International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements

Ran Liu, Biao Yuan & Zhibin Jiang

To cite this article: Ran Liu, Biao Yuan & Zhibin Jiang (2017) Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements, International Journal of Production Research, 55:2, 558-575, DOI: 10.1080/00207543.2016.1213917

To link to this article: http://dx.doi.org/10.1080/00207543.2016.1213917

International Journal of Production Research, 2017
Vol. 55, No. 2, 558–575, http://dx.doi.org/10.1080/00207543.2016.1213917

Taylor & Francis
Taylor & Francis Group

Mathematical model and exact algorithm for the home care worker scheduling and routing problem with lunch break requirements

Ran Liu, Biao Yuan and Zhibin Jiang*

Department of Industrial Engineering & Management, Shanghai Jiao Tong University, Shanghai, P.R. China

(*Received 4 February 2016; accepted 11 July 2016*)

Home health care or home care (HHC/HC) refers to the delivery of social, medical and paramedical services to clients in their own homes. Each day, care workers start from the HHC/HC centre, visit some clients and return to the centre. During the service delivery process, there is usually a lunch break for each worker. In this paper, we address a real-life home care worker scheduling and routing problem with the consideration of lunch break requirements. A three-index mathematical model is constructed for the problem. The problem is decomposed into a master problem and several pricing sub-problems, and is optimally solved by a branch-and-price (B&P) algorithm. Specifically, a sophisticated label-correcting algorithm is designed to address lunch break constraints in pricing sub-problems; some cutting-edge acceleration strategies are applied during the column generation process. Experimental results show that the proposed B&P algorithm is able to produce satisfied solutions within an acceptable runtime and outperforms the mixed integer programming solver CPLEX.

Keywords: healthcare logistics; scheduling; transportation; lunch break; branch-and-price

1. Introduction

Due to the ageing population, congestion of hospitals, and improvement in medical technologies, home health care or home care (HHC/HC) has been a growing medical service industry in many countries, including the United States, France and China. HHC/HC provides customised services for clients in order to help them recover from illness or injury in a personal environment. Clearly, finding an efficient route schedule to meet the requirements of clients is one of the most important daily activities for HHC/HC companies because it is crucial to reduce operating cost and improve customer service quality.

Some researchers have studied the home care worker scheduling and routing problem (HCWSRP). The HCWSRP is to find a set of routes with the minimisation of travel cost in which workers start from the HHC/HC centre, travel to provide services for clients, and return to the centre. Begur, Miller, and Weaver (1997) developed a decision support system for routing HHC nurses that can save schedule preparation time. Bertels and Fahle (2006) solved the HCWSRP with the hybridisation of constraint programming and meta-heuristic. Eveborn, Flisberg, and Rönnqvist (2006) demonstrated a decision support system, LAPS CARE, for the planning of the HC staff. Akjiratikarl, Yenradee, and Drake (2007) proposed a particle swarm optimisation algorithm for scheduling HC workers. Rasmussen et al. (2012) developed a branch-and-price (B&P) algorithm for the HCWSRP with temporal dependencies. Mankowska, Meisel, and Bierwirth (2014) devised an adaptive variable neighbourhood search for the HCWSRP with interdependent services. Yuan, Liu, and Jiang (2015) studied the HCWSRP with stochastic service times and skill requirements, and designed a B&P algorithm to solve it. Although several exact and heuristic algorithms have been proposed for specific HCWSRPs in the aforementioned research, worker fatigue and lunch breaks are not considered. However, because workers generally work approximately 6–8 h in one day in practice, it is necessary for each worker to have a break time to rest and eat lunch. The lunch break usually has a time window (e.g. 12:00–13:00) and a duration (e.g. 30 min). Meanwhile, in real-life applications, the lunch break typically occurs in the vicinity of clients' locations just before or after providing services rather than on the way to the next client (Cheng and Rich 1998; Bard et al. 2014b; Trautsamwieser and Hirsch 2014). In this paper, we address the HCWSRP with lunch break requirements (HCWSRP-LBR) in which visits to clients and lunch breaks of workers have specified time windows. The problem can be regarded as a variant of the vehicle routing problem with time windows (VRPTW) in which an artificial customer without the location should be visited by each

*Corresponding author. Email: zbjiang@sjtu.edu.cn

route. Although both our recently published paper (Yuan, Liu, and Jiang 2015) and this one use the B&P algorithm to solve the HCWSRP, the former mainly considers stochastic service times, and the latter focuses on the inclusion of LBR. Both problems and solution approaches are totally different in the above works.

Little attention has been paid to the HCWSRP with the consideration of LBR. In the daily HCWSRP, Cheng and Rich (1998) considered LBR and developed a two-phase heuristic to solve the problem. In the weekly or medium-term HCWSRP, Shao, Bard, and Jarrah (2012), Bard, Shao, and Wang (2013), Bard, Shao, and Jarrah (2014a), and Bard et al. (2014b) allowed a lunch break if the shift length is larger than a predefined time; however, in their studies, patients are either visited at any time (no time windows) or at a fixed time. Such assumptions make their algorithms inapplicable to our problem. Trautsamwieser and Hirsch (2014) devised a branch-and-price-and-cut algorithm for the medium-term HHC planning problem in which the break is taken at a client if the maximum consecutive working time exceeds a threshold.

The purpose of this paper is to introduce a sophisticated B&P algorithm to exactly solve the HCWSRP-LBR, which has resulted in several contributions: (1) the proposed algorithm can be seen as a special variant of the VRPTW. It differs from the classical VRPTW because of an extra level of difficulty in addressing LBR; (2) a three-index mathematical model is built for the problem; (3) an effective B&P algorithm is developed for the problem. Its effectiveness strongly relies on a number of cutting-edge column generation strategies and a hierarchical branching scheme; and (4) numerical experiments are performed on the instances from the literature and real data to analyse the efficiency of the algorithm.

The remainder of the paper is organised as follows. Section 2 defines the problem. Section 3 introduces the solution approach for it. Section 4 presents computational results and analyses the performance of the algorithm. Section 5 concludes the paper.

2. Mathematical formulation

The HCWSRP-LBR consists of determining a set of routes such that each worker leaves from the HHC/HC centre, serves some clients, takes a lunch break and returns to the centre. The sets of workers and clients are denoted by $K = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$, where m and n are the number of available workers and clients, respectively. Each client $i \in N$ has a service duration d_i and a time window $[a_i, b_i]$, where a_i and b_i are the earliest and latest service start times of this client, respectively. For each worker $k \in K$, artificial nodes 0 and $n + 1$ are the origin and destination of the route (i.e. HHC/HC centre), respectively, and the maximum working time in a day is L, which indicates that $[0, L]$ is the time window of the worker. The travel cost and travel time from client i to j are ct_{ij} and t_{ij}, respectively. For lunch break P, the lunch duration and time window are defined as B and $[a_P, b_P]$ (Cheng and Rich 1998). To maintain the continuity of care in the periodic planning (Nickel, Schröder, and Steeg 2012) a limited number of different workers are assigned to each client. For the pair of client $i \in V$ ($V = N \cup \{0, n + 1\}$) and worker $k \in K$, Q_{ik} is 1 if worker k is allowed to serve client i, and 0 otherwise. We use set N_k to indicate the clients whom worker k can serve. It should be noted that because both the number of workers and the working duration of each worker are limited, in some cases, the service capacity of the company may not be sufficient to provide services for all clients, i.e. some clients cannot be served in any feasible solution. Therefore, as in the research by Rasmussen et al. (2012) and Nickel, Schröder, and Steeg (2012), in this paper, a penalty cp_i is incurred if client i is not served. In practice, when there are always unvisited clients over a period of time, we can conclude that the number of workers is inadequate and the company should hire more workers to satisfy the demands of all clients. The objective of the problem is defined as minimising the sum of the travel cost and the penalty for uncovered clients.

Six types of decision variables are used in the model: x_{ijk} is 1 if worker k directly visits client j after i and 0 otherwise; y_{ik} is 1 if worker k takes a break at client i before service and 0 otherwise; y_{ik}' is 1 if worker k takes a break at client i after service and 0 otherwise; z_i is 1 if client i is not visited and 0 otherwise; ts_{ik} is the service start time of worker k at client i; and ts_{Pk} is the start time of the lunch break of worker k. The problem is formulated as a three-index mixed integer programming model.

$$\min \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} ct_{ij}x_{ijk} + \sum_{i\in N} cp_i z_i \tag{1}$$

$$\text{s.t.} \sum_{k\in K}\sum_{j\in V} x_{ijk} + z_i = 1, \quad \forall i \in N \tag{2}$$

$$\sum_{j\in V} x_{0jk} = 1, \quad \forall k \in K \tag{3}$$

$$\sum_{j\in V} x_{j(n+1)k} = 1, \quad \forall k \in K \tag{4}$$

$$\sum_{i\in V} x_{ijk} - \sum_{i\in V} x_{jik} = 0, \quad \forall k \in K, j \in N \tag{5}$$

$$\sum_{i\in V} y_{ik} + \sum_{i\in V} y'_{ik} = 1, \quad \forall k \in K \tag{6}$$

$$y_{ik} + y'_{ik} \le \sum_{j\in V} x_{ijk}, \quad \forall k \in K, i \in V \tag{7}$$

$$ts_{ik} + (t_{ij} + d_i)x_{ijk} \le ts_{jk} + (1 - x_{ijk})b_i, \quad \forall k \in K, i, j \in V \tag{8}$$

$$ts_{Pk} + B \cdot y_{jk} \le ts_{jk} + (1 - y_{jk})b_P, \quad \forall k \in K, j \in V \tag{9}$$

$$ts_{ik} + (t_{ij} + d_i)(x_{ijk} + y_{jk} - 1) \le ts_{Pk} + (2 - x_{ijk} - y_{jk})b_i, \quad \forall k \in K, i, j \in V \tag{10}$$

$$ts_{Pk} + (B + t_{ij})(x_{ijk} + y'_{ik} - 1) \le ts_{jk} + (2 - x_{ijk} - y'_{ik})b_P, \quad \forall k \in K, i, j \in V \tag{11}$$

$$ts_{ik} + d_i y'_{ik} \le ts_{Pk} + (1 - y'_{ik})b_i, \quad \forall k \in K, i \in V \tag{12}$$

$$a_i \sum_{j\in V} x_{ijk} \le ts_{ik} \le b_i \sum_{j\in V} x_{ijk}, \quad \forall k \in K, i \in N \tag{13}$$

$$0 \le ts_{ik} \le L, \quad \forall k \in K, i \in \{0, n+1\} \tag{14}$$

$$a_P \le ts_{Pk} \le b_P, \quad \forall k \in K \tag{15}$$

$$x_{ijk} \le \min\{Q_{ik}, Q_{jk}\}, \quad \forall k \in K, i, j \in V \tag{16}$$

$$y_{ik}, y'_{ik} \le Q_{ik} \quad \forall k \in K, i \in V \tag{17}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall k \in K, i, j \in V \tag{18}$$

$$y_{ik}, y'_{ik} \in \{0, 1\}, \quad \forall k \in K, i \in V \tag{19}$$

$$z_i \in \{0, 1\}, \quad \forall i \in N \tag{20}$$

The objective function (1) is to minimise the sum of the travel cost and the penalty for unvisited clients. Constraint (2) ensures that each client is either visited by one worker or uncovered. Constraints (3)–(5) guarantee that each worker starts at the origin, visits some clients and ends at the destination. Constraint (6) represents that each worker takes a break in the route. Constraint (7) denotes that worker k can take a break at client i if client i is visited by this worker. Constraint (8) ensures the route feasibility with respect to the service start times at clients. Constraints (9)–(12) guarantee the correct start times of the breaks when workers rest at clients before and after services. Constraints (13)–(15) ensure the time windows of clients, workers and lunch breaks. Constraints (16)–(17) ensure the compatibility between workers and clients. Constraints (18)–(20) define feasible values for decision variables.

Although it is not explicitly imposed that the lunch break depends on the duration of the route, we can find that the lunch break is taken if the duration of the route exceeds a_P. Note that in the above model, a lunch break is allowed at the end of a route, if such a route exists in the optimal solution and its total duration (i.e. the time after serving all

clients plus the travel time from the last client to the centre) is smaller than the earliest possible time of the break a_P, we delete the break in this route. Clearly, this deletion does not change the route cost. Otherwise, the break is kept in the route, i.e. the worker has a lunch before returning to the centre.

3. Solution approach

This section first overviews the B&P algorithm, and then presents its details for solving the problem.

3.1 *Overview of B&P algorithm*

The B&P algorithm is a branch-and-bound method in which the column generation algorithm is employed at each node of the search tree. When the B&P algorithm is used to solve the problem, the model in (1)–(20) is reformulated as a set-partitioning problem or a set-covering problem that is called the *master problem*. Because the basic master problem usually contains exponential variables, the modified problem including a subset of columns and relaxing decision variables (called the *restricted master problem*, RMP) is solved. Based on the values of dual variables of the RMP, a sub-problem (called the *pricing sub-problem*) is solved to find columns with negative reduced cost (for a minimisation problem) and to check the optimality. This process is called the *pricing process*. If a negative reduced cost column exists, such a column is added into the RMP, and the column generation algorithm is repeated until no new column can be found. The B&P algorithm inserts a set of initial columns into the RMP at the root node and then constructs the search tree. At each non-root node of the search tree, the algorithm initialises the RMP with the set of columns inherited from the parent node, except for the columns that have become infeasible because of branching decisions, and the column generation algorithm is used to solve the RMP. If the optimal solution of the RMP is integer, it is also feasible to the master problem; the better one between this solution and the current upper bound is selected as new upper bound, and the pruning operation is performed based on new upper bound. Otherwise, the two-stage branching scheme is adopted to generate new child nodes. The search process stops when all nodes in the tree are explored. Figure 1 describes the whole process of the B&P algorithm for the HCWSRP-LBR.

3.2 *Master problem and pricing sub-problem*

The following notation is defined in the set-partitioning formulation: R_k is the set of all feasible routes visited by worker k; α_{ir} is 1 if client i is visited by route r and 0 otherwise; c_r is the travel cost of route r; decision variable θ_r is 1 if route r is selected in the optimal solution and 0 otherwise. The master problem of the HCWSRP-LBR is described as follows.

$$\min \sum_{k\in K}\sum_{r\in R_k} c_r\theta_r + \sum_{i\in N} cp_i z_i \tag{21}$$

$$\text{s.t.} \sum_{k\in K}\sum_{r\in R_k} \alpha_{ir}\theta_r + z_i = 1, \quad \forall i \in N \tag{22}$$

$$\sum_{r\in R_k} \theta_r \le 1, \quad \forall k \in K \tag{23}$$

$$\theta_r \in \{0,1\}, \quad \forall r \in \bigcup_{k\in K} R_k \tag{24}$$

$$z_i \in \{0,1\}, \quad \forall i \in N \tag{25}$$

The objective function (21) is to minimise total travel cost of the selected routes plus the penalty for uncovered clients. Constraint (22) has the same meaning as constraint (2). Constraint (23) ensures that each worker selects at most one route. Constraints (24)–(25) represent that all decision variables are binary. Because the cardinality of R_k for worker k is extremely large, which precludes its exhaustive enumeration in the set-partitioning model, the modified version with the inclusion of a subset of R_k and the relaxation of decision variables (i.e. the RMP) is used in the column generation algorithm.

Let λ_i and μ_k be the dual vectors associated with constraints (22) and (23), respectively. The reduced cost of route r (i.e. the objective function of the pricing sub-problem) visited by worker k is computed as

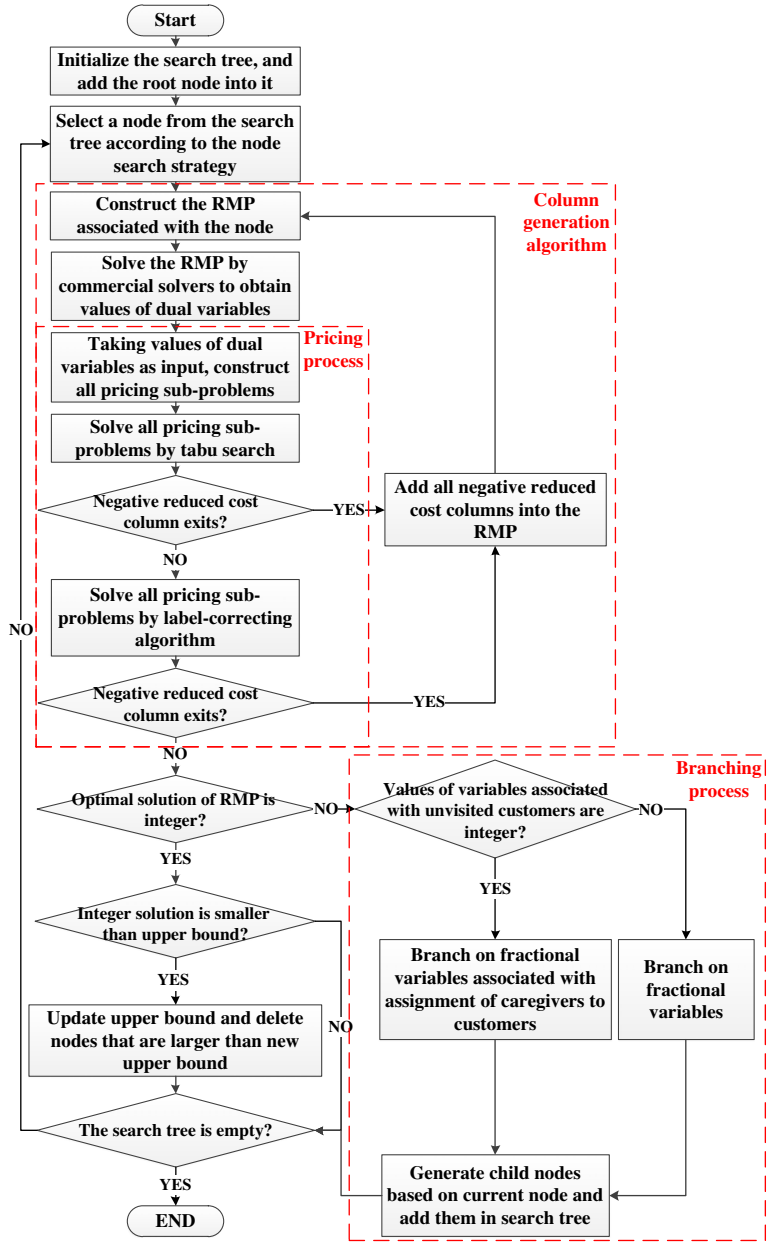

Figure 1. Flowchart of B&P algorithm.

$$c_r - \sum_{i \in N} \alpha_{ir} \lambda_i - \mu_k. \tag{26}$$

The pricing sub-problem in which the route satisfies constraints (3)–(19) is essentially an elementary shortest path problem with time windows and LBR. Since workers have different client sets, each worker corresponds to a sub-problem. There are total m sub-problems in the pricing process.

3.3 *Pricing algorithm*

We solve the sub-problems using two pricing algorithms sequentially, i.e. tabu search (TS) and label-correcting algorithm. Based on preliminary experiments, we find that: (1) the label-correcting algorithm of Righini and Salani (2006, 2008) is able to effectively solve the pricing sub-problems, and (2) how to deal with lunch break constraints is a vital issue in our problem. Note that in one column generation iteration, the m sub-problems are sequentially solved by

pricing algorithms, and all negative reduced cost columns from pricing sub-problems are returned into the RMP to reduce the number of column generation iterations (Contardo, Desaulniers, and Lessard 2015).

3.3.1 *Label-correcting algorithm*

To optimally solve pricing sub-problems, we adopt the label-correcting algorithm proposed by Righini and Salani (2006, 2008). The label-correcting algorithm encodes routs as labels and treats all labels as temporary (non-optimal) until the final step when they all become permanent (optimal) (Ahuja, Magnanti, and Orlin 1993). When it is invoked to solve the pricing sub-problem corresponding to worker k, the network $G_k = \{V_k, A_k\}$, where the node set $V_k = N_k \cup \{0, n+1\}$ and the arc set $A_k = \{(i, j), i, j \in V_k\}$, and its relevant information (e.g. travel cost and dual values) are used as input. Its main components are described in the following subsections.

3.3.1.1 *Label definition.* A label $(S, \eta_b, \eta_a, \tau, C, i)$ at client i is defined as follows. Set S contains the clients that have already been served along the route; η_b equals 1 if the worker has taken a break at a client before serving him/her in the route and 0 otherwise; η_a is 1 if the worker has taken a break at a client after serving him/her and 0 otherwise; τ denotes the service start time of worker at client i; C is the reduced cost of the route; and i is the last reached client. To facilitate the subsequent description of the algorithm, we define notation λ_0 and λ_{n+1} to represent artificial dual values associated with nodes 0 and $n+1$ and set them equal to μ_k when C is updated for the sub-problem corresponding to worker k.

Both forward and backward labels, which indicate routes from client 0 to client i and routes from client $n+1$ to i at client $i \in N$, are generated in the bidirectional extension process. The superscripts fw and bw are used to represent forward and backward labels, respectively.

3.3.1.2 *Label extension.* At the beginning of the label extension, set S is initialised as $\varnothing$, and the values of η_b, η_a, τ, and C are initialised as 0. We first introduce the extension rules to update S and C, which are identical in forward and backward extensions. When the label $(S, \eta_b, \eta_a, \tau, C, i)$ at client i is extended to j, S' and η_b' in new labels at j are updated as follows.

$$S' = S \cup \{j\} \tag{27}$$

$$C' = C + ct_{ij} - 0.5(\lambda_i + \lambda_j) \tag{28}$$

Equation (27) adds client j into set S'. Equation (28) updates the reduced cost of the partial route. When a label is extended to client $n+1$, the reduced cost C of the corresponding route $\{0, v_1, \ldots, v_i, \ldots, n+1\}$ is calculated using Equation (28), where v_i is the ith client and n_r is the number of clients in the route. We can see that the reduced cost obtained by Equation (28) is the same as that by formula (26).

$$\begin{aligned} C &= ct_{0v_1} - 0.5(\lambda_0 + \lambda_{v_1}) + \cdots + ct_{v_{n_r}(n+1)} - 0.5(\lambda_{v_{n_r}} + \lambda_{n+1}) \\ &= c_r - \sum_{i=1}^{n_r} \lambda_{v_i} - 0.5\lambda_0 - 0.5\lambda_{n+1} = c_r - \sum_{i=1}^{n_r} \lambda_{v_i} - \mu_k \end{aligned} \tag{29}$$

According to the values of η_b and η_a, there are three types of labels in the extension process. The first one is that the worker has not taken a break in the label, i.e. $\eta_b = \eta_a = 0$. The second one is that the worker has taken a break at a client before serving this client, i.e. $\eta_b = 1$ and $\eta_a = 0$. The last one is that the worker has taken a break at a client after serving this client, i.e. $\eta_b = 0$ and $\eta_a = 1$. Because only one break is taken in the label, the sum of η_b and η_a is not larger than one. According to the decision on taking a break, η_b^{fw} and η_a^{fw} are set to different values, and then τ^{fw} is updated. The forward extension rules to update the values of η_b^{fw}, η_a^{fw}, and τ^{fw} are described as follows.

Taking a forward label $(S^{fw}, \eta_b^{fw}, \eta_a^{fw}, \tau^{fw}, C^{fw}, i)$ at client i as an example, if both η_b^{fw} and η_a^{fw} equal 0 (i.e. a break hasn't been taken in the label) and the label is extended to j along arc (i, j), three new forward labels $(S^{fw\prime}, \eta_b^{fw\prime}, \eta_a^{fw\prime}, \tau^{fw\prime}, C^{fw\prime}, j)$ at j are generated by the following three rules.

Rule 1: if no break is taken when the label extends from client i to j, $\eta_b^{fw\prime}$, $\eta_a^{fw\prime}$ and $\tau^{fw\prime}$ are calculated using Equations (30)–(32). This extension rule is called *the forward extension without lunch break*.

$$\eta_b^{fw\prime} = \eta_b^{fw} \tag{30}$$

$$\eta_a^{fw\prime} = \eta_a^{fw} \tag{31}$$

$$\tau^{fw\prime} = \max\left\{a_j, \tau^{fw} + d_i + t_{ij}\right\} \tag{32}$$

This new label at client j is feasible if $\tau^{fw\prime} \le b_j$ and $j \notin S^{fw}$, which are the time window and elementary constraints, respectively.

Rule 2: if a break is taken at client i after serving this client when the label extends from clients i to j, $\eta_b^{fw\prime}$, $\eta_a^{fw\prime}$, and $\tau^{fw\prime}$ are computed using Equations (33)–(35). This extension rule is called *the forward extension with lunch break after service*.

$$\eta_b^{fw\prime} = \eta_b^{fw} \tag{33}$$

$$\eta_a^{fw\prime} = 1 \tag{34}$$

$$\tau^{fw\prime} = \max\{a_j, \max\{a_P, \tau^{fw} + d_i\} + B + t_{ij}\} \tag{35}$$

This new label associated with client j is feasible if $\tau^{fw\prime} \le b_j$, $\tau^{fw} + d_i \le b_P$ and $j \notin S^{fw}$, which are the time window and elementary constraints, respectively.

Rule 3: if a break is taken at client j before serving this client when the label extends from clients i to j, $\eta_b^{fw\prime}$, $\eta_a^{fw\prime}$ and $\tau^{fw\prime}$ are updated using Equations (36)–(38). This extension rule is called *the forward extension with lunch break before service*.

$$\eta_b^{fw\prime} = 1 \tag{36}$$

$$\eta_a^{fw\prime} = \eta_a^{fw} \tag{37}$$

$$\tau^{fw\prime} = \max\{a_j, \max\{a_P, \tau^{fw} + d_i + t_{ij}\} + B\} \tag{38}$$

This new label at client j is feasible if $\tau^{fw\prime} \le b_j$, $\tau^{fw} + d_i + t_{ij} \le b_P$ and $j \notin S^{fw}$, i.e. the time window and elementary constraints, respectively.

Figure 2 illustrates three forward extension rules in which solid lines represent the time spent in travel and service activities and lunch breaks, whereas dashed lines indicate the waiting times.

Concerning the forward label (S^{fw}, η_b^{fw}, η_a^{fw}, τ^{fw}, C^{fw}, i) at client i, if the sum of η_b^{fw} and η_a^{fw} equals 1 (i.e. a break has been taken in the label) and it is extended to j along arc (i, j), one new label ($S^{fw\prime}$, $\eta_b^{fw\prime}$, $\eta_a^{fw\prime}$, $\tau^{fw\prime}$, $C^{fw\prime}$, j) is generated using **Rule 1**.

The backward extension process starts from time $T = \max_{i \in N}\{b_i + d_i + t_{i(n+1)}\}$ that indicates the maximum feasible arrival time at client $n + 1$. In the backward label, τ^{bw} at client i represents the minimum time consumed if the departure

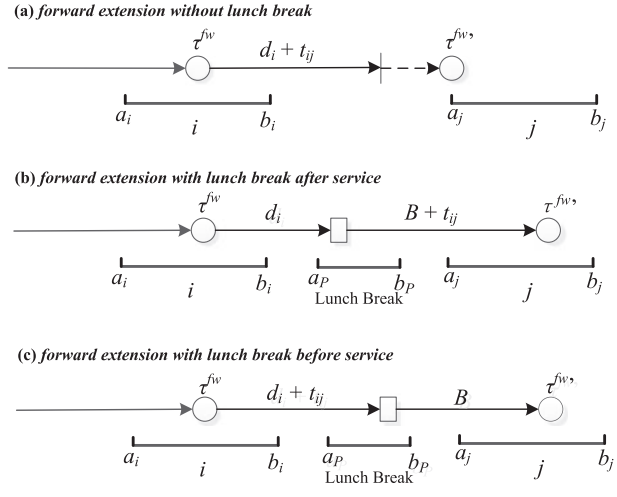

Figure 2. Illustration for three kinds of forward extension rules.

time from i up to the arrival at client $n+1$ is not later than time T. The backward extension process is analogous to the forward one. We describe its details in the supplemental online material.

To restrict the extension of forward and backward labels and reduce the number of duplicate routes, the time resource consumed by labels should be bounded. Note that the lunch break constraint has an impact on the quantity of time resource T but not on the general bidirectional extension process. As the label-correcting algorithm for the VRPTW (Righini and Salani 2006), we impose that each label consumes no more than half of the time resource T, i.e. $\tau^{fw} \leq 0.5\ T$ and $\tau^{bw} \leq 0.5\ T$.

3.3.1.3 *Label join.* The forward and backward labels are joined to generate a complete route from clients 0 to $n+1$, and the feasibility of the route is checked in the connection procedure. Taking a forward label (S^{fw}, η_b^{fw}, η_a^{fw}, τ^{fw}, C^{fw}, i) joined with a backward label (S^{bw}, η_b^{bw}, η_a^{bw}, τ^{bw}, C^{bw}, j) as an example, we analyse two situations to address the lunch break constraints and obtain a feasible route according to the sum of η_b^{fw}, η_a^{fw}, η_b^{bw}, and η_a^{bw}.

If the sum of η_b^{fw}, η_a^{fw}, η_b^{bw} and η_a^{bw} equals 1, a lunch break has been taken either in the forward or backward label. In this case, the joined route is feasible if both labels satisfy conditions (39)–(41).

$$S^{fw} \cap S^{bw} = \emptyset \tag{39}$$

$$\eta_b^{fw} + \eta_a^{fw} + \eta_b^{bw} + \eta_a^{bw} = 1 \tag{40}$$

$$\tau^{fw} + d_i + t_{ij} + d_j + \tau^{bw} \leq T \tag{41}$$

If the sum of η_b^{fw}, η_a^{fw}, η_b^{bw} and η_a^{bw} equals 0, no break has been taken in the forward and backward labels. In this case, the feasibility of taking a break at client i or j should be examined. We employ **Rules 2 and 3** to extend label (S^{fw}, η_b^{fw}, η_a^{fw}, τ^{fw}, C^{fw}, i) to j and generate two labels ($S^{bw\prime}$, $\eta_b^{bw\prime}$, $\eta_a^{bw\prime}$, $\tau^{bw\prime}$, $C^{bw\prime}$, j) and ($S^{bw\prime\prime}$, $\eta_b^{bw\prime\prime}$, $\eta_a^{bw\prime\prime}$, $\tau^{bw\prime\prime}$, $C^{bw\prime\prime}$, j). The joined route is feasible if at least one new label satisfies the following conditions,

$$S^{fw} \cap S^{bw} = \emptyset \tag{42}$$

$$\eta_b^{fw\prime} + \eta_a^{fw\prime} + \eta_b^{bw} + \eta_a^{bw} = 1 \tag{43}$$

$$\max\{\tau^{fw} + d_i, a_P\} + B + t_{ij} + d_j + \tau^{bw} \leq T \tag{44}$$

or

$$S^{fw} \cap S^{bw} = \emptyset \tag{45}$$

$$\eta_b^{fw\prime\prime} + \eta_a^{fw\prime\prime} + \eta_b^{bw} + \eta_a^{bw} = 1 \tag{46}$$

$$\max\{\tau^{fw} + d_i + t_{ij}, a_P\} + B + d_j + \tau^{bw} \leq T \tag{47}$$

The reduced cost of the joined route is calculated using

$$C^{fw} + ct_{ij} + C^{bw} - 0.5(\lambda_i + \lambda_i) \tag{48}$$

3.3.1.4 *Dominance test.* In the label extension, a dominance test is performed to discard the labels that cannot lead to the optimal solution. Taking two different labels ending at client i, i.e. $l = (S, \eta_b, \eta_a, \tau, C, i)$ and $l' = (S', \eta_b', \eta_a', \tau', C', i)$ as an example, the dominance relationship between labels are illustrated as follows.

Label l dominates label l' if

(a) $S \subseteq S'$,
(b) $\tau \leq \tau'$,
(c) $\eta_b + \eta_a \geq \eta_b' + \eta_a'$,
(d) $C \leq C'$,

and at least one of the inequalities is strictly satisfied. Condition (a) indicates that the clients visited by label l is a subset of clients by l'; condition (b) represents that the time resource consumed up to client i by label l is not larger than l';

and condition (d) means that label l is less costly than l' at client i. Condition (c) contains three cases: (i) no break is taken at both labels, i.e. $\eta_b + \eta_a = 0$ and $\eta'_b + \eta'_a = 0$; (ii) a break is taken at both labels, i.e. $\eta_b + \eta_a = 1$ and $\eta'_b + \eta'_a = 1$; and (iii) a break is taken at label l, but not at l', i.e. $\eta_b + \eta_a = 1$ and $\eta'_b + \eta'_a = 0$. In one of the three cases of condition (c), if conditions (a), (b) and (d) are satisfied, we can prove that these conditions still hold when both labels are extended to client $n + 1$. The case in which a break is taken at l', but not at l, i.e. $\eta_b + \eta_a = 0$ and $\eta'_b + \eta'_a = 1$, is not included in condition (c). This is because a break consumes the time resource, and condition (b) may not be satisfied in subsequent extensions. Therefore, the above four conditions guarantee that the labels extended from label l' are covered by those from l with less cost and less time resource.

3.3.1.5 *Overall description of label-correcting algorithm.* The pseudo code of the label-correcting algorithm is shown in **Algorithm 1**. In **Algorithm 1**, Γ_i^{fw} and Γ_i^{bw} are the lists of forward and backward labels associated with node i; l_i is a label associated with node i; E is the set of nodes to be examined; $Extend^{fw}(l_i, j)$ and $Extend^{bw}(l_i, j)$ are the forward and backward extension functions that extend label l_i to node j according to extension rules, respectively; $EFF(\Gamma_j, l_j)$ is the function that inserts label l_j into list Γ_j by applying the dominance test; *Feasible* (l_i, l_j) is the function that checks the feasibility of combining labels l_i and l_j; $Save(l_i, l_j)$ is the function that saves the label obtained from labels l_i and l_j.

Algorithm 1: Label-correcting algorithm for pricing sub-problem

// Initialization
$\Gamma_0^{fw} \leftarrow \{(\varnothing, 0, 0, 0, 0, 0)\}$
$\Gamma_{n+1}^{bw} \leftarrow \{(\varnothing, 0, 0, 0, 0, n + 1)\}$
for all $i \in V \setminus \{0\}$ **do** $\Gamma_i^{fw} \leftarrow \varnothing$
for all $i \in V \setminus \{n + 1\}$ **do** $\Gamma_i^{bw} \leftarrow \varnothing$
$E \leftarrow \{0, n + 1\}$
// Search
repeat
  // Select node
  select $i \in E$
  // Forward extension
  for all $l_i = (S^{fw}, \eta_b^{fw}, \eta_a^{fw}, \tau^{fw}, C^{fw}, i) \in \Gamma_i^{fw}$ **do**
    if $\tau_i^{fw} \leq 0.5 \cdot T$ **then**
      for all $(i, j) \in A_k$ such that $j \notin S^{fw}$ **do**
        $l_j \leftarrow Extend^{fw}(l_i, j)$
        $\Gamma_j^{fw} \leftarrow EFF(\Gamma_j^{fw}, l_j)$
        if Γ_j^{fw} is changed **then** $E \leftarrow E \cup \{j\}$
  // Backward extension
  for all $l_i = (S^{bw}, \eta_b^{bw}, \eta_a^{bw}, \tau^{bw}, C^{bw}, i) \in \Gamma_i^{bw}$ **do**
    if $\tau_i^{bw} \leq 0.5 \cdot T$ **then**
      for all $(j, i) \in A_k$ such that $j \notin S^{bw}$ **do**
        $l_j \leftarrow Extend^{bw}(l_i, j)$
        $\Gamma_j^{bw} \leftarrow EFF(\Gamma_j^{bw}, l_j)$
        if Γ_j^{bw} is changed **then** $E \leftarrow E \cup \{j\}$
  $E \leftarrow E \setminus \{i\}$
until $E = \varnothing$
// Join forward and backward labels
for all $i \in V \setminus \{n + 1\}$
  for all $l_i = (S^{fw}, \eta_b^{fw}, \eta_a^{fw}, \tau^{fw}, C^{fw}, i) \in \Gamma_i^{fw}$
    for all $j \in V \setminus \{0\}$
      for all $l_j \in (S^{bw\prime}, \eta_b^{bw\prime}, \eta_a^{bw\prime}, \tau^{bw\prime}, C^{bw\prime}, j) \in \Gamma_j^{bw}$
        if *Feasible* (l_i, l_j) **then** $Save(l_i, l_j)$

We use the sub-problem of work k as an example to analyse the complexity of the label-correcting algorithm. During the search procedure, once Γ_i^{fw} or Γ_i^{bw} is updated, node i is added into set E. If the data (e.g. travel cost and time windows) in the problem instance are integer, the size of state space of Γ_i^{fw} and Γ_i^{bw} is $2^{|N_k|} \cdot 2 \cdot 2 \cdot 0.5T \cdot 2|N_k|c_{\max}$, where $c_{\max}$ is the largest absolute value of cost of arcs minus half of the dual values of corresponding nodes, i.e. $c_{\max} = \max_{(i,j)\in A_k} |ct_{ij} - 0.5(\lambda_i + \lambda_j)|$, because in the label definition: (1) the maximum size of S is $2^{|N_k|}$, (2) there are two possible values for η_b and η_a and $0.5T$ possible values for τ, (3) the value of C is limited in $[-|N_k|c_{\max}, |N_k|c_{\max}]$. The algorithm selects node i in subsequent iterations and scans arc set A_k. It updates label sets no

more than $2 \cdot 2^{|N_k|} \cdot 2 \cdot 2 \cdot 0.5T \cdot 2|N_k|c_{\max}$ times, the bound on total number of arc scans is $\sum_{i \in N_k} 2 \cdot 2^{|N_k|} \cdot 2 \cdot 2 \cdot 0.5T \cdot 2|N_k|c_{\max}|N_k| = O(2^{|N_k|})$. The bound on the number of joining times of forward and backward labels is also $O(2^{|N_k|})$. Therefore, the label-correcting algorithm runs in $O(2^{|N_k|})$ time.

3.3.2 *ng-route and decremental state space relaxation*

The effectiveness of the label-correcting algorithm is improved by the acceleration strategy proposed by Dayarian et al. (2015), which combines the *ng*-route relaxation (Baldacci, Mingozzi, and Roberti 2011) and decremental state space relaxation (Boland, Dethridge, and Dumitrescu 2006; Righini and Salani 2008).

The *ng*-route relaxation is described as follows. In the pricing sub-problem corresponding to worker k, let $\Theta_i \subseteq N_k$ be the original neighbourhood of client i which contains other $\delta - 1$ nearest clients besides himself/herself and N_r be the set of clients visited by worker k along the partial route r.

In the forward extension, for label l associated with a partial route $r = \{0, v_1, v_2, \ldots, v_{nr}\}$, the set $\Pi^{fw}(l) \subseteq N_r$ including all prohibited extensions from client v_{nr} is defined as

$$\Pi^{fw}(l) = \left\{ v_j \in N_r | v_j \in \bigcap_{h=j+1}^{nr} \Theta_{v_h}, j = 1, \ldots, nr-1 \right\} \cup \{v_{nr}\}. \tag{49}$$

For example, let $l = \{0, 1, 2, 3, 4\}$ be a route ending at client 4, $\Theta_1 = \{1, 8, 9\}$, $\Theta_2 = \{2, 7, 10\}$, $\Theta_3 = \{2, 3, 7\}$, and $\Theta_4 = \{2, 4, 5\}$. Then, $1 \notin \Theta_2 \cap \Theta_3 \cap \Theta_4$, $2 \in \Theta_3 \cap \Theta_4$, $3 \notin \Theta_4$. Therefore, from Equation (49), the $\Pi^{fw}(l) = \{2, 4\}$, which means that label l at client 4 cannot be extended to clients 2 and 4.

In the backward extension, for label l associated with a partial route $r = \{v_i, v_{i+1}, \ldots, v_{nr}, n+1\}$, the set $\Pi^{bw}(l) \subseteq N_r$ including all prohibited extensions from client v_i is defined as

$$\Pi^{bw}(l) = \left\{ v_j \in N_r | v_j \in \bigcap_{h=i}^{j-1} \Theta_{v_h}, j = i+1, \ldots, nr \right\} \cup \{v_i\}. \tag{50}$$

For example, let $l = \{6, 7, 8, 9, n+1\}$ be a route beginning from client 6, $\Theta_6 = \{6, 7, 8\}$, $\Theta_7 = \{7, 8, 10\}$, $\Theta_8 = \{3, 8, 9\}$, $\Theta_9 = \{5, 8, 9\}$. Then $7 \in \Theta_6$, $8 \in \Theta_6 \cap \Theta_7$, $9 \notin \Theta_6 \cap \Theta_7 \cap \Theta_8$. Thus, from Equation (50), the $\Pi^{bw}(l) = \{6, 7, 8\}$, which indicates that the label l at client 6 cannot be extended to clients 6, 7 and 8.

In forward and backward extensions, label l cannot be extended to client j if $j \in \Pi(l)$; this is called the *ng*-rule or *ng*-feasible. Equation (27) is replaced by

$$\Pi' = \left(\Pi \cap \Theta_j\right) \cup \{j\} \tag{51}$$

In the acceleration strategy, instead of original neighbourhoods Θ_i, the label-correcting algorithm begins with initial empty sets $\Theta'_i = \varnothing$, called applied neighbourhoods. At the end of the label-correcting algorithm, all negative reduced cost elementary columns as well as those that have negative reduced cost and satisfy *ng*-rules with respect to Θ_i (called *ng*-feasible columns) are inserted into the RMP. If no column exists above and the optimal column with negative reduced cost is not *ng*-feasible, the applied neighbourhoods Θ'_i of the clients, forming a cycle beginning and ending at client i in the route, are augmented by inserting i into them. The label-correcting algorithm restarts with updated applied neighbourhoods Θ'_i.

In the column generation context, the iterative process stops either the negative reduced cost elementary column or the negative reduced cost *ng*-feasible column is found or the optimal column with negative reduced cost does not exist. We also identify client $j \subseteq \Theta'_i$ who cannot be reached by any feasible extension of a given label associated with i owing to the time window constraints (Feillet et al. 2004) and add him/her to the set $U \subseteq \Theta'_i$ as a new component of the label. Condition (a) in the dominance test is replaced by

(e) $(\Pi \cup U) \subseteq (\Pi' \cup U')$

When joining a forward label and a backward label, Equations (39), (42) and (45) are replaced by a function that checks whether the joined label is *ng*-feasible with respect to Θ_i.

As proposed by Dayarian et al. (2015), Θ'_i is initialised (set to $\varnothing$) at the root node and not elsewhere in the search tree. Because the label-correcting algorithm with the acceleration strategy obtains *ng*-feasible columns that may contain cycles, in constraints (22), α_{ir} denotes the number of times that route r visits client i, and the '=' is replaced by the '≥'.

3.3.3 *TS column generator*

TS is a famous meta-heuristic and is widely used in combinatorial optimisation. It starts with an initial solution, and then at each iteration, it considers a set of solutions in the neighbourhood of the current solution and selects the best one based on the evaluation function. It maintains a tabu list which records the previous search history to avoid getting trapped in local optimal solutions. Since the label-correcting algorithm is time-consuming in the pricing process, we first use TS to find new columns. If TS cannot find new columns, the label-correcting algorithm is invoked. Although the TS in our B&P algorithm is inspired by Desaulniers, Lessard, and Hadjar (2008), there are several differences: (1) lunch break constraints are considered in our TS operators; and (2) only one stop criterion is used in our TS, i.e. the maximum number of iterations $I_{\max}$. In the TS column generation stage, TS runs multiple times using different initial feasible routes that correspond to the basic variables in the optimal solution of the RMP. All negative reduced cost routes are returned into the RMP. At each iteration of TS, we evaluate all non-tabu feasible routs by performing operators on the current route. Two operators are employed to generate new routes: removing a client from the current route and inserting a client not in the current route into it. The reverse operations associated with removed or inserted clients in previous iterations are declared tabu during the search process. In both operators, the lunch break is regarded as an artificial client and not removed from the route; when the break is taken between clients i and j, we check whether the new route is feasible if the break is taken at client i after service or at j before service.

3.4 *Other components*

3.4.1 *Initialisation of RMP*

Except for the root node, the RMP of other nodes in the search tree inherits columns from parent nodes. A set of routes is required as the initial columns of the RMP at the root node. The model in (21)–(25) contains a set of columns which consist of only a client (i.e. decision variable z_i and penalty cp_i). In addition to these columns, we construct another set of initial columns from the feasible solution generated by a heuristic. In the heuristic, the lunch break is treated as an artificial client. The initial routes of workers are the centre – lunch break – the centre. Its details are presented as follows: sort clients in the ascending order of their earliest service start times, insert client i into the route of worker k that can serve him/her ($i \in N_k$) to minimise the increase in total travel cost according to the previously determined client sequence.

3.4.2 *Generation of initial upper bound*

The upper bound at the root node of the search tree is obtained by employing CPLEX to solve the integer programming model in (21)–(25), which uses columns generated in the pricing process of the root node. At other nodes of the search tree, if the optimal solution of the RMP is integer and smaller than the current upper bound, it is updated as the new upper bound, as shown in Figure 1.

3.4.3 *Branching and search strategy*

In the search tree, branching is required when the RMP of each node is solved optimally and the corresponding solution of model in (1)–(20) is not integer. As shown in Figure 1, we sequentially perform two branching strategies if the values of decision variables associated with branching strategies in previous stages are integer. The effectiveness of our branching scheme is validated in the supplemental online material.

(1) Branching on uncovered clients (Ceselli, Righini, and Tresoldi 2014). If some values of z_i ($i \in N$) are fractional, we choose the one which is farthest from an integer to generate two child nodes.

For one child node with $z_i = 1$, which means that client i is prohibited in the solutions, instead of explicitly adding $z_i = 1$ into the model of the RMP, this constraint can be implicitly satisfied by removing the columns with client i from the column sets of the RMP and deleting ingoing and outgoing arcs of client i in the networks of all workers. For the other child node with $z_i = 0$, which indicates that client i must be visited in the solutions, the constraint $z_i = 0$ is added into the model of the RMP, and no modification is required in the column sets of the RMP and the pricing process.

(2) Branching on the assignment of workers to clients (Muter, Cordeau, and Laporte 2014). If all values of z_i ($i \in N$) are integer, we branch on $\rho_{ik} = \sum_{r \in R_k} \alpha_{ir}\theta_r$ ($i \in N$, $k \in K$). If some values of ρ_{ik} are fractional, we select the one that is farthest from an integer to branch in the search process.

If ρ_{ik} equals 1 at the child node, client i must be visited by worker k in the solutions. In this case, this constraint can be achieved by deleting the columns containing client i from the column set $R_{k\prime}$ of worker k' ($k' \neq k$) in the RMP and removing ingoing and outgoing arcs of client i in the network $G_{k'}$ of worker k' ($k' \neq k$). Otherwise, if ρ_{ik} equals 0 at the child node, client i should not be served by worker k in the solutions. This constraint is met by removing the columns with client i from the column set R_k of worker k in the RMP and deleting ingoing and outgoing arcs of client i in the network G_k of worker k.

The node search strategy is such that when choosing a node to be processed next, we choose the one with the smallest lower bound.

4. Computational results

This section reports the results of a series of computational experiments. To the best of our knowledge, there is no public benchmark instance to evaluate the performance of our approach. We first construct different types and scales of test instances. Based on such instances, we then analyse the parameter settings. Finally, we report numerical results on all instances. In the experiments, the penalty parameter for each uncovered client i (cp_i) is set to a large constant 1000 in order to serve as many clients as possible. The RMP involved in the B&P algorithm is solved by CPLEX 12.3. To validate the performance of the proposed algorithm, we solve the model in (1)–(20) using CPLEX. In preliminary experiments, we have tried different parameter settings to tune the performance of CPLEX, e.g. selecting different branching variables and changing branching directions. However, such settings cannot lead to obvious performance improvement, and we choose the default settings. All algorithms are coded in C++. The experiments run on the computer with Intel Xeon E5-2670 2.60-GHz CPU and 128 GB of RAM under Linux. Both CPLEX and our algorithm stop until the optimal solution is obtained or the computing time exceeds three hours.

4.1 *Test instances*

In numerical experiments, we use two sets of instances from the literature and real data to test the proposed algorithm.

The first set of instances is modified from the classical Solomon's VRPTW benchmarks (Solomon 1987). In the modified instances, we generate different sizes of instances: 30 clients with 4 workers, 50 clients with 6 workers, and 100 clients with 12 workers. Travel times and travel costs are equal to the Euclidean distance between the corresponding nodes truncated to integer. Demands of clients are ignored. The subset of clients visited by each worker is listed in the supplemental online material, and its cardinality is 20. The maximum working time of workers is equal to the latest service start time of node 0. The length of the lunch break equals the average service time of clients. The earliest start time of the lunch break is half of the latest service start time of node 0 minus the length of the break, whereas the latest start time is half of the latest service start time of node 0 plus the length of the break. These instances are classified into six classes according to two criteria. The first criterion is the geographical distribution of the clients: clients are clustered in the type C instances, randomly located in the type R instances, and partly clustered, partly randomly located in the type RC instances. The second criterion is the tightness of time windows. In the 1-series instances, time windows are narrower, and in the 2-series instances, time windows are wider. There are totally six classes of instances, i.e. C1, C2, R1, R2, RC1 and RC2. Instances are represented in the following format: C101_30 is the first instance of type C1 where the first 30 clients are used.

The second set of instances is obtained from real-life data provided by an HHC company in Shanghai, China. The company mainly provides HHC services for the elderly. The detailed information of clients, such as locations, service time windows and durations, can be provided by the company. The company allows workers to take a 30-min break between 12:00 and 13:00. To maintain a high level of customer satisfaction, the company appoints two workers to provide services for each client. We select a predefined number of clients from the database to generate different sizes of instances. Three sets of instances with 30, 50 and 100 clients are generated. The number of corresponding available workers is 3, 5 and 10. Instances are represented as the following format: RD30_1 indicates the first instance with 30 clients.

4.2 *Parameter settings and analysis of algorithmic components*

The proposed algorithm has two key parameters: the number of neighbours δ in the *ng*DSSR acceleration strategy and the maximum number of iterations $I_{\max}$ in TS. This section tests δ in the candidate set {5, 10, 15} and $I_{\max}$ in the set {10, 20, 30}. Each combination of values is tested on 12 instances selected from the modified Solomon's VRPTW

Table 1. Computing times in different parameter settings.

	δ		
I_{max}	5	10	15
10	434.6	305.4	479.9
20	439.8	294.0	475.7
30	581.4	320.4	522.2

benchmarks. The average computing time of these instances is used as the indicator to evaluate the result. Table 1 presents the average running times (in seconds) in different parameter settings.

Parameter I_{max} controls the computing time of the TS column generator. As shown in Table 1, when I_{max} increases from 10 to 20, the computing times decrease by an average of approximately 3.5 s, whereas when it equals 30, the computing times increase on an average of 68.0 s. These results indicate that TS is very efficient in the first few column generation iterations, while it may fail to find new columns in the last column generation iterations (Desaulniers, Lessard, and Hadjar 2008). Hence, we think a relatively small value as a good choice of I_{max} in our proposed algorithm.

In terms of parameter δ, it determines the relaxation degree of the pricing sub-problem. When δ is set from a small value to a large one, the computing time becomes larger, and the resulting lower bound of the master problem (i.e. the optimal solution of the RMP) improves (Contardo, Desaulniers, and Lessard 2015). Hence, we should choose a suitable value of δ to balance the quality of the lower bound and the algorithmic computing time. Table 1 demonstrates that when parameters (I_{max}, δ) are set to (20, 10), the average computing time is the shortest. We use this pair of parameter values as the default settings in the proposed algorithm.

Based on such parameter settings, we do a set of preliminary experiments to analyse the impact on the performance of the proposed algorithm of various algorithmic components, such as TS column generator, the proposed acceleration strategy and branching strategy. The details and results of analysis on algorithmic components are reported in the supplemental online material.

4.3 *Results of modified Solomon's VRPTW instances*

In this section, we examine the proposed algorithm on a total of 168 instances originated from Solomon's VRPTW benchmarks. Detailed results are provided in the supplemental online material. Table 2 summarises the results of CPLEX and our B&P algorithm (in two major columns 'CPLEX' and 'B&P') according to the sizes and classes of test instances. For CPLEX results, we provide the number of instances that have been optimally solved (referred to as *solved*) and not solved optimally (*unsolved*), the mean percentage gap between upper and lower bounds of unsolved instances ('*CGap'* calculated as 100% × (upper bound – lower bound)/upper bound), and the average computing time of *solved* instances ('*CCPU'*). In terms of the results of the B&P algorithm, the number of *unsolved* instances and the average gap between upper and lower bounds ('*TGap'*) are presented. For the results of *solved* instances, we then provide the following information: the number of *solved* instances, the average gap and runtime at root node ('*RGap'* and '*RCPU'*), the mean and standard deviation of optimal solutions ('*OptVal Avg'* and '*OptVal Std'*), the number of unvisited clients, branching times, columns, and pricing iterations; and the runtime of the pricing process, solving the model of the RMP by CPLEX, and the whole B&P algorithm ('*PSCPU'*, '*RMPCPU'*, and '*TCPU'*). If the number of *unsolved* and *solved* instances is zero, the corresponding '*CCPU'* and '*TGap'* are left blank.

From the results in Tables 2, we can assert the following observations:

(1) The proposed B&P algorithm can provide good solutions for the HCWSRP-LBR, outperforming CPLEX in terms of the solution quality and computing time. Table 2 shows that out of 168 total instances with different sizes and classes, the proposed B&P algorithm obtains 120 optimal solutions. Considering instances that have not been solved optimally, the mean gap among upper and lower bounds is 7.1%. Compared with the B&P algorithm, CPLEX is able to solve only 25 instances to optimality, and for unsolved instances, its mean gap among bounds is up to 50.2%. The superiority of the B&P algorithm is more apparent in the large test instances with 100 customers. Especially, for large instances with 100 clients, CPLEX finds only one optimal solution, while our B&P algorithm can solve 22 instances optimally. Meanwhile, concerning the computing time, Table 2 shows CPLEX spends much more time solving the problem than the B&P algorithm. These results validate the effectiveness of the proposed algorithm.

Table 2. Results of Solomon's VRPTW instances.

	CPLEX				B&P												
	Unsolved instances		Solved instances		Unsolved instances		Solved instances										
Instance	*Unsolved number*	*CGap %*	*Solved number*	*CCPU (s)*	*Unsolved number*	*TGap %*	*Solved number*	*RGap %*	*RCPU (s)*	*OptVal Avg*	*OptVal Std*	*Unvisited clients*	*Branching times*	*Columns*	*Pricing iterations*	*PSCPU (s)*	*RMPCPU (s)*	*TCPU (s)*
C1_30	4	28.9	5	2068.5	0		9	10.2	10.3	553.8	438.4	0.2	22.0	9241.3	980.9	44.1	2.3	49.3
C2_30	4	17.2	4	996.5	0		8	13.3	758.6	353.5	8.1	0.0	63.9	48,839.9	3550.0	1494.0	11.9	1517.7
R1_30	11	83.7	1	329.4	0		12	17.6	19.1	3796.9	3135.0	3.3	35.3	5413.3	1348.7	222.8	2.2	226.0
R2_30	4	7.0	7	2509.6	0		11	1.0	1103.8	461.3	27.2	0.0	3.5	12,454.5	338.5	1175.5	1.9	1178.8
RC1_30	7	93.6	1	9921.8	1	5.3	7	17.5	41.4	5244.4	1544.0	4.7	113.6	5751.4	3032.6	918.5	4.0	923.6
RC2_30	7	33.9	1	4677.1	1[a]		7	14.6	1482.0	653.3	67.9	0.0	96.3	44,282.7	4940.6	3392.3	14.1	3416.8
C1_50	9	56.0	0		1	3.5	8	11.0	6.6	2945.6	2491.2	2.1	542.6	83,543.3	5210.5	473.1	61.1	546.5
C2_50	6	16.6	2	879.2	1	7.2	7	1.2	43.9	774.3	29.5	0.0	320.6	156,000.7	3524.4	1156.1	94.3	1293.0
R1_50	11	92.6	1	774.9	4	14.0	8	6.0	26.7	12,883.4	4621.1	12.1	123.3	7587.4	974.8	1729.3	9.5	1741.5
R2_50	10	17.7	1	633.3	4	0.7	7	4.1	245.5	948.3	73.0	0.0	77.3	66,540.9	1016.7	1023.1	34.0	1076.5
RC1_50	8	84.4	0		0		8	4.1	71.3	14,184.8	3328.4	13.4	40.3	1646.6	275.3	563.4	1.7	565.8
RC2_50	7	42.6	1	8214.8	2	4.4	6	7.0	699.3	1190.5	118.0	0.0	33.7	28,128.5	530.5	1283.3	17.3	1317.3
C1_100	9	79.7	0		5	7.4	4	3.0	33.6	10,411.8	5556.0	8.0	39.3	45,654.0	564.8	1904.1	46.7	1970.2
C2_100	7	15.4	1	7590.6	1	2.6	7	4.9	152.6	2048.1	107.2	0.0	360.7	188,281.9	4380.1	1243.7	366.2	1765.1
R1_100	12	92.3	0		8	21.9	4	3.8	11.1	29,204.8	8122.9	27.8	733.3	23,247.3	5169.5	2447.4	80.6	2547.0
R2_100	11	24.1	0		10	6.8	1	8.2	7.6	2011.0	0.0	0.0	2292.0	884,386.0	26,091.0	1585.3	2139.7	4418.9
RC1_100	8	93.7	0		5	6.0	3	3.4	7.9	32,344.3	3751.8	30.7	616.3	16,581.0	4162.3	2703.4	51.5	2769.1
RC2_100	8	24.5	0		5	5.3	3	5.4	37.5	2343.7	130.9	0.0	707.7	388,388.0	8732.3	2751.3	895.9	4020.8
Avg		50.2				7.1		7.6	264.4	6797.4	1863.9		345.7		4156.9	1450.6	213.1	1741.3

[a]Since the instance cannot be solved at the root node, the value of TGap is not obtained by the B&P algorithm.

Table 3. Results of real instances.

	CPLEXB&P				B&P													
Instance	*CUB*	*CLB*	*CGap %*	*CCPU (s)*	*RUB*	*RLB*	*RGap %*	*RCPU (s)*	*TUB*	*TLB*	*TGap %*	*Unvisited clients*	*Branching times*	*Columns*	*Pricing iterations*	*PSCPU (s)*	*RMPCPU (s)*	*TCPU (s)*
RD30_1	3475.0	557.2	84.0	–	2543.0	2247.0	11.6	1.6	2533.0	2533.0	0.0	2	9	3260	126	11.4	1.1	12.9
RD30_2	1566.0	592.9	62.1	–	2528.0	1403.2	44.5	1.7	1566.0	1566.0	0.0	1	5	2581	97	6.3	0.7	8.9
RD30_3	1475.0	492.7	66.6	–	1505.0	736.6	51.1	1.9	1475.0	1475.0	0.0	1	17	7825	247	9.3	2.6	13.5
RD30_4	2519.0	1989.8	21.0	–	3509.0	2273.3	35.2	1.2	2519.0	2519.0	0.0	2	9	3626	140	4.9	1.0	6.9
RD30_5	2497.0	602.4	75.9	–	2537.0	1413.1	44.3	1.6	1580.0	1580.0	0.0	1	10	6860	199	8.0	1.8	12.1
RD30_6	2593.0	606.8	76.6	–	1607.0	1408.0	12.4	2.6	1607.0	1607.0	0.0	1	5	2300	94	6.7	0.7	7.6
RD30_7	1527.0	535.6	64.9	–	1537.0	1274.9	17.1	1.4	1527.0	1527.0	0.0	1	7	3855	131	4.4	1.1	6.5
RD30_8	2514.0	556.5	77.9	–	2557.0	2326.0	9.0	1.7	2514.0	2514.0	0.0	2	6	3350	116	6.4	0.9	8.0
RD30_9	385.0	385.0	0.0	2268.5	400.0	376.8	5.8	3.6	385.0	385.0	0.0	0	3	4571	109	6.9	0.9	8.6
RD30_10	1535.0	625.2	59.3	–	1546.0	745.9	51.7	1.3	1535.0	1535.0	0.0	1	17	5847	249	9.1	1.7	11.7
RD50_1	2856.0	810.7	71.6	–	2885.0	1947.0	32.5	7.3	2839.0	2839.0	0.0	2	167	44,966	1999	260.4	38.2	307.0
RD50_2	4878.0	819.7	83.2	–	3935.0	3308.9	15.9	4.0	3883.0	3883.0	0.0	3	152	31,042	1806	280.9	22.9	308.2
RD50_3	2854.0	741.2	74.0	–	2022.0	967.2	52.2	5.3	1935.0	1935.0	0.0	1	264	70,699	3505	446.4	68.2	527.3
RD50_4	1962.0	808.5	58.8	–	1955.0	996.8	49.0	4.9	1916.0	1916.0	0.0	1	101	20,104	970	143.9	24.4	174.2
RD50_5	3921.0	790.5	79.8	–	3920.0	2729.5	30.4	3.0	3878.0	3878.0	0.0	3	1215	233,814	11,422	1193.6	160.3	1380.1
RD50_6	3634.0	654.0	82.0	–	3660.0	1738.3	52.5	5.5	2698.0	2698.0	0.0	2	234	61,628	2594	493.2	50.3	573.3
RD50_7	3815.0	781.0	79.5	–	3862.0	2602.9	32.6	3.7	3815.0	3815.0	0.0	3	1321	258,202	13,244	2203.1	227.7	2470.8
RD50_8	2779.0	730.0	73.7	–	2813.0	1917.0	31.9	4.0	2779.0	2779.0	0.0	2	136	48,359	1725	151.8	36.9	199.3
RD50_9	1831.0	697.6	61.9	–	1857.0	807.4	56.5	4.8	1802.0	1802.0	0.0	1	348	89,252	3997	826.9	56.6	896.1
RD50_10	5832.0	896.5	84.6	–	4872.0	3715.9	23.7	3.0	4858.0	4858.0	0.0	4	2424	343,600	22,685	2460.2	234.1	2724.8
RD100_1	10,486.0	1357.5	87.1	–	6573.0	4529.2	31.1	17.0	5605.0	5322.7	5.0	4	898	150,614	9392	4984.6	663.7	–
RD100_2	8780.0	1513.3	82.8	–	7716.0	6420.3	16.8	15.2	7716.0	7163.8	7.2	6	1560	429,971	17,620	6745.8	1017.4	–
RD100_3	10,621.0	1363.6	87.2	–	5767.0	3455.3	40.1	16.1	4759.0	4502.7	5.4	3	1605	271,239	15,736	8240.8	1072.3	–
RD100_4	9623.0	1489.0	84.5	–	6883.0	5762.5	16.3	14.3	6717.0	6650.9	1.0	5	1376	302,313	16,788	6040.4	867.4	–
RD100_5	8650.0	1432.0	83.4	–	4818.0	3676.8	23.7	14.8	4795.0	4571.3	4.7	3	1635	370,144	18,235	8713.3	1330.3	–
RD100_6	10,503.0	1423.6	86.4	–	4868.0	4032.8	17.2	16.3	4828.0	4828.0	0.0	3	627	106,400	6347	3255.7	377.7	3713.3
RD100_7	9698.0	1491.3	84.6	–	5830.0	4498.6	22.8	12.3	5830.0	5329.0	8.6	4	1686	395,463	17,513	5928.9	907.5	–
RD100_8	6651.0	1457.8	78.1	–	5807.0	4704.8	19.0	10.7	5688.0	5688.0	0.0	4	836	221,265	9955	3850.5	604.3	4571.0
RD100_9	9677.0	1442.9	85.1	–	5829.0	5197.7	10.8	14.8	5767.0	5767.0	0.0	4	905	269,415	12,114	4177.7	524.7	4789.7
RD100_10	6759.0	1535.0	77.3	–	4895.0	3889.7	20.5	22.9	4864.0	4648.4	4.4	3	1235	264,032	14,927	7341.1	736.9	–

(2) Concerning different types of instances, we find that 1-series instances are more difficult to solve by CPLEX than 2-series. For example, out of 87 1-series instances, CPLEX finds 8 optimal solutions, whereas it can find 17 optimal solutions out of 81 2-series instances. Furthermore, for all unsolved 1-series instances, the mean gap between upper and lower bounds is rather larger than that of 2-series. This is because the time windows of 1-series instances are tighter than that of 2-series and more clients are not visited in the solutions (see '*Unvisited Clients'*). Such factors lead to looser lower bounds of CPLEX on 1-series instances. Considering the proposed B&P algorithm, we find that it spends more time in solving 2-series instances than 1-series. This is because facing wider time windows, more clients are involved in the resulting network, more underlying feasible labels are present, and more time is consumed by the pricing process (see '*Columns'* and '*PSCPU'*).

The standard deviation of the optimal solutions of 1-series instances is smaller than that of 2-series. In the modified Solomon's instances, the demand of each client is ignored, and the time window is the unique resource constraint of the problem. When the time window becomes wider, it has less impact on the optimal solutions; the problem is relaxed and degenerated into the capacitated vehicle routing problem.

(3) We find that the pricing process used to generate new columns is the most time-consuming component in the proposed B&P algorithm. On average, it consumes approximately 93% of the total computational time. This means that the effectiveness of the pricing process has an obvious impact on the performance of the proposed algorithm.

4.4 *Results of real-life instances*

In this section, we test 30 instances generated from real-life data. Table 3 shows the details of the results of test instances. For CPLEX results, the upper bound ('*CUB'*), the lower bound ('*CLB'*), the gap ('*CGap'*) between upper and lower bounds, and the computing time ('*CCPU'*) are listed. For the results by the B&P algorithm, the upper bound ('*RUB'*), the lower bound ('*RLB'*), the gap ('*RGap'*) between bounds, the computing time ('*RCPU'*) at the root node, the upper bound ('*TUB'*), the lower bound ('*TLB'*), their gap ('*TGap'*), the computing time ('*TCPU'*) in the search tree are described. Furthermore, the number of unvisited clients, branching times, columns, and pricing iterations; and the computing time of pricing process ('*PSCPU'*) and solving the model of the RMP by CPLEX ('*RMPCPU'*) are described. The '—' represents that the computing time exceeds three hours or the memory in the computing process exceeds the computer memory.

For test instances with 30 customers, the column generation algorithm solves the RMP at the root node optimally within an average of 1.9 s, the mean gap at the root node is 28.3%, and the proposed algorithm exactly solves them in approximately 9.7 s, whereas in three hours, CPLEX solves one instance to optimality, and the average gap over other instances is 65.4%. For test instances with 50 customers, the column generation algorithm enables the average gap at the root node to reach 37.7% within 4.5 s and the B&P algorithm solves all instances optimally in 95.6 s, whereas in the predefined time, CPLEX cannot solve any instance optimally, the average gap is 74.9%. As the size of instances increases, the proposed algorithm spends more time solving them. For test instances with 100 customers, the optimal solution of the RMP at the root node is obtained in an average of 15.4 s by the column generation algorithm, the average gap at the root node is 21.8%, the optimal solution of three instances is acquired by the proposed algorithm, the mean gap of other instances within the predefined time is 5.2%, whereas no instance can be solved optimally in three hours by CPLEX, and the average gap is 83.6%. These results show that the proposed B&P algorithm outperforms CPLEX when solving the HCWSRP-LBR.

5. Conclusions

This paper addresses the HCWSRP-LBR from a real-life application. A three-index mathematical model is used to describe the problem and evaluate the effectiveness of the solution approach. The popular B&P algorithm is proposed to solve the problem. Considering the features of LBR, the label definition, extension rules, and dominance rules are devised in the label-correcting algorithm. The bidirectional extension, decremental state space relaxation, *ng*-route relaxation, and TS are employed to accelerate the column generation process. A hierarchical branching scheme including branching on uncovered clients and the assignment of workers to clients is developed to ensure the integrity of solutions. To validate its effectiveness, the proposed B&P algorithm is compared with CPLEX with default settings by different classes of instances from the literature and real data. Numerical results show that the proposed algorithm produces high-quality solutions within a reasonable computing time span, and is always superior to CPLEX in terms of both the solution quality and computing time.

Acknowledgements

The authors thank the referees and editor for their helpful comments that significantly contributed to improving the quality of the paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Research Grant from National Natural Science Foundation of China [71302013]; Specialized Research Fund for the Doctoral Program of Higher Education of China [20130073120047]; Shanghai Natural Science Foundation of China [13ZR1456400].

Supplemental data

Supplemental data for this article can be accessed http://dx.doi.org/10.1080/00207543.2016.1213917.

References

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice-Hall.

Akjiratikarl, C., P. Yenradee, and P. R. Drake. 2007. "PSO-based Algorithm for Home Care Worker Scheduling in the UK." *Computers & Industrial Engineering* 53 (4): 559–583.

Baldacci, R., A. Mingozzi, and R. Roberti. 2011. "New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem." *Operations Research* 59 (5): 1269–1283.

Bard, J. F., Y. Shao, and H. Wang. 2013. "Weekly Scheduling Models for Traveling Therapists." *Socio-Economic Planning Sciences* 47 (3): 191–204.

Bard, J. F., Y. F. Shao, and A. I. Jarrah. 2014a. "A Sequential GRASP for the Therapist Routing and Scheduling Problem." *Journal of Scheduling* 17 (2): 109–133.

Bard, J. F., Y. F. Shao, X. T. Qi, and A. I. Jarrah. 2014b. "The Traveling Therapist Scheduling Problem." *IIE Transactions* 46 (7): 683–706.

Begur, S. V., D. M. Miller, and J. R. Weaver. 1997. "An Integrated Spatial DSS for Scheduling and Routing Home-health-care Nurses." *Interfaces* 27 (4): 35–48.

Bertels, S., and T. Fahle. 2006. "A Hybrid Setup for a Hybrid Scenario: Combining Heuristics for the Home Health Care Problem." *Computers & Operations Research* 33 (10): 2866–2890.

Boland, N., J. Dethridge, and I. Dumitrescu. 2006. "Accelerated Label Setting Algorithms for the Elementary Resource Constrained Shortest Path Problem." *Operations Research Letters* 34 (1): 58–68.

Ceselli, A., G. Righini, and E. Tresoldi. 2014. "Vehicle Routing Problems with Different Service Constraints: A Branch-and-cut-and-price Algorithm." *Networks* 64 (4): 282–291.

Cheng, E., and J. L. Rich. 1998. *A Home Health Care Routing and Scheduling Problem*. Houston: Rice University, Tech. Rep. TR98-04.

Contardo, C., G. Desaulniers, and F. Lessard. 2015. "Reaching the Elementary Lower Bound in the Vehicle Routing Problem with Time Windows." *Networks* 65 (1): 88–99.

Dayarian, I., T. G. Crainic, M. Gendreau, and W. Rei. 2015. "A Column Generation Approach for a Multi-attribute Vehicle Routing Problem." *European Journal of Operational Research* 241 (3): 888–906.

Desaulniers, G., F. Lessard, and A. Hadjar. 2008. "Tabu Search, Partial Elementarity, and Generalized K -Path Inequalities for the Vehicle Routing Problem with Time Windows." *Transportation Science* 42 (3): 387–404.

Eveborn, P., P. Flisberg, and M. Rönnqvist. 2006. "Laps Care – An Operational System for Staff Planning of Home Care." *European Journal of Operational Research* 171 (3): 962–976.

Feillet, D., P. Dejax, M. Gendreau, and C. Gueguen. 2004. "An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems." *Networks* 44 (3): 216–229.

Mankowska, D., F. Meisel, and C. Bierwirth. 2014. "The Home Health Care Routing and Scheduling Problem with Interdependent Services." *Health Care Management Science* 17 (1): 15–30.

Muter, I., J.-F. Cordeau, and G. Laporte. 2014. "A Branch-and-price Algorithm for the Multidepot Vehicle Routing Problem with Interdepot Routes." *Transportation Science* 48 (3): 425–441.

Nickel, S., M. Schroder, and J. Steeg. 2012. "Mid-term and Short-term Planning Support for Home Health Care Services." *European Journal of Operational Research* 219 (3): 574–587.

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

Righini, G., and M. Salani. 2006. "Symmetry Helps: Bounded Bi-directional Dynamic Programming for the Elementary Shortest Path Problem with Resource Constraints." *Discrete Optimization* 3 (3): 255–273.

Righini, G., and M. Salani. 2008. "New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem." *Networks* 51 (3): 155–170.

Shao, Y., J. F. Bard, and A. I. Jarrah. 2012. "The Therapist Routing and Scheduling Problem." *IIE Transactions* 44 (10): 868–893.

Solomon, M. M. 1987. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints." *Operations Research* 35 (2): 254–265.

Trautsamwieser, A., and P. Hirsch. 2014. "A Branch-price-and-cut Approach for Solving the Medium-term Home Health Care Planning Problem." *Networks* 64 (3): 143–159.

Yuan, B., R. Liu, and Z. Jiang. 2015. "A Branch-and-price Algorithm for the Home Health Care Scheduling and Routing Problem with Stochastic Service times and Skill Requirements." *International Journal of Production Research* 53 (24): 7450–7464.