the spatial distribution of jobs by inserting them into a quadtree. A quadtree is a tree datastructure, whose subtrees have up to four children that, for our purpose, are labelled *north*, *east*, *south* and *west*. Each node (and leaf) represents a job. We insert job j into (sub)tree i (with corresponding job j_i) by comparing j and j_i's GPS coordinates: if j is located to the north of j_i, j is inserted into the *north* child of i, if j is located to the east of j_i, j is inserted into the *east* child of i, and so on. If the respective child does not exist, then j becomes the child of j_i. In this way, we insert all jobs into the quadtree, where initially, the root node contains an empty job with the GPS coordinate of a centre location of Vienna, to assure that the quadtree is reasonably balanced.

Now we can initiate the selection. First, we select k jobs by k times removing the first job that we find using a simple depth-first search. In this way, the k selected jobs are reasonably closely located. Second, we select n nurses with $n = min(k, N)$ where N represents the number of available nurses for the subinstance. We want to find nurses that are reasonably close to the jobs, so we first randomly selected a job j, then order the nurses by distance to job j and finally pick the n closest nurses. This way, we retrieve a cluster with k jobs and n nurses.

Iteratively Altering Clusters In our approach we first create a cluster from a given set of jobs and nurses that are closely located. Second, we invest τ_k time-

units to solve that cluster. If we cannot find a solution within τ_k, we iteratively simplifying the instance until we find a solution or reach an overall timelimit τ:

1. Given jobs M, nurses N, cluster-timeout τ_k and overall timeout τ
2. If $|M| \geq 1$ and timeout τ has not yet been reached: create cluster $c = (M', N')$ where $N' \subseteq N$ and $M' \subseteq M$, otherwise stop.
3. If timeout τ has not yet been reached: attempt to solve cluster c in τ_k secs
   (a) If solving was successful, update the set of overall jobs M to $M - M'$ and the set of overall nurses N to $N - N'$ and goto 2.
   (b) Otherwise, if $|M'| \geq 1$, remove $f(M)$ jobs from M', yielding a new, smaller cluster c=(M'',N') and goto 3.
   (c) Otherwise, if $|N| \geq 1$ add m' nurses to N', yielding a new cluster c=(M',N'') and go to 3.
   (d) Otherwise, increase τ_k by t timeunits

In our current setup, we choose k=25, τ_k=1sec, τ=300secs, t=10secs, $f(M)$=$|M|$-10 if $|M| \geq 10$ and $|M|$-1 otherwise, m'=1. This setup typically produces a valid solution for a 700jobs/500nurses instance within 15-20 seconds — more details and experimental results can be found in Section 5. In case we cannot find an overall solution within τ seconds, the CP approach has failed and we use an alternative approach, as outlined in Section 3.3.

3.3 Random Solution Generation

As an alternative approach, we employ a simple solution generation heuristic that produces a random nurse-job assignment. The generated solution only guarantees that all jobs are executed by a nurse of adequate qualification and that all pre-allocated jobs (such as appraisal interviews) are assigned to the right nurse. No side constraints, such as time window restrictions are considered. We use this technique as a backup to the CP approach and to evaluate the influence of the initial solution during the later improvement phase.

4 Solution Improvement by Different Metaheuristics

A solution to the MHS problem consists of a list of tours—one for each nurse—where a tour contains a sequence of jobs, or is empty in case the nurse is not employed for that particular day. We receive such a (valid) initial solution from the solution generation heuristics discussed in the previous section. However, these generated solutions are typically of mediocre quality and cannot be used for real-world rosters. Therefore, we improve the solution with respect to the objective function (Section 2.1) using different metaheuristics that we discuss in this section.

4.1 Variable Neighborhood Descent

Based on the idea that global optima are locally optimal with respect to all (possible) neighborhoods, *variable neighborhood descent* (VND) REFERENZ tries to systematically examine a predefined set of neighborhood structures N_i, with $1 \leq i \leq l_{max}$ whereupon l_{max} denotes the number of defined structures. To fasten the search, the neighborhood structures are ordered such that potentially smaller neighborhoods $N(x)$ for a given solution x are examined first. As soon as a local optimum is reached (within the i-th neighborhood of x), the search is continued with neighborhood $N_{i+1}(x)$. If, however, an improvement is found in the current neighborhood, the further search is continued with neighborhood structure N_1 again.

Within this work, we define three neighborhood structures implicitely via three different move types:

shift mission A shift mission move shift one mission from one tour to another tour by searching the best matching position in that other tour.

reposition mission When applying a move of that type, one mission is repositioned within its tour, i.e. the best matching position (without reordering the other missions) is determined.

swap nurses By this move, two nurses are swapped with each other, i.e. the tour of the first nurse is then handled by the second nurse and vice versa.

Although the initial neighborhood order (shift missions, swap nurses, reposition mission) leads to rapid improvements during the starting phase of VND, preliminary tests revealed that dynamic neighborhood reordering as applied in REFERENZ leads to more promising results.

4.2 General Variable Neighborhood Search

One of the major drawbacks of VND is the fact that the search might get stuck in local optima. To overcome this, VND is typically embedded as local search phase in a general *variable neighborhood search* (VNS) scheme REFERENZ. Here, the basic principle is to first locally improve a given start solution until a local optimum is reached which in the second step is then randomly changed by so-called shaking moves, i.e. local random perturbations, trying to emerge from local optima while at the same time preserving the solution structure since it is assumed that in most cases local (and global) optima are similar to each other. To broaden search, the randomness introduced during shaking is enlarged each time the local search phase (e.g. VND) did not improve the current best solution.

In our case, shaking is performed by applying $i + 1$ random shift mission moves within the i-th consecutive iteration of VNS without improvement.

4.3 A Hybrid Evolutionary Algorithm

Evolutionary algorithms (EA) [11] imitate biological evolution by applying evolutionary mechanisms, such as reproduction, selection, mutation and recombination, to iteratively derive good solutions from a pool of initial individuals.

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

Fig. 1. Operator setup for the hybrid Evolutionary Algorithm (MA)

During initialization, a pool of parent solutions is generated, the so-called *population*. Typically, the initial population has a very high deversity to cover a large region of search space. From this population, a new population is breeded: first, a set of candidates is selected from the population following a particular *selection heuristic*. Second, these candidates, the *parents*, are used to create offsprings by *recombination* and/or *mutation*, depending on a specified probabilty. Recombination is performed by a *recombination operator* that recombines two (or more) parent solutions to create an offspring, while mutation alters parts of the parent. Subsequently, the resulting offspring replaces the parents in the population according to a *replacement strategy* and thus yields a new generation. In this manner, the population is contineously evolving until a termination criteria is reached (e.g. timeout or a generation limit).

In this work we apply a hybrid EA, or Memetic Algorithm (MA) [12], that uses local search to improve the offsprings before replacement. Since Burke et al [2] presented promising results for the Nurse Rostering Problem (NRP) with an MA, we expected similary good results for our problem. In the following we discuss details about recombination, mutation and local search of our approach. The overall parameter settings are summarized in Figure 1.

Initialization. We create the initial population based on the solution given from the initial solution generation heuristic: for each job j, we select the nurse that has been least often assigned to j in already generated solutions. If there is more than one such nurse, we pick one randomly. This way, the initial population is diverse, but may contain solutions that are not valid.

Recombination Operator. Since solutions are lists of tours (one for each nurse), we implement a special recombination operator: 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_i). Each unfixed job of n_i assigned in P_1 is removed in the offspring solution. Then every remaining job of n_i in the offspring solution is moved to the best nurse n_j, where $n_j \neq n_i$. In the final step, the jobs of n_i in P_1 are assigned to n_i in the offspring solution. This creates one offspring, and the parent with the lowest objective value is set to P_2.

Mutation. To provide higher diversity during the search of the MA, the offspring is mutated by selecting a nurse n_i at random and reassigning all unfixed tours of this nurse to other nurses n_j, where $n_j \neq n_i$, maximizing the objective value.

Local Search. For a given probability, a local search heuristic tries to further improve the offspring. To achieve a better balance between exploration and exploitation, the heuristic aborts after a certain time limit. We apply two local search algorithms: variable neighbourhood descent (VND), as described in Section 4.1, and cyclic search of neighbourhoods (CNS). The best results were obtained using CNS that searches in the next neighbourhood of a given list of neighbourhoods: if a neighbourhood cannot improve the solution within a time limit, the next neighbourhood is selected. If none of the neighbourhoods can find a better solution, the procedure terminates. The population is replaced using a slightly changed steady-state approach[XXX], where an offspring always replaces the most similar solution in the current population that is worse than the offspring. To calculate the similarity of two solutions, we use a simple count of same job to nurse assignments.

4.4 Simulated Annealing Hyper-Heuristic (SAHH)

Simulated Annealing (SA) [XXX] is a local search technique that is inspired by 'annealing', a heat treatment of material in metallurgy, where a controlled heating and cooling process allows atoms to find new positions yielding a configuration of (closely) minimal internal energy. Similarly, in Simulated Annealing, the current solution is replaced by a newly-constructed solution (typically in the neighborhood of the current solution) by a certain probability. This probability depends on the solutions' objective values and the global *temperature* T. If T is large, the replacement works almost randomly, while a small T (close to zero) prefers better solutions, but can risk becoming stuck in local optima. Therefore, the temperature is gradually altered. A Simulated Annealing Hyper Heuristic (SAHH) utilizes other local search techniques (low-level heuristics) as a slaves to improve newly-constructed solutions. Each low-level heurstic has a probability to be selected for the current iteration. These probabilities are changed during the search using an adaptive learning approach, where the selection probability depends on an accept/tested ratio in a predefined learning period.

Bai et al [1] implemented this kind of algorithm for the NRP. As the NRP is related to our problem, we tested this approach too. According to the definition of the SAHH by Bai et al [1], the only parameter changes that have to be made are the used low-level heursitics and the total number of iterations K. As neighbourhoods we use two improvement strategies for each neighbourhood also used in the VND. First we selected the random-improvement strategy with following constraint: we only consider solutions, which number of hard constraints violations are at most the number of violations in the current solution. The second strategy used is next-improvement.

Bai et al. stated for their parameters that around 10% of the solutions should be accepted at the beginning and only 0.5% at the end of the search. Therefore

SAHH Neighbourhood	Settings
shift job neighbourhood	random improvement
	next improvement
swap nurse neighbourhood	random improvement
	next improvement
reposition job neighbourhood	random improvement
	next improvement

Fig. 2. SAHH neighbourhoods and parameters

we calculated our starting/ending temperature to accept 10%/0.5% if the value is 0.5 worse then the current solution.

4.5 Scatter Search

Another population based approach that we evaluate is scatter search [10]. As described by Burke et. al. [6], this metaheuristic creates also good solutions for the NRP. Scatter Search uses a small population of diverse solutions and creates subsets of small size from this pool. These subsets are then combined using a solution combination method to create new solutions which are then improved using a local search algorithm.

Initialization. We create the initial population the same way as for the hybrid EA where we use the solution created by the CP-Solver and create the additional solutions with a random diverse constructor.

Subset Generation. To generate subsets out of the pool of solutions (also called reference set or RefSet), a classical subset generation approach was used [10]: We first create subsets of all pairs of solutions. Next subsets containing of 2-sets plus the best solution not in the set are created (duplicate subsets are removed). Then the same for all 3-sets is done. At last, subsets of size $k = 5$ to the size of the RefSet containing the k best solutions are creates.

Subset Combination. For the combination of solutions for each subset we implemented a path-relinking algorithm. Path-relinking [10] moves from a solution S_i to S_j where a move is defined by a different assignment of a variable. Unlike the classic path-relinking approach we do not perform a local search on a given number of promising solutions but create one solution out of the subset which will then be improved. To create a solution S_new out of a subset of size l we decided to move iterativly from the worst solution S_1 to the best S_l. $S_{new} = S'_l$, where

$$S'_i = S'_{i-1} \rightarrow_{PR} S_i, i = 2, ..., l, S'_1 = S_1$$

and the path-relinking operator $S_i \rightarrow_{PR} S_j$ moves S_i $\mathit{difference}(S_i, S_j)/l$ steps to S_j. To determine which moves are performed first, we calculate the change in the objective per move and perform the move with the best change

(which can also be worse). In order to prevent high runtimes due to a possible large set of moves to be tested, we restrict the number of moves to be calculated with a parameter k.

This combination algorithm was introduced after some preliminary testings with the algorithm described by Burke et al [6]. This tests indicated that the used 'construction by voting' approach does not fit for our model, as the constructed solutions where no good starting points for the local search procedure.

5 Computational Results

We assess our hybrid heuristics on a random selection of real-world instances from 2011 with the same setup. First, we examine the initial solution generation heuristic and second, we compare the impact of the presented metaheuristics.

5.1 Evaluating Initial Solution Generation

We evaluate initial solution generation on a XXX machine with 4 cores with 2 processes running in parallel, each given a maximum of 4GB RAM (with a maximal capacity of 12GB RAM). All techniques are implemented within the same framework in Java and JaCoP[XXX] has been used as constraint solver. In Table XXX we give results averaged over 10 runs.

5.2 Evaluating Metaheuristics

Each metaheuristic is given a maximum of 75 minutes to improve the initial solution. The parameter settings of each metaheuristic is summarized in Table 3.

We evaluate our techniques on a XXX machine with 8 cores where 4 processes are run in parallel, each given a maximum of 4GB RAM (with a maximal capacity of 24GB RAM). All techniques are implemented within the same framework in Java, where we use JaCoP[XXX] as a CP solver. In Table XXX we give results averaged over 10 runs.

6 Conclusions

In this work, we tackle a large-scale real-world Multimodal Homecare Scheduling (MHS) problem using a CP-based construction heuristic combined with one of five different metaheuristical approaches. Our contributions are threefold: first, we show how we can generate valid initial solutions for the MHS in very little time using a Constraint Programming-based clustering heuristic. Second, we study the impact of different metaheuristics on a novel problem setup. Third, and most importantly, this work delivers another success story of how hybrid approaches can effectively tackle large-scale real-world problems.

For future work, we want to extend our current one-day approach to generate multi-day solutions, where we also consider shift and working hour constraints that hold over a longer period of time. Furthermore, we want to explore large neighborhood search[XXX] as another metaheuristic to tackle the MHS problem.

EA 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

SAHH Parameters	**Settings**
K (# iterations)	10000
acceptance prob. (start)	0.05
acceptance prob. (end)	0.005
learn period	$K/500$
nrep (#iter/temp)	#neighbourhoods
min weight	0.1
change of temperature t	Lundy and Meers' nonlinear function $t = t/(1 + \beta t)$, $\beta = ((t_{start} - t_{end}) \cdot nrep/K \cdot t_{start} \cdot t_{end})$

Fig. 3. Parameter settings for each metaheuristic

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. The authors thankfully acknowledge the CareLog project partners Verkehrsverbund Ost-Region GmbH (ITS Vienna Region), Sozial Global AG, and ilogs mobile software GmbH.

References

1. Bai, R., Blazewicz, J., Burke, E.K., Kendall, G., Mccollum, B.: A simulated annealing hyper-heuristic methodology for flexible decision support. Tech. rep., School of Computer Science, University of Nottingham, England (2006)
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. Bertels, S., Fahle, T.: A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Comput. Oper. Res. 33, 2866–2890 (October 2006), http://dl.acm.org/citation.cfm?id=1143203.1143210
4. 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)
5. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, part ii: Metaheuristics. Transportation Science 39(1), 119–139 (2005)
6. 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)
7. Burke, E.K., De Causmaecker, P., Berghe, G.V., Van Landeghem, H.: The state of the art of nurse rostering. Journal of Scheduling 7, 441–499 (November 2004), http://portal.acm.org/citation.cfm?id=1029473.1029477
8. Eveborn, P., Flisberg, P., Ronnqvisb, M.: Laps care - an operational system for staff planning. European Journal of Operational Research 171, 962–976 (2006)
9. Gent, I., Walsh, T.: Csplib: a benchmark library for constraints. Tech. rep., Technical report APES-09-1999 (1999), available from http://csplib.cs.strath.ac.uk/. A shorter version appears in the Proceedings of the 5th International Conference on Principles and Practices of Constraint Programming (CP-99).
10. Glover, F., Laguna, M., Mart, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics 39, 653–684 (2000)
11. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1st edn. (1989)
12. Moscato, P.: Memetic algorithms: a short introduction, pp. 219–234. McGraw-Hill Ltd., UK, Maidenhead, UK, England (1999), http://dl.acm.org/citation.cfm?id=329055.329078
13. 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 (May 2010)
14. Rossi, F., van Beek, P., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA (2006)
15. Steeg, J., Schröder, M.: A hybrid approach to solve the periodic home health care problem. In: Kalcsics, J., Nickel, S. (eds.) Operations Research Proceedings 2007, Operations Research Proceedings, vol. 2007, pp. 297–302. Springer Berlin Heidelberg (2008)