New Decomposition Methods for Home Care Scheduling with Predefined Visits

Florian Grenouilleau[a,*], Nadia Lahrichi[a], Louis-Martin Rousseau[a]

[a]*École Polytechnique de Montréal, 2900 Boulevard Édouard-Montpetit, H3T 1J4, Montréal, Québec, Canada*

Abstract

The continuous aging of the population and the desire of the elderly to stay in their own homes as long as possible has led to a considerable increase in the demand for home visits. In this context, home care agencies try to serve more patients while maintaining a high level of service. They must regularly decide which patients they can accept and how the patients will be scheduled (care provider, visit days, visit times). In this paper we aim to maximize the number of new patients accepted while ensuring a single provider-to-patient assignment and a consistency of the visits times for every patient through the week. To solve this problem, we propose an extension to an existing logic-based Benders decomposition. Moreover, we present a new pattern-based logic-based Benders decomposition and a matheuristic using a large neighborhood search. The experiments demonstrate the efficiency of the proposed approaches and show that the matheuristic can solve all the benchmark instances in less than 20 seconds.

Keywords: Home Care / Scheduling / LBBD / Matheuristic / LNS

*Corresponding author

Email addresses: florian.grenouilleau@cirrelt.net (Florian Grenouilleau), nadia.lahrichi@polymtl.ca (Nadia Lahrichi), louis-martin.rousseau@polymtl.ca (Louis-Martin Rousseau)

1. Introduction

Due to population aging and the Canadian government's plan to decentralize care, the demand for home care services has significantly increased during the last decade (Sinha & Bleakney, 2014). These services allow the patients to stay in their own homes for as long as possible. From the government's point of view, home care services reduce the patient flow in hospitals and reduce the cost of care (Macintyre et al., 2002).

In this context, the home care agencies receive new patient requests everyday and continuously try to better manage their resources in order to serve them while maintaining a high level of service. Due to this resources' management, the scheduling decisions are becoming crucial in order to keep a high acceptance rate and be able to maximize the agency's revenues. In this work, we are interested in the home care scheduling with predefined visits (*HCS-PV*). This problem can be described as follows; each week some patients leave the agencies' system (end of the care plan, problem requiring a hospital admission, etc.) and thus, agencies have to decide how many new patients they can accept and how the patients will be assigned to the providers and scheduled. In order to conserve a high continuity of care (i.e. a strong patient-provider relationship), the schedules for the patients already present in the system (named *existing patients*) have to stay unchanged (same assigned provider, same visit time and days). To efficiently solve the problem, the home care agencies usually split the patients per area and solve the scheduling problems per team of providers (5 to 15 providers). Moreover, most of the agencies are still creating the schedules by hand while trying to take into account all the constraints simultaneously. This usually leads to suboptimal solutions and usually requires a large amount of time for the

schedulers. To help them in this task, more and more researchers attempt to develop efficient optimization methods taking into account the practical constraints met by the home care agencies while producing high quality solutions in a short amount of time in order to be used in practice.

According to the literature, the proposed *HCS-PV* can be described as a variant of the home health care routing and scheduling problem (HHCRSP), a 20 years old problem (Begur et al. (1997), Cheng & Rich (1998)). To the best of our knowledge, no standard version of the *HHCRSP* exists (different constraints and/or objectives) but it is usually respresented as a rich vehicle routing problem. This plurality of modeling, which originates from the countries' home care management policies, makes it difficult to compare the existing methods. In our case, the *HCS-PV* is close to a consistent VRP with home care specific constraints.

The HHCRSP (see Cissé et al. (2017), Fikar & Hirsch (2017) for two comprehensive surveys) was originally solved on a daily planning horizon. It, then, has evolved to integrate more practical constraints such as the maximization of the patients and caregivers' preferences (Braekers et al., 2016), the balance of the workload (Bertels & Fahle, 2006), shared visits (Frifita et al., 2017) or even the time-dependent travel time (Rest & Hirsch, 2016). Thereafter, the HHCRSP has been extended to a weekly horizon that allows for better coping with the reality of some constraints, such as the patients' care plan and/or the continuity of care. Some exact methods (Gamst & Jensen, 2012; Trautsamwieser & Hirsch, 2014; Borsani et al., 2006; Torres-Ramos et al., 2014) have been proposed. Nevertheless, the complexity of the problem often leads to scalability issues. In order to cope with those issues, recent works apply metaheuristic-based methods (Di Gaspero & Urli, 2014; Decerle et al., 2018; Zhang et al., 2018; Yalçındağ et al., 2016; Lin

et al., 2018; Grenouilleau et al., 2019). Those methods allow to solve large problems but do not offer any proof of convergence or optimality gap.

Recently, Heching et al. (2019) have proposed a new method to solve the *HCS-PV* based on a logic-based Benders decomposition (*LBBD*). The proposed method decomposes the problem in two parts, the acceptance and assignments of the patient is done in a master problem while the feasibility of the scheduling constraints (time windows, travel times, etc.) is checked in independent subproblems. Based on their computational experiments, the proposed *LBBD* outperforms the classical mixed-integer formulation in terms of computation time. Nevertheless, on large instances, the proposed *LBBD* does not seem robust and the computation time quickly increases with the size of the instances. In this work, we propose to extend the idea of solving the *HCS-PV* using Benders decompositions by introducing novel master and subproblem decompositions in order to reduce the computation time. This computation time's reduction will allow the home care agencies to solve larger problems and use the proposed methods on a daily basis.

In this paper, the contributions are as follows. We firstly propose a new algorithm for the subproblem of the *LBBD* formulation presented in Heching et al. (2019). It decomposes the subproblem to make it easier to solve. Secondly, we present a new *LBBD* formulation with additional variables. The new variables correspond to visit patterns for the patients; they combine the assigned provider, the visit days, and the visit times in a single variable so that most of the constraints can be handled in the master problem. Finally, we propose a new matheuristic method based on a Dantzig–Wolfe formulation (DWF) and a large neighborhood search (LNS). This matheuristic iteratively solves the problem using *LNS* and then solves the *DWF* using the providers' schedules found during the *LNS* iterations.

Our computational experiments show that the matheuristic finds all the optimal solutions of the benchmark instances in less than 20 seconds.

The remainder of this paper is as follows. Section 2 defines the problem. Section 3 presents the mathematical formulations and Section 4 describes our matheuristic. Section 5 presents the computational results and Section 6 provides concluding remarks.

2. Problem definition

HCS-PV considers a patient set P and maximizes the number of scheduled patients given a set of available providers A on a 5-days horizon. For each scheduled patient, we must determine the assigned provider, the visit days, and the visit time. These decisions must take into account the existing patients; the scheduled visits for these patients cannot be modified for continuity of care purposes. In the home care context, continuity of care involves always sending the same provider at the same time to the same patient, to build a relationship between them and to improve the patient's experience. Figure 1 gives an example of a provider's schedule and a possible slot for a new patient requiring two visits per week.

Various assignment and routing constraints must be taken into account. Each patient is assigned to a single provider, and the visit times must be the same throughout the week (visit consistency). There are also restrictions on the travel time, the available time windows for the patients and providers, and the maximum weekly working time for the providers. Formally, each patient p has a required number of visits $v_p \in [1, 5]$, a level of qualification required Q_p, a visit duration dur_p, a location l_p and a time window $[r_p, d_p]$ in which he/she must be visited. Moreover, some patients have special

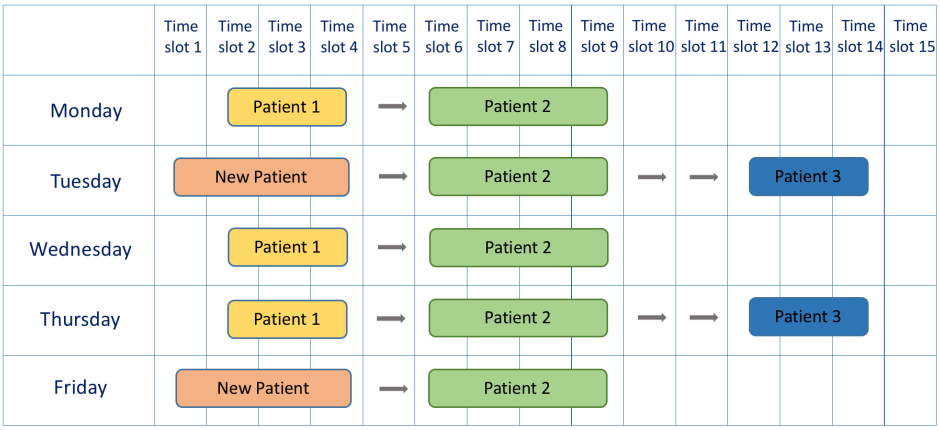

Figure 1: Possible assignment for a new patient requiring two visits per week.

requirements, e.g., they may need a specified duration between visits. For example, a patient could require two visits with at least one day between the visits. In this case, the visit days Monday-Wednesday or Tuesday-Friday will be allowed but the visit days Tuesday-Wednesday or Thursday-Friday will be forbidden. For this constraint we define the set K_p, representing all the combination of visit days which are allowed for patient p. Finally, each provider a has a location l_a, a working time window $[r_a, d_a]$, a qualification level Q_a and a maximum weekly working time $\overline{W_a}$. The working time only comprises the time between the start of the first patient and the end of the last patient for each work day. A summary of these parameters is given in Table 1.

Parameter	Description
P	Set of patients
A	Set of providers
v_p	Number of visits required
Q_p	Required level of qualification
dur_p	Duration per visit
l_p	Patient's location
$[r_p, d_p]$	Patient's time window (same every day)
K_p	Feasible combination of visit days
l_a	Provider's location
$[r_a, d_a]$	Provider's time window (same every day)
Q_a	Provider's qualification level
$\overline{W_a}$	Maximum work time over the week

Table 1: Description of the problem's parameters

3. Mathematical formulations

In this section, we present different formulations for the proposed problem. The goal here is to use those different formulations in order to tackle the problem in different ways and analyze if some computation time reductions are observed. Firstly, we recall the formulation introduced by Heching et al. (2019). This formulation corresponds to a natural Benders decomposition for the problem, with the assignments in the master problem and the scheduling in the subproblems. Secondly, we propose an alternative subproblem for this first formulation using two steps of resolution. Thirdly, we propose a new Benders formulation using pattern of visits as variable, this formulation's objective is to have more constraints in the master problem in order to potentially improve the resolution time. Finally, we propose a classical Dantzig-Wolfe resolution based on the providers' weekly schedules.

3.1. Assignment-based LBBD

This first formulation (Heching et al., 2019) uses a *LBBD* (Hooker & Ottosson, 2003), which derives from the classical Benders decomposition (Benders, 1962). The classical Benders method decomposes the problem into two parts (master problem and subproblem). It iteratively solves the master problem and checks the feasibility and optimality of the solution in the subproblems. If necessary, the subproblem generates feasibility and/or optimality cuts, and these cuts are added to the master problem. The process stops when the generated solution is optimal or the problem is proved infeasible. The Benders subproblems are linear programs, but in the *LBBD* the subproblem is a feasibility check based on the inference dual. In order to ease the comprehension of this first model, the variables presented in the subsequent sections are summarized in Table 4. in the Appendix.

3.1.1. Master problem

In this first *LBBD*, the master problem corresponds to an assignment problem defining the visited patients and their visited days as well as the patient-provider assignments. We define three sets of decision variables: δ_p is set to 1 if patient p is visited and 0 otherwise; $x_{a,p}$ is 1 if patient p is visited by provider a and 0 otherwise; and $y_{a,p,d}$ equals 1 if patient p is visited by provider a on day d. If there are restrictions on which days patients can be scheduled, we manage this with the constraint $\mathbf{y} \in K_p$.

The master problem (MP) is as follows:

$$(MP) : \max \sum_{p \in P} \delta_p \tag{1}$$

$$\text{s.t.} \sum_{a \in A} x_{a,p} = \delta_p \qquad \forall p \in P \tag{2}$$

$$y_{a,p,d} \leq x_{a,p} \qquad \forall a \in A, \forall p \in P, \forall d \in D \tag{3}$$

$$\sum_{a \in A} \sum_{d \in D} y_{a,p,d} = v_p \delta_p \qquad \forall p \in P \tag{4}$$

$$x_{a,p} = 0 \qquad \forall a \in A, \forall p \in P, Q_p \nsubseteq Q_a \tag{5}$$

$$\mathbf{y} \in K_p \qquad \forall p \in P \tag{6}$$

$$\delta_p, x_{a,p}, y_{a,p,d} \in \{0,1\} \quad \forall a \in A, \forall p \in P, \forall d \in D \tag{7}$$

In MP, the objective function (1) maximizes the number of patients visited. Constraints (2) and (3) are the convexity constraints that link the variables, and constraints (4) enforce the required number of visits per patient. Constraints (5) ensure that requirements and skills are respected, and constraints (6) control the sets of days allowed for each patient. Finally, constraints (7) are the binary restrictions.

3.1.2. Subproblem

The subproblem determines if the assignment found by MP is feasible according to the scheduling constraints : synchronization of the visits, travel time, time windows and maximum worktime. In case of infeasibility, *no-good cuts* (on the $y_{a,p,d}$ variables) are added to the master problem. We define a subproblem SP_a for each provider a. Each SP_a corresponds to a multiple-day traveling salesman problem with time windows, and it is solved using constraint programming. For each SP, we define $P_{(SP_a)}$ the

set of assigned patients and $P_{(SP_a),d}$ the set of patients assigned per day d to provider a. In addition, we define sequencing variables $\pi_{d,v}$ which correspond to the patient p's location visited in the v^{th} position on day d, with $p \in P_{(SP_a),d}$. These variables also take into account the fact that each route must start and end at the provider's location l_a. Moreover, we introduce the variables s_p corresponding to the visit time for patient p and the parameter V_d equal to the number of patient visited the day d (i.e, $|P_{(SP_a),d}|$). Finally, we restrain the patients' time windows according to the provider's one by defining $r'_p = max(r_a, r_p)$ and $d'_p = min(d_a, d_p)$. We define a subproblem for each provider as follows:

$$(SP_a) : \max 0 \tag{8}$$

$$\pi_{d,1} = l_a, \pi_{d,V_d+2} = l_a \qquad \forall d \in D \tag{9}$$

$$\text{s.t. } all\_different\{\pi_{d,v} | v = 1, .., V_d + 2\} \qquad \forall d \in D \tag{10}$$

$$r'_p \leq s_p \leq d'_p - dur_p \qquad \forall p \in P_{(SP_a)} \tag{11}$$

$$s_{\pi_{d,v}} + dur_{\pi_{d,v}} + t_{\pi_{d,v},\pi_{d,v+1}} \leq s_{\pi_{d,v+1}} \qquad \forall d \in D, v = 1, .., V_d + 1 \tag{12}$$

$$\sum_{d \in D} (s_{\pi_{d,V_d+1}} + dur_{\pi_{d,V_d+1}} - s_{\pi_{d,2}}) \leq \overline{W_a} \tag{13}$$

$$\pi_{d,v} \in P_{(SP_a),d} \cup l_a \qquad \forall d \in D, v = 1, .., V_d + 2 \tag{14}$$

In this formulation, the objective function (8) is 0 because we simply want to check that a solution exists. Constraints (9) ensure that the provider starts and ends each day at his/her home. This means that for each day, the tour start at $v = 1$, ends at $v = V_d + 2$ and the first patient is visited at the position $v = 2$ and the last patient at position $v = V_d + 1$. Constraints (10) ensure that all the locations are visited and constraints (11) enforce the

patients' time windows. The travel time constraints are taken into account by constraints (12) and the maximum working time by constraints (13). Finally, the variables' domains are defined by constraints (14).

3.2. Two-steps subproblem

As described in the previous section, the subproblem SP_a is defined for each provider a individually and has to cope with all the routing constraints (travel time, visit consistency, overtime, time windows). Depending on the problem's difficulty (size of the time windows, number of patient to schedule), this simultaneous management of all those constraints could lead to an excessive computation time. In order to avoid this issue, we propose an alternative subproblem resolution allowing to split the resolution in two steps. The first step will only check the travel time and time windows constraints, the second will solve the whole subproblem as proposed in the previous section. To describe this two-steps subproblem, we introduce the constraint programming formulation of the daily problem. The daily subproblem ($SP_{a,d}$) is as follows.

$$(SP_{a,d}) : \max 0 \tag{15}$$

$$\text{s.t. } all\_different\{\pi_v | v = 1, .., V_d + 2\} \tag{16}$$

$$\pi_1 = l_a, \pi_{V_d+2} = l_a \tag{17}$$

$$r'_p \leq s_p \leq d'_p - dur_p \qquad \forall p \in P_{(SP_a),d} \tag{18}$$

$$s_{\pi_v} + dur_{\pi_v} + t_{\pi_v,\pi_{v+1}} \leq s_{\pi_{v+1}} \qquad v = 1, .., V_d + 1 \tag{19}$$

$$\pi_v \in P_{(SP_a),d} \cup l_a \cup l_{a'} \qquad v = 1, .., V_d + 2 \tag{20}$$

According to this formulation, the alternative subproblem now first solves

the problem for each day independently ($SP_{a,d}$). The provider's maximum weekly working time and consistency constraints are not considered (constraints 11 and 13). If a feasible route is found for each day, we solve the full subproblem (SP_a), otherwise a feasibility cut is created for each non-feasible day.

3.3. Pattern-based LBBD

As described previously, the master problem proposed in Heching et al. (2019) manages the acceptances/assignments and all the routing constraints are managed in the subproblems. In this section, we present a second Benders decomposition. The idea here is to move most of the constraints from the subproblems to the master one. This modification could help the master problem to better "understand" the problem and so more easily find interesting integer solutions in order to reduce the computation time. As a reminder, in order to define a feasible solution, we must determine for each accepted patient, the assigned provider, the set of visit days, and the visit time. In the second formulation, we combine these decisions into a new variable.

We introduce the concept of a visit pattern ω. That includes all of these four elements: a patient p_ω, an assigned provider a_ω, a set of visit days D_ω, and a visit time s_ω. In comparison with the previous master problem, the pattern ω could be seen as a concatenation of the $x_{a,p}$ and $y_{a,p,d}$ variables. The problem involves assigning a pattern $\omega \in \Omega_p$ to each patient, where Ω_p is a set containing all the feasible visit patterns for patient p, with $\cup_{p \in P} \Omega_p = \Omega$. Because the schedules of the existing patients can't be modified, we can compute in advance the set of feasible patterns for each patient, thus generating the set Ω containing all the feasible patterns. To

do so, for each patient (p), each provider (a), each time index (t) and each combination (C) of $\binom{5}{v_p}$ days respecting the patient's K_p set, we check if the pattern made of provider a, visit time t, and visit days C is feasible for patient p, according to the provider a's available visit slots.

3.3.1. Master problem

We now present the new *LBBD* formulation based on Ω. Variable δ_p still corresponds to the patient acceptance and let x_ω be 1 if visit pattern ω is selected. Finally, $tt_{\omega,\omega'}$ corresponds to the travel time between the patient locations associated with patterns ω and ω'.

$$(PBF) : \max \sum_{p \in P} \delta_p \tag{21}$$

$$\text{s.t. } \delta_p = \sum_{\omega_p \in \Omega_p} x_{\omega_p} \qquad \forall p \in P \tag{22}$$

$$x_\omega + x_{\omega'} \leq 1 \qquad \forall (\omega, \omega') \in \Omega, D_\omega \cap D_{\omega'} \neq \emptyset, s_\omega + tt_{\omega,\omega'} > s_{\omega'} \tag{23}$$

$$\delta_p \in \{0, 1\} \qquad \forall p \in P \tag{24}$$

$$x_\omega \in \{0, 1\} \qquad \forall \omega \in \Omega \tag{25}$$

Let PBF be the new pattern-based formulation. The master problem is a set covering problem defined by (21)–(25). The objective function (21) maximizes the number of patients visited. Constraints (22) are the convexity constraints that link the decision variables, and constraints (23) enforce the travel time between patients. Constraints (24)–(25) are the binary restrictions. This new master problem includes all the constraints (single provider-to-patient assignment, consistent visits, required number of visits, travel time, patient's requirements) except for the restrictions on the

providers' working time, which are enforced in the subproblems.

3.3.2. Subproblem

The subproblems are described as follows. For each provider a, the subproblem retrieves the list of selected patterns and computes for each day the total worktime (time between the start of the earliest visit and the end of the latest one). If the sum of these work times is greater than $\overline{W_a}$, a feasibility cut is added to the master problem. We provide a procedure in polynomial time as described in Algorithm 1.

Algorithm 1: Subproblem solution (for provider a)

```
1 sum_work_time = 0 ;
2 for each day d do
3 |   Retrieve the list L_d of assigned patterns containing d;
4 |   Sort L_d by increasing order of visit times and build the route r_d;
5 |   sum_work_time += r_d's work time;
6 end for
7 if sum_work_time > W_a then
8 |   Create a no-good cut on the assigned patterns;
9 end if
```

3.4. Dantzig-Wolfe decomposition

As described in the introduction, the branch-and-price methods have been widely used to solve the home health care scheduling problem (Gamst & Jensen (2012),Trautsamwieser & Hirsch (2014)). Moreover, in the previous section, we have shown that each patient schedule can be represented by a pattern. For each feasible solution, we can define the provider's schedule by the subset of patterns that has been assigned to this provider. Therefore,

a provider's schedule consists of visit patterns that satisfy travel and work-time constraints. We use this aggregation of patterns (i.e. the schedules) to build the third formulation, a Dantzig-Wolfe decomposition.

We introduce Λ, the set of feasible providers' schedules, and Λ_a the subset of feasible schedules for provider a (with $\Lambda = \cup_{\Lambda_a}$). Let n_λ be the number of patients visited by schedule $\lambda \in \Lambda$. Let $v_{\lambda,p}$ equal to 1 if patient p is visited by schedule λ and 0 otherwise. Finally, we define the decision variable x_λ, which equals 1 if provider's schedule λ is selected and 0 otherwise.

The assignment set partitioning formulation (ASP) is a Dantzig–Wolfe decomposition and is stated as follows:

$$(ASP) : \max \sum_{\lambda \in \Lambda} n_\lambda x_\lambda \tag{26}$$

$$\text{s.t.} \sum_{\lambda \in \Lambda_a} x_\lambda \leq 1 \qquad \forall a \in A \tag{27}$$

$$\sum_{\lambda \in \Lambda} v_{\lambda,p} x_\lambda \leq 1 \qquad \forall p \in P \tag{28}$$

$$x_\lambda \in \{0,1\} \qquad \forall \lambda \in \Lambda \tag{29}$$

The objective function (26) maximizes the number of patients scheduled. Constraints (27) ensure that there is at most one schedule per provider, and constraints (28) ensure that there is at most one schedule per patient. Finally, constraints (29) are the binary restrictions.

4. Visit pattern matheuristic

In this section, we present a visit pattern matheuristic based on the formulation proposed in Section 3.4 and a large neighborhood search (LNS). The *LNS* (Shaw, 1998) is a metaheuristic using the *ruin-and-recreate* principle (Schrimpf et al., 2000). This iterative method destroys some parts of the solution and then repairs it to improve its quality. The current and best solutions are then updated if necessary.

According to the literature (Villegas et al. (2013), Grangier et al. (2017), Grenouilleau et al. (2019)), matheuristics provide a good balance between the solution quality of an exact method and the short computation time of metaheuristics. In the proposed matheuristic, the idea is to use the providers' schedules as variables which can be generated using the *LNS* procedure. During the *LNS*' iterations, all the encountered feasible weekly schedules are kept in a list. After a certain number of iterations, the *LNS* stops and then we solve (26) – (29) using the retrieved schedules. If a better solution is found by the set partitioning, the *LNS*' best solution is updated and another round of *LNS* iterations is run. A similar matheuristic framework has been introduced in (Grenouilleau et al., 2019).

4.1. Overview of visit pattern matheuristic

Algorithm 2 gives an overview of the visit-pattern matheuristic (VPM). We first create an initial solution and then iteratively remove part of the solution using a removal operator and rebuild it using a repair operator. We then analyze the temporary solution (S_t) to see if it improves the best found solution (S^*) or if the acceptance rule (simulated annealing in our context) accepts it as the current solution (S_c). We solve the set partitioning problem based on Λ every 2000 iterations and update the current and best solutions

if necessary. The implementation details are given in Section 4.2

Algorithm 2: VPM

```
1  Create the initial solution S_c;
2  Set the best found solution S* to S_c;
3  Create the empty set of providers' weekly schedules Λ;
4  while termination criterion not met do
5      S_t ← S_c;
6      Apply removal operator to S_t;
7      Apply repair operator to S_t;
8      Add the found schedules to Λ;
9      if S_t is accepted then
10         S_c ← S_t;
11     end if
12     if S_t is better than S* then
13         S* ← S_t;
14     end if
15     if total_iteration % 2000 = 0 then
16         S_sp ← Solve ASP based on Λ;
17         if S_sp better than S* then
18             S* ← S_sp;
19             S_c ← S_sp;
20         end if
21     end if
22 end while
```

4.2. Implementation details

We now present the implementation details of our *LNS* algorithm. We first build the initial solution using a greedy approach (see Algorithm 3). This algorithm starts by creating the actual schedules per provider by inserting all the existing patients. Then, iteratively, a new patient is randomly chosen and all his/her possible insertions (feasible visit patterns assignments) are computed. If at least one assignment is feasible, the patient is accepted and the method assigns the pattern generating the lowest increase in terms of travel time. The procedure is repeated until all the new patients are processed.

Algorithm 3: Initial Solution

```
1  Create P', a copy of the patient set P;
2  Create the solution S_0 with fixed visit patterns per provider;
3  while P' is not empty do
4      Randomly select patient p from P';
5      Remove p from P';
6      Find all the feasible insertions I_p for p's visit patterns;
7      if I_p is not empty then
8          Apply to S_0 the insertion giving the smallest increase of
           travel time;
9      end if
10     return solution S_0;
11 end while
```

In terms of operators, we have adapted the classical removal and destroy operators from Shaw (1998) and Ropke & Pisinger (2006). These operators work on the feasible visit patterns described in Section 3.3. We list the

operators here with brief descriptions. The removal operator (Ropke & Pisinger, 2006) removes q patients per iteration, and we set q to 30% of the number of scheduled patients. We define $C_{p,n}$ to be the increase in the travel time arising from the insertion of patient p's nth best option.

Random removal. This operator randomly selects q scheduled patients and removes their visit patterns from the solution.

Worst removal. This operator computes, for each patient, the improvement in the travel time if the patient's visit pattern is removed. It then removes the q patients with the highest values.

Related removal. This operator randomly selects a patient and removes his/her visit pattern. Then it removes the $q-1$ most closely related patients. In our implementation, the relation between two patients is based on the percentage of shared time windows and the required number of visits: $R(p,p') = \frac{[r_p,d_p]\cap[r_{p'},d_{p'}]}{d_p - r_p} + min(1, \frac{v_{p'}}{v_p})$.

Random repair. This operator randomly selects an unscheduled patient p, computes the possible insertions of p's visit patterns, and applies the insertion with the lowest cost. This operation is repeated until all the unscheduled patients have been tested.

Greedy repair. This operator iteratively computes the possible insertions for the unscheduled patients and applies the insertion associated with the smallest increase of travel time. This operation is repeated until there are no more possible insertions.

Regret repair. This operator iteratively computes the possible insertions of the unscheduled patients and applies the best insertion for the patient with the highest regret value. Patient p's regret value is $C_{p,2} - C_{p,1}$.

During the *LNS*, the acceptance rule determines if the created solution can be accepted as the new current solution. It is based on simulated annealing as described in Ropke & Pisinger (2006). We set our initial temperature to $1.05 * f(S_0)$ and the decreasing temperature c to 0.99975. Finally, the termination of our *LNS* algorithm is based on two termination criteria: we stop after 20,000 iterations or 20 seconds of computation.

5. Computational results

In this section, we present experiments that compare the efficiency of all the proposed methods (two-steps subproblem, the pattern-based formulation and the matheuristic). To provide an extensive comparison, we have re-implemented the method proposed in Heching et al. (2019). We refer to their formulation as *Heching* and we use the 57 provided instances. These instances are based on real-data provided by a home care agency from Pennsylvania, US. Each instance contains 60 patients and 6 providers. Over those 60 patients, between 8 and 30 are "new" and a decision of acceptance or not has to be taken for these patients. The provided instances are split into three sets:

- *Classical*: Instances provided by their industrial partner;
- *Narrow*: Based on the *Classical* instances, with narrower patient time windows;
- *Fewer*: Based on the *Classical* instances, with fewer visits per patient.

We implemented the methods in C++ and performed the tests on a 2.7 GHz Intel Core i5 Macbook, with 16 Gb RAM using only one core. We solve the master problems (1)–(7) and (21)–(25) using Cplex 12.7.1 and the subproblems (8)–(14) and (15)–(20) using CP Optimizer version 12.7. Finally, for the *LBBD*s, the maximum computation time is set to 3600 s per instance.

5.1. Efficiency of the LBBD formulations

We now analyze the impact of the two-steps subproblem (3.2) and the pattern-based formulation (PBF). The results are given in Table 2. The first two columns present the instance name, the number of new patients and the number of patients accepted in the optimal solution. The *CPU* column gives the computation time in seconds. Finally, for *PBF*, *Nb Pattern* gives the number of feasible patterns computed and *TL* is the time limit (3600 s).

We observe that using the two-steps subproblem dramatically reduces the computation time (-36.14%) and outperforms *Heching* for 51 of 55 solved instances. In addition, according to Figure 2, *Heching*' subproblem has a failure rate (*Heching - Inf SubP*) of 80.65% in average while the two-steps subproblem only calls the whole subproblem (*Two_Steps - Call SubP*) 30.70% of the time and the subproblem is infeasible (*Two_Steps - Inf SubP*) only for 28.74% of those calls.

For the *Classical* and *Narrow* instances, the *PBF* dramatically outperforms the model proposed in Heching et al. (2019) even with the two-steps subproblem. The *PBF* solves all the *Classical* instances in less than 9 s and all the *Narrow* instances in less than 2 s. However, for the *Fewer* instances containing more than 25 new patients, we observe a large increase of computation time compared to *Heching*. This issue is due to the increase in the

number of generated patterns and therefore the size of the set partitioning problem. Nevertheless, *PBF* solves all the benchmark instances.

		Hecching	Alt. Subp.		PBF		
Instance	New / Accepted	CPU (s)	CPU (s)	% Improvement	CPU (s)	% Improvement	Nb Pattern
Classic_8	8/8	1.05	0.59	-43.81%	**0.01**	-99.05%	277
Classic_9	9/8	0.96	0.71	-26.04%	**0.01**	-98.96%	276
Classic_10	10/9	1.34	0.83	-38.06%	**0.02**	-98.51%	358
Classic_11	11/10	1.26	1.15	-8.73%	**0.02**	-98.41%	405
Classic_12	12/11	1.53	1.42	-7.19%	**0.02**	-98.69%	441
Classic_13	13/12	2.12	0.85	-59.91%	**0.05**	-97.64%	551
Classic_14	14/12	8.85	5.75	-35.03%	**0.11**	-98.76%	690
Classic_15	15/13	8.79	6.30	-28.33%	**0.11**	-98.75%	724
Classic_16	16/14	14.27	6.06	-57.53%	**0.16**	-98.88%	865
Classic_17	17/16	12.81	11.54	-9.91%	**0.45**	-96.49%	1171
Classic_18	18/16	22.14	14.11	-36.27%	**0.53**	-97.61%	1214
Classic_19	19/17	31.93	30.79	-3.57%	**0.87**	-97.28%	1275
Classic_20	20/17	97.78	47.67	-51.25%	**0.7**	-99.28%	1325
Classic_21	21/19	210.34	86.18	-59.03%	**1.2**	-99.43%	1403
Classic_22	22/20	185.70	96.46	-48.06%	**0.91**	-99.51%	1535
Classic_23	23/21	1048.01	1557.68	48.63%	**5.31**	-99.49%	1913
Classic_24	24/22	TL	TL	/	**5.32**	/	2032
Classic_25	25/24	646.88	676.69	4.61%	**3.3**	-99.49%	2309
Classic_26	26/25	2088.62	532.45	-74.51%	**8.44**	-99.60%	2543
Fewer_12	12/10	1.25	1.03	-17.60%	**0.07**	-94.40%	998
Fewer_13	13/11	1.23	1.11	-9.76%	**0.09**	-92.68%	1158
Fewer_14	14/12	2.15	1.35	-37.21%	**0.12**	-94.42%	1230
Fewer_15	15/13	1.82	1.15	-36.81%	**0.22**	-87.91%	1584
Fewer_16	16/14	2.20	1.58	-28.18%	**0.3**	-86.36%	1671
Fewer_17	17/15	2.92	2.43	-16.78%	**0.56**	-80.82%	1989
Fewer_18	18/16	3.87	2.08	-46.25%	**0.68**	-82.43%	2109
Fewer_19	19/17	4.04	3.35	-17.08%	**1.54**	-61.88%	2484
Fewer_20	20/19	4.78	2.11	-55.86%	**2.02**	-57.74%	2645
Fewer_21	21/20	4.63	2.39	-48.38%	**2.12**	-54.21%	2954
Fewer_22	22/21	4.85	**2.28**	-52.99%	2.53	-47.84%	3459
Fewer_23	23/23	11.05	**1.75**	-84.16%	5.98	-45.88%	3693
Fewer_24	24/24	4.78	**2.55**	-46.65%	4.12	-13.81%	3991
Fewer_25	25/25	19.16	**3.25**	-83.04%	5.61	-70.72%	4536
Fewer_26	26/26	5.09	**1.59**	-68.76%	5.61	10.22%	4875
Fewer_27	27/27	21.70	**4.27**	-80.32%	29.74	37.05%	4950
Fewer_28	28/28	49.97	**13.64**	-72.70%	125.98	152.11%	5108
Fewer_29	29/28	78.30	**21.17**	-72.96%	83.68	6.87%	5196
Fewer_30	30/29	398.6	**221.64**	-44.40%	3530.71	785.78%	5318
Narrow_8	8/7	0.95	0.67	-29.47%	**0.01**	-98.95%	243
Narrow_9	9/8	1.33	0.88	-33.83%	**0.01**	-99.25%	242
Narrow_10	10/9	1.67	0.75	-55.09%	**0.01**	-99.40%	296
Narrow_11	11/10	1.29	0.79	-38.76%	**0.01**	-99.22%	308
Narrow_12	12/11	0.97	1.03	6.19%	**0.01**	-98.97%	348
Narrow_13	13/12	2.16	0.98	-54.63%	**0.02**	-99.07%	394
Narrow_14	14/13	4.51	4.25	-5.76%	**0.04**	-99.11%	543
Narrow_15	15/14	4.08	2.20	-46.08%	**0.04**	-99.02%	568
Narrow_16	16/15	6.80	3.80	-44.12%	**0.08**	-98.82%	692
Narrow_17	17/16	7.38	4.50	-39.02%	**0.14**	-98.10%	823
Narrow_18	18/16	14.90	7.58	-49.13%	**0.24**	-98.39%	842
Narrow_19	19/17	17.81	11.41	-35.93%	**0.25**	-98.60%	846
Narrow_20	20/17	23.46	25.05	6.78%	**0.49**	-97.91%	860
Narrow_21	21/18	34.24	25.47	-25.61%	**0.54**	-98.42%	878
Narrow_22	22/19	73.74	68.79	-6.71%	**0.5**	-99.32%	948
Narrow_23	23/21	190.47	129.74	-31.88%	**1.23**	-99.35%	1212
Narrow_24	24/22	674.33	401.60	-40.44%	**1.13**	-99.83%	1317
Narrow_25	25/23	TL	TL	/	**1.38**	/	1452
Narrow_26	26/25	1303.29	1169.51	-10.26%	**1.89**	-99.85%	1594
Average				**-36.14%**		**-64.30%**	

Table 2: Results for the two-steps subproblem and visit pattern formulation

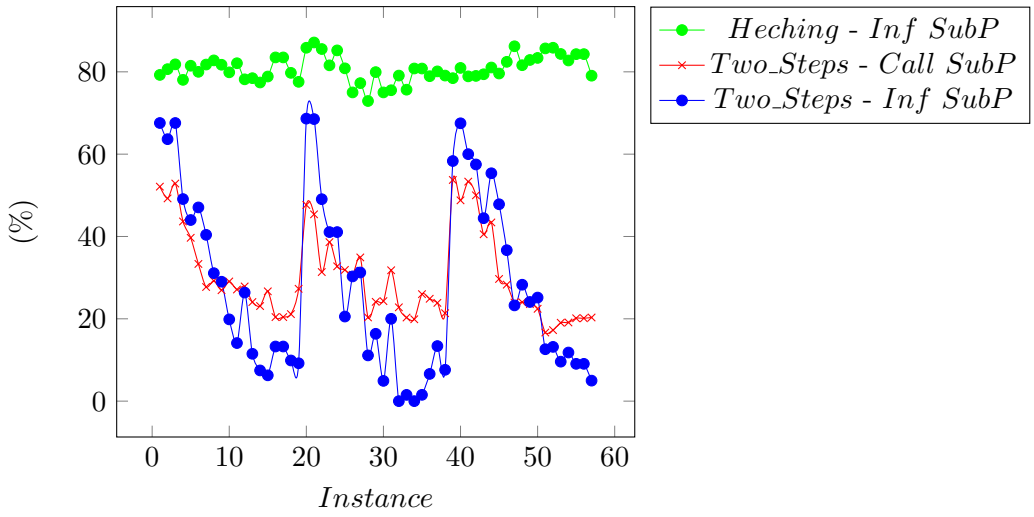

Figure 2: Comparison of the failure rates during the full subproblems resolutions

5.2. Efficiency of the matheuristic

In this section, we test the visit pattern matheuristic (VPM) proposed in Section 4. To do this, we solve the instances with the classical *LNS* (i.e., without the set partitioning resolution) and with the *VPM*. The results are given in Table 3. The columns Best and CPU Best correspond to the best found solution and the time (in seconds) at which this solution was found. The *LNS* optimally solves 37 of the 57 instances in less than 20 s or 20,000 iterations. With the same termination criteria, the *VPM* finds the optimal solution for all the instances. For most of the instances (51), the *VPM* finds the best solution in the first 10 s.

		Heching et al.	LNS			VPM		
Instance	Optimal Solution	CT	Best Solution	CT	Time Best Solution	Best Solution	CT	Time Best Solution
Classic_8	60	1.05	**60**	8.3	**0.06**	**60**	8.44	**0.06**
Classic_9	59	0.96	**59**	8.32	**0.16**	**59**	8.44	**0.16**
Classic_10	59	1.34	**59**	9.18	**0.01**	**59**	9.59	**0.01**
Classic_11	59	1.26	**59**	11.85	**0.11**	**59**	12.1	**0.11**
Classic_12	59	1.53	**59**	13.41	**0.01**	**59**	13.99	**0.01**
Classic_13	59	2.12	**59**	16.21	**0.03**	**59**	14.92	**0.03**
Classic_14	58	8.85	**58**	19.03	**0.05**	**58**	18.76	**0.05**
Classic_15	58	8.79	**58**	18.51	**0.03**	**58**	18.92	**0.03**
Classic_16	58	14.27	**58**	20	**0.06**	**58**	20	**0.06**
Classic_17	59	12.81	**59**	20	17.03	**59**	20	**4.16**
Classic_18	58	22.14	**58**	20	**0.7**	**58**	20	0.75
Classic_19	58	31.93	**58**	20	**3.12**	**58**	20	3.17
Classic_20	57	97.78	**57**	20	**0.02**	**57**	20	0.03
Classic_21	58	210.34	57	20	5.53	**58**	20	**10.49**
Classic_22	58	185.7	57	20	4.01	**58**	20	**5.58**
Classic_23	58	1048.01	57	20	1.12	**58**	20	**6.88**
Classic_24	58	TL	**58**	20	**3.5**	**58**	20	3.52
Classic_25	59	646.88	58	20	8.35	**59**	20	**16.18**
Classic_26	59	2088.62	57	20	0.68	**59**	20	**9.14**
Fewer_12	58	1.25	**58**	20	**0.16**	**58**	20	**0.16**
Fewer_13	58	1.23	**58**	20	**0.04**	**58**	20	**0.04**
Fewer_14	58	2.15	**58**	20	0.02	**58**	20	**0.01**
Fewer_15	58	1.82	**58**	20	**1.01**	**58**	20	1.03
Fewer_16	58	2.2	**58**	20	**0.09**	**58**	20	**0.09**
Fewer_17	58	2.92	**58**	20	**0.09**	**58**	20	0.1
Fewer_18	58	3.87	**58**	20	**0.15**	**58**	20	**0.15**
Fewer_19	58	4.04	**58**	20	**1.04**	**58**	20	**1.04**
Fewer_20	59	**4.78**	**58**	20	0.25	**59**	20	6.29
Fewer_21	59	4.63	**59**	20	**1.69**	**59**	20	1.96
Fewer_22	59	4.85	**59**	20	5.26	**59**	20	**4.83**
Fewer_23	60	11.05	59	20	0.16	**60**	20	**8.79**
Fewer_24	60	4.78	**60**	20	0.84	**60**	20	**0.8**
Fewer_25	60	19.16	**60**	20	16.66	**60**	20	**9.61**
Fewer_26	60	**5.09**	**60**	20	12.62	**60**	20	10.41
Fewer_27	60	21.7	59	20	0.29	**60**	20	**11.19**
Fewer_28	60	49.97	59	20	3.84	**60**	20	**12.2**
Fewer_29	59	78.3	**59**	20	9.99	**59**	20	**9.45**
Fewer_30	59	398.6	58	20	7.55	**59**	20	**13.93**
Narrow_8	60	0.95	**60**	8.22	**0.03**	**60**	7.9	**0.03**
Narrow_9	59	1.33	**59**	8.23	0.25	**59**	8	**0.24**
Narrow_10	59	1.67	**59**	9.79	**0.18**	**59**	9.86	**0.18**
Narrow_11	59	1.29	**59**	9.97	**0.2**	**59**	10.98	0.22
Narrow_12	59	0.97	**59**	11.67	**0.79**	**59**	11.39	**0.79**
Narrow_13	59	2.16	**59**	12.57	**0.11**	**59**	12.51	**0.11**
Narrow_14	59	4.51	58	14.94	0.01	**59**	14.86	**1.73**
Narrow_15	59	4.08	58	15.65	0.97	**59**	16.17	**1.81**
Narrow_16	59	6.8	58	20	0.18	**59**	20	**2.28**
Narrow_17	59	7.38	58	20	0.68	**59**	20	**2.73**
Narrow_18	58	14.9	**58**	20	10.59	**58**	20	**2.98**
Narrow_19	58	17.81	57	20	1.68	**58**	20	**3.3**
Narrow_20	57	23.46	56	20	0.62	**57**	20	**3.47**
Narrow_21	57	34.24	56	20	0.74	**57**	20	**3.49**
Narrow_22	57	73.74	**57**	20	16.93	**57**	20	**3.67**
Narrow_23	58	190.47	57	20	11.72	**58**	20	**9.35**
Narrow_24	58	674.33	57	20	11.86	**58**	20	**10.46**
Narrow_25	58	TL	**58**	20	8.39	**58**	20	**5.48**
Narrow_26	59	1303.29	57	20	1.81	**59**	20	**13.46**

Table 3: Results for the matheuristic

6. Conclusions

The *HCS-PV* is a complex problem that home care agencies have to solve every week. The goal is to assign and schedule a set of new patients given a

set of providers while taking into account the patients already present in the system. Each patient has a required number of visits and can be assigned to only one provider. The visit times must be the same for the entire horizon, and each provider has a maximum working time.

To solve this problem, we have extended the work of (Heching et al., 2019). First, we proposed a two-stage subproblem. Then, we presented a new *LBBD* based on visit patterns that includes more constraints in the master problem. Finally, we introduced a Dantzig-Wolfe formulation and developed a matheuristic based on *LNS*.

When computed on real instances from agencies in Pennsylvania provided in (Heching et al., 2019) our computational experiments show that our two-steps subproblem reduces the average computation time by 34%, while the new pattern-based formulation solves all the benchmark instances (53 out of the 57 instances are solved in less that 10 s). Finally, our matheuristic solves all the instances in less than 20 s. These results lead to two main observations. On the one hand, the pattern-based formulation, due to its speed, could allow the home care agencies to solve larger problems (especially in the $Classic$ and $Narrow$ contexts). Agencies will be able to take into account more providers simultaneously and this could potentially help them improve their offer of service. On the other hand, if the problems become too large, the proposed matheuristic seems to be a good alternative to achieve speed and efficiency. According to the experiments, the proposed matheuristic could allow the agencies to "play" with the algorithm in order to test different parameters (modification of the time windows, set of providers taken into account). This could help them to test different options and optimize their level of service.

In future research, we consider extending the problem to include more

practical constraints such as the possibility of having two providers required for the same visits or breaks in the providers' schedules. Finally, we think that solving this problem for only one week is not realistic enough and so, we would like to work on a dynamic version of this problem in order to solve it on a rolling horizon and analyze the impact of the decisions over time.

Acknowledgment

We thank Aliza Heching, John Hooker, and Ryo Kimura for providing access to the benchmark instances. This work has been supported by the Natural Sciences and Engineering Research Council of Canada.

References

References

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

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*, 238–252.

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

Borsani, V., Matta, A., Beschi, G., & Sommaruga, F. (2006). A home care scheduling model for human resources. *ICSSSM 2006*, *1*, 449–454.

Braekers, K., Hartl, R. F., Parragh, S. N., & Tricoire, F. (2016). A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. *European Journal of Operational Research*, *248*, 428–443.

Cheng, E., & Rich, J. L. (1998). A home health care routing and scheduling problem. *Working paper*, *NA*, NA.

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

Decerle, J., Grunder, O., El Hassani, A. H., & Barakat, O. (2018). A memetic algorithm for a home health care routing and scheduling problem. *Operations research for health care*, *16*, 59–71.

Di Gaspero, L., & Urli, T. (2014). A CP/LNS approach for multi-day homecare scheduling problems. *International Workshop on Hybrid Metaheuristics*, *1*, 1–15.

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

Frifita, S., Masmoudi, M., & 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.

Gamst, M., & Jensen, T. S. (2012). A branch-and-price algorithm for the

long-term home care scheduling problem. *Operations research proceedings 2011*, *1*, 483–488.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2017). A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Computers & Operations Research*, *84*, 116–126.

Grenouilleau, F., Legrain, A., Lahrichi, N., & Rousseau, L.-M. (2019). A set partitioning heuristic for the home health care routing and scheduling problem. *European Journal of Operational Research*, *275*, 295–303.

Heching, A., Hooker, J. N., & Kimura, R. (2019). A logic-based benders approach to home healthcare delivery. *Transportation Science*, *53*, 510–522.

Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming*, *96*, 33–60.

Lin, C.-C., Hung, L.-P., Liu, W.-Y., & Tsai, M.-C. (2018). Jointly rostering, routing, and rerostering for home health care services: A harmony search approach with genetic, saturation, inheritance, and immigrant schemes. *Computers & Industrial Engineering*, *115*, 151–166.

Macintyre, C. R., Ruth, D., & Ansari, Z. (2002). Hospital in the home is cost saving for appropriately selected patients: a comparison with in-hospital care. *International Journal for Quality in Health Care*, *14*, 285–293.

Rest, K.-D., & Hirsch, P. (2016). Daily scheduling of home health care services using time-dependent public transport. *Flexible Services and Manufacturing Journal*, *28*, 495–525.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*, 455–472.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*, 139–171.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431). Springer.

Sinha, M., & Bleakney, A. (2014). *Receiving care at home*. Statistics Canada.

Torres-Ramos, A., Alfonso-Lizarazo, E., Reyes-Rubiano, L., & Quintero-Araújo, C. (2014). Mathematical model for the home health care routing and scheduling problem with multiple treatments and time windows. *Proceedings of the 1st International Conference on Mathematical Methods & Computational Techniques in Science & Engineering*, *1*, 140–145.

Trautsamwieser, A., & Hirsch, P. (2014). A branch-price-and-cut approach for solving the medium-term home health care planning problem. *Networks*, *64*, 143–159.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., & Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, *230*, 231–244.

Yalçındağ, S., Cappanera, P., Scutellà, M. G., Şahin, E., & Matta, A. (2016). Pattern-based decompositions for human resource planning in home health care services. *Computers & Operations Research*, *73*, 12–26.

Zhang, T., Yang, X., Chen, Q., Bai, L., & Chen, W. (2018). Modified ACO for home health care scheduling and routing problem in chinese communities. *Networking, Sensing and Control (ICNSC), 2018 IEEE 15th International Conference on*, *15*, 1–6.

Appendix

	Element	Description
Master Problem	δ_p	Patient p is accepted
	$x_{a,p}$	Provider a is assigned to the patient p
	$y_{a,p,d}$	Patient p is visited by provider a the day d
	y	List of feasible $y_{a,p,d}$ values according to visit day K_p
Subproblem	SP_a	Subproblem for the provider a
	$P_{(SP_a)}$	List of patients currently assigned to a ($x_{a,p}*$)
	$P_{(SP_a),d}$	List of patients visited the day d ($y_{a,p,d}*$)
	$\pi_{d,v}$	Ordering variables representing provider's route on day d
	s_p	Visit time for the patient p
	$t_{\pi_{d,v},\pi_{d,v'}}$	Travel time between the locations $\pi_{d,v}$ and $\pi_{d,v'}$

Table 4: Description of the Benders decomposition's parameters