GENERAL ARTICLE

THE VERSION OF RECORD OF THIS MANUSCRIPT HAS BEEN PUBLISHED AND IS AVAILABLE IN JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY ¡DATE OF PUBLICATION¿ HTTP://WWW.TANDFONLINE.COM/10.1080/01605682.2020.1792365

A Multistage Optimisation Algorithm for the Large Vehicle Routing Problem with Time Windows and Synchronised Visits

Mateusz Polnik[a] and Annalisa Riccardi[a] and Kerem Akartunalı[b]

[a]Mechanical and Aerospace Engineering, University of Strathclyde, Glasgow, UK;
[b]Management Science, University of Strathclyde, Glasgow, UK

ARTICLE HISTORY
Compiled July 14, 2020

Abstract
We propose a multistage algorithm for the Vehicle Routing Problem with Time Windows and Synchronised Visits, which is capable of solving large problem instances arising in Home Health Care applications. The algorithm is based on a Constraint Programming formulation of the daily Home Care Scheduling and Routing Problem. It contains visits with hard time windows and pairwise synchronisation to be staffed by carers who have different skills and work custom shift patterns with contractual breaks. In a computational study, we first experiment with a benchmark set from the literature for the Vehicle Routing Problem with Time Windows and Synchronised Visits. Our algorithm reproduced the majority of the best-known solutions, and strictly improved results for several other instances. Most importantly, we demonstrate that the algorithm can effectively solve real scheduling instances obtained from a UK home care provider. Their size significantly surpass similar scheduling problems considered in the literature. The multistage algorithm solved each of these instances in a matter of minutes, and outperformed human planners, scheduling more visits and significantly reducing total travel time.

KEYWORDS
Vehicle Routing; Scheduling; Constraint Programming; Health Services; Practice of OR

CONTACT Mateusz Polnik. Email: mateusz.polnik@strath.ac.uk; ORCID: 0000-0002-2788-0947
Annalisa Riccardi. ORCID: 0000-0001-5305-9450;
Kerem Akartunalı. ORCID: 0000-0003-0169-3833;

1. Introduction

The Vehicle Routing Problem with Time Windows and Synchronised Visits (VRPTWSyn) is about a fleet of vehicles that are supposed to visit a group of geographically distributed customers. Every visit has a predefined time window for its start and some prescribed duration. No violation of the time window constraints is allowed. Some visits require two vehicles, and they cannot commence until both vehicles arrive. Each vehicle is initially located in a depot, to which the vehicle returns following the last visit. The objective of the problem is to find the assignment of visits to specific vehicles and the routes each vehicle follows while optimising some decision criterion, i.e., to minimise the total travel time for all vehicles.

Relying on the apparent analogy between vehicles and home care workers, we consider a practical application of the VRPTWSyn for daily scheduling and routing in Home Healthcare (HHC). Therefore, we extend the baseline VRPTWSyn problem to capture the following relevant features in the application domain. Carers start and end their routes at their homes. They work in sequences of time intervals, referred to as shift patterns. Periods between the time intervals are contractual breaks. Similar to visits, contractual breaks have time windows, last a fixed duration, and cannot be interrupted. A visit consists of a set of tasks to be executed by a carer. Every task requires a different skill. Only carers who possess relevant skills to complete all tasks can be assigned to the visit. A client may have several consecutive visits throughout the day. The number of different carers who visit a given client is restricted to provide the continuity of care. Finally, the number of requested visits may exceed the capacity of the system. Thus, a visit can be declined subject to a penalty. We refer to the problem as the daily Home Care Scheduling and Routing Problem (HCSRP). Its objective is to find an assignment of visits to carers as well as routes carers should follow to minimise the total distance travelled and the penalty incurred for declined visits.

1.1. Related Work

We focus on daily (single-period) routing and scheduling in HHC. Particular attention is devoted to the modelling of breaks and the treatment of pairwise synchronisation between visits, i.e., visits required to perform at the same time, which is a central feature of the VRPTWSyn. Let us remark that dispatching more vehicles for a visit is the subject of different problems, notably the Manpower Allocation Problem (MAP) (Li et al., 2005) or Technician Routing and Scheduling (TRS) (Kovacs et al., 2012). Aspects related to the problem decomposition (Laesanklang and Landa-Silva, 2017), districting (Benzarti et al., 2013), the assignment of a patient to a health care profes-

sional (Hertz and Lahrichi, 2009), scheduling over multiple days (Nickel et al., 2012), and the treatment of uncertainty (Ehmke et al., 2015; Cappanera et al., 2018) are outside of the scope of this work. For a thorough description of OR in HHC, we refer the interested reader to recent literature reviews of Cissé et al. (2017) and Fikar and Hirsch (2017). Both authors provide a reference of notable contributions in the field, suggest a classification of the optimisation problems considered, briefly explain the solution methods, and attempt to predict future research directions.

Mathematical models for scheduling problems that contain a routing component are derivatives of Integer Programming (IP) models for the Vehicle Routing Problem (VRP) rather than personnel scheduling (Beck et al., 2003). The dominating theme in the literature is to model personnel scheduling and the routing problem using either a set partitioning formulation with side constraints (Dohn et al., 2011; Rasmussen et al., 2012) or a network flow model (Bredström and Rönnqvist, 2008; Cappanera and Scutellà, 2015; Castillo-Salazar et al., 2016; Thomsen, 2006; Trautsamwieser and Hirsch, 2011). The former model is usually solved with the branch-and-cut-and-price method. In practice, however, instances with 100 or more visits frequently arise, which are too computationally expensive for IP solvers (Castillo-Salazar et al., 2016). Problem instances with more than 150 visits are effectively intractable (Paraskevopoulos et al., 2017). Practitioners agreeably remark that short computational times are critical, mainly due to the need for computing several alternative schedules for decision makers (Bertels and Fahle, 2006; Eveborn et al., 2006; Trautsamwieser and Hirsch, 2011) and the dynamic nature of the environment resulting in last-minute cancellation of visits. This has motivated the development of hybrid approaches employing a range of methods, including, among others, an arc insertion heuristic (Thomsen, 2006), hyper-heuristics (Mısır et al., 2015), Constraint Programming (CP) (Bertels and Fahle, 2006; Nickel et al., 2012; Rahimian et al., 2017a), IP (Bredström and Rönnqvist, 2008; Rahimian et al., 2017b), Variable Neighbourhood Search (VNS) (Rahimian et al., 2017b; Trautsamwieser et al., 2011), Adaptive Large Neighbourhood Search (ALNS) (Nickel et al., 2012), Guided Local Search (GLS) (Voudouris and Tsang, 1999), Genetic Algorithm (GA) (Decerle et al., 2018), Simulated Annealing (SA) (Afifi et al., 2016), Particle Swarm Optimisation (PSO) (Mutingi and Mbohwa, 2014) and Tabu Search (TS) (Bertels and Fahle, 2006; Thomsen, 2006).

The need to coordinate actions performed by vehicles is a source of additional computational and modelling challenges. Routing problems in which vehicles do not act independently were studied by Drexl (2012) who coined the term Vehicle Routing Problem with Multiple Synchronisation Constraints (VRPMS). The author proposed a unified taxonomy of arbitrary dependencies apparent in routing problems and com-

piled an extensive reference of practical applications. Drexl (2012) considers coupling constraints restricting the start time of visits as "synchronisation of operations". Commencing a visit simultaneously by two vehicles prevalent in HHC (e.g., to assist a person with reduced mobility (Rasmussen et al., 2012)) is called an "exact operation". Following Drexl (2012), other examples of the synchronisation of operations are temporal precedence dependencies and pure spacial operations (i.e., actions that need to be performed once in the scheduling horizon).

Cheng and Rich (1998) proposed one of the first formulations of single-period scheduling in HHC. Their goal was to reduce the cost of overtime and part-time contractors needed to staff visits with time windows constraints and skill requirements. Restricting eligible vehicles to serve a customer due to skill considerations is known as the Skill Vehicle Routing Problem (Skill-VRP) (Cappanera et al., 2011). Skill requirements are sometimes complemented or replaced by preferences, which measure satisfaction from assigning a carer to a given visit. Contrary to skills, not respecting preferences does not lead to infeasible schedules but has a detrimental impact on the cost of the solution. Scheduling in HHC is considered an example of the Vehicle Routing Problem with Resource Constraints (VRPRC) (Paraskevopoulos et al., 2017). Other notable representatives of this class are TRS (Kovacs et al., 2012) and MAP (Li et al., 2005).

Fikar and Hirsch (2017), Cissé et al. (2017) and Duque et al. (2015) provide recent surveys of single and multi-period scheduling in HHC. The authors organised the reviewed articles based on the features considered in the formulation of the constraints, the components included in the objective function, and the solution methods applied. Predictably, the features of the HCSRP we solve in this paper have been studied by other researchers. However, the treatment of breaks in the literature and the formulations supporting the synchronisation of visits have some noteworthy limitations which we explained below.

Thompson and Pullman (2007) considered breaks as visits, for which a carer should return to a depot. However, this is only justifiable for long breaks in urban areas, where travel times to reach the depot are short. Kergosien et al. (2009) reserved specific time slots in carers' schedules as not available for work. Nevertheless, it is inflexible for scenarios, where waiting time preceding a visit could be accommodated as a break. Nickel et al. (2012) avoided scheduling breaks by using shifts shorter than the period after which labour law enforces a mandatory break, (e.g., 6 hours in Austria and Germany). Finally, Trautsamwieser et al. (2011) proposed a more general approach, although limited to one lunch break per carer. The researchers used a network flow-based formulation, where visits and breaks were two separate classes of nodes. Break nodes could only be reached from visit nodes, and a carer must return immediately to

the preceding visit node from which the break node was reached to continue the tour. This way carers retained their location in the network. Liu et al. (2017) developed a similar formulation also restricted to a single lunch break supported by a branch-and-price framework which could solve instances up to 100 visits to optimality. Overall, the techniques applied to the modelling of breaks impose simplifying assumptions, such as fixed start time, return to a depot, no more than one break allowed. Consequently, they do not seem well-posed for multiple breaks and in particular for short (relief) breaks. The technique of modelling breaks we adopted is free of such assumptions. We allow for an arbitrary number of breaks which do not require travelling and whose start times could be flexibly adjusted within predefined time windows.

Besides the need to model working hours and breaks, HHC are further complicated by visits with synchronisation constraints. Table 1 provides an overview of the literature considering visits which require simultaneous presence of two vehicles. The majority of the papers considered were covered in literature reviews (Cissé et al., 2017; Drexl, 2012; Fikar and Hirsch, 2017). We decided not to include works related to the MAP (Dohn et al., 2009; Li et al., 2005; Lim et al., 2004) and TRS (Kovacs et al., 2012) because they consider visits with synchronisation of more than two workers, and it was not mentioned precisely how many visits required multiple staff. For the same reason, we did not include (Eveborn et al., 2006), who presented a construction heuristic for home care scheduling and routing. For each reviewed article, we mention the problem discussed in the paper, the solution method, and some features in the problem formulation that are vital for HHC applications. Among them, we distinguish shift patterns and contractual breaks, the continuity of care, and the assignment of carers to visits based on skills or preferences.

All articles in Table 1 model visits with hard time windows. The authors who focused on the VRPTWSyn, which is a generic problem, do not include HHC specific features. Of the papers analysed, only Bachouch et al. (2011) considered a scheduling horizon longer than a single day, and it is also the only work that supports synchronised visits, the continuity of care, and lunch breaks. Contrary to shift patterns with contractual breaks and the continuity of care, either skills or preferences are commonly present in problem formulations.

Overall, the largest instances with synchronised visits solved in the literature are much smaller compared to the scheduling problems presented in HHC applications that consider exclusively single carer visits (Fikar and Hirsch, 2017), i.e., 500-700 visits, see (Duque et al., 2015; Hiermann et al., 2015; Trautsamwieser and Hirsch, 2011). Our use case, supported by the requirements of a home care organisation, demonstrates that solution methods which can solve problem instances of similar size, albeit with pairwise synchronisation, are needed in the real world, and it is the primary

Table 1.: Overview of literature on VRPTWSyn. Columns of the table report: the source article, the class of the problem, the solution method, support for specific features in the model, and the size of the largest problem instance solved. Among the features relevant to HHC applications, we distinguish shift patterns and contractual breaks (B), the continuity of care (CC), and the assignment of carers to visits based on skills or preferences (SP). The support for breaks may be limited to one lunch break (L) or offering full flexibility in the number of breaks and their time windows (Y). The continuity of care indicates if there is a restriction on the number of different carers who visit a given client (Y). The support for skills and preferences can include only skills (S), only preferences (P), or both (SP). For the largest instance size solved, we report the number of carers (C), the number of visits that require one carer (V^1), and the number of visits with pairwise synchronisation constraints (V^2). The table is sorted in the ascending order of the number of synchronised visits.

Source	Problem	Solution Method	Features			Largest Instance		
			B	CC	SP	C	V^1	V^2
Kergosien et al. (2009)	HCSRP	MIP	-	-	S	40	32	8
Bredström and Rönnqvist (2007)	VRPTWSyn	Branch and Price	-	-	P	16	64	16
Afifi et al. (2016)	VRPTWSyn	Heuristic + LS + LS	-	-	P	16	64	16
Parragh and Doerner (2018)	VRPTWSyn	LNS	-	-	-	16	64	16
En-nahli et al. (2016)	HCSRP	LNS	-	-	-	16	64	16
Frifita et al. (2017)	HCSRP	VNS	-	-	-	16	64	16
Decerle et al. (2018)	HCSRP	GA	-	-	-	16	64	16
Bredström and Rönnqvist (2008)	VRPTWSyn	MIP-driven heuristic	-	-	P	9	0	18
Mutingi and Mbohwa (2014)	HCSRP	GA, TS, SA, PSO	-	-	S	5	0	20
Bachouch et al. (2011)	HCSRP	Commercial Product	L	Y	S	5	80	20
Liu et al. (2019)	VRPTWSyn	LNS	-	-	P	50	160	40
Rasmussen et al. (2012)	HCSRP	Branch and Price	-	-	SP	15	104	46
Mankowska et al. (2014)	HCSRP	Heuristic + LNS	-	-	S	40	100	200
This Paper	HCSRP	Heuristic + LNS + GLS	Y	Y	S	63	388	240

motivation for this work.

1.2. Contributions of the Paper

The main contributions of the paper are:

(1) We present a CP formulation for the VRPTWSyn and its extension to model the daily HCSRP problem. The latter formulation provides a broad set of features crucial to the HHC domain, i.e., skill matching, the continuity of care, and shift patterns with an arbitrary number of breaks, which may have time

windows. Consequently, the solutions of the formulation can be fairly compared with human planners whose schedules account for those constraints.

(2) The multistage optimisation algorithm we propose can solve both formulations. The algorithm is competitive with other solution methods presented in the VRPTWSyn literature; it reproduced the majority of the best-known results for the benchmark set (Bredström and Rönnqvist, 2007) and strictly improved solutions for five instances. Furthermore, the multistage algorithm is capable of solving real instances of the daily HCSRP with synchronised visits, which are considerably larger than those presented in the literature before.

Finally, we emphasise that the utility of the contributions is not limited to our specific application, e.g., Paraskevopoulos et al. (2017) remark that studies in the whole class of the Vehicle Routing Problem with Resource Constraints (VRPRC) seldom consider working regulations. In the case study, we model shift patterns with contractual breaks adopted by a real HC organisation in the UK and show it does not have a detrimental impact on the ability to solve large HCSRP problem instances.

1.3. Organisation of the Paper

The next section presents CP formulations for the VRPTWSyn and the HCSRP, respectively. The formulations, albeit concise, are difficult to solve for large problem instances by stand-alone CP solvers. Therefore, we introduce in Section 3 a multistage optimisation algorithm. Section 4 details computational results demonstrating the utility of the proposed solution approach. Concluding remarks are presented in Section 5.

2. Constraint Programming Formulation

Before presenting the formulations, we introduce the notation and describe relevant global constraints which will make the CP models for the problems we consider more concise and faster to solve.

2.1. Notation

We adopt the following notation. Capital letters written using regular font represent sets. Bold lower-case letters denote vectors. We use the vector notation for index-based access to the content of a vector, i.e., v_i is the value of the element at the i-th position in the vector $\mathbf{v}$. Symbols $\mathbb{Z}$ and $\mathbb{B}$ denote integer and binary numbers, respectively. Given some integer constant n, the syntax $\mathbb{Z}_{0\leq n}$ represents a set of integer numbers

between 0 and n with the boundary values included. Let bool$(\cdot)$ be an operator which accepts a logical expression and converts it into a binary decision variable. Finally, let us define a generic placeholder operator $\iota_n(\cdots)$ for some integer n. The operator accepts an arbitrary number of input arguments and returns the element passed at the n-th position in the argument list.

2.2. Global Constraints

Global constraints are common modelling structures apparent in many Discrete Optimisation problems, i.e., packing constraints. They act as abstract interfaces to algorithms which propagate bounds for integer variables faster than an equivalent reformulation of the given constraint into a system of mixed-integer inequalities.

We employ the global constraints explained below. In their definitions, we assume $\mathbf{x}$ is a vector of some prescribed size s which contains integer variables that represent routes. The value of the variable x_i indicates the next node visited after the node i. A route terminates at the node i if the value assigned to the variable x_i is outside the bounds of the vector $\mathbf{x}$.

NO-CYCLE($\mathbf{x}$) - No cycles are present in the routes encoded by the vector $\mathbf{x}$.
ALL-DIFFERENT($\mathbf{x}$) - Elements of the vector $\mathbf{x}$ are assigned different values.
AT-MOST($\mathbf{y}, z, f$) - No more than f elements of the vector $\mathbf{y} \in \mathbb{Z}^s$ are assigned value z.
PATH-CUMUL($\mathbf{x}, \mathbf{y}, \gamma$) - The constraint computes the quantity of some arbitrary resource acquired along routes encoded by the vector $\mathbf{x}$. The amount of the resource available at every node is saved in the vector $\mathbf{y} \in \mathbb{Z}^s$. The function $\gamma : [0, \ldots, s-1] \times \mathbb{Z} \to \mathbb{Z}$ computes the marginal change in the resource acquired following a visit of a given node and the current quantity of the resource. The PATH-CUMUL constraint can be used to model the progression of time along a route, i.e., the start time of a succeeding visit is greater or equal to the sum of the start time of the previous visit, its duration, and the travel time to the succeeding visit.
MEMBER(x, S) - Value assigned to the integer variable x is an element of the set S.

2.3. Vehicle Routing Problem with Synchronised Visits

We present a multi-depot formulation for the VRPTWSyn which is obtained by adding the support for visits with synchronisation constraints to the generic CP model for the Vehicle Routing Problem with Time Windows (VRPTW) proposed by Perron (2011).

Let V be the set of visits. A visit $v \in V$ can commence within the time window

$[\underline{s}_v, \overline{s}_v]$ and has duration d_v. Every regular visit with a synchronisation constraint is represented by multiple elements in the set V. Their number equals the number of requested vehicles. To query visits which require the same number of vehicles, we define a generic set V^i for some integer i as the subset of V restricted to elements representing visits which require exactly i vehicles to arrive. Furthermore, we define M^i to be the partition of the vector V^i in which each subset contains elements representing the same visit.

Let C be the set of vehicles. Every vehicle c has an individual depot. We represent it using separate nodes b_c and e_c that correspond to the start depot and the terminal depot, respectively. To derive a more straightforward and generic formulation, we consider depots as visits. Consequently, for every vehicle c, the interval $[\underline{s}_{b_c}, \overline{s}_{b_c}] = [\underline{s}_{e_c}, \overline{s}_{e_c}]$ indicates the working hours of the vehicle. The duration of the visit at a depot is zero, i.e., $d_{b_c} = d_{e_c} = 0$.

Consider a complete graph $G(N = V \cup \bigcup_{c \in C}\{b_c, e_c\}, E)$ where N is the set of nodes, and E is the set of edges. The function $\delta : N \times N \to \mathbb{Z}_{\geq 0}$ computes the travel time between two nodes. It is convenient to wrap it with the function $\tau(n_i, n_j) = d_{n_i} + \delta(n_i, n_j)$ which computes marginal increase in time needed to complete the visit n_i and arrive to the node n_j.

We define the following decision variables. For each node $n \in N$, x_n contains the node succeeding the node n on a route of some vehicle. Furthermore, for all nodes $n \in N$, the variable s_n stores the start time if the node represents a visit, the departure time if the node corresponds to a start depot, and the arrival time if the node is a terminal depot. Similarly, the variable w_n indicates the vehicle visiting the node n.

Without loss of generality, we assume that sets C and N represent vehicles and nodes as integer indices which will be used to access decision variables stored in vectors. With these necessary prerequisites, we propose the following formulation.

$$\min \sum_{n \in N} \delta(n, x_n) \tag{1}$$

$$\text{s.t.} \quad \text{ALL-DIFFERENT}(\mathbf{x}) \tag{2}$$

$$\text{NO-CYCLE}(\mathbf{x}) \tag{3}$$

$$\text{PATH-CUMUL}(\mathbf{x}, \mathbf{s}, \tau) \tag{4}$$

$$\text{PATH-CUMUL}(\mathbf{x}, \mathbf{w}, \iota_2) \tag{5}$$

$$s_{v_i} = s_{v_j} \qquad \forall (v_i, v_j) \in M^2 \tag{6}$$

$$w_{v_i} \leq w_{v_j} \qquad \forall (v_i, v_j) \in M^2 \tag{7}$$

$$\underline{s}_n \leq s_n \leq \overline{s}_n \qquad \forall n \in N \tag{8}$$

$$\mathbf{x} \in \mathbb{Z}^{|N|}_{0 \leq |N|-1}, \quad \mathbf{s} \in \mathbb{Z}^{|N|}_{\geq 0}, \quad \mathbf{w} \in \mathbb{Z}^{|N|}_{0 \leq |C|-1} \tag{9}$$

We remark that pre-processing is employed, e.g., to fix variables associated with the start and the terminal depot of each vehicle. The objective function (1) minimises the total travel time for all vehicles. Constraint (2) ensures that each visit has a different successor (hence no visit is performed more than once), and Constraint (3) eliminates routes which contain cycles. Constraint (4) models time propagation between consecutive nodes on a route. In a similar vein, the index of the vehicle servicing a route is passed through connected nodes by Constraint (5). Constraints (6) controls the start time for visits with pairwise synchronisation. Constraint (7) is a symmetry breaker which ensures the first element of the pair of visits is assigned to the vehicle represented by a lower integer number. Constraints (8) requires start times to respect predefined time windows. Finally, Constraint (9) declares types and domains for decision variables.

2.4. Daily Home Care Scheduling and Routing Problem

Next, we extend the VRPTWSyn formulation to model the HCSRP problem.

Let ρ be the penalty for a declined visit, which is incurred once for every regular visit from the problem domain. All declined visits will be assigned to the auxiliary carer $c_\emptyset$.

Every carer $c \in C$ has some set of skills P^c. Similarly, every visit $v \in V$ requires some set of skills R^v. Furthermore, every two visits that must start simultaneously have the same set of skills, i.e., $R^{v_1} = R^{v_2} \quad \forall (v_1, v_2) \in V^2$. A carer who does not possess all relevant skills is not eligible for making the visit. For conciseness, for every visit $v \in V$, we introduce an additional symbol C^v which denotes the subset of carers who possess all requested skills to perform the visit v, i.e., $C^v = \{c \in C \mid R^v \subseteq P^c\}$.

The symbol U represents the set of clients. Let V^u be the set of visits requested for the client u. Every client u has an upper limit $\overline{f}_u$ on the maximum number of different carers who can visit the client during a day, which reinforces the continuity of care.

For each carer $c \in C$, let B^c be an ordered set of time intervals when the carer is not supposed to work, which includes contractual breaks and periods outside working hours. The first element of the set represents the time before working hours. Its start time is fixed, and its duration can be reduced to allow commencing a shift earlier. Next, the set B^c contains a sequence of contractual breaks $[b_1, \ldots, b_n]$. Every break b has a flexible start time which must commence within the range $[\underline{s}_b, \overline{s}_b]$ and lasts fixed duration d_b. Finally, the last element of the set B^c is the period representing

the time outside working hours. Its start time window has limited flexibility restricted by the maximum overtime allowance. The total number of contractual breaks, their time and duration vary between carers. In the set of problem instances we studied, carers with the longest working hours were entitled to four breaks while carers who worked just a few hours had no breaks. Although time intervals in the set B^c share similar properties to visits, we do not treat them this way due to the assumption that taking a break or time outside working hours does not involve travelling. Instead, we follow the approach proposed by Perron (2011) in which routing decisions precede scheduling breaks. The latter are scheduled using a custom global constraint whose simplified implementation is presented in Appendix B.

We introduce the following auxiliary variables. For each visit v, let a_v be a binary variable which indicates whether the visit is covered in the schedule or has been declined. Furthermore, for each pair of the client u and the carer c, the variable $f_{c,u}$ indicates whether the carer visits the client at least once.

Let us represent Constraints (2-9) from the VRPTWSyn formulation using the feasible set $\mathcal{F}$. It includes schedules with declined visits because the domain of the decision variables $\mathbf{w}$ has been expanded to account for the auxiliary carer $c_\emptyset$.

$$\min \sum_{n \in N} \delta(n, x_n) + \rho \sum_{v \in V^1} (1 - a_v) + \frac{1}{2}\rho \sum_{v \in V^2} (1 - a_v) \tag{10}$$

$$\begin{aligned}
\text{s.t.} \qquad w_v = c_\emptyset \Leftrightarrow x_v = v & \qquad \forall v \in V & (11)\\
a_v = \text{BOOL}(x_v \in N^t \setminus \{v\}) & \qquad \forall v \in V & (12)\\
a_{v_i} = a_{v_j} & \qquad \forall (v_i, v_j) \in M^2 & (13)\\
\text{MEMBER}(w_v, C^v) & \qquad \forall v \in V & (14)\\
\text{BREAKS}(\mathbf{x}, \mathbf{s}, \mathbf{d}, \delta, b_c, e_c, B^c) & \qquad \forall c \in C & (15)\\
\sum_{v \in V^u} \text{BOOL}(w_v = c) \leq |V^u| f_{c,u} & \qquad \forall c \in C \quad \forall u \in U & (16)\\
\sum_{c \in C} f_{c,u} \leq \overline{f}_u & \qquad \forall u \in U & (17)\\
(\mathbf{x}, \mathbf{s}, \mathbf{w}) \in \mathcal{F} & & (18)\\
\mathbf{a} \in \mathbb{B}^{|V|}, \quad \mathbf{f} \in \mathbb{B}^{|C| \times |U|} & & (19)
\end{aligned}$$

The objective function (10) minimises the total distance travelled by carers and penalties incurred by declined visits. Constraint (11) ensures that if a visit is not performed its corresponding decision variable will represent a self-loop, which is required for correct calculation of the total travel time. Constraints (12-13) update the status

of variables indicating whether visits are scheduled. Only carers with relevant skills are allowed to perform a visit due to Constraint (14). Sufficient time for scheduling breaks is ensured by Constraint (15). Constraints (16) and (17) provision the continuity of care. Constraint (18) includes the VRPTWSyn formulation. Finally, Constraint (19) declares variable types and their domains.

The breaks constraint is developed purposefully for this formulation. Its basic implementation transforms all activities the carer is expected to perform (i.e., visits, breaks, travelling) into intervals with the relevant time window restrictions and temporal dependencies known during the constraint propagation. For instance, visits must be performed according to the order established by the route the carer follows and preceded by travelling. If the intervals cannot be arranged in a non-overlapping manner without violating the time window restrictions, then the solution is infeasible.

Preliminary computational experiments using the CP formulation to solve large instances (450-500 visits out of which 100 requires two carers) demonstrated that the CP solver fails to find schedules staffing more than 60-70% of all visits. We identified the synchronisation constraints as the computational bottleneck and therefore, developed the multi-stage approach, which is discussed next.

3. Three-Stage Optimisation Algorithm

The three-stage optimisation algorithm can solve either the VRPTWSyn or the HC-SRP problem, depending on the choice of the formulation presented in the previous section. The algorithm optimises a given formulation in three subsequent stages. A solution obtained at one stage is provided as the initial guess for optimisation in the next stage. Every stage optimises the problem instance using the routing library from the or-tools framework (Google, 2018). The library simplifies the formulation of a VRP and configures the CP solver from the same toolbox to solve the problem instance using a highly customisable hybrid method. The solution approach combines a heuristic initialisation, LS and LNS operators, constraint propagation, and optionally a meta-heuristic. The pseudocode of the hybrid method is illustrated in Algorithm 1. For ease of exposition, the pseudocode presents a specialised variant of the generic optimisation method adapted to solve minimisation problems.

The algorithm uses the following symbols and functions. Let S be the set of search operators and Π be the solution pool. A solution $\mathbf{x}$ is added to the pool Π using the function ADD($\Pi, \mathbf{x}$). The function NEXT(Π) retrieves a solution from the pool Π to initialise search in the next iteration. The best solution stored in the pool Π is queried using the function BEST(Π). For implementation of the stopping criterion, the function ELAPSED-TIME() returns the time since the start of the optimisation

Data: Initial feasible solution $\mathbf{x}^{(i)}$
Parameters: Problem formulation, Search operators S, Solution pool Π
ADD(Π, $\mathbf{x}^{(i)}$)
$\mathbf{x}^{(s)}, \mathbf{x}^{(n)} \leftarrow \emptyset, \mathbf{x}^{(i)}$
$t \leftarrow 0$
repeat
  $\mathbf{x}^{(s)} \leftarrow \mathbf{x}^{(n)}$
  for $s \in S$ **do**
    $X \leftarrow$ NEIGHBOURHOOD$(s, \mathbf{x}^{(s)})$
    while $X \neq \emptyset$ **do**
      for $\mathbf{x}^{(c)} \in X$ **do**
        if ELAPSED-TIME() $- t \geq \Delta$ **then**
          return BEST(Π)
        end
        if IS-FEASIBLE($\mathbf{x}^{(c)}$) **and** OBJ($\mathbf{x}^{(c)}$) $<$ OBJ(BEST(Π))) **then**
          ADD($\Pi, \mathbf{x}^{(c)}$)
          $t \leftarrow$ ELAPSED-TIME()
          $X \leftarrow$ NEIGHBOURHOOD$(s, \mathbf{x}^{(s)})$
          break
        end
      end
    end
  end
  $\mathbf{x}^{(n)} \leftarrow$ NEXT(Π)
until $\mathbf{x}^{(s)} = \mathbf{x}^{(n)}$
return BEST(Π)

Algorithm 1: Specialisation of the generic method for solving VRPs implemented in the or-tools framework (Google, 2018). The method is adapted to solve minimisation problems with stopping criterion defined as the maximum time without improving the incumbent solution.

and the symbol Δ denotes the maximum time without improvement of the incumbent solution. If the time limit is exceeded, the algorithm will terminate. The function IS-FEASIBLE($\mathbf{x}$) tests the feasibility of the solution $\mathbf{x}$ and the function OBJ($\mathbf{x}$) computes the objective. Finally, the function NEIGHBOURHOOD$(s, \mathbf{x})$ returns the set of candidate solutions for the given search operator s in the neighbourhood of the solution $\mathbf{x}$.

The method takes an initial feasible solution as input and iteratively improves it through the exploration of solutions in neighbourhoods induced by search operators. The control flow resembles a VNS without the perturbation step. The role of the search operators is reduced to the enumeration of the candidate solutions that may not even be fully initialised. It is the responsibility of the CP solver to find values for the unassigned variables, to check the feasibility of the candidate solution and to evaluate its cost. The specialisation of the algorithm we present accepts candidate solutions only if they strictly improve the incumbent solution. The generic method implemented in the or-tools framework (Google, 2018) can be customised by introducing a meta-heuristic and tentatively accepting solutions that do not improve the objective. Furthermore, users have the flexibility to alter stopping criteria and change the implementation of the solution pool. By default, no meta-heuristic is used, and the solution pool stores only the current best solution. The algorithm continues exploration until stopping conditions are triggered. Furthermore, if no meta-heuristic is used, the optimisation is stopped when no search operator can improve the incumbent solution.

Each of three stages calls to the algorithm above using the OPTIMISE procedure. We next elaborate on the details of each stage.

3.1. First Stage

The first stage aims to find an initial solution which schedules as many visits with synchronisation constraints as possible. Such a schedule will then be passed as a partial solution (with all other visits declined) to the second stage and re-optimised.

Following the approach of human planners to staff visits with synchronisation constraints, we create teams of two carers who work and travel together. The team selection procedure we adopted follows a greedy strategy that creates pairs of carers to maximise the number of skills and working hours both team members share in common. The algorithm traverses the list of carers sorted by the number of working hours in descending order. If a carer is not a member of a team, we search for another carer who has the most compatible working hours and skills. If no such carer is available, i.e., the number of shared working hours is less than some minimum threshold (e.g., 2 hours and 15 minutes), a team is not formed, and the carer will not be considered for team selection any more. The pseudocode of the algorithm is presented in Appendix A.

After the teams are constructed, the first stage proceeds to solve the CP formulation for the subproblem $F(V^{(T)}, C^{(T)})$ of finding a schedule to staff visits with synchronisation constraints ($V^{(T)}$), using the set of teams ($C^{(T)}$). For every pair of visits which are required to commence simultaneously, one element of the pair belongs to

the set $V^{(T)}$. The synchronisation and symmetry breaking constraints are implicitly satisfied. Hence, they are removed from the CP formulation. Algorithm 2 illustrates the pseudocode of the first stage optimisation.

Parameters: Problem formulation $F(V^{(T)}, C^{(T)})$, Search operators S^1
Result: First stage solution $\mathbf{x}^{(1)}$
$\mathbf{x} \leftarrow [1, \ldots, |N^{(T)}|]$
for $c \in C^{(T)}$ **do**
  $x_{b_c} \leftarrow e_c$
end
$Q \leftarrow \emptyset$ *// initialise a priority queue*
for $v \in V^{(T)}$ **do**
  for $c \in C^{(T)}$ **do**
    ADD$(Q, \delta(b_t, v) + \delta(v, e_t) - \delta(b_t, e_t), [b_t, e_t, v])$ *// insertion* $b_t - [v] - e_t$
  end
end
while $Q \neq \emptyset$ **do**
  $a, b, v \leftarrow$ POP(Q) *// get the next cheapest insertion*
  if $x_a \neq b$ **then**
    // nodes a and b are no longer directly connected
    continue *// insertion not possible*
  end
  $\mathbf{x}^{(c)} \leftarrow \mathbf{x}$ *// perform the insertion on a copy*
  $x_a^{(c)}, x_v^{(c)} \leftarrow v, b$ *// insert v between nodes a and b*
  if IS-FEASIBLE$(\mathbf{x}^{(c)})$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}^{(c)}$ *// accept the insertion*
    for $v \in V^{(T)}$ **do**
      if $x_v = v$ **then**
        // visit v is still available for insertion
        ADD$(Q, \delta(a, v) + \delta(v, c) - \delta(a, c), [a, c, v])$ *// insertion* $a - [v] - c$
        ADD$(Q, \delta(c, v) + \delta(v, b) - \delta(c, b), [c, b, v])$ *// insertion* $c - [v] - b$
      end
    end
  end
end
return OPTIMISE$(F(V^{(T)}, C^{(T)}), \mathbf{x}, S^1)$

Algorithm 2: First stage optimisation.

The first stage applies the parallel cheapest insertion heuristic to construct the

initial guess. The implementation of the heuristic in the routing library (Google, 2018) is a loose adaptation of the heuristic as proposed by Savelsbergh (1990). The algorithm is based on a min priority queue Q. The function $\text{ADD}(Q, \mathbf{v}, c)$ adds the element $\mathbf{v}$ with the declared cost c to the priority queue Q. Conversely, the function $\text{POP}(Q)$ removes the element at the top from the priority queue Q. The algorithm starts with the set of empty routes which join the begin and end depots for each vehicle. For all unstaffed visits, the algorithm considers an insertion between every pair of subsequent nodes for all routes. The available insertion operations are stored in the priority queue ordered according to the marginal increase in the total travel time. If depots' locations for both vehicles in a team are different, we use mean travel time instead. The algorithm iteratively attempts to perform the next possible cheapest insertion that does not make the solution infeasible. Every completed insertion creates new opportunities to insert the remaining visit, which are added to the priority queue. The algorithm proceeds until no more insertions are possible.

The initial guess is then optimised using Algorithm 1 and the following LS operators implemented in the framework (Google, 2018). All search operators work on the set of routes encoded using the variables $\mathbf{x}$ from the CP formulation. Remaining variables are assigned by the CP solver using constraint propagation. For ease of exposition, we arrange the search operators into groups depending on the objects they process. The first group of operators work with nodes which belong to the same route, i.e., to move a node to a different position, to swap positions of two nodes, to move a chain of nodes between two most costly arcs to a new position. This group also includes well-known operators proposed for the Travelling Salesman Problem: 2-Opt, OrOpt, and the operator of Lin and Kernighan (1973). In the first stage, we use only one operator that works on two routes, i.e., to swap chains of consecutive nodes at the beginning of two routes. Finally, several operators work on a route and the set of declined visits. We do not list them all here because they are simple variants of a generic operator that attempts to insert a declined visit into a route optionally removing another node from the route to make the insertion feasible. The optimisation is stopped as soon as no search operator can find a better solution.

3.2. Second Stage

The second stage optimises the original problem instance $F(V, C)$ starting from the solution produced by the first stage. Algorithm 3 illustrates the pseudocode of the second stage optimisation.

We start by transforming the solution restricted to visits with synchronisation constraints found by the first stage into an initial guess for the second stage optimisation. Teams are disbanded into individual vehicles, and every routing node visited by a

Data: First stage solution $\mathbf{x}^{(1)}$
Parameters: Problem formulation $F(V, C)$, Search operators $S^1 \cup S^2$
Result: Second stage solution $\mathbf{x}^{(2)}$
$\mathbf{x}^{(i)} \leftarrow [1, \ldots, N]$
for $n \in N^{(T)}$ **do**
  $n_1, n_2 \leftarrow \text{FIRST}(n), \text{SECOND}(n)$
  $x_{n_1}, x_{n_2} \leftarrow \text{FIRST}(x_n^{(1)}), \text{SECOND}(x_n^{(1)})$
end
return $\text{OPTIMISE}(F(V, C), \mathbf{x}^{(i)}, S^1 \cup S^2)$

Algorithm 3: Second stage optimisation.

team is split into two separate routing nodes for each vehicle. Functions $\text{FIRST}(n)$ and $\text{SECOND}(n)$ for a node n in the first stage given as input return corresponding routing nodes n_1 and n_2 in the second stage. For simplicity, we use the same set of functions for the mapping between teams and individual vehicles. All visits without synchronisation constraints are left unassigned in the initial guess, and the solver is free to change the matching between vehicles and visits. On top of the LS operators described in the first stage (S^1), we enable two additional LNS operators (S^2) implemented in the or-tools framework (Google, 2018). The first operator destroys the assignment of variables that represent two chains of up to three subsequent arcs selected across all routes. The chains could be located on the same route and may even overlap. The second operator destroys one route entirely as well as the whole set of declined visits. The CP solver is responsible for performing the repair operation. Similarly to the first stage, the optimisation is stopped once the incumbent solution cannot be improved by an exhaustive examination of neighbourhoods induced by all search operators.

3.3. Third Stage

The third stage optimisation attempts to reduce further the cost of the solution obtained in the second stage. For this purpose, we adopt the GLS meta-heuristic, which works by applying an adaptive penalty for using a given arc on a route. The implementation of the meta-heuristic in the framework (Google, 2018) adds the following term to the objective function:

$$\lambda \sum_{n \in N^s} (p_{n,x_n} \delta(n, x_n)) \tag{20}$$

where λ is a scaling factor set to 0.1 and $p_{a,b}$ is the penalty for using the arc (a, b). We refer to the formulation with the modified objective function as $F^{(\mathrm{GLS})}(V, C, \mathbf{p})$. The vector $\mathbf{p}$ stores the current values of penalties.

Algorithm 4 presents the pseudocode of the third stage optimisation.

Data: Second stage solution $\mathbf{x}^{(2)}$
Parameters: Problem formulation $F(V, C)$, Search operators $S^1 \cup S^2 \cup S^3$, Solution pool Π
Result: Third stage solution $\mathbf{x}^{(3)}$
ADD($\Pi, \mathbf{x}^{(3)}$)
$\mathbf{p} \leftarrow [0]^{|N|\times|N|}$ ▷ *initialise GLS penalties*
$t \leftarrow 0$
while ELAPSED-TIME() $- t \leq \Delta$ **do**
  $\mathbf{x}^{(i)} \leftarrow$ NEXT(Π)
  $\mathbf{x}^{(c)} \leftarrow$ OPTIMISE($\mathbf{x}^{(i)}, F^{(\mathrm{GLS})}(V, C, \mathbf{p}), S^1 \cup S^2 \cup S^3, L$)
  if OBJ($\mathbf{x}^{(c)}$) < OBJ(BEST(Π)) **then**
    $t \leftarrow$ ELAPSED-TIME()
  end
  ADD($\Pi, \mathbf{x}^{(c)}$)
  $\mathbf{d} \leftarrow []$
  for $n \in N$ **do**
    ▷ *compute the utility for each arc in the solution*
    if $x_n^{(c)} \neq n$ **then**
      $\mathbf{d} \leftarrow \mathbf{d} || [\delta(n, x_n)/(p_{n,x_n} + 1), n, x_n]$ ▷ *append a row to the 2D vector*
    end
  end
  ▷ *increment penalties for arcs with the highest utility*
  DESCENDING-SORT($\mathbf{d}, \iota_1$)
  $u_{\max} \leftarrow d_{0,0}$
  for $(u, a, b) \in \mathbf{d}$ **do**
    if $u \neq u_{\max}$ **then**
      break
    end
    $p_{a,b} \leftarrow p_{a,b} + 1$
  end
end
return BEST(Π)

Algorithm 4: Third stage optimisation.

The third stage is warm-started using the best solution found by the second stage. The GLS penalties are initially set to zero for each arc. The method proceeds iteratively until the maximum time without improvement to the incumbent solution is reached, which we defined as the stopping criterion for the third stage. In every iteration, the solution most recently added to the pool is passed as the initial guess to start optimisation using Algorithm 1. Besides search operators enabled in the first and the second stage ($S^1 \cup S^2$), the solver exploits an additional LS operator from the framework (Google, 2018) which moves forward and backward chains of subsequent nodes only if the move contributes to decreasing the cost of a route. The optimisation is stopped after the time limit without improvement of the incumbent solution is reached. The obtained solution is then added to the solution pool and used to decide which penalties should be updated. For every arc (a, b) traversed in the most recent solution, the meta-heuristic computes the expression $\delta(a, b)/(p_{a,b} + 1)$ called the utility of a feature. Then, the function SORT-DESCENDING($\mathbf{d}, \iota_1$) sorts the vector $\mathbf{d}$ in descending order by the utility, and the penalties are increased by one for all arcs for which the utility attained the highest value. Consequently, a penalty for using a costly arc is not exacerbated if the arc frequently appears in solutions.

4. Computational Results

Computational experiments will verify whether the multistage algorithm is competitive with the solution methods proposed for the VRPTWSyn and demonstrate if it remains a viable approach for considerably larger and more complex HCSRP instances which arise in the real world. To provide diversified results, we run computational tests with the benchmark set of 50 VRPTWSyn instances (Bredström and Rönnqvist, 2007) as well as 14 real-world HCSRP instances obtained from a home care provider, who operates in a major city in the UK. Before describing the problem instances and the results, we discuss the implementation of the algorithm and the configuration parameters.

4.1. Implementation and Configuration Details

The multistage algorithm and the CP formulations were implemented in C++ as a single-thread program using the or-tools library (Google, 2018). The computations were run on a workstation with AMD Ryzen 7 processor and 32 GB of RAM. Termination of the third stage optimisation is triggered by not improving the incumbent solution within the prescribed time limit. We set it to three minutes for the VRPTWSyn instances and five minutes for the HCSRP instances. These values were

derived experimentally after observing that longer time limits do not contribute to obtaining better results. The first stage and the second stage do not need external stopping criteria because they do not use meta-heuristics and terminate as soon as none of the search operators can improve the incumbent solution.

4.2. Vehicle Routing Problem with Time Windows and Synchronised Visits

The VRPTWSyn benchmark set compiled by Bredström and Rönnqvist (2007) consists of 50 instances derived from 10 base problems. Each base problem contains either 20, 50, or 80 visits. The ratio of visits with synchronisation constraints is 10%. The instances are obtained from the base problems by applying different time window configurations: no time windows (F), small (S), medium (M), large (L), and time windows equal to the span of the scheduling horizon (A). The number of vehicles is restricted. Each vehicle can work at most 9 hours, which overlaps with the span of the scheduling horizon. Among the objective functions defined for the benchmark, we use the minimisation of the total travel time.

For some of the problem instances, for which proving optimality remains an open problem, the multistage algorithm strictly improved the best results published in the literature. To quantify the improvement, we calculate a relative marginal difference between the best solution reported in the literature and the solution found by the multistage algorithm. We refer to this quantity as Δ and formally define it as $\Delta = 100 \cdot (O_{\text{Lit}} - O_{\text{MS}})/O_{\text{Lit}}$, where O_{MS} and O_{Lit} denote objective function values of the schedule found by the multistage algorithm and the best schedule reported in the literature, respectively.

In Table 2, we present the best solutions known in the literature, along with the results found by the proposed multistage algorithm. An asterisk in the second column indicates whether the solution has been proven optimal. All solutions of the benchmark problems found by the multistage optimisation algorithm are included in the data set (Polnik et al., 2018). The multistage algorithm reproduced 39 out of 50 best solutions from the literature and strictly improved results for five instances (9S, 6A, 7A, 8A, 9A). For four instances (10S, 10M, 9L, 10L), the multistage algorithm found a slightly worse solutions than the best-known solution, with the difference being less than 0.7%. Finally, the multistage algorithm failed to find a feasible solution in two cases (8F and 9F). To the best of our knowledge, the two instances were solved only by Parragh and Doerner (2018). Overall, comparing the optimality of the solutions produced, the multistage algorithm is competitive with the other most successful approaches proposed in the literature (Afifi et al., 2016; Liu et al., 2019; Parragh and Doerner, 2018). None of these methods unequivocally dominates others in this

Table 2.: Column (P) indicates the problem instance. The number of vehicles, visits without synchronisation constraints, and visits with pairwise synchronisation are displayed in Columns (C), (V^1), and (V^2). Column (O_{Lit}) contains the objective value of the best solution published in the literature. Column (O_{MS}) reports the objective of the best solution returned by the multistage algorithm, and Column (T_{MS}) displays the computational time required to find that solution. Finally, Column (Δ) presents the relative improvement of the multistage solution divided by the best-known result from the literature.

P	C	V^1	V^2	O_{Lit}	O_{MS}	T_{MS} [s]	Δ	P	C	V^1	V^2	O_{Lit}	O_{MS}	T_{MS} [s]	Δ
1F	4	16	2	5.13*[a]	5.13	0.1	0	6M	10	40	5	7.7[e]	7.7	41.89	0
2F	4	16	2	4.98*[a]	4.98	0.01	0	7M	10	40	5	7.48[e]	7.48	23.49	0
3F	4	16	2	5.19*[a]	5.19	3.51	0	8M	10	40	5	8.54*[a]	8.54	119.03	0
4F	4	16	2	7.21*[a]	7.21	0.01	0	9M	16	64	8	10.92[e]	10.92	129.36	0
5F	4	16	2	5.37*[a]	5.37	0.76	0	10M	16	64	8	7.62[e]	7.67	1590.21	-0.66
6F	10	40	5	14.45*[b]	14.45	12.74	0	1L	4	16	2	3.39*[d]	3.39	0.18	0
7F	10	40	5	13.02*[b]	13.02	7.74	0	2L	4	16	2	3.42*[d]	3.42	7.43	0
8F	10	40	5	34.94*[c]	-	-	-	3L	4	16	2	3.29*[d]	3.29	0.07	0
9F	16	64	8	43.48*[c]	-	-	-	4L	4	16	2	5.13*[d]	5.13	0.3	0
10F	16	64	8	12.08*[c]	12.08	57.49	0	5L	4	16	2	3.34*[d]	3.34	0.37	0
1S	4	16	2	3.55*[a]	3.55	0.05	0	6L	10	40	5	7.14*[d]	7.14	53.92	0
2S	4	16	2	4.27*[a]	4.27	0.04	0	7L	10	40	5	6.88[d]	6.88	52.03	0
3S	4	16	2	3.63*[a]	3.63	0.12	0	8L	10	40	5	8[e]	8	2656.16	0
4S	4	16	2	6.14*[a]	6.14	7.38	0	9L	16	64	8	10.43 [f]	10.5	190.14	-0.67
5S	4	16	2	3.93*[a]	3.93	0.24	0	10L	16	64	8	7.36 [f]	7.38	256.76	-0.27
6S	10	40	5	8.14*[d]	8.14	1.82	0	1A	4	16	2	2.95[c]	2.95	0.02	0
7S	10	40	5	8.39*[d]	8.39	9.16	0	2A	4	16	2	2.88[c]	2.88	0.02	0
8S	10	40	5	9.54*[d]	9.54	86.72	0	3A	4	16	2	2.74[c]	2.74	0.11	0
9S	16	64	8	11.93 [e]	11.92	2521.88	0.08	4A	4	16	2	4.29 [g]	4.29	0.02	0
10S	16	64	8	8.54 [f]	8.58	1423.75	-0.47	5A	4	16	2	2.81[c]	2.81	0.5	0
1M	4	16	2	3.55*[d]	3.55	0.16	0	6A	10	40	5	6.48[b]	5.77	56.66	10.96
2M	4	16	2	3.58*[d]	3.58	3.25	0	7A	10	40	5	5.71[c]	5.7	160.85	0.18
3M	4	16	2	3.33*[d]	3.33	1.05	0	8A	10	40	5	6.52[c]	6.51	86.08	0.15
4M	4	16	2	5.67*[d]	5.67	0.81	0	9A	16	64	8	8.51[c]	8.5	116.37	0.12
5M	4	16	2	3.53*[a]	3.53	0.51	0	10A	16	64	8	6.31[c]	6.31	676.65	0

[a] Bredström and Rönnqvist (2008); [b] Decerle et al. (2018); [c] Parragh and Doerner (2018); [d] Bredström and Rönnqvist (2007); [e] Afifi et al. (2016); [f] Liu et al. (2019);[g] Gayraud (2015);

benchmark, as for each solution approach, we can find at least one instance for which a given method did not find the best-known schedule.

From the computational time perspective, the multistage algorithm is slower, in particular on instances with 16 vehicles, compared to methods developed by Afifi et al. (2016); Liu et al. (2019); Parragh and Doerner (2018). In the most extreme case, our algorithm ran 45 minutes (8L), whereas the ALNS of Parragh and Doerner (2018)

runs always below 45 seconds. Among factors that contribute to slower performance, we remark that we are using a general-purpose algorithm with a standard CP solver, rather than a meta-heuristic tailored for this specific problem. On the other hand, the flexibility of the multistage algorithm allows us to extend its use broadly, including a real-world application.

4.3. Daily Home Care Scheduling and Routing Problem

We obtained the HCSRP instances from a home care organisation that provides services in a major city in the UK. The benchmark suite comprises instances for the first two weeks of October 2017 for the largest district where the company operates. The period and the month overlap with the pilot deployment of our algorithm in the home care organisation. Furthermore, other researchers, e.g., Duque et al. (2015), conducted computational studies using data collected in the same period. Depending on the day, between 444 and 508 visits were requested, and the number of available carers varied from 47 to 71. The records include the expected duration of the visit, tasks to be performed, and the number of carers needed. Approximately 20% of visits required pairwise synchronisation.

Shift patterns and contractual breaks of each carer are extracted from historical schedules, which is a common way of retrieving carers' working hours in the literature, see, e.g., (Duque et al., 2015; Nickel et al., 2012). Postal addresses of visits locations were replaced by a distance matrix for pedestrians computed using the Open Source Routing Machine (Luxen and Vetter, 2011). The anonymised problem instances are available online for testing and benchmarking purposes, see (Polnik et al., 2018). To the best of our knowledge, they are the largest HCSRP instances with pairwise synchronisation considered in the literature.

When solving the problem instances, we allow the start times of visits and breaks to be moved up to 90 minutes from their nominal values. Furthermore, each carer can work no more than 60 minutes overtime, 30 minutes before the start of the shift, and 30 minutes after its end. The penalty term was set to the sum of the five longest travel times between every pair of visits' locations in the given problem instance. The resultant value was high enough to prevent attempts to decline a visit and compensate for the penalty incurred by the reduction of the travel time. Finally, clients with single carer visits only can be visited by at most two different carers per day to provide the continuity of care. The limit is extended to four carers for clients who receive multiple carer visits.

Table 3 compares the schedules obtained by human planners and the multistage algorithm. Human planners create schedules manually, starting from a baseline schedule in which the same carer or a team of carers perform a recurring visit for a given

Table 3.: The first set of columns summarise attributes of the problem instances: problem identifier (P), the number of carers (C), single carer visits (V^1), visits with pairwise synchronisation constraints (V^2), and the penalty incurred for declining a visit (ρ). The next three sets of columns contrast solutions obtained by human planners (HP) and the multistage algorithm in the second stage (MS_2) and the third stage (MS_3): the cost of the solution (Objective), the number of declined visits (Declined), and the total travel time for all carers (Travel). The last column reports the computational time when the best solutions were reported in the second and the third stage of the algorithm.

P	C	V^1	V^2	ρ	Objective			Declined			Travel [h]			Time [s]	
					HP	MS_2	MS_3	HP	MS_2	MS_3	HP	MS_2	MS_3	MS_2	MS_3
1	47	338	224	7.03	120.64	29.07	21.11	7	0	0	71.46	29.07	21.11	309	1079
2	62	389	232	7.07	204.9	51.6	36.28	19	3	2	70.59	30.39	22.14	858	2520
3	60	386	240	7.07	200.44	31.37	22.93	17	0	0	80.27	31.37	22.93	73	1322
4	63	388	240	7.07	192.52	52.55	32.73	17	4	2	72.35	24.27	18.59	968	2316
5	62	388	232	7.07	165.43	40.09	32.9	13	2	2	73.54	25.95	18.76	80	1464
6	64	393	224	7.07	147.93	25.09	17.8	10	0	0	77.24	25.09	17.8	503	1222
7	47	343	224	7.03	163.89	57.31	40.64	12	4	3	79.57	29.21	19.56	1335	3486
8	47	336	226	7.03	94.85	45.28	35.8	2	3	2	80.8	24.2	21.74	93	208
9	58	387	224	7.07	168.75	58.66	52.47	13	4	4	76.85	30.38	24.19	1534	2202
10	60	375	236	6.97	159.33	41.8	37.34	11	2	2	82.65	27.85	23.4	96	707
11	58	380	232	6.97	178.24	42.87	36.34	14	2	2	80.66	28.93	22.4	1541	2677
12	71	382	234	6.97	200.45	47.99	38.52	17	3	2	81.96	27.08	24.58	1601	2735
13	58	383	218	6.97	176.86	26.72	17.98	14	0	0	79.27	26.72	17.98	110	1042
14	53	329	230	6.89	176.64	44.91	35.04	14	2	2	80.14	31.13	21.26	1080	2201

client. If a carer is not available, e.g., due to different work arrangements, the first-line manager will have to find a replacement among carers present in the area that day. The alternative assignment is typically selected based on the number of visits already allocated to a given carer.

Considering the objective function, the best solutions reported in the second stage of the multistage algorithm significantly outperformed schedules created by human planners. The second stage solved each instance in less than 30 minutes of computational time. The third stage reduced the cost of a schedule even further at the expense of additional computational time. The best solutions reported by the algorithm typically had three times less declined visits compared to human planners and reduced time spent on travelling by half. Hindle and Hindle (2010) remark that travel-related costs are most likely to negatively impact service levels. Consequently, this extra capacity is a very desirable outcome for first-line managers, e.g., to minimise delays in commencing visits, to perform even more visits (Trautsamwieser and Hirsch, 2011),

and to increase work satisfaction allowing carers to finish their shifts early (Thomsen, 2006).

Both the multistage algorithm and human planners declined some visits. The oversubscription is a well-known fact to first-line managers, who resolve such issues by widening the time windows of the visit, shortening its duration, or rescheduling for some other time. If the manager concludes that not enough workforce is available, individual carers may work more overtime, or some visits may be transferred to an external agency.

Overall, the problems with synchronised visits considered in the literature are significantly smaller than their counterparts, which contain only single carer visits (Fikar and Hirsch, 2017). However, as our example demonstrates, the need for solving large instances with synchronisation constraints arises in the real world. The multistage algorithm we presented solved such problems in a reasonable computational time, i.e., less than 30 minutes, and without ignoring important features in the HHC application domain, i.e., the continuity of care, scheduling of contractual breaks for carers, etc. Such aspects were of critical importance to the home care organisation we supported. The company has evaluated our software during a pilot deployment, and its results are consistent with the findings discussed in this section. Most importantly, the scheduling problems the organisation is dealing with are practically solvable by optimisation software which provides the opportunity for the travel time reduction and does not compromise the continuity of care human planners deliver. As a result, the company board is supporting the adoption of automated scheduling methods in planning the daily operations of the organisation.

5. Conclusion

We proposed a multistage algorithm that solves a Constraint Programming formulation of the Vehicle Routing Problem with Time Windows and Synchronised Visits and its extended version adapted to the Home Healthcare domain. The improved formulation supports flexible shift patterns with an arbitrary number of contractual breaks, skill requirements for visits, and the continuity of care. These features are vital, considering a real application in scheduling home care visits and enable a fair comparison with human planners.

The multistage algorithm is competitive with the other methods proposed in the literature for the pure Vehicle Routing Problem with Time Windows and Synchronised Visits. Our solution method reproduced the majority of the best-known solutions for the popular benchmark set (Bredström and Rönnqvist, 2007) and strictly improved results for several instances. Furthermore, we evaluated the algorithm and the extended

formulation on a suite of real-world instances obtained from a home care provider who operates in a large city in the UK. To the best of our knowledge, these instances are at least two times bigger than the problems with pairwise synchronisation previously considered in the literature. We show that such instances are effectively solvable in acceptable computational times, and the new solutions significantly outperform human planners.

Our current research focuses on improving schedules by altering the CP formulation at the last stage of the optimisation algorithm, i.e., to reduce the number of carers needed to staff the visits or to improve the on-time arrival of carers to a visit.

Acknowledgement(s)

Among many who offered us their time and expertise during this project, we owe special gratitude to employees of Cordia Services LLP and Glasgow City Council. Furthermore, we are grateful to Professor Mikael Rönnqvist for kindly sharing the benchmark problems (Bredström and Rönnqvist, 2007). Finally, we would like to thank two anonymous referees, whose comments helped to improve the presentation of the paper.

Disclosure statement

The authors declare that they have no conflict of interest.

Funding

This work was funded by The Data Lab through an Innovation Centre Grant [grant number 17357].

Appendix A. Team Construction Heuristic

Algorithm 5 presents the team selection policy we adopted.

We extend symbols introduced in Section 2 as follows. Let A be the set of carers who can join a team. The minimum duration both carers should work together to form a team is restricted by $\underline{t}$, which equals 2 hours and 15 minutes. Carers who throughout the selection process either joined a team or do not meet criteria to form a team are stored in the set U. The set T contains the teams constructed by the algorithm. The function ω accepts an arbitrary number of arguments representing carers and returns the total time all of them work together.

input: C
$T \leftarrow \emptyset$; $U \leftarrow \emptyset$
foreach $c_i \in$ SORT-DESCENDING(C, ω) **do**
  if $c_i \in U$ **then**
    continue
  end
  $U \leftarrow U \cup \{c_i\}$; $A \leftarrow C \setminus U$
  if $A = \emptyset$ **then**
    return T
  end
  $m \leftarrow c_\emptyset$
  foreach $c_k \in A$ **do**
    if $\omega(c_i, c_k) \geq \underline{t}$ **and** $\omega(c_i, c_k) \geq \omega(c_i, m)$ **and** $|P^{c_u} \cap P^{c_k}| \geq |P^{c_i} \cap P^m|$ **then**
      $m \leftarrow c_k$
    end
  end
  if $m \neq c_\emptyset$ **then**
    $U \leftarrow U \cup \{m\}$; $T \leftarrow T \cup \{(c_i, m)\}$
  end
end
return T

Algorithm 5: Team Selection

Overall, the algorithm follows a greedy strategy maximising the working hours and the number of skills a team has. The flow of the algorithm is explained in Section 3.1 in the main text. The proposed heuristic algorithm runs in $O(|C|^2 log(|C|))$ time due to $O(nlog(n))$ complexity of sorting.

Appendix B. Breaks Constraint

Algorithm 6 presents a simplified implementation of a custom CP constraint to enforce that carers do not work during contractual breaks and time out of office hours. The implementation of the algorithm extensively uses variables of the interval type designed to model temporal activities that commence within a predefined time window and finish after a specified duration.

Besides symbols introduced in Section 2, the algorithm employs the following functions. The function H(e) converts the element e of some set into an interval, e.g., if v

input: $\mathbf{x}$, $\mathbf{s}$, $\mathbf{d}$, b_c, e_c, B^c
if not PATH-CONNECTED($\mathbf{x}$, $\mathbf{s}$, $\mathbf{d}$, δ, b_c, e_c) **then**
  return
end
$I \leftarrow \emptyset$; $v_i \leftarrow b_c$
while $v_i \neq e_c$ **do**
  $v_j \leftarrow x_{v_i}$
  STARTS-AFTER-END(H($\delta(v_i, v_j)$), H(d_{v_i}))
  STARTS-AFTER-END(H(d_{v_j}), H($\delta(v_i, v_j)$))
  if LB(H(d_{v_i})$_e$) + $\delta(v_i, v_j)$ > UB(s_{v_j}) **then**
    fail
  end
  if $v_i \neq b_c$ **then**
    $I \leftarrow I \cup \{\text{H}(d_{v_i})\}$
  end
  $I \leftarrow I \cup \{\text{H}(\delta(v_i, v_j))\}$; $v_i \leftarrow v_j$
end
foreach $b_i \in B^c$ **do**
  $I \leftarrow I \cup \{\text{H}(d_{b_i})\}$
  foreach $b_j \in B^c$ **do**
    if LB(s_{b_i}) < LB(s_{b_j}) **then**
      STARTS-AFTER-END(H(b_j), H(b_i))
    end
  end
end
ALL-DISJUNCTIVE(I)

Algorithm 6: BREAKS Constraint

belongs to the set of visits V, then H(v) creates its corresponding time interval. Given some interval i, the function END(i) returns the variable that denotes the time when the interval i ends. Interval variable type is accompanied with a set of built-in constraints. Our implementation employs the ALL-DISJUNCTIVE($\mathbf{i}$) constraint to assert that no pair of intervals in the vector $\mathbf{i}$ overlaps and the STARTS-AFTER-END(a, b) constraint which defines a precedence relation between the end of the interval b and the start of the interval a. Functions LB and UB return the lower and upper bounds for the given variable, respectively. Finally, the function PATH-CONNECTED($\mathbf{x}, \mathbf{s}, \mathbf{d}, b_c, e_c$) tests whether variables encode a valid route from the begin depot b_c to the end depot e_c.

If the precondition holds, we create a time interval for every activity a carer performs on a route besides breaks and time outside working hours. $\mathrm{H}(\delta(v_i, v_j))$ denotes the interval corresponding to travel between visits v_i and v_j. The interval $\mathrm{H}(d_{v_i})$ represents performing a visit v_i. The length of the interval equals the duration of the visit, and the beginning of the interval respects the time window for commencing a visit. Relevant precedence constraints between intervals are defined whenever possible. Travel to the next visit location can be started after the previous visit is completed. Similarly, the succeeding visit cannot commence unless the travel is finished. The statement $\mathrm{LB}(\mathrm{END}(\mathrm{H}(v_i)) + \delta(v_i, v_j) > \mathrm{UB}(s_j)$ guards against negative intervals. If the time window of the succeeding visit is violated, the assertion will fail, and the solution will be declared invalid. Furthermore, the start times of breaks must respect their relative order during the day. Ultimately, none of the intervals can overlap, which is enforced by the ALL-DISJUNCTIVE constraint.

References

Afifi, S., Dang, D.-C., and Moukrim, A. (2016). Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. *Optimization Letters*, 10(3):511–525.

Bachouch, R. B., Guinet, A., and Hajri-Gabouj, S. (2011). A decision-making tool for home health care nurses' planning. *Supply Chain Forum: An International Journal*, 12(1):14–20.

Beck, J. C., Prosser, P., and Selensky, E. (2003). Vehicle Routing and Job Shop Scheduling: What's the Difference? In *International Conference on Automated Planning and Scheduling*, pages 267–276.

Benzarti, E., Sahin, E., and Dallery, Y. (2013). Operations management applied to home care services: Analysis of the districting problem. *Decision Support Systems*, 55(2):587–598.

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

Blais, M., Lapierre, S. D., and Laporte, G. (2003). Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Society*, 54(11):1141–1147.

Bredström, D. and Rönnqvist, M. (2007). A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints. Discussion Papers 2007/7, Norwegian School of Economics, Department of Business and Management Science.

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

Cappanera, P., Gouveia, L., and Scutellà, M. G. (2011). The skill vehicle routing problem. In *Proceedings of the 5th International Conference on Network Optimization*, INOC'11, pages 354–364, Berlin, Heidelberg. Springer-Verlag.

Cappanera, P. and Scutellà, M. G. (2015). Joint assignment, scheduling, and routing models to home care optimization: A pattern-based approach. *Transportation Science*, 49(4):830–852.

Cappanera, P., Scutellà, M. G., Nervi, F., and Galli, L. (2018). Demand uncertainty in robust home care optimization. *Omega*, 80:95 – 110.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239(1):39–67.

Cheng, E. and Rich, J. (1998). A home health care routing and scheduling problem. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.8869`.

Cissé, M., Yalçındağ, S., Kergosien, Y., Şahin, E., Lenté, C., and Matta, A. (2017). Or problems related to home health care: A review of relevant routing and scheduling problems. *Operations Research for Health Care*, 13-14:1 – 22.

Decerle, J., Grunder, O., Hassani, A. H. E., and Barakat, O. (2018). A memetic algorithm for a home health care routing and scheduling problem. *Operations Research for Health Care*, 16:59 – 71.

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

Dohn, A., Rasmussen, M. S., and Larsen, J. (2011). The Vehicle Routing Problem with Time Windows and Temporal Dependencies. *Networks*, 58(4):273–289.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science*, 46(3):297–316.

Duque, P. M., Castro, M., Sörensen, K., and Goos, P. (2015). Home care service planning. The case of Landelijke Thuiszorg. *European Journal of Operational Research*, 243(1):292–301.

Ehmke, J. F., Campbell, A. M., and Urban, T. L. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, 240(2):539 – 550.

En-nahli, L., Afifi, S., Allaoui, H., and Nouaouri, I. (2016). Local search analysis for a vehicle routing problem with synchronization and time windows constraints in home health care services. *IFAC-PapersOnLine*, 49(12):1210 – 1215.

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

Fikar, C. and Hirsch, P. (2017). Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86 – 95.

Frifita, S., Masmoudi, M., and Euchi, J. (2017). General variable neighborhood search for home healthcare routing and scheduling problem with time windows and synchronized visits. *Electronic Notes in Discrete Mathematics*, 58:63 – 70.

Gayraud, F. (2015). *Vehicle routing problem with synchronization constraints in home care support services - Mathematical formulation and hybridization based on metaheuristics*. Theses, Université Blaise Pascal - Clermont-Ferrand II.

Google (2018). Google Optimization Tools. `https://developers.google.com/optimization/`. Accessed: 2018-06-19.

Hertz, A. and Lahrichi, N. (2009). A patient assignment algorithm for home care services. *Journal of the Operational Research Society*, 60(4):481–495.

Hiermann, G., Prandtstetter, M., Rendl, A., Puchinger, J., and Raidl, G. R. (2015). Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research*, 23(1):89–113.

Hindle, G. A. and Hindle, A. (2010). Developing geographical indicators of mileage-related costs: a case study exploring travelling public services in english local areas. *Journal of the Operational Research Society*, 61(5):714–722.

Kergosien, Y., Lenté, C., and Billaut, J.-C. (2009). Home health care problem: An extended multiple Traveling Salesman Problem. In *4th Multidisciplinary International Conference on Scheduling: Theory and Applications*, Dublin, Ireland.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling*, 15(5):579–600.

Laesanklang, W. and Landa-Silva, D. (2017). Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. *Annals of Operations Research*, 256(1):93–127.

Li, Y., Lim, A., and Rodrigues, B. (2005). Manpower Allocation with Time Windows and Job-teaming Constraints. *Naval Research Logistics (NRL)*, 52(4):302–311.

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

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Liu, R., Tao, Y., and Xie, X. (2019). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Com-*

puters & Operations Research, 101:250 – 262.

Liu, R., Yuan, B., and Jiang, Z. (2017). Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements. *International Journal of Production Research*, 55(2):558–575.

Luxen, D. and Vetter, C. (2011). Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 513–516, New York, NY, USA. ACM.

Mankowska, D. S., Meisel, F., and Bierwirth, C. (2014). The home health care routing and scheduling problem with interdependent services. *Health Care Management Science*, 17(1):15–30.

Mutingi, M. and Mbohwa, C. (2014). Multi-objective homecare worker scheduling: A fuzzy simulated evolution algorithm approach. *IIE Transactions on Healthcare Systems Engineering*, 4.

Mısır, M., Smet, P., and Berghe, G. V. (2015). An analysis of generalised heuristics for vehicle routing and personnel rostering problems. *Journal of the Operational Research Society*, 66(5):858–870.

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

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects. *European Journal of Operational Research*, 263(3):737 – 754.

Parragh, S. N. and Doerner, K. F. (2018). Solving routing problems with pairwise synchronization constraints. *Central European Journal of Operations Research*, 26(2):443–464.

Perron, L. (2011). Operations research and constraint programming at google. In Lee, J., editor, *Principles and Practice of Constraint Programming – CP 2011*, pages 2–2, Berlin, Heidelberg. Springer Berlin Heidelberg.

Polnik, M., Riccardi, A., and Akartunalı, K. (2018). Dataset of Home Care Scheduling and Routing Problems with Synchronized Visits. `https://doi.org/10.15129/2d4885e1-bc24-414b-83ce-a846fb5c9689`.

Rahimian, E., Akartunalı, K., and Levine, J. (2017a). A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers & Operations Research*, 82:83–94.

Rahimian, E., Akartunalı, K., and Levine, J. (2017b). A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research*, 258(2):411–423.

Rasmussen, M. S., Justesen, T., Dohn, A., and Larsen, J. (2012). The Home Care Crew Scheduling Problem: Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3):598–610.

Savelsbergh, M. (1990). A parallel insertion heuristic for vehicle routing with side constraints. *Statistica Neerlandica*, 44(3):139–148.

Thompson, G. M. and Pullman, M. E. (2007). Scheduling workforce relief breaks in advance versus in real-time. *European Journal of Operational Research*, 181(1):139–155.

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

Trautsamwieser, A., Gronalt, M., and Hirsch, P. (2011). Securing home health care in times of natural disasters. *OR Spectrum*, 33(3):787–813.

Trautsamwieser, A. and Hirsch, P. (2011). Optimization of daily scheduling for home health care services. *Journal of Applied Operational Research*, 3(3):124–136.

Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469 – 499.