The Home Care Crew Scheduling Problem:
Preference-Based Visit Clustering and Temporal Dependencies

Rasmussen, Matias Sevel; Justesen, Tor; Dohn, Anders Høeg; Larsen, Jesper

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

The Home Care Crew Scheduling Problem: Preference-Based Visit Clustering and Temporal Dependencies

Report 11.2010

DTU Management Engineering

Matias Sevel Rasmussen
Tor Justesen
Anders Dohn
Jesper Larsen
May 2010

DTU Management Engineering
Department of Management Engineering

The Home Care Crew Scheduling Problem: Preference-Based Visit Clustering and Temporal Dependencies

Matias Sevel Rasmussen* Tor Justesen Anders Dohn
Jesper Larsen

Department of Management Engineering
Technical University of Denmark

May, 2010

Abstract

In the Home Care Crew Scheduling Problem a staff of caretakers has to be assigned a number of visits to patients' homes, such that the overall service level is maximised. The problem is a generalisation of the vehicle routing problem with time windows. Required travel time between visits and time windows of the visits must be respected. The challenge when assigning visits to caretakers lies in the existence of soft preference constraints and in temporal dependencies between the start times of visits.

We model the problem as a set partitioning problem with side constraints and develop an exact branch-and-price solution algorithm, as this method has previously given solid results for classical vehicle routing problems. Temporal dependencies are modelled as generalised precedence constraints and enforced through the branching. We introduce a novel visit clustering approach based on the soft preference constraints. The algorithm is tested both on real-life problem instances and on generated test instances inspired by realistic settings. The use of the specialised branching scheme on real-life problems is novel. The visit clustering decreases run times significantly, and only gives a loss of quality for few instances. Furthermore, the visit clustering allows us to find solutions to larger problem instances, which cannot be solved to optimality.

*Corresponding author: Email: mase@man.dtu.dk. Address: Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 424, DK-2800 Kgs. Lyngby, Denmark. Tel.: +45-45254442. Fax: +45-45933435.

Keywords: Home care, Health care, Home health care, Crew scheduling, Vehicle routing, Vehicle routing with time windows, Visit clustering, Clustering, Preferences, Generalised precedence constraints, Temporal dependency, Temporal dependencies, Synchronisation, Real-life application, Branch-and-price, Column generation, Dantzig-Wolfe decomposition, Set partitioning, Scheduling, Routing, Integer programming

1 Introduction

The Home Care Crew Scheduling Problem (HCCSP) described in this paper has its origin in the Danish health care system. The home care service was introduced in 1958 and since then, there has been a constant increase in the number of services offered. The primary purpose is to give senior citizens and disabled citizens the opportunity to stay in their own home for as long as possible. The HCCSP is the problem of scheduling caretakers in a way that maximises the service level, possibly even at a reduced cost.

When a citizen applies for home care service, a preadmission assessment is initiated. The result of the assessment is a list of granted services. The services may include cleaning, laundry assistance, preparing food, and support for other everyday tasks. They may also include assistance with respect to more personal needs, e.g. getting out of bed, bathing, dressing, and dosing medicine. As a consequence of the variety of services offered, people with many different competences are employed as caretakers.

Given a list of services for each of the implicated citizens, a long-term plan is prepared. In the long-term plan, each service is assigned to specific time windows, which are repeated as frequently as the preadmission assessment prescribes. The citizens are informed of the long-term plan, and hence they know approximately when they can expect visits from caretakers. From the long-term plan, a specific schedule is created on a daily basis. In the daily problem, caretakers are assigned to visits. A route is built for each caretaker, respecting the competence requirements and time window of each visit and working hours of the caretaker.

In the following, we restrict ourselves to looking at the daily scheduling problem only. The problem is a crew scheduling problem with strong ties to vehicle routing with time windows. However, we have a number of complicating issues that differentiates the problem from a traditional vehicle routing problem. One complication is the multi-criteria nature of the objective function. It is, naturally, important to minimise the overall operation costs. However, the operation costs are not very flexible in the daily scheduling problem. More important is it to maximise the level of service that can be provided. The service level depends on a number of different factors. Often, it is impossible to fit all visits into the schedule in their designated time windows. Hence, some visits may have to be rescheduled

or cancelled. In our solutions, a visit is either scheduled within the given restrictions or marked as uncovered. The manual planner will deal with uncovered visits appropriately. The main priority is to leave as few visits uncovered as possible. Also, all visits are associated with a priority and it is important to only reschedule and cancel less significant visits. Furthermore, it is important to service each citizen from a small subgroup of the whole workforce (the so-called preferred caretakers), as this establishes confidence with the citizen. Another complication compared to traditional vehicle routing, is that we have temporal dependencies between visits. The temporal dependencies constrain and interconnect the routes of the caretakers.

HCCSP generalises the Vehicle Routing Problem with Time Windows (VRPTW) for which column generation solution algorithms have proven successful, see Kallehauge et al. (2005). Therefore, we model HCCSP as a Set Partitioning Problem (SPP) with side constraints and develop a branch-and-price solution algorithm. Temporal dependencies are modelled by a single type of constraints: generalised precedence constraints. These constraints are enforced through the branching. Different visit clustering schemes are devised for the problem. The schemes are based on the existence of soft preference constraints. The visit clustering schemes for the exact branch-and-price framework are novel. The visit clustering will naturally compromise optimality, but will allow us to solve larger instances. We will compare the different visit clustering schemes by testing them both on real-life problem instances and on generated test instances inspired by realistic settings. To our knowledge, we are the first to enforce generalised precedence constraints in the branching for real-life problems. The contribution of this paper is hence twofold. Firstly, we devise visit clustering schemes for the problem, and secondly, we enforce generalised precedence constraints in the branching for the first time for real-life problems.

Optimisation methods for crew scheduling are widely used and described in the literature, especially regarding air crew scheduling. However, when it comes to scheduling of home care workers the literature is sparse. This work builds on top of two recent Master's theses, Lessel (2007) and Thomsen (2006). The most recent of these, Lessel (2007), uses a two-phase approach which first groups the visits based on geographical position, competences, and preferences. A caretaker is associated to each group and the second phase considers each group as a Travelling Salesman Problem with Time Windows (TSPTW).

The other thesis, Thomsen (2006), treats the problem as a Vehicle Routing Problem with Time Windows and Shared Visits (VRPTWSV) and uses an insertion heuristic to feed a tabu search with initial solutions. The models and solution methods in Lessel (2007) and Thomsen (2006) can only handle connected visits where two caretakers are at the same time at the citizen.

With offset in the Swedish home care system, Eveborn et al. (2006) describe a system in operation. They use a Set Partitioning Problem model

and solve the problem heuristically by using a repeated matching approach. The matching combines caretakers with visits. Eveborn et al. (2006) report that the travelling time savings in a moderate guess are 20% and that the time savings on the planning correspond to 7% of the total working time. Bredström and Rönnqvist (2008) show a mathematical model that can handle synchronisation constraints and precedence constraints between pairs of visits. The model is a VRPTW with the additional synchronisation and precedence constraints that tie the routes together. They solve the model using a heuristic. Bredström and Rönnqvist (2007) develop a branch-and-price algorithm to solve the model of Bredström and Rönnqvist (2008), but without the precedence constraints. The model is decomposed to an SPP and the integrality requirement on the binary decision variables is relaxed. Also, the synchronisation constraints are removed from the SPP. Instead, integrality and synchronisation are handled by the branching, and to our knowledge they are the first to use a non-heuristic solution approach to home care problems. Their subproblem is an Elementary Shortest Path with Time Windows (ESPPTW).

Bertels and Fahle (2006) use a combination of linear programming, constraint programming and metaheuristics for solving what they call the Home Health Care Problem. However, they do not incorporate connected visits, which makes their approach less interesting for our situation. Begur et al. (1997) describe a decision support system (DSS) in use in the United States. The DSS provides routes for caretakers by using GIS systems. Their model is a Vehicle Routing Problem (VRP) without time windows and without shared visits, which again is not suitable for our needs. Cheng and Rich (1998) describe the Home Health Care Routing and Scheduling Problem which they model as a Vehicle Routing Problem with Time Windows (VRPTW). They distinguish between full-time and part-time caretakers. They use a two-phase heuristic approach, in which they first find an initial solution using a greedy heuristic. Next, the solution is improved using local search. The model does not include temporal connections between visits.

Related to the HCCSP is the Manpower Allocation Problem with Time Windows (MAPTW). A demanded number of servicemen must be allocated to each location within the time windows. Primarily the number of used servicemen must be minimised, and secondarily the used travel time. The jobs have different locations, skill requirements, and time windows. This problem is dealt with by Lim et al. (2004). More closely related to HCCSP is the Manpower Allocation Problem with Time Windows and job-Teaming Constraints (MAPTWTC). Li et al. (2005) present a construction heuristic combined with simulated annealing for solving MAPTWTC instances. Their model adds synchronisation constraints to the model of Lim et al. (2004), but does not include precedence constraints. MAPTWTC is also solved in Dohn et al. (2009a), again with multiple teams cooperating on tasks. An exact solution approach is introduced. They decompose to a set partitioning

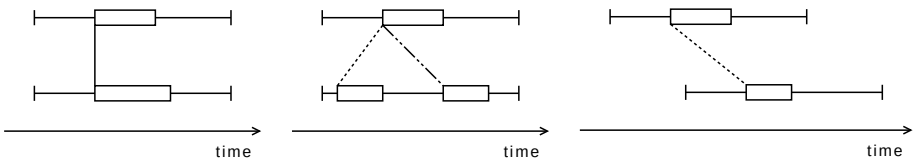
problem and develop a branch-and-price algorithm. The subproblem in the column generation is an ESPPTW.

The HCCSP can be seen as a VRPTW, but with the addition of the complicating connections between visits, and with another objective than the regular minimisation of total distance. When only synchronised visits are considered as connection type, the problem can be referred to as shared visits, yielding the VRPTWSV. The literature on VRPTW is huge. We refer to Kallehauge et al. (2005) and Cordeau et al. (2002). Recently, a variant of VRPTW very similar to the HCCSP has been described in Dohn et al. (2009b). The authors formalise the concept of temporal dependencies in the Vehicle Routing Problem with Time Windows and Temporal Dependencies (VRPTWTD) and investigate the effectiveness of different formulations and solution approaches.

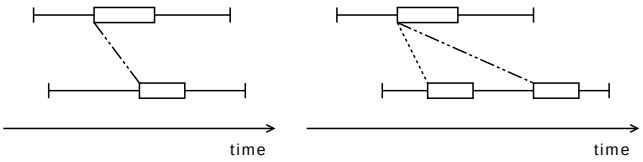
The remainder of the paper is organised as follows. In Section 2, we present an IP formulation of HCCSP. In Section 3, we introduce a decomposed version of this formulation. In Section 4, we present the specialised branching scheme used. Section 5 introduces clustering of visits and other methods to decrease the solve time of the pricing problem. Real-life and generated test instances are described in Section 6. In Section 7, we present results from test runs on these instances. Finally, in Section 8 we conclude on our work and suggest topics for future research.

2 Problem formulation

The set of caretakers is denoted $\mathcal{K}$, and the set of visits at the citizens is denoted $\mathcal{C} = \{1, \ldots, n-1\}$. For each visit $i \in \mathcal{C}$ a time window is defined as $[\alpha_i, \beta_i]$, where $\alpha_i \geq 0$ and $\beta_i \geq 0$ specify the earliest respectively latest possible start time of the visit. For algorithmic reasons, we introduce artificial visits 0^k and n^k as the *start visit* respectively *end visit* for caretaker $k \in \mathcal{K}$, and we define $\mathcal{N}^k = \mathcal{C} \cup \{0^k, n^k\}$ as the set of all potential visits for caretaker k. The duty length for each caretaker $k \in \mathcal{K}$ is given by the time window $[\alpha_{0^k}, \beta_{0^k}] = [\alpha_{n^k}, \beta_{n^k}]$, i.e. caretaker $k \in \mathcal{K}$ cannot start his or her duty before time $\alpha_{0^k} \geq 0$ and must have finished his or her last visit before time $\beta_{0^k} \geq 0$. The travel distance between a pair of visits (i, j) is given by the parameter s_{ij}^k. The parameter depends on $k \in \mathcal{K}$ as the caretakers use different means of transportation. If it is not possible to travel directly between visits i and j for caretaker k, then $s_{ij}^k = \infty$. We define $s_{ii}^k = \infty$, $\forall k \in \mathcal{K}, \forall i \in \mathcal{N}^k$. The parameter s_{ij}^k includes the duration (service time) of visit i. Travelling between any two visits i and j gives rise to the costs c_{ij}^k dependent on the caretaker $k \in \mathcal{K}$. For any combination of $i \in \mathcal{C}$ and $k \in \mathcal{K}$ the parameter $\rho_i^k = 1$ if k can be assigned to visit i, $\rho_i^k = 0$ otherwise. Also, for any combination of $i \in \mathcal{C}$ and $k \in \mathcal{K}$ the preference parameter $\delta_i^k \in \mathbb{R}$ gives the cost for letting caretaker k handle visit i. A negative cost



(a) Synchronisation. (b) Overlap. (c) Minimum difference.



(d) Maximum difference. (e) Min+max difference.

Figure 1: Five types of temporal dependencies. Each of the five subfigures shows the time windows of two visits i (top) and j (bottom) with a temporal dependency between them. Assuming some start time for visit i, the dotted line shows the earliest feasible start time for visit j, and the dashed dotted line shows the latest feasible start time. For synchronisation (a) the two lines coincide, and are drawn as one full line.

means that we would like caretaker k to handle visit i, whereas a positive cost means that we would prefer not to let caretaker k handle visit i. The parameter γ_i is the priority of visit $i \in \mathcal{C}$, the higher, the more important.

As described in Section 1, visits may be temporally dependent due to different home care needs. In order to make it easier for the manual planner to assign substitutes to the uncovered visits, it is required that visits, which have a temporal dependency to an uncovered visit, still respect the temporal dependency. In other words, a temporal dependency is still respected even if one of the visits is uncovered. Five types of temporal dependencies are often seen in practice. The five types can be seen in Figure 1. These temporal dependencies can be modelled by introducing generalised precedence constraints of the form

$$\sigma_i + p_{ij} \leq \sigma_j \ ,$$

where σ_i denotes the start time of visit i, and $p_{ij} \in \mathbb{R}$ quantifies the required gap. The set of pairs of visits $(i, j) \in \mathcal{C} \times \mathcal{C}$ for which a generalised precedence constraint exists is denoted $\mathcal{P}$.

As can be seen, this constraint simply implies that j starts minimum p_{ij} time units after i. An often encountered example of a temporal dependency is that of synchronisation, see Figure 1(a), where two visits are required to start at the same time. The way to handle this is to add both (i, j) and (j, i) to $\mathcal{P}$ with $p_{ij} = p_{ji} = 0$. As also described by Dohn et al. (2009b), Table 1 shows how to model all the temporal dependencies of Figure 1 with

generalised precedence constraints. It can be seen that (a), (b) and (e) each requires two generalised precedence constraints, whereas (c) and (d) only need one each.

	Temporal dependency	p_{ij}	p_{ji}
(a)	Synchronisation	0	0
(b)	Overlap	$-\text{dur}_j$	$-\text{dur}_i$
(c)	Minimum difference	$\text{diff}_{\min}$	N/A
(d)	Maximum difference	N/A	$-\text{diff}_{\max}$
(e)	Minimum+maximum difference	$\text{diff}_{\min}$	$-\text{diff}_{\max}$

Table 1: Values for p_{ij} for the five temporal dependencies of Figure 1. dur_i is the service time of visit i, $\text{diff}_{\min}$ is the minimum difference required and $\text{diff}_{\max}$ is the maximum difference required.

2.1 Integer programme

To model the problem, three sets of decision variables are defined: the binary routing variables x_{ij}^k, the scheduling variables $t_i^k \in \mathbb{Z}_+$ and the binary coverage variables y_i. $x_{ij}^k = 1$ if caretaker $k \in \mathcal{K}$ goes directly from visit $i \in \mathcal{N}^k$ to $j \in \mathcal{N}^k$, and $x_{ij}^k = 0$ otherwise. The scheduling variable t_i^k is the time the caretaker $k \in \mathcal{K}$ starts handling visit $i \in \mathcal{N}^k$. $t_i^k = 0$ if caretaker k is not assigned to visit i. $y_i = 1$ if visit $i \in \mathcal{C}$ is not covered by any caretaker and $y_i = 0$ otherwise. The weights w_1, w_2 and w_3 are used to control the objective function.

HCCSP can now be formulated as the integer programme given below. The formulation is very similar to the formulation in Bredström and Rönnqvist (2007).

$$\min w_1 \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}^k} \sum_{j\in\mathcal{N}^k} c_{ij}^k x_{ij}^k + w_2 \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{N}^k} \delta_i^k x_{ij}^k + \quad w_3 \sum_{i\in\mathcal{C}} \gamma_i y_i \tag{1}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{N}^k} x_{ij}^k + y_i = 1 \qquad \forall i \in \mathcal{C} \tag{2}$$

$$\sum_{j\in\mathcal{N}^k} x_{ij}^k \leq \rho_i^k \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{C} \tag{3}$$

$$\sum_{j\in\mathcal{N}^k} x_{0^k,j}^k = 1 \qquad \forall k \in \mathcal{K} \tag{4}$$

$$\sum_{i\in\mathcal{N}^k} x_{i,n^k}^k = 1 \qquad \forall k \in \mathcal{K} \tag{5}$$

$$\sum_{i\in\mathcal{N}^k} x_{ih}^k - \sum_{j\in\mathcal{N}^k} x_{hj}^k = 0 \qquad \forall k \in \mathcal{K}, \forall h \in \mathcal{C} \tag{6}$$

$$\alpha_i \sum_{j\in\mathcal{N}^k} x_{ij}^k \leq t_i^k \leq \beta_i \sum_{j\in\mathcal{N}^k} x_{ij}^k \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{C} \cup \{0^k\} \tag{7}$$

$$\alpha_{n^k} \leq t_{n^k}^k \leq \beta_{n^k} \qquad \forall k \in \mathcal{K} \tag{8}$$

$$t_i^k + s_{ij}^k x_{ij}^k \leq t_j^k + \beta_i (1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, \forall i, j \in \mathcal{N}^k \tag{9}$$

$$\alpha_i y_i + \sum_{k\in\mathcal{K}} t_i^k + p_{ij} \leq \sum_{k\in\mathcal{K}} t_j^k + \beta_j y_j \qquad \forall (i,j) \in \mathcal{P} \tag{10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall i, j \in \mathcal{N}^k \tag{11}$$

$$t_i^k \in \mathbb{Z}_+ \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}^k \tag{12}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{C} \tag{13}$$

The objective (1) is multi-criteria. Often, minimising uncovered visits (the third term) is prioritised over maximising caretaker-visit preferences (the second term), which again is prioritised over minimising the total travelling costs (the first term). This can be accomplished by setting $w_1 = 1$, $w_2 = \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}^k} \sum_{j\in\mathcal{N}^k} c_{ij}^k$ and $w_3 = w_2 |\mathcal{C}| \max_{k\in\mathcal{K}, i\in\mathcal{C}} \delta_i^k$. Constraints (2) ensure that each visit is covered exactly once or left uncovered, and caretakers can only handle allowed visits (3). Constraints (4)–(6) ensure that the caretakers begin at the start visit, end at the end visit, and that routes are not segmented. Constraints (7) and (8) control that time windows are respected. Furthermore, Constraints (7) set $t_i^k = 0$ when k does not visit i. Travelling distances are respected due to Constraints (9). Constraints (10) are the generalised precedence constraints. The y-variable terms ensure that generalised precedence constraints are respected even when visits are cancelled. Finally, Constraints (11)–(13) set the domains of the decision variables.

The HCCSP formulation can be seen as a generalisation of an uncapacitated, multiple-depot VRPTW. The aim is to push flow for each caretaker from start visit to end visit through as many profitable nodes as possible

while respecting time windows and minimising costs. Also, it is only allowed for one caretaker to go through each node.

The HCCSP generalises the Travelling Salesman Problem (TSP). TSP is well-known to be $\mathcal{NP}$-hard as its decision problem version is $\mathcal{NP}$-complete, see Problem ND22 in Garey and Johnson (1979). Therefore, also HCCSP is $\mathcal{NP}$-hard, and we can therefore not expect to solve the problem efficiently, i.e. in polynomial time. The $\mathcal{NP}$-hardness of the HCCSP is the reason why we develop a branch-and-price solution algorithm.

3 Decomposition

We will Dantzig-Wolfe decompose the HCCSP described in the previous section and model it as a set partitioning problem with side constraints. Then we will solve the model using dynamic column generation in a branch-and-price framework. This approach has presented superior results on VRPTW and the similarities to HCCSP are strong enough to suggest the same approach here. There is a vast amount of literature on column generation based solution methods for VRPTW, see e.g. Kallehauge et al. (2005) for a recent literature review and an introduction to the method. In a branch-and-price framework the problem is split into two problems, a master problem and a subproblem. The subproblem generates feasible caretaker schedules, which are then subsequently combined in a feasible way in the master problem. In the master problem, given a large set of feasible schedules to choose from, one schedule is chosen for each caretaker. Given a set of caretakers $\mathcal{K}$, each caretaker must choose a schedule from the set $\mathcal{R}^k$, which is the set of potential schedules for caretaker k. Together, the schedules must cover as many visits as possible from the set $\mathcal{C}$.

A feasible schedule r for a caretaker $k \in \mathcal{K}$ is defined as a route starting at 0^k and ending at n^k and respecting all constraints in the IP formulation from Section 2.1 which do not link multiple routes. The schedule includes the starting times of the visits. The parameter c_r^k gives the cost for caretaker $k \in \mathcal{K}$ for schedule $r \in \mathcal{R}^k$, and $c_i = w_3\gamma_i$ gives the cost for leaving visit $i \in \mathcal{C}$ uncovered. The binary parameter $a_{ir}^k = 1$ if visit i is included in schedule r for caretaker k and $a_{ir}^k = 0$ otherwise. Moreover, t_{ir}^k is the start time of visit i in schedule r for caretaker k, if i is included in r for k. If i is not included in r for k, $t_{ir}^k = 0$.

3.1 Master problem

We introduce the binary decision variable λ_r^k where $\lambda_r^k = 1$ if schedule $r \in \mathcal{R}^k$ is chosen for caretaker $k \in \mathcal{K}$, and $\lambda_r^k = 0$ otherwise. Furthermore, we introduce the binary decision variable Λ_i where $\Lambda_i = 1$ if visit $i \in \mathcal{C}$ is uncovered, and $\Lambda_i = 0$ otherwise. HCCSP can now be solved by finding a minimum cost combination of schedules such that all constraints are fulfilled.

The master problem of the Dantzig-Wolfe decomposition of HCCSP is given by the mathematical programme shown below.

$$\min \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} c_r^k \lambda_r^k + \sum_{i \in \mathcal{C}} c_i \Lambda_i \tag{14}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} a_{ir}^k \lambda_r^k + \Lambda_i = 1 \qquad \forall i \in \mathcal{C} \tag{15}$$

$$\sum_{r \in \mathcal{R}^k} \lambda_r^k = 1 \qquad \forall k \in \mathcal{K} \tag{16}$$

$$\alpha_i \Lambda_i + \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{ir}^k \lambda_r^k + p_{ij} \leq \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{jr}^k \lambda_r^k + \beta_j \Lambda_j \qquad \forall (i,j) \in \mathcal{P} \tag{17}$$

$$\lambda_r^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall r \in \mathcal{R}^k \tag{18}$$

$$\Lambda_i \in \{0,1\} \qquad \forall i \in \mathcal{C} \tag{19}$$

The total costs of the schedules plus the costs of leaving visits uncovered are minimised (14). The cost of a schedule contains the remaining components of the original objective and is therefore determined by the travel costs and by the service level of the visits in the schedule. Constraints (15) ensure that all visits are either included in exactly one schedule or considered uncovered. One schedule must be assigned to each caretaker (16), and the generalised precedence constraints must be respected (17). Again, the Λ-variable terms in Constraints (17) ensure that precedence constraints are respected even for uncovered visits. Integrality of the decision variables is ensured by Constraints (18) and (19). Any feasible solution to the decomposed problem is a feasible solution to the original problem, and any optimal solution to the decomposed problem is an optimal solution to the original problem.

To be able to solve the master problem in an LP-based branch-and-price framework, the integrality constraints on λ_r^k and Λ_i are relaxed. Also, the precedence constraints (17) are relaxed, as we thereby have no constraints interconnecting the starting times in the schedules of different caretakers. This gives a simpler pricing problem, which will be explained further in Section 3.2. The two relaxed constraints will be handled in the branching.

As the number of feasible schedules for each caretaker is enormous, the set $\mathcal{R}^k$ of schedules for caretaker $k \in \mathcal{K}$ is restricted to only contain a small subset $\mathcal{R}'^k$ of promising schedules, which will be generated by the column generating pricing problem. We abbreviate the relaxed and restricted master problem as RMP. For each primal solution to the RMP, we obtain a dual solution $[\pi, \omega]$, where π_i, $i \in \mathcal{C}$, and ω_k, $k \in \mathcal{K}$, are the dual variables of Constraints (15) and (16), respectively. The dual variables are used in the column generation technique to generate new schedules that lead to an improvement of the solution to the RMP.

3.2 Pricing problem

The pricing problem is used to find the feasible schedule with the most negative reduced cost (if any exists). As the caretakers have different working hours and competences, the pricing problem is split into $|\mathcal{K}|$ independent pricing problems. The pricing problem is an Elementary Shortest Path Problem with Time Windows (ESPPTW), which has been proved $\mathcal{NP}$-hard in Dror (1994). The pricing problem for a caretaker $k \in \mathcal{K}$ is formulated as an integer programme below. Any feasible solution to the pricing problem with negative cost represents a column with negative reduced cost in the RMP.

$$\min \sum_{i \in \tilde{\mathcal{N}}^k} \sum_{j \in \tilde{\mathcal{N}}^k} \left(w_1 c_{ij}^k + w_2 \delta_i^k - \pi_i \right) x_{ij} - \omega_k \tag{20}$$

$$\text{s.t.} \sum_{j \in \tilde{\mathcal{N}}^k} x_{0^k,j} = 1 \tag{21}$$

$$\sum_{i \in \tilde{\mathcal{N}}^k} x_{i,n^k} = 1 \tag{22}$$

$$\sum_{i \in \tilde{\mathcal{N}}^k} x_{ih} - \sum_{j \in \tilde{\mathcal{N}}^k} x_{hj} = 0 \qquad \forall h \in \mathcal{C}^k \tag{23}$$

$$\alpha_i \sum_{j \in \tilde{\mathcal{N}}^k} x_{ij} \le t_i \le \beta_i \sum_{j \in \tilde{\mathcal{N}}^k} x_{ij} \qquad \forall i \in \mathcal{C}^k \cup \{0^k\} \tag{24}$$

$$\alpha_{n^k} \le t_{n^k} \le \beta_{n^k} \tag{25}$$

$$t_i + s_{ij}^k x_{ij} \le t_j + \beta_i (1 - x_{ij}) \qquad \forall i, j \in \tilde{\mathcal{N}}^k \tag{26}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \tilde{\mathcal{N}}^k \tag{27}$$

$$t_i \in \mathbb{Z}_+ \qquad \forall i \in \tilde{\mathcal{N}}^k \tag{28}$$

Here, we have introduced the decision variables x_{ij} and t_i which are the same as in (1)–(13), without the k index. For a given $k \in \mathcal{K}$, the subset of visits $\mathcal{C}^k = \{i \in \mathcal{C} : \rho_i^k = 1\}$ is the set of visits allowed for k. Moreover, $\tilde{\mathcal{N}}^k = \mathcal{C}^k \cup \{0^k, n^k\}$, and we define $\delta_{0^k}^k = \delta_{n^k}^k = \pi_{0^k} = \pi_{n^k} = 0$.

The relatively simple expression (20) for the reduced costs of a column is one of the reasons why the generalised precedence constraints are relaxed. One could, as done in van den Akker et al. (2006) and in Dohn et al. (2009b) have kept the generalised precedence constraints in the RMP. This would have implied a more complicated pricing problem, as the pricing problem then incorporates a means of adjusting the starting times in a schedule based on the dual variables. In van den Akker et al. (2006) they do not solve their pricing problem by an exact method, but use a heuristic method. Benchmark results from Dohn et al. (2009b) show that in many cases it is as good to relax the generalised precedence constraints, as to keep them in the master problem.

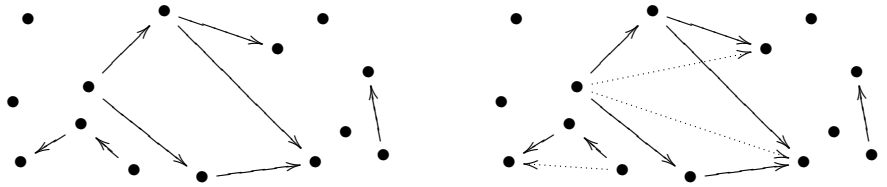
We solve the pricing problem with a labelling algorithm built on ideas from Chabrier (2006) and Feillet et al. (2004).

4 Branching

The generalised precedence constraints and the integrality constraints that were relaxed from the master problem are handled in the branching part of the branch-and-price algorithm. To handle both types of constraints, we need to present two branching methods. One to handle the violation of an integrality constraint and another to handle the violation of a precedence constraint. The branching scheme used to handle precedence constraint violations is a time window branching scheme. This also enforces integrality to a certain point as shown in Gélinas et al. (1995). Nonetheless, one cannot solely rely on time window branching to enforce integrality, so we use an additional branching scheme to force the solution to integrality. First, we will present a preprocessing technique.

4.1 Preprocessing of time windows

The visits $\mathcal{C}$ can be grouped according to how they are connected by generalised precedence constraints. Define the directed *temporal dependency graph* $G = (V, A)$ by $V = \mathcal{C}$ and $A = \mathcal{P}$. The graph G consists of one or more sub-graphs, which corresponds to the connected components in the undirected version of G. An example of such a graph is shown in Figure 2(a). From the existing generalised precedence constraints, additional



(a) Example of a temporal dependency graph.

(b) The same graph expanded with derived arcs.

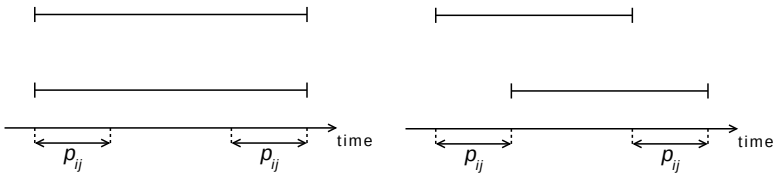
Figure 2: A temporal dependency graph.

derived generalised precedence constraints can be found. In every subgraph with three or more nodes, we look for triples $i, j, k \in \mathcal{C}$ where $(i, j), (j, k) \in \mathcal{P}$ with $i \neq j, j \neq k, k \neq i$. If $(i, k) \notin \mathcal{P}$, then the pair is added to $\mathcal{P}$ with $p_{ik} = p_{ij} + p_{jk}$. If $(i, k) \in \mathcal{P}$ the offset is updated to $p_{ik} := \max\{p_{ik}, p_{ij} + p_{jk}\}$ in order to get the tightest constraint. The process is repeated until no further constraints can be derived or tightened. The example will now look as in Figure 2(b). This derivation of generalised precedence constraints will

make it possible to reduce more time windows, as there will be a greater number of precedence constraints on which to perform the following pairwise reduction technique.

If two visits $i, j \in \mathcal{C}$ are connected via a (possibly derived) generalised precedence constraint $(i, j) \in \mathcal{P}$, it might be possible to tighten the time windows of i and j, such that $[\alpha'_i, \beta'_i] = [\alpha_i, \min\{\beta_i, \beta_j - p_{ij}\}]$ and $[\alpha'_j, \beta'_j] = [\max\{\alpha_j, \alpha_i + p_{ij}\}, \beta_j]$ are the new time windows as illustrated in Figure 3.

This preprocessing is repeated until no time windows are tightened. The preprocessing technique can be used in every node of the branch-and-bound tree. It should be noted that this time window reduction can only be carried out, because it is required that also temporal dependencies with cancelled visits must be respected. If this was not the case, then the cancellation of a visit i with $(i, j) \in \mathcal{P}$ would lead to the time window of j being "reset" (assuming it was previously reduced by preprocessing).



(a) Before preprocessing. (b) After preprocessing.

Figure 3: Adjustment of time windows in accordance to a generalised precedence constraint. Each of the subfigures shows the time windows of two visits i (top) and j (bottom).

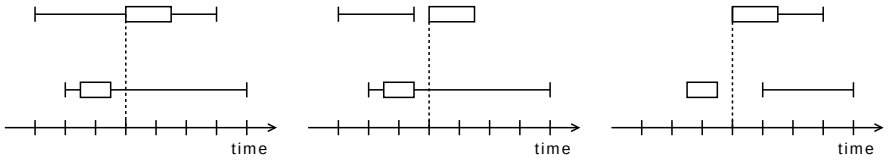
4.2 Branching on generalised precedence constraints

A generalised precedence constraint $(i, j) \in \mathcal{P}$ is violated if there exists positive variables $\lambda^{k_1}_{r_1} > 0$ and $\lambda^{k_2}_{r_2} > 0$ (the relaxation allows for $k_1 = k_2$ and $r_1 = r_2$, but we will prevent that in the subproblem) in the solution to the RMP for which

$$t^{k_1}_{i,r_1} + p_{ij} \not\leq t^{k_2}_{j,r_2} \ .$$

Therefore, to remedy this, we will alter the time windows in the branches. In the left branch the time window of visit i is set to $[\alpha_i, t_{\text{split}} - 1]$, where t_{split} is the split time. In the right branch the time window of visit i is set to $[t_{\text{split}}, \beta_i]$. The preprocessing technique described in the previous section is used again, which will result in the time window of visit j in the right branch being changed to $[t_{\text{split}} + p_{ij}, \beta_j]$. All previously generated schedules violating these new time windows are removed. The split time is selected such that $t^{k_2}_{j,r_2} - p_{ij} + 1 \leq t_{\text{split}} \leq t^{k_1}_{i,r_1}$. Hence, the branching scheme divides the

solution space into two sets, where the solution that violates the precedence constraint for (i, j) becomes infeasible in each of them. Synchronisation constraints are often seen in home care instances. The branching scheme suggested here combined with preprocessing of time windows is as strong as the scheme tailored for synchronisation described in Ioachim et al. (1999). This is elaborated in Dohn et al. (2009b), where it is also described how to pick the best split time in the given interval. An illustration of the generalised precedence constraint branching scheme can be found in Figure 4.



(a) Parent node. (b) Left child node. (c) Right child node.

Figure 4: Example of the branching applied when a generalised precedence constraint is violated. Each of the subfigures shows the time windows of two visits i (top) and j (bottom), and the start times of the visits in an RMP solution. The violated constraint for (i, j) has $p_{ij} = 2$. The dotted line shows the chosen split time, and the distance between the ticks on the time line is two time units.

4.3 Integer branching

In the following, we will let A^k denote the $|\mathcal{C}| \times |\mathcal{R}'^k|$-matrix where the elements are given by the parameter a^k_{ir} for a given caretaker $k \in \mathcal{K}$, i.e. each column in A^k represents a schedule $r \in \mathcal{R}'^k$. Now, consider the structure of the constraint matrix of the RMP which is shown in Figure 5. For clarity, we only show ones of the constraint matrix and introduce $m = |\mathcal{K}|$ and $\bar{n} = n - 1$.

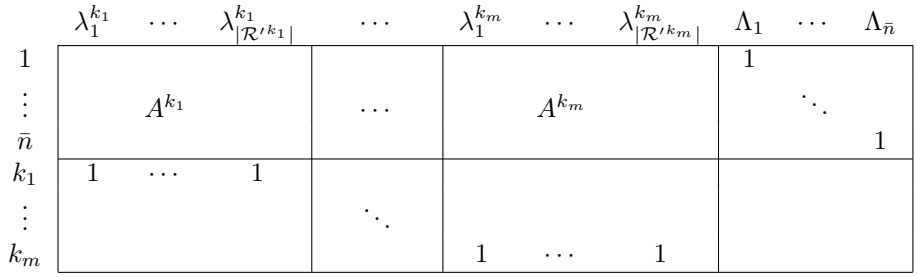


Figure 5: Constraint matrix for the RMP.

We observe that the RMP has strong integer properties due to the generalised upper bound constraints (16) for each caretaker, see e.g. Rezanova and Ryan (2010) for further details and references. That is, for all caretakers $k \in \mathcal{K}$, their submatrix of the constraint matrix is perfect. Consequently, fractionality in the LP solutions will never appear within one caretaker's block of schedules. Any fractions in the RMP can therefore only occur between blocks of columns, belonging to different caretakers. Hence, if the LP solution is fractional, it is because two or more caretakers compete for one or more visits in their schedules. Let $i \in \mathcal{C}$ denote a visit for which caretaker $k \in \mathcal{K}$ is competing with one or more other caretakers. Since the visit can only be handled by one caretaker, then in an integer solution either k handles i or k does *not* handle i.

We will exploit the strong integer properties of the constraint matrix of the RMP to apply a so-called constraint branching strategy, see Ryan and Foster (1981). We introduce the sum $S_i^k = \sum_{r \in \mathcal{R}'^k} a_{ir}^k \lambda_r^k$. If a fractional solution occurs, the constraint branching strategy is now to find a visit-caretaker pair (i, k) of a visit $i \in \mathcal{C}$ and a caretaker $k \in \mathcal{K}$ for which $0 < S_i^k < 1$. In the 1-branch visit i is forced to be handled by k and in the 0-branch prohibit visit i from being handled by k. Notice that since at least one of the unique λ_r^k is fractional then at least one sum S_i^k is also fractional. This can be shown by a contradiction argument, see Dohn and Kolind (2006).

Whenever the sum S_i^k is fractional for two or more visit-caretaker pairs (i, k), we have to select one of these as the candidate for branching in the node. If S_i^k is close to 1, forcing $S_i^k = 1$ will probably not change the solution drastically, so only a small increase in the lower bound can be expected in this branch. On the other hand, as the LP solution suggests that caretaker k should handle i in an optimal solution, ruling out this option ($S_i^k = 0$) is likely to cause a major increase in the lower bound. If S_i^k is close to 0, a similar line of reasoning also shows a skewed branching. In order to keep the branch-and-bound tree balanced, we select the "most fractional" candidate, i.e. the candidate closest to one half. More formally, we select $(i^*, k^*) = \arg\min_{(i,k) \in \mathcal{C} \times \mathcal{K} : 0 < S_i^k < 1} \left| S_i^k - \frac{1}{2} \right|$.

5 Clustering of visits and arc removal

The HCCSP exhibits a structural feature that can be used to group or *cluster* visits. HCCSP has, as opposed to VRPTW, a preference parameter for each caretaker-visit combination. Moreover, test runs have suggested that the ESPPTW solver is a bottleneck in the branch-and-price algorithm. Therefore, we have developed schemes that reduce the sizes of the ESPPTW networks, which will in turn decrease the running time of the algorithm. For some larger instances, visit clustering is even needed to find feasible

solutions.

When no reduction of the ESPPTW networks is used, every caretaker $k \in \mathcal{K}$ can visit every $i \in \mathcal{C}$ where $\rho_i^k = 1$. However, in a good solution (assuming the objective function weights are set as suggested in Section 2.1), a caretaker k will only handle visits i where δ_i^k is favourable. Therefore, we have devised three ways of clustering visits for a caretaker according to the preference parameter δ_i^k, thereby effectively reducing the network sizes for each caretaker. Again, let $\mathcal{C}^k = \{i \in \mathcal{C} : \rho_i^k = 1\}$. All schemes operate with a cluster of visits $\bar{\mathcal{C}}^k \subseteq \mathcal{C}^k$ for a caretaker. The first scheme only puts visits in the cluster, when it is directly profitable, i.e. $\bar{\mathcal{C}}^k = \{i \in \mathcal{C}^k : \delta_i^k < 0\}$.

In the second scheme preference parameters for caretaker k are ordered non-decreasingly as $\delta_{i_1}^k \leq \cdots \leq \delta_{i_\xi}^k \leq \cdots \leq \delta_{i_{|\mathcal{C}^k|}}^k$ with ties broken arbitrarily. Define the set $\Delta_\xi^k = \{\delta_{i_1}^k, \ldots, \delta_{i_\xi}^k\}$ given the parameter ξ. The second scheme then defines the cluster as $\bar{\mathcal{C}}^k = \{i \in \mathcal{C}^k : \delta_i^k \in \Delta_\xi^k\}$. The cluster contains the ξ most profitable visits.

The two first clustering schemes do not guarantee that all visits are in a cluster. Therefore, all remaining visits $i \in \mathcal{C} \setminus \bigcup_{k \in \mathcal{K}} \bar{\mathcal{C}}^k$ are added to all clusters.

The third scheme seeks to exploit the integer properties of the problem described in Section 4.3. If the caretakers cannot compete for visits, the LP solutions will be naturally integer, and hence the run times will decrease significantly. Therefore, we make the visit clusters pairwise disjoint, i.e. $\forall k_1, k_2 \in \mathcal{K}, k_1 \neq k_2 : \bar{\mathcal{C}}^{k_1} \cap \bar{\mathcal{C}}^{k_2} = \emptyset$. Again, the preference parameters for each caretaker are ordered non-decreasingly. Hereafter, the scheme iterates over the caretakers in a round-robin fashion and puts the most profitable visit in the caretaker's cluster (if it is not already put in another caretaker's cluster). Suppose visit j is already in $\bar{\mathcal{C}}^k$ for caretaker k, then there are two conditions under which another visit i is not permitted in the cluster. If i cannot be carried out before j, and also j cannot be carried out before i, then the visits can never be scheduled in the same route. This is detected whenever $\alpha_i + s_{ij}^k > \beta_j \wedge \alpha_j + s_{ji}^k > \beta_i$. The second condition is when there is a temporal dependency, which disallows any route with both visits. This is the case when $(i,j),(j,i) \in \mathcal{P}$ and $-s_{ji}^k < p_{ij} \wedge -s_{ij}^k < p_{ji}$.

The use of clustering will sacrifice optimality, and later we will look into how big the gap to optimality is, and compare it against the benefit of improved run time. The closest to this idea we have seen in the VRP literature is the *petal method*, see e.g. Foster and Ryan (1976), which clusters the visits based on geographical position.

5.1 Expansion of visit clusters

The clustering of visits can lead to visits being uncovered not because it is optimal, but due to the clustering. Hopefully, these are only a very few

visits. In order to remedy this, the clusters are made dynamic, in the sense that expansion of the clusters is allowed. For any branch-and-bound node, uncovered visits can be detected, by looking at the LP optimal solution. If there are uncovered visits, they are added to all clusters, and the LP problem is solved again. We suggest two versions of the cluster expansion. Either cluster expansion can happen only in the root node, or it can happen in any node of the branch-and-bound tree. Especially the latter adds a twist of unpredictability (though still deterministic) to the problem, because the problem basically can be changed anywhere in the branch-and-bound tree. It can happen that the lower bound for a child is lower than the lower bound for its parent, which is avoided when expansion is only allowed in the root node.

5.2 Removal of idle arcs

We will here present another method to reduce the network sizes. The time where the caretaker is neither visiting a citizen nor travelling is called *idle time*. This is time where the caretaker is basically just waiting for the time window of the next visit to open. Therefore, another means to reduce the sizes of the ESPPTW networks, is to remove arcs where the minimum idle time $\phi_{ij}^k = \alpha_j - (\beta_i + s_{ij}^k)$ between two visits is high. Again, proof of optimality is sacrificed, but in a good solution, we probably would not see the use of many arcs with large idle time.

6 Test instances

We have had access to four authentic test instances from two Danish municipalities. These are named hh, ll1, ll2 and ll3. Based on the authentic instances we have generated 60 extra instances. These instances are constructed by generating new sets of generalised precedence constraints for each of the four authentic instances, while still keeping the original sets of caretakers and visits and original travelling times. The new generalised precedence constraint sets are based on the five types of temporal dependencies from Figure 1, and we have created five sets named td0–4. The generalised precedence constraints in the set td0 are of the temporal dependency type synchronisation (a). The set td1 is of the type overlap (b). The set td2 consists of the types minimum difference (c) and maximum difference (d). When values are drawn randomly, these categories collapse to one. The set td3 is of type minimum+maximum difference (e). Finally, td4 is a random combination of the other four types. Three sets of generalised precedence constraints were generated for each of these fives sets: Sets A, B, and C, where the number of generalised precedence constraints approximately is, respectively, 10%, 20%, and 30% of the number of visits. The sets of generalised precedence constraints were generated as in Dohn et al.

(2009b). Characteristics for these test instances can be seen in Table 2. The notation td[0-4] expands to td0, td1, td2, td3, and td4. It is compacted in the table, because the instances share the same characteristics.

Furthermore, Bredström and Rönnqvist (2007) have very kindly given us access to their 30 data instances. These data instances are generated based on realistic settings and only contain synchronisation constraints. All visits have the same priority, and no visits are excluded for any of the caretakers, i.e. $\rho_i^k = 1, \forall i \in \mathcal{C}, \forall k \in \mathcal{K}$. The preference parameter δ_i^k is drawn randomly between -10 and 10. For all of the instances, all caretakers in that instance have the same duty hours. Characteristics for these test instances can also be seen in Table 2. Again, br-[06-08][S,M,L]-td0 means that for each of br-06, br-07, and br-08, there are three instances: One with small (S) time windows, one with medium (M) time windows, and one with large (L) time windows.

	hh	ll1	ll2	ll3	hh-td[0-4]-A	hh-td[0-4]-B	hh-td[0-4]-C	ll1-td[0-4]-A	ll1-td[0-4]-B	ll1-td[0-4]-C	ll2-td[0-4]-A	ll2-td[0-4]-B	ll2-td[0-4]-C	ll3-td[0-4]-A	ll3-td[0-4]-B	ll3-td[0-4]-C	br-[01-05][S,M,L]-td0	br-[06-08][S,M,L]-td0	br-[09-10][S,M,L]-td0
$\lvert\mathcal{K}\rvert$	15	8	7	6	15	15	15	8	8	8	7	7	7	6	6	6	4	10	16
$\lvert\mathcal{C}\rvert$	150	107	60	61	150	150	150	107	107	107	60	60	60	61	61	61	20	50	80
$\lvert\mathcal{P}\rvert$	6	0	0	0	16	30	46	10	22	32	6	12	18	6	12	18	4	10	16

Table 2: Characteristics for the test instances. $|\mathcal{K}|$ is number of caretakers, $|\mathcal{C}|$ is number of visits and $|\mathcal{P}|$ is number of generalised precedence constraints.

7 Computational results

The aim in this section is to compare the different visit clustering techniques presented in Section 5. We will also try to measure the effects of removal of idle time arcs and cluster expansion. Using clustering will sacrifice optimality, and we will here investigate how big the gap to optimality is, and compare it against the benefit of improved run time.

We measure three quality parameters, which are also the terms of the objective function: uncovered visits, caretaker-visit preferences, and total travel costs. The weights of the objective function are set as suggested in Section 2.1, so a hierarchical ordering is obtained. We seek to minimise the number of uncovered visits and maximise the preference level of the solution. The total travel costs are measured in minutes for all caretakers for the whole daily schedule. We subtract the durations of the visits in the total

travel time, hence giving preference to longer visits, and thereby maximising the so-called face-to-face time. More formally, we define the travel cost as $c_{ij}^k = s_{ij}^k - 2 \cdot \text{dur}_i$. Hence, if it were only possible to cover either visit i or the two visits j and h, coverage of visit i is preferred, whenever $\gamma_i \geq \gamma_j + \gamma_h$ and $\text{dur}_i > \text{dur}_j + \text{dur}_h$, assuming the travel time for both options is the same. Minimising the total travel costs are not as important as minimising the two other measurements, but low travel costs are naturally preferred. In order to be able to make comparisons this third measure is ignored, when we are performing tests on the instances from Bredström and Rönnqvist (2007).

The algorithm is implemented in the branch-and-cut-and-price framework from COIN-OR, see Lougee-Heimer (2003), using the COIN-OR open-source LP solver CLP. All tests are run on 2.2 GHz processors. As an outcome of preliminary tests, we return up to five negative reduced cost columns per caretaker per iteration. For all of the test runs we have set a time out limit of one hour. The implementation of the ESPPTW solver ensures that generalised precedence constraints, that make two visits mutually exclusive, are respected within the individual routes. This tightens the lower bounds and reduces the number of branch-and-bound nodes.

We have grouped the instances into 35 test groups based on their size, the type of temporal dependency included, and the number of temporal dependencies. The test groups can be seen in Table 3. For each of these groups, 13 different settings for the algorithm are compared. The settings are written as CS-RA-ER, abbreviating *clustering scheme*, *removal of arcs*, and *expansion in root only*, respectively. CS = 0 corresponds to no use of visit clustering. CS = 1 gives all-preferred clusters, i.e. clusters for caretaker k where $\delta_i^k < 0$ for all visits i in the cluster, as described in Section 5. CS = 2 gives fixed-size clusters of a fixed size ξ. CS = 3 gives pair-wise disjoint clusters. Before the preference parameters are sorted they are shuffled randomly in order to make the tie-breaking arbitrary. The setting RA is a binary parameter, which is 1, if we remove arcs based on idle time, and 0 otherwise. The setting ER is also a binary parameter, which is 1, if we only allow cluster expansion in the root node of the branch-and-bound tree, and 0 if cluster expansion is allowed in every node of the branch-and-bound tree. Thus, the settings 0-0-0 (as well as the redundant settings 0-0-1) give the optimal solution. However, some of the instances are not possible to solve to optimality within the time limit, they can only be solved using clustering. For CS = 2, we use a fixed cluster size $\xi = 12$. With a fixed cluster size of 12, the pricing problems (at least initially, before cluster expansion) are solved very fast. When arcs are removed, the largest allowed minimum idle time ϕ_{ij}^k is set to 10 minutes, both based on preliminary tests. The abbreviation BR is used when we show results from Bredström and Rönnqvist (2007).

In Table 3 the comparison is shown. The numbers are averages over all instances in the test group. In total, we have 1190 test runs, which gives a good statistical foundation. Let T and Z denote the run time and the

objective value, respectively, of a given test run, and let T_{ref} and Z_{ref} denote the run time and the objective value of a reference solution, respectively. The time difference is then calculated as T/T_{ref} in percent, and the objective gap is calculated as $|(Z - Z_{\text{ref}})/Z_{\text{ref}}|$ in percent. When the objective is better than the reference objective a minus sign is added. The reference settings are the leftmost, in most cases it will be the settings 0-0-0, which is a very intuitive reference. All the instances in the test groups [hh,ll1] and [hh,ll1]-td[0-4]-[A,B,C] have not been possible to solve to optimality. Any attempt has, when the one hour time out limit is reached, ended up with around 70% or more of the visits uncovered in the best solution in the branch-and-bound tree. Therefore, the reference settings for these test groups will be 1-0-0. When using relative gaps for comparison, one should be careful, because relative gaps are highly dependent on the objective measure. In our case we have a very high penalty on uncovered visits, so a single uncovered visit as opposed to no uncovered visits would lead to a large gap. Also, for all instances based on ll1, there will be eight visits that are impossible to cover, as they cannot be completed within the working hours of any of the caretakers. This fixed cost for all generated routes makes the gaps smaller.

As mentioned earlier, the dynamic expansion of visit clusters makes the algorithm behave somewhat unpredictable. In the cases where we see the time difference being close to 100% and the objective gap at the same time being close to 0%, it is very likely that the clusters are expanded to nearly the entire set $\mathcal{C}^k$, thereby getting close to CS = 0. On the other hand, when the gap is small and the time difference significantly below 100%, then a good initial clustering is used. With regard to the time-quality trade-off, the all-preferred (CS = 1) and the fixed-size (CS = 2) clustering schemes both have instance groups where they are performing best. If dynamic cluster expansion was not used, then it would be expected that the fixed-size clustering scheme would be the fastest on larger instances (e.g. instances based on hh or ll1), as the cluster size is kept small. This does not happen, though, due to the clusters being expanded aggressively. The aggressive expansion happens when a lot of visits are uncovered and therefore added to every caretaker's cluster. For the hh and ll1 instance groups, we therefore see that the fixed-size clustering is slower, but better than the all-preferred clustering. In some test runs with the fixed-size clustering scheme in the test groups [ll2,ll3]-td0-A and [ll2,ll3]-td3-B, the initial clustering has lead to very large branch-and-bound trees. This is visible in the averages. For the pair-wise disjoint clustering scheme the picture is more clear. As expected it is very fast, but it does not come without a price, as the solution qualities for this scheme generally are the worst.

Focusing on the impact of removal of idle time arcs (RA), Table 3 does not disclose much. It it very hard to find a pattern in the impact of this setting. Removal of idle time arcs may reduce the ESPPTW networks, but the removal could also lead to more visits being uncovered in the LP

Group	Settings 0-0-0	1-0-0	1-0-1	1-1-0	1-1-1	2-0-0	2-0-1	2-1-0	2-1-1	3-0-0	3-0-1	3-1-0	3-1-1	BR
							Time difference (%)							
br-[01-05]	100	39	96	48	52	47	54	60	57	6	4	6	5	192
br-[06-08]	100	60	58	48	55	11	20	13	57	1	1	1	1	157
br-[09-10]	100	81	81	87	87	21	50	11	27	1	1	1	0	144
[hh,ll1]	N/A	100	87	70	106	105	104	48	45	8	5	5	4	N/A
[hh,ll1]-td0-A	N/A	100	246	99	322	1395	1384	1872	1770	77	35	66	36	N/A
[hh,ll1]-td0-B	N/A	100	44	102	43	540	1362	160	1235	30	23	30	18	N/A
[hh,ll1]-td0-C	N/A	100	45	72	88	266	645	168	971	63	41	56	42	N/A
[hh,ll1]-td1-A	N/A	100	100	74	74	138	155	110	235	47	45	43	38	N/A
[hh,ll1]-td1-B	N/A	100	78	89	75	161	126	123	133	40	34	38	33	N/A
[hh,ll1]-td1-C	N/A	100	89	97	109	101	123	89	104	56	37	48	29	N/A
[hh,ll1]-td2-A	N/A	100	97	89	80	29	29	39	36	5	5	5	6	N/A
[hh,ll1]-td2-B	N/A	100	2788	216	253	121	124	100	91	15	15	15	13	N/A
[hh,ll1]-td2-C	N/A	100	115	107	848	211	173	140	133	81	52	75	58	N/A
[hh,ll1]-td3-A	N/A	100	43	91	90	510	440	258	247	5	4	6	5	N/A
[hh,ll1]-td3-B	N/A	100	58	201	177	15	17	17	16	3	2	2	1	N/A
[hh,ll1]-td3-C	N/A	100	104	104	104	11	8	8	5	1	1	2	1	N/A
[hh,ll1]-td4-A	N/A	100	61	50	176	103	106	71	64	18	11	19	12	N/A
[hh,ll1]-td4-B	N/A	100	207	67	119	351	351	106	90	8	5	11	7	N/A
[hh,ll1]-td4-C	N/A	100	99	16	60	102	102	15	24	4	3	4	2	N/A
[ll2,ll3]	100	15	15	17	19	8	29	6	29	0	0	0	0	N/A
[ll2,ll3]-td0-A	100	14	16	37	71	2014	2014	1310	2014	1	1	1	1	N/A
[ll2,ll3]-td0-B	100	40	38	42	42	1	1	1	1	0	0	0	0	N/A
[ll2,ll3]-td0-C	100	31	29	30	29	3	3	3	3	2	1	2	2	N/A
[ll2,ll3]-td1-A	100	98	98	20	24	19	24	11	16	0	0	0	0	N/A
[ll2,ll3]-td1-B	100	65	63	21	34	20	4	5	5	0	0	0	0	N/A
[ll2,ll3]-td1-C	100	94	94	91	91	1	1	1	0	0	0	0	0	N/A
[ll2,ll3]-td2-A	100	12	12	11	11	1	17	1	11	0	0	0	0	N/A
[ll2,ll3]-td2-B	100	12	12	83	77	98	98	98	94	0	0	0	0	N/A
[ll2,ll3]-td2-C	100	7	8	11	10	66	62	79	76	1	1	1	1	N/A
[ll2,ll3]-td3-A	100	97	97	89	97	1	97	1	96	0	0	0	0	N/A
[ll2,ll3]-td3-B	100	21	22	12	12	940	1971	463	1970	0	0	1	1	N/A
[ll2,ll3]-td3-C	100	97	97	97	97	1	1	0	0	0	0	0	0	N/A
[ll2,ll3]-td4-A	100	99	98	102	102	79	98	45	11	0	0	0	0	N/A
[ll2,ll3]-td4-B	100	99	99	82	79	0	0	0	0	0	0	0	0	N/A
[ll2,ll3]-td4-C	100	93	93	50	92	0	0	0	0	0	0	0	0	N/A
Avg. (0-0-0)	100	56	59	52	57	175	239	111	235	1	1	1	1	N/A
Avg. (1-0-0)	N/A	100	180	115	157	704	884	480	884	14	10	13	10	N/A
							Objective gap (%)							
br-[01-05]	0	204	203	205	205	3	3	5	4	806	937	1070	1201	0
br-[06-08]	0	0	0	0	0	121	147	93	147	524	747	526	776	0
br-[09-10]	0	0	0	-1	-1	155	179	158	179	289	914	292	914	6
[hh,ll1]	N/A	0	26	26	26	0	0	3	3	75	83	83	90	N/A
[hh,ll1]-td0-A	N/A	0	4	-6	4	-20	-13	-19	-11	7	32	7	32	N/A
[hh,ll1]-td0-B	N/A	0	18	7	18	-7	0	-5	2	23	44	23	54	N/A
[hh,ll1]-td0-C	N/A	0	21	10	24	3	7	-1	6	15	40	32	53	N/A
[hh,ll1]-td1-A	N/A	0	0	-2	-2	-29	-2	-27	0	-4	4	6	13	N/A
[hh,ll1]-td1-B	N/A	0	19	0	28	-9	2	-9	6	15	34	17	36	N/A
[hh,ll1]-td1-C	N/A	0	20	0	20	-14	16	-10	12	20	35	20	35	N/A
[hh,ll1]-td2-A	N/A	0	2	4	4	15	15	-8	-8	14	14	46	46	N/A
[hh,ll1]-td2-B	N/A	0	2	6	6	-4	-2	-2	0	31	33	39	39	N/A
[hh,ll1]-td2-C	N/A	0	2	6	6	-6	2	-4	4	11	36	15	42	N/A
[hh,ll1]-td3-A	N/A	0	9	2	9	0	0	-2	-2	59	61	61	63	N/A
[hh,ll1]-td3-B	N/A	0	0	2	2	0	0	2	2	58	63	60	72	N/A
[hh,ll1]-td3-C	N/A	0	18	2	20	2	5	2	5	11	16	20	27	N/A
[hh,ll1]-td4-A	N/A	0	5	-5	5	12	5	-2	5	34	43	34	46	N/A
[hh,ll1]-td4-B	N/A	0	13	0	11	2	11	2	11	37	59	39	63	N/A
[hh,ll1]-td4-C	N/A	0	16	-2	10	2	8	0	2	18	54	18	66	N/A
[ll2,ll3]	0	0	0	0	0	456	570	456	570	1710	1710	3990	3990	N/A
[ll2,ll3]-td0-A	0	0	0	0	0	174	174	0	35	626	626	764	764	N/A
[ll2,ll3]-td0-B	0	0	0	0	0	246	246	246	246	369	390	635	717	N/A
[ll2,ll3]-td0-C	0	0	0	0	0	153	153	153	153	170	204	255	408	N/A
[ll2,ll3]-td1-A	0	0	0	0	0	456	570	456	570	1597	1597	2052	2052	N/A
[ll2,ll3]-td1-B	0	0	0	0	0	343	570	115	570	1711	2508	1597	3078	N/A
[ll2,ll3]-td1-C	0	0	0	0	0	229	570	570	798	1483	2053	1711	2167	N/A
[ll2,ll3]-td2-A	0	0	0	0	0	456	570	0	114	2622	2622	5015	5015	N/A
[ll2,ll3]-td2-B	0	0	0	0	0	103	103	21	21	185	185	451	451	N/A
[ll2,ll3]-td2-C	0	0	0	0	0	68	68	0	0	442	442	664	664	N/A
[ll2,ll3]-td3-A	0	0	0	0	0	0	114	456	570	2622	2622	3534	3534	N/A
[ll2,ll3]-td3-B	0	0	0	0	0	1	114	1	114	2508	2508	2964	2964	N/A
[ll2,ll3]-td3-C	0	0	0	0	0	1	1	1	1	1711	1711	2395	2395	N/A
[ll2,ll3]-td4-A	0	0	0	0	0	266	266	266	266	1012	1012	2077	2077	N/A
[ll2,ll3]-td4-B	0	0	0	0	0	266	266	266	266	1119	1119	1970	1970	N/A
[ll2,ll3]-td4-C	0	0	0	0	0	266	266	266	266	959	959	2343	2343	N/A
Avg. (0-0-0)	0	11	11	11	11	198	261	186	257	1182	1309	1806	1973	N/A
Avg. (1-0-0)	N/A	0	5	1	5	100	137	93	135	647	722	988	1086	N/A

Table 3: Comparison of different settings for the algorithm for the test groups. Settings are written as CS-RA-ER. Test groups br-[x-x][S,M,L]-td0 are shortened to br-[x-x]. The average 'Avg. (0-0-0)' uses the settings 0-0-0 as reference settings and does not include the test groups [hh,ll1] and [hh,ll1]-td[0-4]-[A,B,C], as test runs are not carried out in these groups with the settings 0-0-0. The average 'Avg. (1-0-0)' uses the settings 1-0-0 as reference settings and includes all test groups, but not the settings 0-0-0.

solution and therefore added to all caretaker's clusters. This would increase the network sizes.

The table shows that when cluster expansion is allowed in every node in the branch-and-bound tree (ER = 0), the solution quality tends to be just better than when expansion is only allowed in the root node (ER = 1). This is expected, but still, if many visits are uncovered in the root LP solution, this could lead to large clusters, and thereby better solution quality.

Looking at the numbers for the test groups ending with A, B, and C, there does not seem to be a correlation between the performance of the different clustering schemes and the number of generalised precedence constraints for an instance. None of the clustering schemes stand out with a consequently good or bad performance in either size A, B, or C. Likewise, there does not seem to be a correlation between the type of temporal dependency and the performance of the schemes. This is also sensible, as the clustering is preference-based and as such independent of types and numbers of temporal dependencies.

If we compare our results against the results from Bredström and Rönnqvist (2007), we are significantly faster in all test groups. We are able to verify their optimal solution values for the groups br-[01-05][S,M,L]-td0 and br-[06-08][S,M,L]-td0, and we are able to improve the best known solution values for the group br-[09-10][S,M,L]-td0 by 6% on average. For some instances we can prove optimality of the improved solutions. The settings 1-1-0 and 1-1-1 give better solution quality on average for the group br-[09-10][S,M,L]-td0 than the setting 0-0-0. This is possible, because we reach the time out limit on some test runs, and therefore the returned solution is not necessarily optimal, but only the best solution in the branch-and-bound tree at time out.

Table 4 shows detailed statistics for individual test runs. The test runs shown here are chosen, because they are representative for the numbers from Table 3. It should be noted, that we have integerised the preference parameter in the instances from Bredström and Rönnqvist (2007), by scaling it with a factor of 10^4. In the process some rounding took place, and furthermore the results reported in Bredström and Rönnqvist (2007) are rounded. Therefore, reported numbers for the settings 0-0-0 and BR in Table 4 will not necessarily match on all digits. Also, one should keep in mind that our lower bounds are tighter.

The detailed statistics for the test runs show that there is a clear connection between the run times and the sizes of the branch-and-bound trees, which is intuitively very sensible and expected. It should here be noticed that Bredström and Rönnqvist (2007) use significantly fewer branch-and-bound nodes than our algorithm. This is most probably due to their branching candidate selection which seems to perform very well.

Overall it can be said, that, as expected, run times can be decreased by using visit clustering, but this implies a decreased solution quality. It

Test case	Settings	Root LP value	Objective value	Uncovered visits	B&B nodes	B&B depth	Subproblems solved	Generated columns	LP solve time (s)	Subproblem time (s)	Running time (s)
br-05S-td0	0-0-0	-76277.00	-76277	0	11	4	284	687	0.31	0.17	0.60
br-05S-td0	1-0-0	-71289.00	923612	1	3	1	64	124	0.02	0.02	0.05
br-05S-td0	1-1-0	-71289.00	923612	1	3	1	88	154	0.04	0.02	0.07
br-05S-td0	1-0-1	-71289.00	923612	1	3	1	64	124	0.02	0.03	0.05
br-05S-td0	1-1-1	-71289.00	923612	1	3	1	88	154	0.03	0.02	0.07
br-05S-td0	2-0-0	-76277.00	-76277	0	9	3	200	436	0.12	0.08	0.26
br-05S-td0	2-1-0	-76277.00	-76277	0	9	3	152	366	0.11	0.06	0.22
br-05S-td0	2-0-1	-76277.00	-76277	0	9	3	200	436	0.14	0.06	0.28
br-05S-td0	2-1-1	-76277.00	-76277	0	9	3	152	366	0.12	0.06	0.22
br-05S-td0	3-0-0	933849.00	933849	1	1	0	16	38	0.00	0.00	0.01
br-05S-td0	3-1-0	933849.00	933849	1	1	0	16	34	0.00	0.00	0.01
br-05S-td0	3-0-1	933849.00	933849	1	1	0	16	38	0.00	0.00	0.02
br-05S-td0	3-1-1	933849.00	933849	1	1	0	16	34	0.00	0.01	0.02
br-05S-td0	BR	-76290.00	-76290	0	1	-	139	-	-	-	0.64
br-06M-td0	0-0-0	-380509.00	-379854	0	353	33	8989	8941	54.91	35.10	107.18
br-06M-td0	1-0-0	-380509.00	-379854	0	419	58	10284	8359	34.44	26.10	72.35
br-06M-td0	1-1-0	-379287.00	-378589	0	431	48	10904	8148	32.17	26.71	70.64
br-06M-td0	1-0-1	-380509.00	-379854	0	419	58	10284	8359	34.38	26.32	72.53
br-06M-td0	1-1-1	-379287.00	-378589	0	431	48	10904	8148	32.17	26.49	69.67
br-06M-td0	2-0-0	-376764.00	649853	1	77	38	1910	1399	2.67	4.16	8.22
br-06M-td0	2-1-0	-374594.00	-332648	0	109	54	2520	1701	4.04	5.23	11.37
br-06M-td0	2-0-1	-376764.00	641531	1	91	40	2240	1642	2.83	4.49	8.95
br-06M-td0	2-1-1	-374594.00	653339	1	177	43	4070	2254	5.11	7.72	15.67
br-06M-td0	3-0-0	-362005.00	686191	1	51	25	1060	530	0.22	1.73	2.42
br-06M-td0	3-1-0	-352443.00	-301188	0	33	16	840	464	0.14	1.41	1.87
br-06M-td0	3-0-1	-362005.00	1662244	2	47	13	880	436	0.20	1.31	1.81
br-06M-td0	3-1-1	-352443.00	3680521	4	31	15	520	282	0.08	0.81	1.12
br-06M-td0	BR	-386860.00	-379880	0	101	-	1861	-	-	-	247.88
hh	1-0-0	6851842.00	6851850	5	187	21	16755	12032	27.74	587.64	639.90
hh	1-1-0	6851859.00	6851867	5	141	15	11910	9090	17.65	422.74	458.90
hh	1-0-1	6851842.00	6851850	5	171	19	15915	11629	22.73	517.76	564.61
hh	1-1-1	6851859.00	6851867	5	139	15	11640	8792	19.51	613.39	694.34
hh	2-0-0	6858829.00	6858843	5	167	21	16890	20070	42.30	549.17	617.44
hh	2-1-0	7860857.00	7860868	6	121	21	6630	6896	17.90	235.45	266.05
hh	2-0-1	6858829.00	6858843	5	167	21	16305	19558	43.97	569.96	639.66
hh	2-1-1	7860857.00	7860868	6	115	21	5880	6012	13.21	186.02	209.15
hh	3-0-0	11869075.00	10870068	9	23	11	1650	1594	1.06	45.22	48.64
hh	3-1-0	12871112.00	11872103	10	21	10	1080	1012	0.48	29.91	31.94
hh	3-0-1	11869075.00	13869078	10	21	10	1140	1223	0.60	29.90	32.29
hh	3-1-1	12871112.00	14871115	11	19	9	810	855	0.36	22.95	24.50
ll1-td1-B	1-0-0	36920146.00	37926295	17	21	10	952	1487	2.05	31.07	35.56
ll1-td1-B	1-1-0	36920146.00	37926294	17	17	8	992	1494	2.08	23.22	26.91
ll1-td1-B	1-0-1	36920146.00	44921229	18	25	12	792	1128	1.43	21.31	24.26
ll1-td1-B	1-1-1	36920146.00	48924236	19	27	13	888	1230	1.80	18.80	22.20
ll1-td1-B	2-0-0	33245315.00	35937179	15	65	32	2216	3688	33.05	86.91	128.25
ll1-td1-B	2-1-0	33245315.00	34937306	17	51	25	1856	3524	28.04	60.63	95.23
ll1-td1-B	2-0-1	33245315.00	40932305	17	39	19	1448	2962	21.55	65.10	94.11
ll1-td1-B	2-1-1	33245315.00	41932341	18	73	36	2104	3535	32.74	65.87	106.49
ll1-td1-B	3-0-0	38767254.00	39934260	16	17	8	856	1074	0.88	15.47	17.73
ll1-td1-B	3-1-0	38767255.33	39934260	16	21	10	816	1035	1.02	13.13	15.56
ll1-td1-B	3-0-1	38767254.00	48931257	19	19	9	552	825	0.53	8.86	10.31
ll1-td1-B	3-1-1	38767255.33	48932218	19	25	12	576	721	0.58	8.39	9.96
ll2-td4-C	0-0-0	940394.00	940400	1	341	22	15393	29336	204.53	103.45	333.50
ll2-td4-C	1-0-0	940394.00	940400	1	153	14	6279	8202	29.95	19.74	56.44
ll2-td4-C	1-1-0	940398.75	940400	1	27	7	938	1488	4.19	3.10	8.33
ll2-td4-C	1-0-1	940394.00	940400	1	153	14	6202	8114	29.61	19.24	55.40
ll2-td4-C	1-1-1	940398.75	940400	1	27	7	854	1331	3.88	2.83	7.72
ll2-td4-C	2-0-0	940412.00	4941459	2	3	1	203	582	0.31	0.79	1.26
ll2-td4-C	2-1-0	940415.00	4941423	2	3	1	203	553	0.23	0.70	1.09
ll2-td4-C	2-0-1	940412.00	4941459	2	3	1	203	582	0.31	0.79	1.25
ll2-td4-C	2-1-1	940415.00	4941423	2	3	1	203	553	0.25	0.67	1.09
ll2-td4-C	3-0-0	3949532.00	3949532	4	1	0	84	233	0.03	0.22	0.30
ll2-td4-C	3-1-0	25951558.00	25951558	8	1	0	56	152	0.01	0.14	0.21
ll2-td4-C	3-0-1	3949532.00	3949532	4	1	0	84	233	0.02	0.23	0.31
ll2-td4-C	3-1-1	25951558.00	25951558	8	1	0	56	152	0.02	0.15	0.22

Table 4: Key statistics for selected test runs. Settings are written as CS-RA-ER.

is difficult to point out the best settings as well as to quantify the speed gain/quality loss trade-off. As mentioned earlier, it is seen that the pair-wise disjoint clustering is by no doubt the fastest, but if quality is also taken into account, the all-preferred clustering scheme tends to perform best. Settings with CS = 1 have at least equally good and in most cases much better run times when compared to the settings 0-0-0 and do only have a significant loss in quality for the test group br-[01-05][S,M,L]-td0. The loss in quality is due to three out of 15 instances in the group having a single uncovered visit in the solutions with CS = 1. Focusing on the all-preferred clustering scheme, it seems to be slightly better for the solution quality to allow cluster expansion in every node of the branch-and-bound tree.

Lastly, it should be mentioned that we have also compared our solutions of hh, ll1, ll2, and ll3 with the current practice. Current practice is based partly on an automated heuristic and partly on manual planning. Unfortunately, it is not straight forward to make a comparison. It is clearly indicated, though, that we are able to enhance the service level. There is a significant decrease in the number of uncovered visits and a truly dramatic decrease in the number of necessary constraint adjustments. Constraint adjustments are another way of dealing with an uncovered visit, so that it is possible to fit the visit into the schedule anyway. Possible options are to: reduce the duration of the visit, extend the time window of the visit or extend the work shift of one of the caretakers. This is done a lot in practice, and it makes comparison very difficult. However, any of these adjustments will naturally decrease the overall quality of the schedule. In the presented solution method, we have chosen to keep all the original constraints intact, and let the constraint adjustment be a manual post-processing task. This decision is supported by the fact that it is hard to put a quantitative penalty on all possible adjustments before solving.

8 Conclusion and future work

Initiated by the method's successful use in the VRPTW context, we have formulated the Home Care Crew Scheduling Problem as a set partitioning problem with side constraints and developed a branch-and-price solution algorithm. All temporal dependencies are modelled as generalised precedence constraints, and these constraints are enforced through the branching. To our knowledge, we are the first to enforce generalised precedence constraints in the branching for real-life problems. Based on the preference parameters, we have devised different visit clustering schemes. The visit clustering schemes for the exact branch-and-price framework are novel. We have compared the visit clustering schemes in order to survey how much they decrease run times, and how much they compromise optimality. The visit clustering schemes have been tested both on real-life problem instances and

on generated test instances inspired by realistic settings. The tests have shown that by using clusters with only preferred visits, run times were significantly decreased, while there was only a loss of quality for few instances. The clustering schemes have allowed us to find solutions to instances that could not be solved to optimality. Summarised, our main contributions are: Development of visit clustering schemes for the Home Care Crew Scheduling Problem, and enforcement of generalised precedence constraints in the branching for real-life problems.

We see a number of directions in which future work on this problem could go. One direction is improvement of the algorithm presented in this paper. New visit clustering schemes could be devised accompanied by cluster expansion schemes. For the clustering scheme with a fixed cluster size, it could be interesting to look into what determines a good cluster size for a given instance. It might be possible to express the cluster size as a function of number of visits and number of caretakers.

Other very interesting and yet unexplored planning problems in home care are long-term planning and disruption management. In the long-term planning problem, the goal is to present a plan that spans e.g. half a year. The long-term problem does not decide how the visits should be assigned to the specific caretakers, but only how to distribute the visits optimally on the weekdays and possibly in time windows.

In a disruption management or recovery situation the original plan has become infeasible due to unforeseen circumstances. Therefore, rescheduling of the caretakers for the remains of the planning period (most likely the rest of the day) must take place. The goal of the rescheduling is to provide a new, feasible plan very fast, i.e. within minutes, with as few alterations to the original plan as possible. In many cases the disruption will only directly influence a smaller subset of the caretakers, and an approach could be inspired by what Rezanova and Ryan (2010) do for train driver rescheduling.

References

S. Begur, D. Miller, and J. Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35–48, 1997.

S. Bertels and T. Fahle. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers and Operations Research*, 33(10):2866–2890, 2006.

D. Bredström and M. Rönnqvist. A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints. Technical report, Department of Finance and Management Science, Norwegian School of Economics and Business Administration, 2007.

D. Bredström and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19–31, 2008.

A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers and Operations Research*, 33(10):2972–2990, 2006.

E. Cheng and J. L. Rich. A home health care routing and scheduling problem. Technical report, Department of CAAM, Rice University, 1998.

J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. Vrp with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7, pages 176–213. Society for Industrial and Applied Mathematics, 2002.

A. Dohn and E. Kolind. A practical branch and price approach to the crew scheduling problem with time windows. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2006.

A. Dohn, E. Kolind, and J. Clausen. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers and Operations Research*, 36(4):1145–1157, 2009a.

A. Dohn, M. S. Rasmussen, and J. Larsen. The vehicle routing problem with time windows and temporal dependencies. Technical report, Department of Management Engineering, Technical University of Denmark, 2009b.

M. Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–8, 1994.

P. Eveborn, P. Flisberg, and M. Rönnqvist. Laps care—an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962–976, 2006.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44(3):216–29, 2004.

B. A. Foster and D. M. Ryan. An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly*, 27(2):367–384, 1976.

M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., New York, 1979.

S. Gélinas, M. Desrochers, J. Desrosiers, and M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119(1):75–90, 1999.

B. Kallehauge, J. Larsen, O. B. Madsen, and M. Solomon. The vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, GERAD 25th anniversary series, chapter 3, pages 67–98. Springer, New York, 2005.

C. R. Lessel. Ruteplanlægning i hjemmeplejen. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2007.

Y. Li, A. Lim, and B. Rodrigues. Manpower allocation with time windows and job-teaming constraints. *Naval Research Logistics*, 52(4):302–311, 2005.

A. Lim, B. Rodrigues, and L. Song. Manpower allocation with time windows. *Journal of the Operational Research Society*, 55(11):1178–1186, 2004.

R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

N. J. Rezanova and D. M. Ryan. The train driver recovery problem-a set partitioning based model and solution method. *Computers and Operations Research*, 37(5):845–856, 2010.

D. M. Ryan and B. Foster. An integer programming approach to scheduling. *Computer Scheduling of Public Transport. Urban Passenger Vehicle and Crew Scheduling. Proceedings of an International Workshop*, pages 269–280, 1981.

K. Thomsen. Optimization on home care. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2006.

J. van den Akker, J. Hoogeveen, and J. van Kempen. Parallel machine scheduling through column generation: Minimax objective functions. *Lecture Notes in Computer Science*, 4168:648–659, 2006.

In the Home Care Crew Scheduling Problem a staff of caretakers has to be assigned a number of visits to patients' homes, such that the overall service level is maximised. The problem is a generalisation of the vehicle routing problem with time windows. Required travel time between visits and time windows of the visits must be respected. The challenge when assigning visits to caretakers lies in the existence of soft preference constraints and in temporal dependencies between the start times of visits.

We model the problem as a set partitioning problem with side constraints and develop an exact branch-and-price solution algorithm, as this method has previously given solid results for classical vehicle routing problems. Temporal dependencies are modelled as generalised precedence constraints and enforced through the branching. We introduce a novel visit clustering approach based on the soft preference constraints. The algorithm is tested both on real-life problem instances and on generated test instances inspired by realistic settings. The use of the specialised branching scheme on real-life problems is novel. The visit clustering decreases run times significantly, and only gives a loss of quality for few instances. Furthermore, the visit clustering allows us to find solutions to larger problem instances, which cannot be solved to optimality.

ISBN 978-87-90855-80-2

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk