Long-term home care scheduling

Gamst, Mette; Jensen, Thomas Sejr

Publication date:
2011

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

Long-term home care scheduling

Report 12.2011

DTU Management Engineering

Mette Gamst
Thomas Sejr Jensen
September 2011

DTU Management Engineering
Department of Management Engineering

Long-term home care scheduling

M. Gamst* and T. Sejr Jensen□
*DTU Management Engineering, gamst@man.dtu.dk
□University of Southern Denmark, thomassejr@gmail.com

September 28, 2011

Abstract

In several countries, home care is provided for certain citizens living at home. Home care offers cleaning, grocery shopping, helping with personal hygiene and medicine, helping citizens to get in and out of bed, etc. The long-term home care scheduling problem is to generate work plans such that a high quality of service is maintained, the work hours of the employees are respected, and the overall cost is kept as low as possible. The problem covers several days of home care scheduling. A solution provides detailed information on visits and visit times, for each employee on each day.

We propose a branch-and-price algorithm for the long-term home care scheduling problem. The pricing problem generates a one-day plan for an employee, and the master problem merges the plans with respect to regularity constraints. The method is capable of generating plans with up to 99 visits during one week. This truly illustrates the complexity of the problem.

1 Introduction

In many countries a large number of citizens who live in their homes but are not able to do so without help, receive regular services from so-called home care centres. The citizens may receive a substantial number of visits during the week. There are many types of such visits: some are simple tasks like cleaning, bringing out food, doing the laundry etc., while others involve personal hygiene, medication, getting out of and into bed etc.

The goal of the home care center is to plan all these visits such that the perceived quality of service is high without overloading the individual home care personnel. The overall cost of performing this service must be kept as low as possible. This is highly nontrivial as the perceived quality of service is often in direct conflict with the cost of performing the service.

In practice, a two-level approach is used for planning home health care. The first phase is a *master-plan* which is a long-term plan. The second phase is *daily planning* which uses the master-plan as a starting point but incorporates last minute changes such as employees calling in sick, ad hoc visits, and other unforeseen events. This paper focuses on construction of the master-plan. The master-plan specifies when citizens are visited and the employees that conduct each visit.

Every visit in the master-plan is repeated regularly, as specified by its *period*. A visit with period p is repeated after p time, e.g., a period of one day means that the visit is conducted every day. Since every visit in the master-plan is repeated indefinitely, the same goes for the entire master-plan. In this sense the master-plan is *periodic*, which means that each week in the master-plan is rolled out to an infinite number of actual calendar weeks. If for example the length of the master-plan is four weeks, every fourth calendar week will be identical (until the master-plan is updated due to addition or removal of visits).

Quality of service partly consists of *regularity*, i.e., that a visit is always conducted at the same time of the day and that a citizen is visited by the same (small group of) employee(s). Especially the latter is of high importance to many citizens who feel safer when being serviced by persons they are familiar with.

The other aspect of quality of service is *skill set requirements*, i.e., assigning suitable employees to a visit. How suitable an employee is for a visit depends on the employee's professional skills and personality features of the employee and citizen.

Efficiency is quantified by the total time spent traveling between visits for all employees. Clearly, the total travel time should be as low as possible.

The long-term home care scheduling problem is $\mathcal{NP}$-hard and differs from previous work on home care scheduling by calculating detailed work plans for a longer period without having pre-fixed visits to specific days, and with taking quality of service into account. The latter aspect causes inter-dependent constraints between the daily work plans. It is thus not possible to calculate the daily plans independently from one another, even if all visits are fixed to specific days. We propose at branch-and-price algorithm for solving the problem to optimality. The pricing problem generates a plan for an employee on a given day, and the master problem merges the plans into an overall solution.

The proposed solution method has been tested on real-life data provided by Papirgården, which is a home care provider located on Funen, Denmark. It has about 25 employees and serves citizens spread over a diameter of about 3.5 kilometers. This means that all employees

travel by bicycle and an employee can, under normal circumstances, drive between any pair of citizens in less than 15 minutes. Computational results show that while outperforming CPLEX used for solving a basic formulation of the problem, the branch-and-price algorithm is unable to calculate large plans due to time and space usage. Instead the algorithm is well-suited for benchmarking heuristics. This truly illustrates the complexity of the problem.

This paper is organized as follows. First, a review of related work from the literature is presented in Section 2. In Section 3, the problem is formally defined and formulated mathematically. Section 4 contains the proposed branch-and-price solution method. The method is computationally evaluated in Section 5, and final conclusions are given in Section 6.

2 Related work

In this section, an overview is given on work from the literature on home care scheduling and related problems.

An early application of operations research methods to home health care was described by Begur et al. in [1]. The daily planning problem was modeled as a site dependent VRPTW, i.e., a VRPTW where a customer can only be visited by a subset of the vehicles. The problem was solved using the sequential savings heuristic developed by Clarke and Wright [4].

Eveborn et al. developed a decision support system for solving daily planning. The system is called Laps Care, and the recent paper [8] documents its success in real-life use. Laps Care modeled daily planning as a VRP which was solved heuristically. The heuristic consisted of iteratively combining paths using generalized matching, and splitting paths into single visits which are then combined with the remaining paths. In another paper about Laps Care [7], the authors assumed the existence of a base master-plan on which the daily plan was based, but they did not consider how to construct and maintain the master-plan.

The master's theses of Thomsen [17], Lessel [11], and Godskesen [9] also dealt with daily planning, and they all used heuristic solution methods. Thomsen and Godskesen modeled the problem as a so called *rich* VRP, i.e., a VRP with many real-life constraints and objectives which leads to complicated models, see e.g. [18].

Nikolajsen [13] considered a routing problem spanning several days, but each visit was conducted only once and therefore dependencies between individual days were not introduced. The work is related to the Periodic VRP (PVRP), where the planning horizon spans several days and each customer must be visited on a specified number of these days. The specified number of days for each customer is denoted its schedule: if every customer must be visited every day, PVRP reduces to VRP since the same path can be used every day. The schedules make PVRP differ from the long-term home care scheduling problem; in the latter visits at a citizen can be conducted on any (unspecified) day. PVRP was introduced by Russell and Igo [15] and defined formally by Christofides and Beasley [3]. A recent survey focusing on PVRP mentioned many real-life applications of PVRP, none of which was home health care [16].

Dohn et al. [10] modeled daily planning and solved the problem using branch-and-price. The master problem was a set partitioning model, where each column corresponded to a path for an employee. The pricing problem was the $\mathcal{NP}$-hard elementary shortest path problem with resource constraints and was solved using a labeling algorithm.

Bredström and Rönnqvist [2] introduced a mixed-integer mathematical formulation for the combined vehicle routing and scheduling problem with time windows and temporal constraints. They showed how the formulation is applicable on the daily planning problem.

3 Formal Problem Definition

This section introduces notation for planning home health care. Employees are introduced in Section 3.1, the planning horizon in Section 3.2, visits in Section 3.3, activities in Section 3.4, and finally the objectives of the problem in Section 3.5. The entities are gathered in a mathematical formulation of the long-term home care scheduling problem in Section 3.6. Note that the long-term home care scheduling problem and thus all entities described in the following operate in a discrete time space.

3.1 Employees

Let E denote the set of employees. Let H_j denote the work hours for employee j in a given period of k days. An interval $[a_{jh}, b_{jh}] \in H_j$ means that employee j is on duty from time a_{jh} to time b_{jh} on day $h \in \{1, \ldots, k\}$.

3.2 Planning horizon

The master-plan covers a period of time with length $L \in \mathbb{N}$, given as a number of discrete time steps. In order to include all visits properly, L should be set to the least common multiple of the visit periods. For example, if the plan includes visits, which are repeated every 2nd and 7th day, then L is 14 days. If the plan includes visits, which are repeated every 3rd and 7th day, then L is 21 days.

Some visits, for example those that are repeated infrequently, cause a large L and consequently a larger problem instance. This can be remedied in two ways. One way is to exclude such visits from the master-plan and only handle them during daily planning. Another way is to reduce the visit's period by dividing it by an integer k which divides evenly into the original period. This implies that the visit is repeated too often in the master-plan, so only every k'th repetition is included in daily planning. If, for example, a visit has a period of 21 days, dividing by 3 reduces the period to 7 days and only every 3rd repetition is included in daily planning.

3.3 Visits

The set of visits is denoted V. Each visit $i \in V$ is repeated at regular intervals, e.g., once per day or once per week, as indicated by its period which is denoted p_i. The travel time between two visits $i, j \in V$ is denoted $c_{ij} \in \mathbb{N}$ and is measured in time steps. The duration of visit i is $d_i \geq 0$, and its time window of feasible visiting times is $[a_i, b_i]$

Not all employees are equally suited for conducting a certain visit i, and some are not even allowed to do so. Therefore, $\mathbf{pr_i}$ is a vector of non-negative costs of letting each employee conduct i. If an employee is unsuited for conducting the visit, then the corresponding cost is infinite.

Finally, let C denote the set of all citizens receiving service and V_c the set of all visits for citizen $c \in C$.

3.4 Activities

Visit j with period p_j is repeated L/p_j times in the planning horizon of length L, and these repetitions are scheduled independently of each other. Therefore, visit j is rolled out to L/p_j

activities, and these activities are scheduled instead of j itself. Let A denote the set of all activities. The set of activities for visit $j \in V$ is denoted by A_j. Two consecutive activities i and k are denoted $(i,k): \ i,k \in A_j$.

Every activity $i \in A_j$ inherits most data from visit $j \in V$, so the duration of i is $d_i = d_j$, the time window is $W_i = W_j$, i.e., $[a_i, b_i] = [a_j, b_j]$, and the cost vector of letting employees conduct the activity is $\mathbf{pr_i}$. The travel time between two activities $i \in A_j$ and $k \in A_{j'}$ is $c_{ik} = c_{jj'}$.

In addition to the activities in A there is a special activity 0 representing the *depot*, which is the location where employees start and end their work days. The duration of the depot activity is zero, and it can be performed at all times.

A solution specifies start time and assigned employees for each activity. Let $s_i \in \mathbb{N}$ denote the start time of activity $i \in A$, let e_i denote the employee assigned to activity $i \in A$, and let $E_c \subseteq E$ denote the set of employees which are assigned to citizen $c \in C$. In the formal problem definition, we denote the daily schedule for an employee a *path*. A path consists of a list of activities with corresponding start visit times. The master-plan thus consists of paths for all employees on all days covering all activities.

3.5 Objective functions

This section defines the objective functions, i.e., the aspects of quality of service, how busy the home care employees are, and travel time. We choose to aggregate the objectives even though this is not trivial due to the relatively large number of objectives and the different units of measurement. How to weigh the objectives is not discussed any further here but is instead considered in Section 5.

3.5.1 Travel time

Let $A(h) \subseteq A$ denote the set of activities conducted in work shift $h \in H_j$ of employee $j \in E$. Let $\sigma_i \in A(h)$ denote the successor of activity i. The total travel time is computed as:

$$f_{TT}(S) = \sum_{j \in E} \sum_{h \in H_j} \sum_{i \in A(h)} c_{i\sigma_i} \tag{1}$$

Since the start and end depots are considered as activities in $A(h)$, we define the successor of the end depot to also be the end depot and we let the corresponding travel time be zero.

3.5.2 Employee priority

An activity $i \in A$ should be conducted by the most suitable employees, as specified by the non-negative cost vector $\mathbf{pr_i}$. Let $pr_i(j) \geq 0$ denote the cost of employee $j \in E$ conducting activity $i \in A$. If employee $j \in E$ does not have the required skill set for conducting the activity, then $pr_i(j) = \infty$. The employee priority objective is defined as:

$$f_{EP}(S) = \sum_{i \in A} \sum_{j \in E} pr_i(j). \tag{2}$$

3.5.3 Busyness

An employee is said to be *busy* if the time period in the schedule violates constraints on time windows and/or travel time. Specifically, busyness appears in the following cases:

1. The start times at two activities are too close, i. e., $s_{\sigma_i} - (s_i + d_i + c_{i\sigma_i}) < 0 \Leftrightarrow s_i + d_i > s_{\sigma_i} - c_{i\sigma_i}$
2. The start time at an activity is too late, i.e., let b_{jh} be the end time window of employee j performing activity i on day h then $s_i + d_i > b_{jh}$.

In both cases, busyness is penalized in the objective function. There are two reasons for treating busyness as an objective rather than a hard constraint in the master-plan. First of all, it reflects how the plan is constructed manually. Second, it introduces slack in the model such that constraints concerning time windows for visits and overtime for employees can be enforced without constraining the solution space to the point where no feasible solution exists. The busyness objective is computed as:

$$f_B(S) = \sum_{j \in E} \sum_{h \in H_j} \sum_{i \in A(h)} \max(0, s_i + d_i - \min\{s_{\sigma_i} - c_{i\sigma_i}, b_{jh}\})$$

If a work shift is empty, its busyness is zero by definition. Any busyness in the master-plan is taken care of during daily planning, which spreads out activities or assign some activities to another employee.

3.5.4 Employee regularity

The employee regularity objective counts the number of different employees visiting each citizen:

$$f_{ER}(S) = \sum_{c \in C} |E_c|. \tag{3}$$

3.5.5 Visit periods

Recall that visits are rolled out to activities such that the latter are scheduled independently, allowing more flexibility in the master-plan. Let $(i, k) : \ i, k \in A_j$ be two consecutive activities for visit $j \in V$. If the time period between s_i and s_k differs from the given time period of the visit, p_j, then the objective is penalized:

$$f_{VP}(S) = \sum_{j \in V} \sum_{(i,k):\ i,k \in A_j} |s_i + p_j - s_k| \tag{4}$$

3.6 Mathematical Formulation

The long-term home care scheduling problem can now be defined mathematically. Recall the notation introduced previously. Furthermore, define:

$$[a_{ij}^h, b_{ij}^h] = [\max\{a_i, a_{jh}\}, \min\{b_i - d_i, b_{jh}\}]$$

As mentioned in Section 3.5 we optimize an aggregated objective function, where each part is weighed appropriately. Let $\mathbf{w}$ be a non-negative vector of such weights, where $w^{TT} \geq 0$ is

the weight for travel times, $w^{EP} \geq 0$ is for employee priority, $w^{B} \geq 0$ is for busyness, $w^{ER} \geq 0$ is for employee regularity, and $w^{VP} \geq 0$ is the weight vector for visit periods. Also, let M be some large number.

In addition to start time variables $s_i \geq 0$, the variables are as follows. Let $x_{ik}^{jh} \in \{0,1\}$ equal one iff employee $j \in E$ travels from activity i to k on day h. Let $u_{ik} \geq 0$ denote the difference in the start times between two consecutive activities $(i,k) : \ i,k \in A_j$ for visit $j \in V$. Let $z_{ij} \geq 0$ denote the busyness of employee $j \in E$ caused by activity $i \in A$. Finally, let $y_c^j \in \{0,1\}$ equal one iff employee $j \in E$ visits citizen $c \in C$.

The long-term home care scheduling problem is formulated as:

$$\min \sum_{j\in E}\sum_{h\in H_j}\sum_{i\in A}\sum_{k\in A}\left(w^{TT}\ c_{ik} + w^{EP}\ pr_i(j)\right)\ x_{ik}^{jh} + \sum_{i\in A}\sum_{j\in E} w^{B}\ z_{ij} + \sum_{c\in C}\sum_{j\in E} w^{ER}\ y_c^j + \sum_{j\in V}\sum_{(i,k):\ i,k\in A_j} w^{VP}\ u_{ik} \tag{5}$$

$$\text{s. t.} \quad \sum_{k\in A}\sum_{j\in E}\sum_{h\in H_j} x_{ik}^{jh} = 1 \qquad \forall i \in A\backslash\{0\} \tag{6}$$

$$\sum_{k\in A} x_{ik}^{jh} - \sum_{k\in A} x_{ki}^{jh} = 0 \qquad \forall i \in A\backslash\{0\},\ \forall j \in E,\ \forall h \in H_j \tag{7}$$

$$\sum_{k\in A} x_{0k}^{jh} = \sum_{k\in A} x_{k0}^{jh} = 1 \qquad \forall j \in E,\ \forall j \in H_j \tag{8}$$

$$s_i + d_i + c_{ik} - M(1 - x_{ik}^{jh}) \leq s_k + z_{kj} \qquad \forall i,k \in A, \forall j \in E,\ \forall h \in H_j \tag{9}$$

$$s_i \geq a_{ij}^h - M(1 - \sum_{k\in A} x_{ik}^{jh}) \qquad \forall i \in A, \forall j \in E, \forall h \in H_j \tag{10}$$

$$s_i \leq b_{ij}^h + M(1 - \sum_{k\in A} x_{ik}^{jh}) \qquad \forall i \in A, \forall j \in E, \forall h \in H_j \tag{11}$$

$$s_i + p_j - s_k \leq u_{ik} \qquad \forall (i,k) :\ i,k \in A_j, \forall j \in V \tag{12}$$

$$s_k - (s_i + p_j) \leq u_{ik} \qquad \forall (i,k) :\ i,k \in A_j, \forall j \in V \tag{13}$$

$$\sum_{h\in H_j}\sum_{k\in A} x_{ik}^{jh} \leq y_c^j \qquad \forall c \in C, \forall v \in V_c, \forall i \in A_v, \forall j \in E \tag{14}$$

$$s_i + M(1 - \sum_{j\in E}\sum_{h\in H_j} x_{ik}^{jh}) \leq s_k \qquad \forall i,k \in A \tag{15}$$

$$x_{ik}^{jh} \in \{0,1\} \qquad \forall i,k \in A,\ \forall j \in E,\ \forall h \in H_j \tag{16}$$

$$s_i \geq 0 \qquad \forall i \in A \tag{17}$$

$$z_{ij} \geq 0 \qquad \forall i \in A,\ \forall j \in E \tag{18}$$

$$u_{ik} \geq 0 \qquad \forall (i,k) :\ i,k \in A_j,\ \forall j \in V \tag{19}$$

$$y_c^j \in \{0,1\} \qquad \forall c \in C,\ \forall j \in E \tag{20}$$

The objective function (5) minimizes a weighted sum of the objectives described in Section 3.5. First part of the objective function consists of travel times and employee priority. If $x_{ik}^{jh} = 1$ for activities $i,k \in A$, employee $j \in E$ and day $h \in H_j$, then the objective must pay the travel time from i to k and the cost of letting employee j conduct activity i. Next, the objective function consists of three parts: busyness, employee regularity and visit periods.

Constraints (6) ensure that all activities are conducted. Constraints (7) ensure that a path of an employee is connected, and constraints (8) force paths to start and end in the depot.

Constraints (9) measure busyness: if a visit at activity k starts too late, then busyness is added to the variable $z_{kj} \geq 0$. Furthermore, constraints (9) eliminate subtours. Constraints (10) and (11) say that any variable s_k must satisfy the time windows of activity $k \in A$ and employee $j \in E$.

Constraints (12) and (13) measure the amount of time difference between two consecutive visits $(i,k) : \ i,k \in A_j, \ j \in V$. If the gap between the start times at i and k differs from p_j, then the variable u_{ik} is set to the absolute time deviation.

Constraints (14) count the number of employees visiting a citizen: if employee $j \in E$ visits activity $i \in A_v, \ v \in V_c, c \in C$, then the variable y_c^j is set to one. Constraints (15) limit the amount of allowed busyness such that visit times on a path are non-decreasing. Finally, bounds (16) - (20) force variables to take on feasible values.

4 Exact Solution Approach

A branch-and-price algorithm for solving the long-term home care scheduling problem is presented in this section. The mathematical formulation (5) - (20) is Dantzig-Wolfe decomposed [5]. The pricing problem generates a path for a given employee on a given day, and the master problem merges the paths into an overall feasible solution.

Let p be a path and P the set of all generated paths. The master problem contains three types of variables. Variable $x_p \in \{0,1\}$ denotes whether or not path p is part of the solution. Variables $u_{ik} \geq 0$ and $y_c^j \in \{0,1\}$ are as defined for the original formulation (5) - (20).

Each path p has a number of constants attached. Constant δ_p^i is set to one, if path $p \in P$ visits activity $i \in A$, otherwise δ_p^i is zero. Let constant $\delta_{bp}^{ij} \geq 0$ denote the amount of busyness in path $p \in P$ for employee $j \in E$ and activity $i \in A$. Let δ_{sp}^i denote the start time of path $p \in P$ at activity $i \in A$, and let δ_{sp}^i be undefined if $\delta_p^i = 0$. Let δ_p^{ij} be set to one, if path $p \in P$ is generated for employee $j \in E$ and visits activity $i \in A$, otherwise δ_p^{ij} is zero. Constant δ_p^{jh} is set to one, if path $p \in P$ is generated for employee $j \in E$ on day $h \in H_j$, otherwise δ_p^{jh} is zero. Finally, let $c_p \geq 0$ denote the total travel time in plan p.

The master problem is formulated as:

$$\min \quad \sum_{p\in P} w^{TT} c_p x_p + \sum_{i\in A}\sum_{j\in E}\sum_{p\in P} w^{EP}\ pr_i(j)\ \delta_p^{ij} x_p \quad + \sum_{i\in A}\sum_{j\in E}\sum_{p\in P} w^{B}\ \delta_{bp}^{ij}\ x_p + \sum_{c\in C}\sum_{j\in E} w^{ER}\ y_c^j + \sum_{j\in V}\sum_{(i,k):\ i,k\in A_j} w^{VP}\ u_{ik} \tag{21}$$

$$\text{s. t.} \quad \sum_{p\in P} \delta_p^i x_p = 1 \qquad \forall i \in A \tag{22}$$

$$\sum_{p\in P} \delta_{sp}^i x_p + p_j - \sum_{p\in P} \delta_{sp}^k x_p \le u_{ik} \qquad \forall (i,k):\ i,k\in A_j,\ \forall j \in V \tag{23}$$

$$\sum_{p\in P} \delta_{sp}^k x_p - (\sum_{p\in P} \delta_{sp}^i x_p + p_j) \le u_{ik} \qquad \forall (i,k):\ i,k\in A_j,\ \forall j \in V \tag{24}$$

$$\sum_{p\in P} \delta_p^{ij} x_p \le y_c^j \qquad \forall c \in C, \forall v \in V_c, \forall i \in A_v, \forall j \in E \tag{25}$$

$$\sum_{p\in P} \delta_p^{jh} x_p \le 1 \qquad \forall j \in E,\ \forall h \in H_j \tag{26}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{27}$$

$$u_{ik} \ge 0 \qquad \forall (i,k):\ i,k\in A_j,\ \forall j \in V \tag{28}$$

$$y_c^j \in \{0,1\} \qquad \forall c \in C, \forall j \in E \tag{29}$$

The objective function (21) corresponds to the objective in the original formulation (5), only with variables x_p instead of x_{ik}^{jh}. The objective function consists of the weighted sum of travel times, employee priorities, busyness, employee regularity and visit periods.

Constraints (22) ensure that every activity is visited. Constraints (23) and (24) measure time deviation similar to constraints (12) and (13) in the original formulation. Constraints (25) measure employee regularity. Constraints (26) ensure that at most one path per employee per day is part of a solution (note that the original formulation forces every employee to leave the depot every day, but it is feasible for an employee to travel from the depot activity back to the depot activity, see constraints (7) and (8). This corresponds to not assigning a path to an employee in constraint (26)).

The number of columns in the master problem is reduced by fixing certain activities to certain days. Specifically, if a visit must be repeated every day, then the corresponding seven activities are fixed to Monday, Tuesday, Wednesday, etc., respectively. The pricing problem only allows such activities to be part of paths on appropriate days.

4.1 Pricing problem

Associate the dual variables $\pi_i^{(22)} \in \mathbb{R}$ with constraints (22), $\pi_{ik}^{(23)} \le 0$ with constraints (23), $\pi_{ik}^{(24)} \le 0$ with constraints (24), $\pi_{icj}^{(25)} \le 0$ with constraints (25), and $\pi_{jh}^{(26)} \le 0$ with constraints (26).

The pricing problem tries to generate a path with negative reduced cost for a given em-

ployee $j \in E$ on a given day $h \in H_j$. The reduced cost is defined as:

$$\begin{aligned}\bar{c}_{jh} = & \sum_{i,k \in A} w^{TT}\ c_{ik} + \sum_{i \in A} w^{EP}\ pr_i(j) + \sum_{i \in A} w^{B}\ z_{ij} \\ & - \sum_{i \in A} \pi_i^{(22)} - \sum_{v \in V} \sum_{(i,k):\ i,k \in A_v} \left((s_i - s_k)\pi_{ik}^{(23)} + (s_k - s_i)\pi_{ik}^{(24)} \right) \\ & - \sum_{c \in C} \sum_{v \in V_c} \sum_{i \in A_v} \pi_{icj}^{(25)} - \pi_{jh}^{(26)} \leq 0 \end{aligned} \tag{30}$$

where $z_{ij} \geq 0$ denotes the amount of busyness for employee $j \in E$ at activity $i \in A$, and $s_i \geq 0$ denotes the start time at activity i. The following notation is introduced to simplify Inequality (30). Let:

$$\bar{c}_0^i = \begin{cases} s_i(\pi_{ik}^{(23)} - \pi_{ik}^{(24)}) & \exists k : (i,k) :\ i,k \in A_v,\ v \in V \\ 0 & \text{otherwise} \end{cases}$$

and:

$$\bar{c}_1^i = \begin{cases} s_i(\pi_{ki}^{(24)} - \pi_{ki}^{(23)}) & \exists k : (k,i) :\ i,k \in A_v,\ v \in V \\ 0 & \text{otherwise} \end{cases}$$

The reduced cost for visiting an activity $i \in A_v,\ v \in V_c,\ c \in C$ can now be expressed as:

$$\bar{c}_{jh}^i = \ w^{EP}\ pr_i(j) - \pi_i^{(22)} - \pi_{icj}^{(25)} - \bar{c}_0^i - \bar{c}_1^i$$

A master-plan makes sure that an activity $i \in A$ is visited exactly once, hence we know that exactly one employee leaves activity $i \in A$ exactly once on exactly one day. For that reason, it is feasible to define the reduced cost between any two activities $i, k \in A$ as:

$$\bar{c}_{jh}^{ik} = \ w^{TT}\ c_{ik} + \bar{c}_{jh}^i$$

Now, the reduced cost (30) is rewritten as:

$$\bar{c}_{jh} = \ \sum_{i \in A} \sum_{k \in A} \bar{c}_{jh}^{ik} + w^B \sum_{i \in A} z_{ij} \leq \pi_{jh}^{(26)} \tag{31}$$

The pricing problem is solved for each employee $j \in E$ on day $h \in H_j$, hence $\pi_{jh}^{(26)}$ is a constant and isolated on the right-hand side. If the pricing problem generates a path where $\bar{c}_{jh} < \pi_{jh}^{(26)}$, then the path has negative reduced cost and the corresponding column may be added to the master problem.

Now, using constraints similar to the original formulation (5) - (20), the pricing problem for employee $j \in E$ on day $h \in H_j$ is formulated as:

$$\min \quad \sum_{i\in A}\sum_{k\in A} \bar{c}^{ik}_{jh} x_{ik} + w^B \sum_{i\in A} z_{ij} \tag{32}$$

$$\text{s. t.} \quad \sum_{k\in A} x_{ik} - \sum_{k\in A} x_{ki} = 0 \qquad \forall i \in A\backslash\{0\} \tag{33}$$

$$\sum_{k\in A} x_{0k} = \sum_{k\in A} x_{k0} = 1 \tag{34}$$

$$s_i + d_i + c_{ik} - M(1 - x_{ik}) \leq s_k + z_{kj} \qquad \forall i,k \in A \tag{35}$$

$$s_i \geq a^h_{ij} - M(1 - \sum_{k\in A} x_{ik}) \qquad \forall i \in A \tag{36}$$

$$s_i \leq b^h_{ij} + M(1 - \sum_{k\in A} x_{ik}) \qquad \forall i \in A \tag{37}$$

$$s_i + M(1 - x_{ik}) \leq s_k \qquad \forall i,k \in A \tag{38}$$

$$x_{ik} \in \{0,1\} \qquad \forall i,k \in A \tag{39}$$

$$s_i \geq 0 \qquad \forall i \in A \tag{40}$$

$$z_{ij} \geq 0 \qquad \forall i \in A \tag{41}$$

The objective function (32) minimizes the reduced cost as defined in (31). Constraints (33) and (34) ensure path connectivity and that the path starts and ends in the depot. Constraints (35) - (37) make sure that the start time s_i at activity $i \in A$ is set appropriately, that any busyness is added to the variable z_{ij} and that subtours are eliminated. Constraints (38) limit the amount of allowed busyness by ensuring that start times on the path are non-decreasing. Finally, the bounds (39) - (41) force variables to take on feasible values.

The pricing problem is recognized as a shortest path problem with time constraints and potentially negative edge weights defined by (32). This is also denoted the Elementary Shortest Path Problem with Resource Constrained (ESPPRC). An instance of ESPPRC consists of a number of resources and a weighted graph whose edges and vertices consume resources and have a lower and upper bound on total consumption of each resource. The task is to find a shortest simple path from node s to node t. The resource consumption at every node and every edge on this path must be within the specified bounds. In this application of ESPPRC, a node corresponds to an activity, an edge travels between two activities, edges have no resource bounds, and the resource bounds on vertices are time windows of the activities and employees. The weight of an edge is determined by (32).

Because ESPPRC is $\mathcal{NP}$-hard, see Dror [6], we first try to solve the pricing problem heuristically. If the heuristic cannot find any path with negative reduced cost for any employee on any day, then the pricing problem is solved to optimality.

Labeling algorithm

Both the heuristic and exact solution approaches for the pricing problem use a *labeling* algorithm. The approach associates a set of labels with each activity. A *label* for activity i represents a path from the (source) depot activity s to i. Associated with a label ℓ are the following attributes:

- The last visited activity $v(\ell)$

- The start time $t(\ell)$ of the last visited activity $v(\ell)$
- The set of activities, which ℓ can be extended to, denoted $extendables(\ell)$
- The reduced cost of the label $reduced\_cost(\ell)$

A label can be extended to activities, which have not yet been conducted, and whose time window is still open. The reduced cost of a label is defined in (32) and is based on the path leading up to $v(\ell)$.

Algorithm 1 is a generic label-setting algorithm that finds a shortest paths with negative reduced cost from node s to t. A *bucket* is in this context a set of labels, and we use one bucket for each activity.

Algorithm 1 Generic labeling algorithm which computes a set $\mathcal{P}$ containing up to k resource constrained shortest paths with negative reduced cost from node s to t. The set of all buckets is denoted by $\mathcal{B}$.

```
1:  P ← ∅
2:  ℓ_init ← initialize_label(s)
3:  B(s) ← B(s) ∪ {ℓ_init}
4:  while a non-empty bucket in B exists and |P| < k do
5:    L ← dequeued non-empty bucket from B
6:    for all ℓ ∈ L do
7:      for all activity i ∈ extendables(ℓ) do
8:        for all feasible start times s_i of i do
9:          ℓ' ← create_label(i, s_i)
10:         if i = t then
11:           if reduced_cost(ℓ') < π_jh^(26) then
12:             p ← get_path(ℓ)
13:             if keep_path(p) then
14:               P ← P ∪ {p}
15:             end if
16:           end if
17:         else
18:           B(i) ← B(i) ∪ {ℓ}
19:           remove_dominated(B(i), ℓ)
20:         end if
21:       end for
22:     end for
23:   end for
24: end while
```

In the first three lines of the Algorithm, an empty set of solutions is initialized along with a label in the (source) depot activity. The latter is added to the bucket of labels in the depot activity. The algorithm then extracts some non-empty bucket in Line (5). In Line (6)-(9) the algorithm extends the path of each label in the bucket with feasible activities and start times, which results in new labels. If a path has reached the (target) depot activity t, the reduced cost is negative, and we wish to save the path, then it is added to the set of solutions, see Line (10)-(14).

We wish to save a path p if:

- No more than 100 paths are already saved. The number "100" is reached through parameter tuning, or
- More than 100 paths are saved and the new path has smaller reduced cost than another saved path

If the (target) depot activity t has not been reached in Line (10), then the label is added to the appropriate bucket. Finally, in Line (19) the algorithm checks if the new label ℓ *dominates* any labels in the bucket or if it is *dominated* by any labels in the bucket.

A label ℓ *dominates* another label ℓ' if the following conditions hold:

- The labels ℓ and ℓ' end at the same activity: $v(\ell) = v(\ell')$
- The reduced cost of ℓ is no greater than that of ℓ': $reduced\_cost(\ell) \leq reduced\_cost(\ell')$
- The path of ℓ ends no later than that of ℓ': $t(\ell) \leq t(\ell')$
- Label ℓ can at least extend to the same activities as ℓ'

These criteria ensure that if at least one path with negative reduced cost exists, then the labeling algorithm is guaranteed to find it.

When using Algorithm 1 as a heuristic, only the first label added to every bucket is processed. Keeping only one label in each bucket speeds up the pricing algorithm significantly and it is still capable of finding columns with negative reduced cost in the first iterations of the branch-and-price algorithm.

4.2 Branching strategy

Branching is necessary when the optimal solution in a branch node is fractional. Fractional solutions occur in the following situations:

Fractional citizen visits: That is $0 < y_c^j < 1$ for some citizen $c \in C$ and employee $j \in E$.
In this case two branching children are generated with added cut:

$$y_c^j = 0 \quad \text{resp.} \quad y_c^j = 1.$$

This does not change the pricing problem, because the cut is not on the x_p-variables.

An activity is visited by several employees or on several days: That is, $0 < x_p,\ x_{p'} < 1$ for paths $p, p' \in P$, constants $\delta_p^i = \delta_{p'}^i = 1$ for some activity $i \in A$, and constants $\delta_p^{jh} = \delta_{p'}^{j'h'} = 1$ for some employee(s) $j, j' \in J$ and day(s) $h, h' \in H_j$ with either $j \neq j'$ or $h \neq h'$.
In this case two branching children are generated with the following rules:

$$\sum_{p \in P} \delta_p^{jh} x_p = 0 \quad \text{resp.} \quad \sum_{p \in P} \delta_p^{j'h'} x_p = 0$$

The branching rule is maintained in the pricing problem, which ensures that employee j (resp. j') never visits activity i on day h (resp. h'). The branching rule does not complicate the structure of the pricing problem.

An employee travels on an edge on a given day a fractional number of times: That is, $0 < x_p,\ x_{p'} < 1$ for paths $p, p' \in P$, and constants $\delta_p^{jh} = \delta_{p'}^{jh} = 1$ for some employee $j \in J$ and day $h \in H_j$. Let i be the first activity from which the paths p and p' differ. Let $i, k, k' \in A$, such that p travels from i to k at time s_{ik} and such that p' travels from i to k' at time $s_{ik'}$. Furthermore, let $k \neq k'$, or $s_{ik} \neq s_{ik'}$.
Two branching children are generated with the following rules:

$$\sum_{p \in P} \delta_p^{jh} \delta_p^{s_{ik}} x_p = 0 \quad \text{resp.} \quad \sum_{p \in P} \delta_p^{jh} \delta_p^{s_{ik'}} x_p = 0$$

where constant $\delta_p^{s_{ik}}$ (resp. $\delta_p^{s_{ik'}}$) is set to one, if path p travels from i to k at time s_{ik} (resp. from i to k' at time $s_{ik'}$), otherwise it is set to zero. The branching rule is maintained by the pricing problem, which ensures that employee j never travels from i to k (resp. k') at time s_{ik} (resp. $s_{ik'}$) on day h. The branching rule does not complicate the structure of the pricing problem.

Together the three branching strategies are finite and eventually ensure an integer solution. The strategy generates branching children in the order given above, and best first is used as search strategy in the branch-and-bound tree. Strong branching is applied: for each branching candidate, a lower bound on its LP relaxation of each of the children is obtained. The candidate that leads to children with the lowest bounds is selected.

4.3 Incumbent

Before the branch-and-price process can begin, an initial solution to the long-term home care scheduling problem must be generated. The solution, also denoted the *incumbent*, is used for finding initial values for the dual variables.

Algorithm 2 tries to assign activities to the first employee on the first day, with respect to time windows and fixed days. If unable to assign an activity to this employee and day, the algorithm tries to assign it to the employee on the next day. Eventually, the algorithm tries to assign the activity to the next employee on the first day, etc. An activity is assigned to an employee in a feasible way and such that busyness is avoided when possible, see Line (8) and (14), i.e., the algorithm only allows busyness to occur when having reached the last employee on the last day of the instance. Lines (13) - (15) measure travel time and busyness for returning to the depot.

The algorithm always finds a feasible solution if one such exists. The reasons for this are that activities are sorted according to their end times in non-decreasing order, and that busyness is allowed.

5 Computational Results

The exact solution method is tested on a number of real-life benchmark instances. In this section, the benchmark instances are first introduced. This is followed by computational results for the branch-and-price algorithm.

5.1 Real-life test instance

The proposed solution method is tested on real-life data provided by Papirgården, a home care center in Funen, Denmark. The real-life instances consist of up to 99 activities to be

Algorithm 2 Construction heuristic used in the branch-and-price algorithm to generate an initial solution.

```
1: for all employees j ∈ E do
2:   for all days h ∈ H_j do
3:     t ← a_jh (start of work shift)
4:     i ← 0 (the depot)
5:     sort activities in non-descending order of b_i (end of time window)
6:     for all activities k ∈ A do
7:       if k is unscheduled and can be scheduled on day h then
8:         Update t according to c_ik and b^h_kj
9:         start k at time t and assign employee j
10:        i ← k
11:      end if
12:    end for
13:    k ← 0 (the depot)
14:    Update t according to c_ik and b^h_kj
15:    start k at time t and assign employee j
16:  end for
17: end for
```

conducted in 7 days by at most 2 employees. The time window of an activity is set to either 7.30–9.00, 9.00–11.00, 11.30–13.00, or 13.00–15.00. Employees work from 7.00 – 13.00, 7.00 – 14.00 or 7.00 – 15.00. Time is discretized into either 5 or 10 minute time steps.

The distances between citizens are found with the Google Maps API, which means that the actual distance is used in the master-plan rather than straight line distances. When computing travel times, we assume that an employee travels at 15 kilometers per hour (all employees travel by bicycle). Unless two activities are at the same citizen, two minutes are added to the travel time between them to account for the time it takes to enter and leave a residence. All travel times are ceiled to nearest integer.

Weights must be set for the aggregated objective function. Papirgården has recommended the following priorities: highest priority is given to minimizing busyness and employee regularity, followed by minimizing travel times. Visit regularity has fourth priority, and employee skill requirements are of no importance for Papirgården, because its employees have similar skill sets. Visit regularity is given low priority, because the time window of an activity is relatively small and because high priority is given to reduce busyness. Including the priorities in the objective function is done by assigning large numbers to the weights, i.e., $w^{TT} = 500$, $w^{B} = 750$, $w^{EP} = 0 \cdot 5/\tau$, $w^{ER} = 750 \cdot 5/\tau$ and $w^{VR} = 50$, where τ is the number of minutes per time step. The travel time, busyness, and visit regularity objectives depend on the time discretization, while this is not the case for employee priority and employee regularity. The weights of the latter two are thus multiplied with a time step dependent factor.

5.2 Results

The branch-and-price algorithm has been implemented using the COIN Bcp framework, see Lougee-Heimer [12], and is tested on an Intel 2.13GHz Xeon CPU with 4 cores and 8 GB RAM. Note that all results stem from using one core. CPLEX 12.1 was used as standard MIP

solver for both solving the original formulation (5)-(20) and for solving the master problem in the branch-and-price algorithm.

CPLEX is unable to solve the original formulation (5)-(20) for instances with more than 11 activities and is thus not evaluated further. For an instance with 12 activities, CPLEX was stopped after 75 000 seconds. It had generated a search tree with 4 057 600 nodes, but only reduced the gap between the upper and lower bound from an initial 14.8% to 8.25%. Applying CPLEX on the original formulation is not a successful strategy, hence the rest of this section concentrates on the branch-and-price algorithm.

Test results are summarized in Table 1 and 2. The first table displays results for the optimal branch-and-price algorithm, while the second table displays results for the branch-and-price algorithm with only heuristically generated columns.

An instance is named "$|E| - |A| - \tau$", where τ is the number of minutes per time step, i.e., either 5 or 10 minutes. All instances have a planning horizon of 7 days. A time limit of 30 minutes has been imposed on the runs, and an "*" in the last column indicates that an provably optimal solution was not found within these 30 minutes (some runs exceeded this limit slightly because elapsed time is not checked everywhere in the program).

As can be seen in Table 1, only four instances with 5 minute time steps can be solved to optimality within half an hour. A coarser discretization helps, but the branch-and-price algorithm still suffers from a large time usage.

An interesting observation for instances with 10 minutes time steps is that the instances with 30 activities time out, whereas the instances with 33 and 40 activities are solved to optimality. This is due to the fact that the instances with 30 activities contain many visits with a period of one week, which can be scheduled on any day. The other instances have more visits with a period of one day whose activities are thus fixed to specific days.

The number of columns is large for several instances, which is caused partly by large time windows and partly by busyness, i.e., that time windows may be violated. The tree grows large for many instances not solved to optimality, hence branching also constitutes a bottleneck.

As can be seen in Table 2, the branch-and-price algorithm is generally faster when only generating columns heuristically. Note that the gap in this table denotes the gap between the *heuristic* upper and lower bound, and not between the heuristic upper and optimal lower bound. Some instances still suffer from large tree sizes and many columns, but the far majority of instances are solved in seconds. The objective values generally suffer from the heuristic approach. Comparing the two tables shows that the heuristic solutions are between 4% and 85% solution. The average gap between provably optimal and heuristic solutions is 47%. Even though solving the pricing problem heuristically reduces the overall running time significantly, the branch-and-price approach is still unable to solve the instance with a 10 minute time step, 2 employees and 99 instances.

Considering the complexity of the master-plan problem, it is no surprise that the exact branch-and-price algorithm can solve only limited sized instances. The results truly illustrate the complexity of the problem.

6 Conclusion

In this paper, we presented a branch-and-price algorithm for the long-term home care scheduling problem. The pricing problem consisted of calculating a work plan on a given day for a

Instance	Cols.	Rows	Tree size	Tree depth	Gap	Value	Time
1-20-5	6 327	199	1	0	0.00	27 000.0	3.97
1-25-5	8 671	250	4 687	59	0.03	29 750.0	1 801.33*
1-30-5	12 883	295	2 079	71	0.04	38 000.0	1 801.97*
1-33-5	11 447	328	1 615	75	0.00	38 800.0	1 655.35
1-40-5	28 365	397	171	51	0.00	43 000.0	812.94
1-44-5	40 445	433	179	89	0.01	97 100.0	1 802.57*
1-50-5	31 480	493	27	13	0.07	87 000.0	1 803.35*
1-55-5	37 944	544	159	28	0.00	57 000.0	1 804.30*
1-58-5	39 997	571	9	4	0.08	156 200.0	1 807.19*
1-80-5	8 737	787	1	0	No LB	259 250.0	1 906.50*
2-20-5	8 917	346	5	2	0.00	27 000.0	8.58
2-25-5	10 089	432	2 405	69	0.03	29 750.0	1 803.39*
2-30-5	16 383	512	695	66	0.05	38 400.0	1 803.29*
2-33-5	18 316	566	701	90	0.03	39 300.0	1 803.90*
2-40-5	23 921	684	455	70	0.05	42 100.0	1 801.46*
2-44-5	34 527	748	125	62	0.08	84 600.0	1 835.07*
2-50-5	30 642	850	71	35	0.07	85 000.0	1 806.21*
2-55-5	34 414	936	89	32	0.03	51 500.0	1 817.40*
2-58-5	30 582	984	17	8	0.02	143 700.0	2 174.36*
2-80-5	8 261	1 354	1	0	No LB	237 500.0	1 908.73*
1-20-10	4 877	199	59	28	0.00	17 250.0	6.37
1-25-10	5 507	250	277	51	0.00	18 125.0	27.36
1-30-10	10 934	295	2 449	101	0.01	25 300.0	1 801.07*
1-33-10	10 688	328	43	21	0.00	23 875.0	42.04
1-40-10	14 037	397	63	31	0.00	27 750.0	90.16
1-44-10	27 925	433	567	98	0.03	33 750.0	1 803.50*
1-50-10	19 907	493	3	1	0.00	36 500.0	337.04
1-55-10	19 912	544	3	1	0.00	60 375.0	123.64
1-58-10	40 434	571	111	55	0.00	64 450.0	1 803.82*
1-80-10	19 711	787	1	0	No LB	161 100.0	1 847.78*
2-20-10	5 913	346	39	19	0.00	17 250.0	9.19
2-25-10	6 594	432	81	40	0.00	18 125.0	16.30
2-30-10	10 109	512	1 245	56	0.00	25 200.0	1 802.52*
2-33-10	14 485	566	89	44	0.00	23 750.0	146.03
2-40-10	15 937	684	77	36	0.00	27 250.0	162.93
2-44-10	27 208	748	271	101	0.01	33 400.0	1 805.10*
2-50-10	25 457	850	21	10	0.00	35 750.0	654.53
2-55-10	29 331	936	49	24	0.01	80 625.0	1 841.08*
2-58-10	21 593	984	31	15	0.01	96 750.0	1 826.54*
2-80-10	22 136	1 354	3	1	0.02	149 750.0	2 475.74*

Table 1: Results for instances with either 1 or 2 employees and either 5 or 10 minutes per time step when using the optimal branch-and-price algorithm. The Table shows total number of generated columns and total number of constraints in the master problem, number of nodes in the branch-and-bound tree, depth of the tree, gap between lower and upper bound, objective value of best found solution, and running time in seconds. An "*" in the running time indicates that the algorithm timed out.

Instance	Cols.	Rows	Tree size	Tree depth	Gap	Value	Time
1-20-5	1 794	199	1	0	0.00	38 750.0	0.75
1-25-5	608	250	1	0	0.00	44 500.0	0.18
1-30-5	2 852	295	1	0	0.00	86 000.0	1.45
1-33-5	848	328	1	0	0.00	59 750.0	0.45
1-40-5	903	397	1	0	0.00	67 500.0	0.55
1-44-5	1 710	433	1	0	0.00	97 100.0	1.95
1-50-5	1 235	493	1	0	0.00	97 500.0	1.45
1-55-5	1 599	544	1	0	0.00	112 000.0	1.20
1-58-5	2 919	571	1	0	0.00	156 200.0	4.76
1-80-5	2 954	787	1	0	0.00	259 250.0	10.96
1-99-5	2 449	970	1	0	0.00	328 800.0	22.10
2-20-5	1 917	346	1	0	0.00	31 750.0	0.66
2-25-5	1 416	432	3	1	0.00	34 750.0	0.27
2-30-5	5 637	512	481	25	0.00	73 500.0	117.37
2-33-5	4 171	566	7	3	0.00	41 750.0	1.49
2-40-5	4 252	684	7	3	0.00	49 500.0	1.98
2-44-5	7 236	748	481	28	0.00	84 600.0	242.35
2-50-5	8 629	850	101	23	0.00	85 000.0	48.85
2-55-5	7 660	936	35	17	0.00	99 500.0	18.33
2-58-5	8 770	984	123	29	0.00	143 700.0	116.47
2-80-5	12 677	1 354	185	35	0.00	237 500.0	388.09
2-99-5	15 601	1 670	159	30	0.00	285 750.0	1 232.10
1-20-10	1 286	199	1	0	0.00	24 750.0	0.29
1-25-10	324	250	1	0	0.00	29 625.0	0.11
1-30-10	1 845	295	1	0	0.00	54 250.0	0.71
1-33-10	433	328	1	0	0.00	36 125.0	0.27
1-40-10	733	397	1	0	0.00	48 750.0	0.32
1-44-10	1 473	433	1	0	0.00	61 400.0	1.61
1-50-10	1 002	493	1	0	0.00	67 500.0	0.71
1-55-10	1 409	544	1	0	0.00	86 875.0	0.69
1-58-10	2 270	571	1	0	0.00	106 650.0	1.89
1-80-10	2 255	787	1	0	0.00	161 100.0	4.44
1-99-10	3 076	970	1	0	0.00	215 425.0	16.13
2-20-10	1 770	346	1	0	0.00	20 750.0	0.43
2-25-10	714	432	3	1	0.00	23 125.0	0.15
2-30-10	4 585	512	751	25	0.00	48 000.0	115.49
2-33-10	2 717	566	7	3	0.00	24 875.0	0.66
2-40-10	3 267	684	7	3	0.00	37 500.0	1.00
2-44-10	5 338	748	713	30	0.00	55 150.0	220.10
2-50-10	6 914	850	191	26	0.00	61 250.0	58.38
2-55-10	6 090	936	67	23	0.00	80 625.0	21.94
2-58-10	7 242	984	143	25	0.00	96 750.0	76.81
2-80-10	8 552	1 354	221	36	0.00	149 500.0	265.50
2-99-10	11 410	1 670	557	58	0.01	195 250.0	1 802.64*

Table 2: Results for instances with either 1 or 2 employees and either 5 or 10 minutes per time step when only generating columns heuristically in the branch-and-price algorithm. The table shows total number of generated columns and total number of constraints in the master problem, number of nodes in the branch-and-bound tree, depth of the tree, gap between the heuristic lower and upper bound, objective value of best found solution, and running time in seconds. An "*" in the running time indicates that the algorithm timed out.

given employee. The master problem merged plans into an overall optimal solution. The pricing problem was $\mathcal{NP}$-hard and solved through a labeling algorithm. Initially, the pricing problem was solved heuristically by only considering a small subset of labels. When this approach was unsuccessful, the labeling algorithm solved the pricing problem to optimality.

The branch-and-price algorithm was implemented and tested on a number of real-life instances provided by the Papirgården home care service in Funen, Denmark. The branch-and-price algorithm outperformed applying CPLEX to the original formulation. The algorithm, however, showed performance difficulties for larger instances due to the large number of combinations of visits, visit times and employees.

Improving the branch-and-price approach would require methods for reducing the number of columns and limiting the search tree size. The authors attempted stabilizing the value of dual variables using the interior point method of Rousseau et al. [14], but with no avail. Other stabilization methods could be investigated, as better values for the dual variables could reduce the number of generated columns. Different primal and incumbent heuristics have been implemented and tested without improving the bounds or pruning larger parts of the search tree. Future work should focus on finding better bounds, through primal and incumbent heuristics and through the branching strategy.

A different approach could also be taken to the Dantzig-Wolfe decomposition. If the master problem was to decide the time of visits, then the number of columns would be reduced significantly. This, however, would come at a price, because the complexity of the master problem would be affected negatively.

Acknowledgements
We would like to thank Papirgåden for sharing knowledge on home care scheduling and providing real-life data. Furthermore, we thank the Villum-Kann-Rasmussen foundation for their support of this work.

References

[1] S. V. Begur, D. M. Miller, and J. R. Weaver. An integrated spatial DSS for scheduling and routing home-health-care nurses. *Interfaces*, 27(4):35 – 48, 1997.

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

[3] N. Christofides and J. E. Beasley. The period routing problem. *Networks*, 14(2):237 – 256, 1984.

[4] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568 – 581, 1964.

[5] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[6] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operational Research*, 42(5):977 – 978, 1994.

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

[8] P. Eveborn, M. Rönnqvist, H. Einarsdóttir, M. Eklund, K. Lidén, and M. Almroth. Operations research improves quality and efficiency in home care. *Interfaces*, 39(1):18 – 34, 2009.

[9] S. E. Godskesen. Automated planning of work for home nurses. Master's thesis, Department of Mathematics and Computer Science, University of Southern Denmark, 2006.

[10] A. D. Hansen, M. S. Rasmussen, T. Justensen, and J. Larsen. The home care crew scheduling problem. In *ICAOR 2008, 1st International Conference on Applied Operational Research, Conference Proceedings*. American University of Armenia, 2008.

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

[12] R. Lougee-Heimer. The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47(1):57 – 66, 2003.

[13] J. Nikolajsen. Algorithms for optimizing work schedules for service personel in the home care sector. Master's thesis, Department of Mathematics and Computer Science, University of Southern Denmark, 2009.

[14] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, pages 660 – 668, 2007.

[15] R. A. Russell and W. Igo. An assignment routing problem. *Networks*, 9(1):1 – 17, 1979.

[16] R. Sharda, S. Voß, P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 73 – 102. Springer US, 2008.

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

[18] M. Wen. *Rich Vehicle Routing Problems and Applications*. PhD thesis, Department of Management Engineering, The Technical University of Denmark, 2010.

In several countries, home care is provided for certain citizens living at home. The long-term home care scheduling problem is to generate work plans spanning several days such that a high quality of service is maintained and the overall cost is kept as low as possible. A solution to the problem provides detailed information on visits and visit times for each employee on each of the covered days.
We propose a branch-and-price algorithm for the long-term home care scheduling problem. The pricing problem generates one-day plans for an employee, and the master problem merges the plans with respect to regularity constraints. The method solves instances with up to 99 visits during one week. This truly illustrates the complexity of the problem.

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

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

www.man.dtu.dk