European Journal of Operational Research 244 (2015) 379–391

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The discrete time window assignment vehicle routing problem

Remy Spliet [a,*], Guy Desaulniers [b]

[a] Econometric Institute, Erasmus University, Burgemeester Oudlaan 50, 3000 DR Rotterdam, Netherlands
[b] Department of Mathematics and Industrial Engineering and GERAD, Polytechnique Montréal, Montréal, H3T 1J4, Québec, Canada

ARTICLE INFO

Article history:
Received 27 November 2013
Accepted 11 January 2015
Available online 28 January 2015

Keywords:
Vehicle routing
Time window assignment
Column generation
Uncertain demand

ABSTRACT

In this paper we introduce the discrete time window assignment vehicle routing problem (DTWAVRP) that can be viewed as a two-stage stochastic optimization problem. Given a set of customers that must be visited on the same day regularly within some period of time, the first-stage decisions are to assign to each customer a time window from a set of candidate time windows before demand is known. In the second stage, when demand is revealed for each day of the time period, vehicle routes satisfying vehicle capacity and the assigned time windows are constructed. The objective of the DTWAVRP is to minimize the expected total transportation cost. To solve this problem, we develop an exact branch-price-and-cut algorithm and derive from it five column generation heuristics that allow to solve larger instances than those solved by the exact algorithm. We illustrate the performance of these algorithms by means of computational experiments performed on randomly generated instances.

1. Introduction

In distribution networks, it is common for a supplier and a customer to agree on a time window in which a delivery will be made. This time window is often used repeatedly within some period of time in which multiple deliveries are made at regular intervals. At the moment of choosing a time window for a customer, its demand is usually unknown and may fluctuate for different deliveries. When the demands of all customers become known for a given day, a vehicle routing problem with time windows (VRPTW) must be solved to construct a delivery schedule within the agreed time windows.

The time window assignment vehicle routing problem (TWAVRP) was recently introduced by Spliet and Gabor (2014). This problem can be viewed as a two-stage stochastic optimization problem (see Birge & Louveaux, 1997). Given a set of customers to be visited on the same day regularly during some period of time (e.g., a season), the TWAVRP consists of assigning in the first stage a time window to each customer before customer demand is known. In the second stage, when demand is revealed for each day of the time period, vehicle routes that respect the assigned time windows are constructed. The assigned time windows have a predetermined width and can start at any time within an exogenous time window that can be customer-dependent. The objective of the TWAVRP is to minimize the expected total transportation cost over the whole time period that is equivalent to minimizing the expected cost for a single day.

In this paper, we study the discrete TWAVRP (DTWAVRP) that differs from the TWAVRP by considering for each customer a finite set of candidate time windows from which one has to be selected. For example, a customer might divide the day in blocks of 2 hours commencing on the hour and require one of these blocks to be the assigned time window. We have encountered such time window assignment problems (discrete or not) while collaborating with Dutch retail chains, and believe they are common in this industry. Here, the retailers (customers) are heavily dependent on the time window to be fixed in advance and kept for some time. For instance, a retailer might receive all its deliveries on the same day of the week and more or less the same hour of the day for an entire year. This is crucial for many operational purposes like inventory management and the scheduling of personnel. Considering a discrete set of time windows is often more practical for the retailers, especially to ease the personnel scheduling process which must take into account various regulations. Furthermore, it can give them the opportunity to express preferences for the time windows. Maximizing the satisfaction of these preferences might be taken into account as a secondary objective during the optimization process, an option that is not considered in this paper.

As is common for two-stage stochastic optimization problems, we suggest to approximate the probability distributions of the customer demands by a finite set of possible scenarios of demand realizations. In this framework, the DTWAVRP is clearly NP-hard as, in the case of one scenario and one candidate time window per customer, it reduces to the VRPTW. When it involves several scenarios, the DTWAVRP

* Corresponding author. Tel.: +31 104081342.
E-mail addresses: spliet@ese.eur.nl (R. Spliet), guy.desaulniers@gerad.ca (G. Desaulniers).

http://dx.doi.org/10.1016/j.ejor.2015.01.020
0377-2217/

corresponds to solving several VRPTWs (one per scenario) that are linked together by the choice of the time windows. The VRPTW is a well-studied problem for which many exact and heuristic algorithms have been developed (see, e.g., the surveys of Baldacci, Mingozzi, & Roberti, 2012; Bräysy & Gendreau, 2005a, 2005b; Kallehauge, Larsen, Madsen, & Solomon, 2005).

We believe that in the scientific literature, the problem of assigning time windows before knowing demand has been largely overlooked so far. It was only tackled recently by Spliet and Gabor (2014) who designed an exact branch-price-and-cut algorithm for the TWAVRP that can solve instances with up to 25 customers and three scenarios. In their solution approach, the pricing problem is modeled as an elementary shortest path problem with linear node costs (induced because the set of time windows is not discrete) and capacity and time window constraints. Due to the complexity of this problem route relaxation is required, but only basic route relaxation can be utilized efficiently: allowing all cyclic routes and eliminating 2-cycles. For the DTWAVRP we also develop a branch-price-and-cut algorithm, but in this case we are able to employ the more sophisticated *ng*-route relaxation introduced by Baldacci, Mingozzi, and Roberti (2011). The complexity of the pricing problem of Spliet and Gabor (2014) also prohibits the use of certain types of valid inequalities as these add to the complexity. For instance subset row inequalities, introduced by Jepsen, Petersen, Spoorendonk, and Pisinger (2008), are not used in the algorithm for the TWAVRP, whereas they are used in our algorithm for the DTWAVRP. Furthermore, as opposed to the algorithm for the TWAVRP, we add a heuristic pricing algorithm to speed up computations and develop several column generation heuristics.

Introduced by Groër, Golden, and Wasil (2009), the consistent vehicle routing problem (ConVRP) shares some similarities with the DTWAVRP when considering demand scenarios. In this deterministic problem, each customer must be visited on different days of a given horizon (not all customers must be serviced each day) following a consistent schedule, that is, the arrival times at a customer from one day to another cannot differ by more than a limited amount of time. Moreover, it is required that each customer is always visited by the same driver. Groër et al. (2009) found optimal solutions to ConVRP instances involving up to 12 customers and three scenarios using a commercial mixed integer programming solver. They reported computation times of up to several days. Furthermore, they developed a local search heuristic to solve instances with over 3700 customers.

Jabali, Leus, van Woensel, and de Kok (2013) considered another related problem that involves the assignment of time windows in a vehicle routing problem with stochastic travel times and deterministic demands. They developed a tabu search algorithm for solving it. Also, Agatz, Campbell, Fleischmann, and Savelsbergh (2011) studied a stochastic problem faced by e-tailers providing home delivery that consists of selecting which time slots to offer per zip code for making deliveries. They developed a local search heuristic.

The main contributions of this paper are as follows. First, we propose a new problem, the DTWAVRP. Second, we develop a state-of-the-art exact branch-price-and-cut algorithm to solve it and report computational results obtained on randomly generated instances to evaluate the effectiveness of some of its components. Third, from this exact algorithm, we derive several column generation heuristics that can find good solutions to the DTWAVRP in limited computation times. This allows to find solutions to instances that are larger than those solved by the exact algorithm. We report computational results to compare the performance of five such algorithms. Finally, we also provide additional results to assess the benefits of using several demand scenarios to assign time windows to the customers instead of using a single one defined by the average demands as it is usually done in practice.

In the next section, we define the DTWAVRP using demand scenarios and we present an integer programming formulation for it. In Section 3, we introduce the proposed branch-price-and-cut algorithm. In Section 4, we describe the five column generation heuristics. The results of our computational experiments are reported in Section 5. Finally, conclusions are drawn in Section 6.

2. Problem definition

Consider a complete graph $G = (N, A)$, where $N = \{0, \dots, n+1\}$ is a set of locations such that 0 and $n+1$ represent the depot at the start and the end of a day, respectively, and $N' = \{1, \dots, n\}$ the set of customers. Let $c_{ij} \geq 0$ be the cost to travel along arc (i, j) and $t_{ij} \geq 0$ the corresponding travel time (including, if any, the service time at i). Both the travel costs and travel times satisfy the triangle inequality. Furthermore, an unlimited number of vehicles of equal capacity Q is available.

Let the random variable $\mathbf{d}_i$ with known distribution represent the demand of customer $i \in N'$. We distinguish between two stages, in the first stage the demand realization is not yet known, but in the second stage it is. Furthermore, the realization d_i of demand is such that $0 < d_i \leq Q$.

Associate with each customer i a set W_i of candidate time windows that may or may not overlap. In the first stage, one time window $w = [\underline{w}, \overline{w}] \in W_i$ must be selected for each customer such that service at customer i starts between $\underline{w}$ and $\overline{w}$. For the starting and ending depot only one time window exists. Note that waiting at a location is allowed, i.e., a vehicle can arrive prior to the time window lower bound and start service later.

The DTWAVRP is a two-stage stochastic programming problem that consists of assigning one time window to each customer in the first stage and determining in the second stage an optimal vehicle routing schedule satisfying the assigned time windows. The expected total travel cost must be minimized.

We can express the DTWAVRP more formally as follows. Let y_{iw}, for $i \in N'$ and $w \in W_i$, be the time window assignment variables. These are the first-stage decision variables indicating whether time window w is selected for customer i. Note that we do not associate costs with the first-stage decisions. We denote by VRPTW$(y, \mathbf{d})$ the minimal transportation costs as a function of a time window assignment described by y and demand $\mathbf{d}$. This is the recourse function that provides the second stage costs which are obtained by solving the corresponding VRPTW to optimality. Below we first introduce additional notation to express VRPTW$(y, \mathbf{d})$ using a set-partitioning formulation.

Using the set of candidate time windows for each customer, we can construct an auxiliary graph $\hat{G} = (\hat{N}, \hat{A})$, where $\hat{N} = \{(i, w) \mid i \in N, w \in W_i\}$ contains a copy of each customer node i for each of its possible time windows $w \in W_i$. Moreover, $\hat{A}$ contains an arc between two nodes (i, w) and (i', w'), $i \neq i'$, if and only if $\underline{w} + t_{ii'} \leq \overline{w}'$.

We use the term route to refer to a pair $(\hat{P}, t)$ where $\hat{P}$ is a path in $\hat{G}$ starting at 0 and ending at $n+1$ and t is a vector containing the service start time at each location on the path. Associated with each route r is the parameter e_{iwr} indicating the number of times customer i is visited within time window $w \in W_i$ on route r (note that $e_{iwr} \in \{0, 1\}$ if P is an elementary path). To each route r whose path contains the arcs $\{a_1, \dots, a_k\}$ we assign the cost $c_r = \sum_{i=1}^{k} c_{a_i}$.

Let $R(d)$ be the set of all feasible routes corresponding to demand realization d. A route $(\hat{P}, t)$ is considered feasible if (i) it satisfies the capacity constraint, (ii) t is such that, for each customer i on the route, service begins within a time window in W_i, and (iii) if location j is visited directly after i on route r then $t_i + t_{ij} \leq t_j$.

Let the variables x_r, $r \in R(d)$, be route variables indicating whether route r is selected. We formulate the recourse function VRPTW(y, d) using the following set-partitioning formulation.

$$\text{VRPTW}(y,d) = \min \sum_{r\in R(d)} c_r x_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r\in R(d)} e_{iwr} x_r = y_{iw} \quad \forall i \in N',\ \forall w \in W_i, \tag{2}$$

$$x_r \in \{0,1\} \quad \forall r \in R(d) \tag{3}$$

The objective function (1) represents total traveling costs. Constraints (2) impose that each customer is visited exactly once within the selected time window as indicated by y. Note that all constraints (2) for which y_{iw} takes value 0 can be removed if the set $R(\mathbf{d})$ is filtered accordingly. Finally, (3) are the integrality requirements on the x variables.

Using the recourse function $\text{VRPTW}(y,d)$ and denoting by E the expected value function, the DTWAVRP can be formulated as the following minimization problem:

$$\min \quad E(\text{VRPTW}(y,\mathbf{d})) \tag{4}$$

$$\text{s.t.} \quad \sum_{w\in W_i} y_{iw} = 1 \quad \forall i \in N' \tag{5}$$

$$y_{iw} \in \{0,1\} \quad \forall i \in N',\ \forall w \in W_i \tag{6}$$

Here, the objective function (4) aims at minimizing the expected costs. Constraints (5) ensure that for every customer exactly one time window is selected and (6) are the integrality requirements on the y variables.

2.1. Deterministic equivalent problem

We assume in this paper that the demand follows a discrete distribution and present next the deterministic equivalent problem (DEP) of the DTWAVRP. We do this for the following two reasons. First, in many applications goods are delivered in integer amounts, on pallets or in roll cages. Hence only a finite (but possibly large) number of demand realizations are possible. Second, the DEP can also be used for a sample average approximation strategy when the number of realizations is too large or demand follows a continuous distribution. In this case exact optimization can likely not be done in a limited amount of computation time due to the complexity of the recourse function, which entails solving a VRPTW.

To present the DEP we introduce S which is a set of demand scenarios, where each scenario is characterized by a realization of the demand at every customer. Let d_i^s be the demand of customer i in scenario $s \in S$ such that $0 < d_i^s \leq Q$. The probability that scenario s occurs is p^s. For ease of notation, we replace $R(d^s)$ by $R(s)$ to denote the set of feasible routes corresponding to realization d^s.

Introducing the variables x_r^s, $s \in S$, $r \in R(s)$, as the route variables indicating whether route r is selected in scenario s, the DEP of the DTWAVRP can be formulated as the following integer linear program.

$$\min \quad \sum_{s\in S} p^s \sum_{r\in R(s)} c_r x_r^s \tag{7}$$

$$\text{s.t.} \quad \sum_{w\in W_i} y_{iw} = 1 \quad \forall i \in N' \tag{8}$$

$$\sum_{r\in R(s)} e_{iwr} x_r^s = y_{iw} \quad \forall i \in N',\ \forall w \in W_i,\ \forall s \in S \tag{9}$$

$$x_r^s \in \{0,1\} \quad \forall s \in S,\ \forall r \in R(s) \tag{10}$$

$$y_{iw} \in \{0,1\} \quad \forall i \in N',\ \forall w \in W_i \tag{11}$$

The objective function (7) minimizes the expected costs over all scenarios resulting from a time window assignment. Constraints (8) ensure that exactly one time window is selected for each customer. Constraints (9) enforce that each customer is visited exactly once within its selected time window in each scenario. Integrality requirements on the x and y variables are provided by (10) and (11).

2.2. Integrality requirements

Next, let us discuss how to reformulate the integrality requirements. Consider the linear programming (LP) relaxation of formulation (7)–(11) where the integrality requirements on the x and y variables are relaxed continuously. For each scenario, let the arc flow in $\hat{G}$ be the cumulative value by which each arc $a \in \hat{A}$ is selected in a solution to this LP relaxation. More precisely, if e_a^r represents the number of times route r traverses arc a, the arc flow in $\hat{G}$ on arc a in scenario s is given by $\hat{F}_a^s = \sum_{r\in R(s)} e_a^r x_r^s$. It is straightforward that when the arc flow in $\hat{G}$ is integer for every scenario, it also provides an integer solution to the DTWAVRP.

Moreover, a solution to the LP relaxation also corresponds to an arc flow in G for each scenario. Observe that when the arc flow in $\hat{G}$ is integer for every scenario, so is the arc flow in G. However, when the arc flow in G is integer, the arc flow in $\hat{G}$ might not be. Nevertheless, in this case an integer solution can always be derived by selecting for each customer the time window with the earliest start time among the time windows that are fractionally selected in the LP relaxation. The corresponding routes can now straightforwardly be found using the integer arc flow in G and the selected time windows. This result is stated more formally in Proposition 1.

Proposition 1. *Let (x,y) be a solution to the LP relaxation of formulation (7)–(11). When the corresponding arc flow in G is integer for every scenario, there exists an integer solution (x^*,y^*) to the DTWAVRP of equal value.*

The proof of Proposition 1 can be found in Appendix A. In this proof, it is verified that the integer solution found as described above is indeed feasible. We would like to emphasize that integrality of the time window assignment variables is not required at all. Also, integrality of the arc flow in the auxiliary graph $\hat{G}$ is not required, but merely integer arc flow in the aggregate graph G. In the rest of this paper, we relax the integrality conditions (10) and (11), and instead only impose integrality on the arc flow in G. As a result, the amount of branching necessary in our algorithm is reduced.

3. Exact algorithm

In this section, we provide an exact algorithm to solve the DEP of the DTWAVRP. We first describe the column generation algorithm that we use to solve the LP relaxation of (7)–(11). In particular, we present the *ng*-route relaxation introduced by Baldacci et al. (2011) and discuss acceleration strategies to speed up the pricing algorithm. Next, we suggest valid inequalities to strengthen the LP bound. Finally, we describe the branch-price-and-cut algorithm.

3.1. Column generation algorithm

In practice, the LP relaxation of (7)–(11), also called the master problem, contains a very large number of variables. To overcome this difficulty, we solve the master problem using a column generation algorithm as introduced by Ford and Fulkerson (1958) for multi-commodity flow problems and by Dantzig and Wolfe (1960) for general linear programming problems. This algorithm iteratively solves a restricted master problem (RMP) and a pricing problem. The RMP is the master problem where only a subset of the route variables are included. It is solved to find a feasible primal solution and the values of the dual multipliers associated with constraints (8) and (9). The pricing problem is solved to identify route variables with negative reduced costs that have not yet been added to the RMP. When a route with a negative reduced cost is identified, its variable is added to the RMP and the procedure is repeated. If no route with a negative reduced cost exists, the current solution to the RMP is also optimal for the master problem.

For the DTWAVRP, the pricing problem can be decoupled into several problems, one for each scenario. The pricing problem for scenario s aims at finding a feasible route for that scenario with the least reduced cost. Let λ be the vector of unrestricted dual multipliers associated with constraints (9). The reduced cost of a route $r \in R(s)$ is given by

$$p^s c_r - \sum_{i \in N'} \sum_{w \in W_i} \lambda^s_{iw} e_{iwr}. \tag{12}$$

This pricing problem can be modeled as an elementary shortest path problem with resource constraints defined on network $\hat{G}$. To do so, associate with each node $(i, w) \in \hat{N}$ the demand d^s_i and with each arc $((i, w), (i', w')) \in \hat{A}$ the (reduced) cost $p^s c_{ii'} - \lambda^s_{i'w'}$ and the travel time $t_{ii'}$. The pricing problem consists of finding a shortest elementary $(0, n+1)$-path in $\hat{G}$ that respects time windows and vehicle capacity (the resource constraints). Note, however, that elementarity is required for the customers. This means that for each customer $i \in N'$ at most one node $(i, w) \in \hat{N}$ can be included in an elementary path.

To solve the pricing problem, we use the labeling algorithm proposed by Desrochers (1988) and enhanced by Feillet, Dejax, Gendreau, and Gueguen (2004) which we modify to consider elementarity of the customers instead of the nodes in network $\hat{G}$. In this algorithm, constructed partial paths are represented by labels. Let l be a label representing a partial path from the starting depot to a specific node $(i, w) \in \hat{N}$. Let $c(l)$ be the total reduced cost of the partial path represented by label l, $t(l)$ its earliest start of service time at customer i in time window w, and $q(l)$ its total load. Finally, let $f_j(l)$, $j \in N'$, be a binary parameter equal to 1 if customer j has already been visited on the partial path associated with label l or if this path cannot be feasibly extended to reach any node representing customer j as this would violate capacity or time window constraints. In this respect, we define the function $U^s_j(l)$ that takes value 1 if $q(l) + d^s_j > Q$ or $t(l) + t_{ij} > \overline{w}$ for all $w \in W_j$, indicating whether l can be extended to j.

The labeling algorithm starts with a single label associated with depot node 0. Next, labels are extended along the arcs in $\hat{G}$. A label l associated with a node (i, w) can be extended to a node (i', w') only if $((i, w), (i', w')) \in \hat{A}$ and $f_{i'}(l) = 0$. To perform this extension and create a label l', we use the following extension functions:

$$c(l') = c(l) + p^s c_{ii'} - \lambda^s_{i'w'} \tag{13}$$

$$s(l') = \max\{t(l) + t_{ii'}, \underline{w}'\} \tag{14}$$

$$q(l') = q(l) + d^s_{i'} \tag{15}$$

$$f_j(l') = \begin{cases} 1 & \text{if } j = i' \\ \max\{f_j(l), U^s_j(l')\} & \text{otherwise} \end{cases} \qquad \forall j \in N'. \tag{16}$$

Label l' is deemed feasible if $s(l') \leq \overline{w'}$. Otherwise, it is discarded. Note that it is not necessary to check if $q(l') \leq Q$ because $f_{i'}(l) = 0$.

In order to avoid the enumeration of all partial paths, a dominance procedure is applied. The aim of this procedure is to remove all non-Pareto optimal labels. A label that is not Pareto optimal is said to be dominated. Label l' is dominated if there exists a label l associated with the same customer and $c(l) \leq c(l')$, $t(l) \leq t(l')$, $q(l) \leq q(l')$ and $f_j(l) \leq f_j(l')$ for all $j \in N'$. We want to emphasize the fact that we check dominance for labels at the same customer instead of at the same node as we require elementarity for the customers and not for the nodes. This increases the number of dominated labels.

This labeling algorithm might provide multiple routes with negative reduced costs. In our implementation of the column generation algorithm, we add to the RMP all route variables with a negative reduced cost identified by the labeling algorithm.

3.2. Route relaxations

As solving the elementary shortest path problem with resource constraints is computationally expensive, it is common to relax elementarity. Generating routes containing cycles and adding them to the formulation does not alter the optimal integer solution as each customer is visited exactly once. However, the LP bound becomes weaker. For the VRPTW, Desrochers, Desrosiers, and Solomon (1992) were the first to suggest a branch-and-price algorithm using a non-elementary shortest path problem as the pricing problem. They eliminate 2-cycles, i.e., cycles of the form $i - j - i$, to strengthen the LP bound at the expense of limited additional computation time. Irnich and Villeneuve (2006) extended this approach by providing an algorithm to solve the shortest path problem with resource constraints and k-cycle elimination, for arbitrary values of k.

Recently, Baldacci et al. (2011) proposed the *ng*-route relaxation. For each customer $i \in N'$ a neighborhood $N_i \subseteq N'$ with $i \in N_i$ is introduced. An *ng*-path is not necessarily an elementary path. Indeed, cycles starting and ending at a customer i are allowed if this cycle contains a customer i' such that $i \notin N_{i'}$. Similar to, e.g., Baldacci et al. (2011) and Ribeiro, Desaulniers, and Desrosiers (2012), we construct neighborhoods of a fixed size Δ_{ng} for each customer $i \in N'$. They contain the closest customers with respect to travel costs, including i itself. This way, any cycle in an *ng*-path will be relatively long or expensive.

In our branch-price-and-cut algorithm we use the *ng*-route relaxation which means that not all routes might be elementary. We adjust the labeling algorithm for the elementary shortest path problem with resource constraints to solve a shortest *ng*-path problem with resource constraints by modifying the extension functions for the customer resources. When extending label l from a node (i, w) to (i', w') to create label l', customer resource $f_j(l')$ is set to zero if $j \notin N_{i'}$ even when $f_j(l) = 1$. Hence, expression (16) is replaced by:

$$f_j(l') = \begin{cases} 1 & \text{if } j = i' \\ \max\{f_j(l), U^s_j(l')\} & \text{if } j \in N_{i'} \setminus \{i'\} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in N'. \tag{17}$$

In this case, label l' is declared feasible if $t(l') \leq \overline{w'}$ and $q(l') \leq Q$. Here the latter condition must be checked because it can be violated even if $f_{i'}(l) = 0$.

During the label dominance check at a node $(i, w) \in \hat{N}'$, only the customer resources for $j \in N_i$ need to be considered, that is, the dominance rule involves only the conditions $f_j(l) \leq f_j(l')$, $\forall j \in N_i$, for the customer resources. This is sufficient because $f_j(l) = f_j(l') = 0$, $\forall j \in N' \setminus N_i$. Using *ng*-paths typically increases the number of dominated labels and, thus, speeds up the labeling algorithm. Low values of Δ_{ng} yield a fast labeling algorithm at the expense of a decreased LP bound, whereas high values slow down the labeling algorithm but increase the value of the LP bound. Observe that all cycles are allowed in an *ng*-path when $\Delta_{ng} = 1$, and only elementary paths are allowed when $\Delta_{ng} = n$.

3.3. Acceleration strategies

It is well known that, in a column generation algorithm, there is no need to solve the pricing problems to optimality at each iteration. As long as negative reduced cost columns are found, the pricing problems can be solved heuristically and it is even possible to skip some pricing problems. The algorithm remains exact if the pricing problems are solved to optimality in the last column generation iteration when solving a linear relaxation. Below, we discuss two strategies to potentially generate negative reduced cost columns in fast computation times.

Algorithm 1 Column generation algorithm, reusing routes.

```
Initialize R(s) for all s ∈ S
repeat
    Solve the RMP using the routes in R(s) for scenario s ∈ S
    Set S̃ := S
    while S̃ ≠ ∅ do
        Choose s ∈ S̃ and remove it from S̃
        Solve the pricing problem for scenario s to find a set R of
        routes with negative reduced costs
        Add all routes in R to R(s)
        for all s̃ ∈ S̃ do
            Let R̃ ⊆ R be the subset of routes that are feasible and have
            a negative reduced cost for scenario s̃
            if R̃ ≠ ∅ then
                Add the routes in R̃ to R(s̃)
                Remove s̃ from S̃
until No new routes are added to the RMP
```

3.3.1. Reusing routes

At each iteration of the column generation algorithm, a pricing problem is solved for each scenario. Because these pricing problems are very similar, solutions to the pricing problem of one scenario might also be feasible for another. Reusing a solution in this way potentially decreases the number of pricing problems that have to be solved at each iteration. Therefore, we propose the column generation algorithm described in Algorithm 1, in which routes are reused for other scenarios when they are feasible and have a negative reduced cost.

Note that the order in which the scenarios are solved at each iteration might affect the performance of the algorithm. However, in preliminary experiments, we tested different scenario selection strategies and none seems to outperform the others. The computational results presented in Section 5 were obtained by using a fixed order of the scenarios over all iterations. In our experiments this corresponds to the order in which the scenarios are generated as described in Section 5.1. This effectively means that the lowest demand scenario is selected first, followed by the second lowest, etc.

3.3.2. Tabu search column generator

The column generation algorithm can be further accelerated by using a heuristic to solve the pricing problem. A heuristic might be able to identify feasible routes with negative reduced costs in less time than it takes to solve the pricing problem exactly. When using a heuristic at each iteration of the column generation algorithm, the exact algorithm is only used to find new routes or prove optimality when the heuristic fails.

As done by, e.g., Desaulniers, Lessard, and Hadjar (2008), we developed a tabu search algorithm to solve the pricing problem. In this algorithm an initial route is considered, which is iteratively replaced by a neighboring route. The neighborhood of each route contains all feasible elementary routes that can be obtained by performing one move. We consider two types of moves: adding a single node at any position in the route and removing a single node from the route.

At each iteration, the best neighbor in terms of reduced cost is selected as the new route. Note that this might yield a route with a larger reduced cost than that of the previous route. To avoid cycling, selecting the inverse of the move used to obtain the current route is tabu for TS_{tabu} iterations. If the reduced cost of the new route is negative, it is added to the RMP. To diversify the search, at every TS_{It} iterations, it is restarted using a completely new route. The initial route and those used to restart the search correspond to the routes selected in the current solution of the RMP for the scenario associated with the pricing problem. When such a route is not elementary, the first visit to each customer is maintained and all other visits to the same customer are removed from the route. The algorithm stops when all selected routes have been used to restart, or a total of TS^{max} new routes have been added to the RMP during the current search.

3.4. Valid inequalities

For the vehicle routing problem, many valid inequalities have been studied: for example, capacity, comb, hypotour and multistar inequalities (Lysgaard, Letchford, & Eglese, 2004), k-path inequalities (Kohl, Desrosiers, Madsen, Solomon, & Soumis, 1999) and subset row inequalities (Jepsen et al., 2008). These inequalities are also applicable to each scenario in the DTWAVRP.

We have tested all the above mentioned valid inequalities in our algorithm. However, preliminary experiments showed that adding capacity inequalities and subset row inequalities provide the lowest computation time. Below, we describe these inequalities in more detail.

Let δ_{Vr} be the number of times route r traverses an arc (i, j) such that $i \in V$ and $j \notin V$ for $V \subseteq N'$. Let $b(V)$ be the minimum number of vehicles needed to visit all customers in $V \subseteq N'$. The capacity inequalities are as follows:

$$\sum_{r \in R(s)} \delta_{Vr} x_r^s \geq b(V) \quad \forall V \subseteq N',\ \forall s \in S \tag{18}$$

As it is common, we replace $b(V)$ by the lower bound $\lceil \frac{\sum_{i \in V} d_i^s}{Q} \rceil$. The separation problem of these rounded capacity inequalities is strongly NP-hard. We use the heuristic of Lysgaard et al. (2004) to separate them, more precisely, we use the implementation that can be found in the package by Lysgaard (2003).

When capacity inequalities are added to the master problem, the pricing problems remain the same. However, the reduced cost of a route may be altered. Let μ_V^s be the dual variable associated with the capacity inequality for subset V in scenario s. We modify the pricing problem for scenario s by subtracting μ_V^s from the reduced cost of the arcs $((i, w), (i', w')) \in \hat{A}$ such that $i \in V$ and $i' \notin V$.

The subset row inequalities are a special case of the Chvátal-Gomory rank 1 cuts. They were introduced by Jepsen et al. (2008). Let $e_{ir} = \sum_{w \in W_i} e_{iwr}$ be the number of times i is visited on route r. The subset row inequalities can be expressed as follows:

$$\sum_{r \in R(s)} \left\lfloor \frac{1}{k} \sum_{i \in V} e_{ir} \right\rfloor x_r^s \leq \left\lfloor \frac{|V|}{k} \right\rfloor \quad \forall V \subseteq N',\ 2 \leq k \leq |V|,\ \forall s \in S. \tag{19}$$

The subset row separation problem is NP-complete. As suggested by e.g. Jepsen et al. (2008) and Desaulniers et al. (2008), we separate only subset row inequalities for subsets V of size three, using $k = 2$, by enumeration. In this case, the inequalities ensure that for any set of three customers, at most one route can be selected that includes more than one of these customers.

Adding subset row inequalities to the formulation for scenario s changes the corresponding pricing problem. Let σ_V^s be the dual variable associated with the subset row inequality for subset V in scenario s. For every k customers in V visited by a path in the pricing problem, σ_V^s is subtracted from the reduced cost of that path.

The labeling algorithm is adjusted to include the dual values of the subset row inequalities in the reduced cost of each path. For the pricing problem associated with scenario s, the labels are modified by incorporating a new resource h_V^s for every generated subset row inequality associated with a subset V and scenario s. When extending a label to a customer in V, h_V^s is increased by one. When this resource reaches k, then σ_V^s is subtracted from the reduced cost and the resource is reset to 0. Hence, $h_V^s(l)$ gives the number of visits to the nodes in V by the partial path corresponding to label l since the last time σ_V^s was subtracted from the reduced cost.

As proposed by Jepsen et al. (2008), the dominance check is modified as follows. When trying to establish whether a label l dominates

a label l', instead of checking whether $c(l) \leq c(l')$, we check whether $c(l) - \sum_{V:h_V^s(l)>h_V^s(l')} \sigma_V^s \leq c(l')$. Note that the subset row resources $h_V^s(l)$ and $h_V^s(l')$ are not compared during the dominance check. This way, more labels might be dominated.

As adding subset row inequalities slows down the labeling algorithm, we limit the number of inequalities added simultaneously as proposed by Desaulniers et al. (2008). In each iteration only a maximum number of SR_i^{max} subset row inequalities might be added for subsets that include customer i. Furthermore, we limit the number of subset row inequalities added at once by SR_{It}^{max}. Finally, we limit the total number of added subset row inequalities to SR^{max}. To ensure that the limited number of subset row inequalities are likely to make an impact on the LP bound, we only add subset row inequalities that are violated by at least SR_{Vio}^{min}.

3.5. Branch-price-and-cut

We propose the following branch-price-and-cut algorithm to solve the DTWAVRP to optimality. Lower bounds are found by solving the LP relaxation using column generation (see Algorithm 1) and adding valid inequalities. Capacity inequalities are separated in each iteration of the column generation algorithm where no new routes with negative reduced costs are identified. Because of their negative impact on the computation time of the algorithm that solves the pricing problem, subset row inequalities are only generated when no violated capacity constraint can be found. Branching is performed on the arcs in G, as by Proposition 1, integer arc flow in G is sufficient to identify an integer optimal solution.

We perform special ordered subset branching on the arcs (SOS branching). More specifically, for scenario s and customer i, let $\delta_s^-(i)$ and $\delta_s^+(i)$ be the sets of in and out arcs of a node representing customer i, respectively. Next, a customer i', a scenario s' and an arc type $o' \in \{-,+\}$ is selected such that the number of arcs a in $\delta_{s'}^{o'}(i')$ for which the arc flow is greater than zero, $F_a^{s'} > 0$, is the largest set. Let $\delta_{s'}^{o'}(i') = \{a_1, \ldots, a_k\}$ be ordered with respect to the arc flow in G, such that $F_{a_u}^{s'} \geq F_{a_v}^{s'}$ if $u < v$. The arcs are divided into two groups, V and its complement $\bar{V}$, where $V = \{a_1, \ldots, a_u\}$ is such that $\sum_{a\in V} F_a^{s'} \geq 0.5$ and $\sum_{a\in V\setminus\{a_u\}} F_a^{s'} < 0.5$. In one branch we disallow the use of the arcs in V and in the other we disallow the use of the arcs in $\bar{V}$. This does not alter the nature of the pricing problem, in fact the number of arcs in the network decreases.

In our branch-price-and-cut algorithm, upper bounds are obtained when computing an integer RMP solution. The search tree is explored using a best-first strategy, that is, the node with the lowest lower bound is selected to process next.

4. Column generation heuristics

In this section, we introduce five column generation heuristics for the DTWAVRP that are derived from the above branch-price-and-cut algorithm and that can be used to find good solutions in less computation time than the exact algorithm. According to the terminology used in the survey of Joncour, Michel, Sadykov, Sverdlov, and Vanderbeck (2010), we propose one *restricted master* heuristic, two *diving* heuristics, and two *rounding* heuristics. Our goal here is to show how the branch-price-and-cut algorithm can be adapted to tackle larger instances. Other types of heuristics could be devised but such an endeavor does not fall within the scope of this paper.

The proposed heuristics rely on the formulation (7)–(11). These heuristics focus on the first-stage decisions, which means that they do not consider integrality requirements on the route variables but only on the time window assignment variables. This speeds up the solution process, while the fractional route variables may still guide the search well enough to find good time window assignments for the DTWAVRP. Given a time window assignment, the corresponding solution value of the DTWAVRP can be evaluated by solving a VRPTW for each scenario using these time windows. Note that because we consider an unlimited amount of vehicles, a feasible routing schedule can be found for any time window assignment.

4.1. Restricted master heuristic

The restricted master heuristic is based on a MIP solver used as a black-box after generating routes at the root node. First, the LP relaxation of the DTWAVRP is solved using the column generation algorithm described in Section 3. Valid inequalities are then separated and, if violated ones are found, they are added to the RMP which is then reoptimized by column generation. This process is iterated until no violated inequalities are found. Next, the resulting RMP (in the last column generation iteration) is used to find an integer time window assignment. Hence, no new routes are generated beyond this stage. To solve the RMP, a MIP solver is invoked but only integrality of the time window assignment variables is required, that is, we do not require the selection of integer routes.

4.2. Diving heuristics

In the first diving heuristic, called the TWDiving heuristic, the LP relaxation of the DTWAVRP is solved using the exact column generation algorithm, alternating as before with the generation of valid inequalities. Next, we fix the assignment of the highest fractionally selected time window among all customers, i.e., the time window variable that has a fractional value closest to 1 is fixed to 1. We iterate these two steps of column generation and variable fixing until an integer time window assignment is found. Here again, we do not require the route selection to be integer.

At each iteration of the TWDiving heuristic, the column generation algorithm is used to solve to optimality the LP relaxation of the DTWAVRP with a partially fixed time window assignment. Even though the optimal solution to the LP relaxation at each iteration might provide a good indication of the time windows to assign, there is no guarantee that this gives a better indication than a suboptimal solution. Therefore, we also consider a second diving heuristic in which, at each column generation iteration, the pricing problem is only solved using the tabu search algorithm described in Section 3.3.2. In this case, the exact pricing algorithm is never invoked during the execution of the heuristic. This speeds up the heuristic, while the solution quality may not suffer too much as most routes that are important for determining a good time window assignment might be generated by the tabu search algorithm anyway. We refer to this heuristic as the TWDiving-Tabu heuristic.

4.3. Rounding heuristics

The TWRounding heuristic is a rounding heuristic in which we round time window assignment variables that are fractionally selected. First, the LP relaxation of the DTWAVRP is solved using the column generation algorithm (possibly adding valid inequalities). Next, to each customer we assign the highest fractionally selected time window, i.e., for each customer separately, the time window variable that has a fractional value closest to 1 is set to 1. This way an integer time window assignment is obtained, but no integer route selection.

As for the TWDiving heuristic, we also consider a variant of the TWRounding heuristic in which the pricing problem is only solved using the tabu search algorithm. We refer to this heuristic as the TWRounding-Tabu heuristic.

5. Computational results

In this section we present the results of our computational experiments. First, we elaborate on the instances that were used. Next, we

report the results obtained with the exact branch-price-and-cut algorithm (Subsection 5.2) and with the five column generation heuristics (Subsection 5.3). Finally, we provide results to compare a multiple-scenario approach with the approach currently used in practice that considers a single scenario (Subsection 5.4).

All our tests were performed on an Intel(R) Core(TM) i5-2450M CPU 2.5 gigahertz processor. The algorithms were coded in C++ and the IBM ILOG Cplex optimizer, version 12.4, was used to solve the RMP in the column generation algorithm and also to derive an integer solution in the restricted master heuristic. Unless stated otherwise, a time limit of one hour was enforced to solve each instance.

5.1. Test instances

The instances used for our experiments were randomly generated.[1] For each instance, n customers are generated using a uniform distribution over a square with sides of length 5. The depot is located in the center of the square. Travel costs and times are computed as the Euclidean distance between two locations rounded to two digits. Vehicle capacity is 30. The depot has time window [6,22]. We construct three sets of candidate time windows which we randomly assign to each customer, such that each set of candidate time windows is assigned with fixed frequency. We assign the set $\{[10, 12], [12, 14], [14, 16]\}$ to 10 percent of the customers, the set $\{[8, 10], [10, 12], [12, 14], [14, 16], [16, 18]\}$ to 60 percent of the customers, and $\{[7, 9], [9, 11], [11, 13], [13, 15], [15, 17], [17, 19], [19, 21]\}$ to 30 percent of the customers.

For each instance, we generate three demand scenarios, each occurring with equal probability. The scenarios are generated such that the first scenario has low demand, the second scenario has medium demand, and the final scenario has high demand. We accomplish this by randomly generating a basic demand $\overline{d}_i$ for all $i \in N'$ according to a normal distribution with expectation 5 and variance 1.5. Next we generate multipliers u_i^1, u_v^2 and u_v^3 for all $i \in N'$ uniformly distributed on $[0.7, 0.8]$, $[0.95, 1.05]$ and $[1.2, 1.3]$, respectively. Finally, we generate the demand for each customer $i \in N'$ and each scenario $s \in \{1, 2, 3\}$ by computing $d_i^s = \lceil u_i^s \overline{d}_i \rceil$. Generating scenarios in this way resembles demand behavior that is encountered in the case, for instance, of ice cream vendors. When the weather is exceptionally good or bad, demand for ice cream goes up or down respectively. Moreover, all vendors in the network are affected similarly by the weather, leading to an increase or decrease of demand for all vendors simultaneously.

These settings are inspired by experience with a Dutch retail chain. The time window and capacity constraints ensure that no more than roughly seven or eight customers can be visited by a single vehicle in any scenario. We have generated 10 instances for eight different sizes, namely, 10, 15, 20, 25, 30, 40, 50 and 60 customers, making a total of 80 instances.

5.2. Results with the exact algorithm

This section is divided in two parts. We present results for the column generation algorithm first, before reporting results for the full branch-price-and-cut algorithm. For this section, our tests were restricted to instances involving 25 customers or less.

5.2.1. Column generation experiment results

In this section we present the results obtained by the column generation algorithm when solving the LP relaxation of (7)–(11). As initial routes in the RMP, we use routes visiting a single node, i.e., routes of the form $(0, w_0) - (i, w) - (n + 1, w_{n+1})$ for all $(i, w) \in \hat{N}$ and for all scenarios $s \in S$. We will distinguish between using only the exact algorithm to generate routes, and using the tabu search heuristic to generate routes as well. No valid inequalities are added during these experiments.

Table 1 shows the results of using Algorithm 1, without the tabu search column generator, for the cases when all cycles are allowed, when only ng-paths are allowed for a neighborhood size of $\Delta_{ng} = 5$, and when only elementary paths are allowed. Recall that the same implementation of this algorithm can be used for these route relaxations by setting $\Delta_{ng} = 1$, $\Delta_{ng} = 5$, and $\Delta_{ng} = n$, respectively. The first and second columns of Table 1 show the instance number and the number of customers in this instance. For each instance and each type of pricing problem, we report the total time in seconds needed by the column generation algorithm to solve the LP relaxation (T.Time), the time spent on solving pricing problems (P.Time), the number of column generation iterations needed (Iter.), and the LP optimal value (LP).

Observe that almost all of the computation time is spent on solving the pricing problems. For four of the instances with 25 customers, the time limit is exceeded before solving the LP relaxation, when using only elementary paths. When comparing the use of elementary paths versus allowing all cycles, we observe that the computation times are in general significantly faster when all cycles are allowed (instance 1 is an obvious exception) but the LP values are significantly lower. When using ng-paths with $\Delta_{ng} = 5$, the LP values are very close to those obtained when using only elementary paths. Moreover, for the largest instances, the computation times are significantly lower. Hence, using ng-paths provides bounds that are comparable to those obtained when using only elementary paths, in much less time.

Table 2 shows the results of using the column generation algorithm in which the tabu search algorithm is used to find routes with negative reduced costs. We use the settings $\text{TS}_{\text{tabu}} = 5$, $\text{TS}_{\text{lt}} = 15$ and $\text{TS}^{\max} = 150$.

When comparing the results in Tables 1 and 2, one can observe a significant decrease in computation time when using the tabu search heuristic in the elementary route case. In this case, all instances are now solved within the time limit of one hour. When the ng-route relaxation is used, a smaller decrease in computation time is observed. When all cycles are allowed, using the tabu search algorithm leads to an increase in computation time in many instances. Recall that the tabu search heuristic generates only elementary routes. Therefore the routes produced by this heuristic may be less useful when cycles are allowed. Note that we also developed a similar tabu search algorithm for generating ng-routes. It was not successful because checking whether a route is an ng-route is computationally expensive.

All results presented in the next section were obtained using the tabu search column generator as well as the ng-route relaxation. Moreover, preliminary experiments with various values of Δ_{ng} showed that the algorithm yields its best results with $\Delta_{ng} = 5$.

5.2.2. Branch-price-and-cut experiment results

Next, we present the results of our experiments using the exact branch-price-and-cut algorithm. Table 3 reports the results obtained when only capacity inequalities are considered. The column Opt.Gap shows the percentage difference between the best obtained upper and lower bounds after termination of the algorithm. The column UB gives the best upper bound found. The column LP Gap shows the percentage difference between the value of the LP relaxation, without adding valid inequalities, and the best upper bound found. The column Root Gap specifies the same difference but after adding valid inequalities. The column Nodes indicates the number of nodes processed in the search tree and the column CI gives the number of added capacity inequalities. Note that a dash indicates that no integer solution was obtained within the time limit.

Observe that the total computation time increases rapidly with the number of customers in the instances. Three of the instances with 20 customers and eight with 25 customers could not be solved within 1 hour. For four 10-customer instances, the LP bound is already tight.

[1] All instances can be found online at http://people.few.eur.nl/spliet.

Table 1
Column generation experiment results, without tabu search.

Inst.	$\|N'\|$	All cycles allowed				ng-paths with $\Delta_{ng} = 5$				Only elementary paths			
		T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP
1	10	25.43	25.29	18	9.80	6.86	6.57	28	12.78	7.99	7.71	28	12.79
2	10	3.39	3.28	17	15.19	3.04	2.73	32	16.67	3.67	3.26	30	16.67
3	10	1.06	1.03	16	12.02	1.40	1.23	33	16.53	1.76	1.59	32	16.53
4	10	1.73	1.67	21	14.87	4.31	3.93	37	15.70	6.66	6.30	36	15.83
5	10	1.81	1.64	32	17.61	1.97	1.63	48	19.65	2.31	1.98	45	19.65
6	10	0.86	0.81	21	16.09	1.59	1.33	32	18.06	1.83	1.61	33	18.06
7	10	2.01	1.81	28	11.36	2.15	1.81	30	12.17	4.38	4.20	31	12.17
8	10	1.50	1.31	28	15.16	2.29	1.82	38	17.09	2.32	2.04	43	17.09
9	10	1.92	1.79	31	16.91	1.98	1.54	47	19.78	1.83	1.53	39	19.78
10	10	0.89	0.81	20	14.92	1.40	1.09	26	17.17	1.47	1.19	33	17.17
11	15	8.27	7.92	29	20.77	12.61	12.03	33	22.22	44.99	44.45	33	22.23
12	15	4.54	4.35	27	22.12	8.86	8.37	33	24.86	14.27	13.82	34	24.86
13	15	9.24	9.00	22	18.41	12.67	12.03	26	21.36	28.39	27.86	28	21.41
14	15	38.55	38.08	26	15.31	42.85	41.43	40	18.08	134.41	132.99	39	18.08
15	15	8.72	8.44	24	21.34	16.46	15.84	40	24.15	33.68	33.06	32	24.26
16	15	22.93	22.37	29	16.65	29.53	28.37	38	19.11	103.72	102.27	43	19.11
17	15	16.72	16.04	30	20.49	30.65	28.92	52	21.45	141.79	140.29	49	21.53
18	15	16.44	16.18	21	19.56	21.42	20.25	38	22.49	38.74	37.81	44	22.55
19	15	6.77	6.58	28	20.51	8.94	8.38	33	22.58	19.91	19.25	33	22.65
20	15	10.28	10.01	22	17.00	15.44	14.70	35	18.29	69.42	68.50	30	18.30
21	20	15.66	15.21	29	24.78	36.11	34.63	42	27.46	176.25	174.95	32	27.54
22	20	73.96	72.20	37	23.30	223.78	221.14	35	25.23	2061.23	2058.81	39	25.27
23	20	45.19	44.29	35	22.49	153.04	150.59	42	25.90	1162.78	1160.08	47	25.91
24	20	22.42	21.59	27	29.50	38.41	37.25	35	31.13	166.98	165.80	31	31.16
25	20	32.95	32.14	27	25.54	56.43	54.43	44	27.56	145.28	143.21	40	27.70
26	20	17.49	16.79	34	24.07	51.48	48.92	43	26.95	402.75	399.91	43	26.97
27	20	58.47	57.60	25	24.98	156.16	154.70	29	26.76	1334.75	1333.41	34	26.82
28	20	45.41	44.24	34	25.15	63.34	61.01	42	26.22	1029.01	1026.10	46	26.23
29	20	64.93	63.87	26	26.30	94.58	93.30	28	28.61	367.40	366.21	30	28.74
30	20	35.35	34.41	27	21.71	73.54	71.36	39	23.24	1626.60	1624.22	43	23.25
31	25	73.94	71.87	34	33.65	178.12	175.02	40	35.02	2108.34	2105.72	34	35.14
32	25	90.18	87.86	30	29.48	150.37	145.88	47	31.37	1170.66	1165.34	51	31.56
33	25	169.32	166.78	34	27.97	276.76	273.67	31	30.45	3600.00	3600.00	1	–
34	25	39.55	38.91	28	30.56	94.21	92.20	36	33.18	1584.22	1582.39	35	33.27
35	25	93.62	92.38	33	27.86	193.52	189.53	45	30.03	3600.00	3600.00	1	–
36	25	129.92	127.83	30	29.76	236.79	232.68	48	31.62	2823.07	2819.57	40	31.69
37	25	110.81	108.98	30	24.62	250.16	245.17	51	27.17	3600.00	3600.00	1	–
38	25	60.72	58.81	35	32.16	151.21	146.47	49	34.07	942.12	938.66	43	34.14
39	25	124.61	122.74	32	31.69	239.12	235.84	39	33.46	2476.24	2473.37	37	33.51
40	25	121.01	119.37	30	28.09	269.54	266.53	36	29.66	3600.00	3600.00	1	–

For 14 more instances the gap is completely closed by adding capacity cuts, including the instance with the largest (observed) LP gap.

Table 4 shows the results of using the branch-price-and-cut algorithm while adding both the capacity inequalities and the subset row inequalities. Recall that subset row inequalities are only separated when no violated capacity inequalities are identified. We limit the subset row inequalities that we add as described in Section 3.4. We use the settings $SR_i^{max} = 5$, $SR_{lt}^{max} = 10$, $SR^{max} = 30$ and $SR_{Vio}^{min} = 0.1$. In this table, the column SRI indicates the number of generated subset row inequalities.

Three instances (28, 30 and 40) that were previously unsolved are now solved by adding subset row inequalities. Out of the twenty other instances in which subset row inequalities were added, seven instances were solved faster than without adding them, eight remained unsolved, while the others required more computation time. The LP gap of one additional instance, instance 26, is closed after adding subset row inequalities. Adding subset row inequalities improves the lower bounds that are obtained and yields less nodes in the branching tree. However, the additional time spent on solving the pricing problems as a result of adding these inequalities often outweighs the gains of these improved bounds, especially for the smaller instances.

5.3. Results with the heuristics

Next, we present the results of our computational experiments using the five column generation heuristics. In these heuristics, the column generation algorithm is set as specified in Section 5.2 except for the TWDiving-Tabu and TWRounding-Tabu heuristics that only used the tabu search algorithm to generate columns. As mentioned in Section 4, valid inequalities are separated in each heuristic but preliminary experiments (with no inequalities, only capacity inequalities, only subset row inequalities or both types) showed that it is better to not use the same strategy for all heuristics. Indeed, for the restricted master heuristic, the quality of the solutions is dependent on the number and quality of the generated routes. Adding valid inequalities increases the number of routes that are generated by the column generation algorithm and potentially has a positive effect on the quality of the routes. Therefore, for the restricted master heuristic we separate both capacity inequalities and subset row inequalities (using the same parameter setting described in Section 5.2). For the other four heuristics, the results of the preliminary tests showed that no single setting produced the best results over all instances with respect to solution quality and computation time. Adding more inequalities typically yields better solutions but increases computation time. Furthermore, it was not clear which type of valid inequalities provides the best results. Nevertheless, we chose to only present results of experiments in which we separate only the subset row inequalities during the execution of the TWDiving, TWDiving-Tabu, TWRounding and TWRounding-Tabu heuristics.

Table 5 presents the results of solving DTWAVRP instances using the five proposed column generation heuristics. The column Opt. indicates the optimal solution value for the instances that were solved to

Table 2
Column generation experiment results, with tabu search.

Inst.	$\|N'\|$	All cycles allowed				ng-paths with $\Delta_{ng} = 5$				Only elementary paths			
		T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP
1	10	21.01	19.92	103	9.80	2.87	2.53	47	12.78	1.90	1.62	34	12.79
2	10	8.42	7.41	97	15.19	1.31	0.99	46	16.67	1.59	1.26	50	16.67
3	10	4.34	3.67	86	12.02	1.25	0.81	63	16.53	1.23	0.86	50	16.53
4	10	5.40	4.51	94	14.87	2.75	2.12	79	15.70	2.34	1.84	73	15.83
5	10	4.59	3.58	105	17.61	1.61	1.08	71	19.65	1.53	0.98	68	19.65
6	10	4.57	3.84	90	16.09	0.98	0.70	46	18.06	1.00	0.67	46	18.06
7	10	7.00	6.15	124	11.36	1.23	0.95	67	12.17	1.31	1.05	70	12.17
8	10	7.66	6.53	119	15.16	1.83	1.53	79	17.09	1.48	1.11	75	17.09
9	10	6.12	4.93	120	16.91	1.79	1.42	63	19.78	2.18	1.67	73	19.78
10	10	4.29	3.38	99	14.92	0.95	0.47	53	17.17	1.11	0.53	62	17.17
11	15	20.00	17.55	110	20.77	3.35	2.53	29	22.22	4.07	3.17	31	22.23
12	15	13.21	11.24	86	22.12	3.39	2.56	33	24.86	3.46	2.71	33	24.86
13	15	18.49	16.72	103	18.41	3.39	2.69	37	21.36	3.17	2.59	34	21.41
14	15	45.01	40.18	164	15.31	8.39	5.87	79	18.08	8.02	5.64	77	18.08
15	15	26.60	23.46	116	21.34	7.10	5.53	57	24.15	6.32	4.99	50	24.26
16	15	37.03	32.53	143	16.65	9.94	8.32	67	19.11	9.31	7.38	58	19.11
17	15	30.03	26.42	119	20.49	7.16	5.43	52	21.45	7.69	6.42	44	21.53
18	15	24.37	21.83	103	19.56	8.08	6.49	69	22.49	6.96	6.21	49	22.55
19	15	18.44	16.61	106	20.51	3.92	3.15	41	22.58	3.37	2.75	35	22.65
20	15	21.36	18.28	122	17.00	5.87	4.57	51	18.29	6.79	5.48	54	18.30
21	20	35.90	30.64	124	24.78	8.56	6.49	42	27.46	6.27	4.57	33	27.54
22	20	72.59	65.06	133	23.30	46.22	41.32	100	25.23	104.72	101.42	66	25.27
23	20	84.13	74.98	196	22.49	12.45	9.64	50	25.9	16.23	13.56	47	25.91
24	20	35.55	30.53	116	29.50	12.06	9.48	54	31.13	11.36	9.05	47	31.16
25	20	52.34	45.87	134	25.54	9.75	7.49	46	27.56	10.28	8.25	45	27.70
26	20	48.31	41.46	128	24.07	14.71	11.68	55	26.95	29.75	26.68	64	26.97
27	20	56.07	50.81	113	24.98	29.86	27.34	56	26.76	44.48	42.46	43	26.82
28	20	46.50	41.39	117	25.15	14.62	11.58	61	26.22	10.67	8.10	46	26.23
29	20	60.34	53.14	153	26.30	12.86	10.64	45	28.61	8.36	6.77	31	28.74
30	20	67.89	61.36	142	21.71	14.99	12.28	56	23.24	16.10	13.68	47	23.25
31	25	80.04	71.14	143	33.65	42.71	37.88	78	35.02	47.55	44.38	44	35.14
32	25	133.01	116.34	225	29.48	39.72	32.69	91	31.37	64.93	59.54	69	31.56
33	25	135.53	124.32	155	27.97	47.00	41.11	79	30.45	344.21	339.67	67	30.60
34	25	73.83	66.83	154	30.56	15.55	12.67	44	33.18	20.44	18.07	38	33.27
35	25	129.25	120.55	143	27.86	34.23	29.90	62	30.03	41.50	38.37	41	30.04
36	25	110.64	100.48	162	29.76	26.38	21.71	63	31.62	44.77	41.11	50	31.69
37	25	192.65	176.89	194	24.62	62.90	55.74	91	27.17	163.25	157.09	72	27.22
38	25	81.14	71.37	151	32.16	24.82	20.04	70	34.07	21.90	18.46	50	34.14
39	25	136.41	124.78	164	31.69	56.36	50.76	80	33.46	91.06	87.11	56	33.51
40	25	129.25	119.31	186	28.09	28.77	25.09	61	29.66	121.38	118.08	55	29.71

optimality with the exact algorithm. Furthermore, the ten rightmost columns show the solution value and computation time of solving the instances with each heuristic. Note that the computation time for the heuristics does not include the time to evaluate the solution, that is, the time to solve a VRPTW for each scenario using the time windows provided by the heuristics.

First, observe that the restricted master heuristic terminates before reaching the time limit for the instances with up to 25 customers, but for the larger instances, it only terminates within the time limit for one instance. Moreover, note that there are instances in which the time limit is exceeded, but still a solution value is reported. In these cases, the best integer solution found by the branch-and-bound algorithm after one hour of computation time is used. In the other cases where the time limit is exceeded, no integer time window assignment was found.

Furthermore, the TWDiving heuristic terminates before reaching the time limit for all instances with up to 25 customers. It fails to do so for 2 of the 30 customer instances, 8 of the 40 customer instances, and all of the instances with 50 and 60 customers. The TWRounding heuristic terminates within the time limit for all instances except for 4 of the 60 customer instances. The TWDiving-Tabu and TWRounding-Tabu heuristic terminate well within the time limit for all instances. The TWRounding-Tabu heuristic yields the lowest computation times.

To compare the heuristics with respect to solution quality, we distinguish between small and large instances, that is, the instances with up to 25 customers and the instances with 30 or more customers, respectively. For the 40 small instances, the five heuristics (in order of their presentation) provide the best solution for 27, 20, 19, 8 and 8 instances, respectively. Thus, the restricted master heuristic provides the best solution most often. The cost of its solutions is on average 0.29 percent above the optimal value for all instances that have been solved to optimality. For the other four heuristics, the average optimality gap is 0.51 percent for TWDiving, 0.39 percent for TWDiving-Tabu, 4.30 percent for TWRounding, and 3.17 percent for TWRounding-Tabu. Clearly, the diving heuristics produce solutions of much better quality than the counterpart rounding heuristics. However, they are more time-consuming. It seems that using only the tabu search column generator does not affect solution quality.

For the 40 large instances, the five heuristics (in the same order) provide the best solution for 3, 4, 32, 1 and 0 instances, respectively. In this case, the TWDiving-Tabu heuristic yields the best solution most often. The restricted master heuristic and TWDiving heuristic are unable to produce solutions in many of these instances due to exceeding the time limit. In consequence, the TWDiving-Tabu heuristic seems to be the heuristic offering the best compromise between solution quality and computation time.

5.4. Comparison with current practice

In practice, time windows are typically assigned based on historical average demand. In this case, a VRP with multiple time windows can be solved (by branch-price-and-cut), where the time windows

Table 3
Branch-price-and-cut experiment results, with capacity inequalities only.

Inst.	$\|N'\|$	UB	Tot.Time	Opt. Gap	LP gap	Root gap	Nodes	CI
1	10	12.83	4.85	0	0.4	0	2	1
2	10	16.84	2.18	0	1.04	0	1	6
3	10	16.60	1.29	0	0.47	0	1	3
4	10	15.96	20.48	0	1.61	0.58	23	24
5	10	19.65	1.59	0	0	0	1	0
6	10	18.13	1.54	0	0.38	0	1	6
7	10	12.17	1.23	0	0	0	1	0
8	10	17.09	1.84	0	0	0	1	0
9	10	20.14	3.41	0	1.78	0	4	7
10	10	17.17	0.95	0	0	0	1	0
11	15	23.04	726.36	0	3.56	0.47	345	59
12	15	25.27	6.00	0	1.6	0	1	31
13	15	22.12	16.10	0	3.44	0	2	27
14	15	18.46	78.40	0	2.06	0	10	4
15	15	24.87	701.79	0	2.93	1.01	329	29
16	15	19.82	82.74	0	3.59	0.09	7	16
17	15	21.96	344.62	0	2.36	0.52	69	55
18	15	22.93	66.48	0	1.9	0	19	12
19	15	23.14	33.64	0	2.39	0	8	44
20	15	18.84	35.44	0	2.9	0	7	26
21	20	27.99	32.94	0	1.87	0	2	29
22	20	25.63	1438.67	0	1.59	0.34	125	56
23	20	26.53	3080.33	0	2.39	0.13	216	87
24	20	32.36	894.09	0	3.8	0.36	223	106
25	20	28.84	105.59	0	4.44	0	3	54
26	20	26.99	150.07	0	0.12	0.11	21	4
27	20	27.55	3600.00	0.06	2.87	0.33	233	81
28	20	26.53	3600.00	0.02	1.16	0.41	346	114
29	20	29.49	654.39	0	2.97	0.29	46	112
30	20	23.64	3600.00	0.57	1.69	0.93	307	118
31	25	35.47	1460.91	0	1.26	0.1	44	184
32	25	–	3600.00	–	–	–	346	110
33	25	–	3600.00	–	–	–	163	138
34	25	–	3600.00	–	–	–	504	118
35	25	–	3600.00	–	–	–	97	132
36	25	–	3600.00	–	–	–	140	648
37	25	–	3600.00	–	–	–	98	135
38	25	34.83	204.20	0	2.16	0	5	88
39	25	–	3600.00	–	–	–	110	178
40	25	–	3600.00	–	–	–	152	166

Table 4
Branch-price-and-cut experiment results, with capacity and subset row inequalities.

Inst.	$\|N'\|$	UB	Tot.Time	Opt. Gap	LP gap	Root gap	Nodes	CI	SRI
1	10	12.83	4.88	0	0.4	0	2	1	0
2	10	16.84	2.02	0	1.04	0	1	6	0
3	10	16.60	1.32	0	0.47	0	1	3	0
4	10	15.96	26.31	0	1.61	0.58	16	24	17
5	10	19.65	1.59	0	0	0	1	0	0
6	10	18.13	1.56	0	0.38	0	1	6	0
7	10	12.17	1.23	0	0	0	1	0	0
8	10	17.09	1.82	0	0	0	1	0	0
9	10	20.14	3.44	0	1.78	0	4	7	0
10	10	17.17	0.96	0	0	0	1	0	0
11	15	23.04	507.00	0	3.56	0.27	119	41	30
12	15	25.27	5.91	0	1.6	0	1	31	0
13	15	22.12	15.92	0	3.44	0	2	27	0
14	15	18.46	78.51	0	2.06	0	10	4	0
15	15	24.87	1364.49	0	2.93	1.01	221	28	30
16	15	19.82	70.60	0	3.59	0.08	3	16	10
17	15	21.96	501.76	0	2.36	0.27	17	31	30
18	15	22.93	66.83	0	1.9	0	21	12	5
19	15	23.14	32.93	0	2.39	0	8	44	0
20	15	18.84	35.03	0	2.9	0	7	26	0
21	20	27.99	32.94	0	1.87	0	2	29	0
22	20	25.63	1225.18	0	1.59	0.12	19	43	26
23	20	26.53	3079.46	0	2.39	0.13	216	87	0
24	20	32.36	594.39	0	3.8	0.21	86	105	25
25	20	28.84	105.65	0	4.44	0	3	54	0
26	20	26.99	138.12	0	0.12	0	3	1	10
27	20	–	3600.00	–	–	–	7	75	27
28	20	26.53	3284.31	0	1.16	0.2	91	61	30
29	20	29.49	343.38	0	2.97	0.07	11	84	20
30	20	23.55	2425.43	0	1.33	0.33	147	96	30
31	25	35.47	820.01	0	1.26	0.01	7	162	17
32	25	–	3600.00	–	–	–	157	98	30
33	25	–	3600.00	–	–	–	30	118	30
34	25	–	3600.00	–	–	–	174	78	30
35	25	–	3600.00	–	–	–	46	113	24
36	25	–	3600.00	–	–	–	68	110	30
37	25	–	3600.00	–	–	–	25	101	30
38	25	34.83	214.90	0	2.16	0	5	88	2
39	25	–	3600.00	–	–	–	81	172	27
40	25	30.73	489.06	0	3.49	0	4	101	10

are the candidate time windows of each customer. The time window in which a customer is visited in the computed solution is then declared as the assigned time window. We refer to this approach as the average demand based time window assignment, ADTWA.

In this section, we present the results of computational experiments in which we compare the 1-scenario ADTWA approach with several sample average approximation approaches. To perform this comparison, we generated 30 new instances involving 20, 25, and 30 customers (10 instances for each size). We considered 25 different demand scenarios that were created using the following procedure. Basic demand $\bar{d}_i$, for all $i \in N'$, is generated as described in Section 5.1. For every customer $i \in N'$ and scenario $s \in S$, a perturbation factor ϵ_i^s to the basic demand is generated using a uniform distribution on $[-1.5, 1.5]$. Next, for each scenario $s \in S$, we generate a multiplier u^s uniformly distributed on $[0.625, 1.375]$. Finally, demand for each customer $i \in N'$ and each scenario $s \in S$ is computed as $d_i^s = \left\lceil u^s(\bar{d}_i + \epsilon_i^s) \right\rceil$.

As sample average approximation approaches, the TWDiving-Tabu heuristic is used, first considering a sample of three scenarios and secondly a sample of five scenarios. Furthermore, the exact branch-price-and-cut algorithm is used considering a sample of three scenarios, for the 20-customer instances only. The sample of three scenarios is generated as described in Section 5.1. The sample of five scenarios is generated in the same way except that five multipliers u_i^s per customer $i \in N'$ are generated using uniform distributions on $[0.65, 0.75]$, $[0.8, 0.9]$, $[0.95, 1.05]$, $[1.1, 1.2]$ and $[1.25, 1.35]$, respectively. Note that, for a given instance, the same basic demand $\bar{d}_i$, $i \in N'$, is used in each scenario. In particular, it is considered as the average demand in the ADTWA approach.

No time limit was imposed for these experiments. To compute the expected total transportation cost resulting from the assigned time windows obtained with each method, we solved for each of the 25 scenarios the corresponding VRPTW to optimality. The average optimal value over these scenarios gives the expected total transportation cost. We would like to emphasize that this means that the exact approach for the approximate 3-scenario DTWAVRP instance might not yield the lowest expected costs when evaluated over 25 scenarios.

Table 6 shows the results of these experiments. For each instance, we report in the column BPC the expected transportation cost (ETC) of the time window assignment obtained by applying the branch-price-and-cut algorithm considering three scenarios. In the first two TWDiving-Tabu columns, we provide the ETC yielded by the solutions produced by the TWDiving-Tabu heuristic with three and five scenarios, respectively. The difference between these ETCs is given in the third TWDiving-Tabu column. In the column ADTWA, the ETC of the ADTWA approach solution is reported. The last column shows the difference between the ETCs of the solutions computed by the ADTWA approach and the TWDiving-Tabu heuristic with five scenarios. For each instance, the best ETC value is highlighted in boldface.

First, observe that the ADTWA approach provides the worst results for all instances. Moreover, the TWDiving-Tabu heuristic with five

Table 5
Heuristic experiment results.

Inst.	$\|N'\|$	Opt.	Restricted master		TWDiving		TWDiving-Tabu		TWRounding		TWRounding-Tabu	
			Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
1	10	12.83	12.83	4.92	12.83	8.83	12.83	0.96	12.87	4.13	12.83	1.03
2	10	16.84	16.84	2.34	16.84	8.75	16.98	2.27	16.85	3.49	16.85	1.20
3	10	16.60	16.60	1.40	16.60	4.25	16.60	0.81	16.60	1.37	16.60	0.81
4	10	15.96	15.96	5.07	15.96	11.10	15.96	0.57	15.99	6.38	15.96	0.56
5	10	19.65	19.65	1.73	19.65	1.63	19.70	0.67	19.65	1.75	19.71	0.61
6	10	18.13	18.13	1.77	18.13	1.51	18.13	0.62	18.13	1.50	18.13	0.61
7	10	12.17	12.17	1.35	12.17	1.29	12.17	0.56	12.17	1.28	12.17	0.55
8	10	17.09	17.09	1.98	17.09	2.05	17.09	0.51	17.09	1.86	17.09	0.50
9	10	20.14	20.14	3.49	20.14	4.36	20.14	0.59	20.14	2.47	20.14	0.56
10	10	17.17	17.17	1.04	17.17	1.04	17.17	0.48	17.17	0.97	17.17	0.45
11	15	23.04	23.06	20.42	23.21	99.56	23.04	13.30	25.28	26.49	24.09	6.97
12	15	25.27	25.27	6.45	25.27	71.19	25.27	8.04	27.06	17.60	26.28	5.15
13	15	22.12	22.23	26.96	22.34	63.36	22.22	11.52	24.70	16.91	22.51	5.54
14	15	18.46	18.46	17.55	18.46	141.38	18.46	7.69	19.85	36.83	18.51	5.87
15	15	24.87	25.24	21.31	25.33	67.31	24.99	13.69	26.16	23.84	25.17	7.30
16	15	19.82	20.09	46.42	19.82	491.29	19.82	17.62	22.38	209.38	20.16	11.26
17	15	21.96	22.05	50.82	22.10	206.31	22.10	13.18	22.05	81.67	24.74	6.46
18	15	22.93	22.93	16.20	23.07	109.91	22.93	8.95	23.16	28.49	23.57	4.37
19	15	23.14	23.17	15.56	23.26	65.62	23.34	12.09	23.83	19.91	23.34	6.61
20	15	18.84	18.95	16.09	19.11	130.55	19.11	17.53	19.86	50.59	19.59	8.18
21	20	27.99	27.99	27.95	28.14	211.34	28.01	22.24	29.84	35.66	29.95	10.73
22	20	25.63	25.63	200.91	25.87	1875.65	25.65	26.41	29.20	349.99	26.07	13.45
23	20	26.53	26.53	59.89	26.68	1770.08	26.64	29.64	28.62	139.21	28.48	13.99
24	20	32.36	32.53	936.09	32.60	301.94	32.99	32.26	32.87	41.64	33.87	12.68
25	20	28.84	28.84	60.65	29.06	216.76	29.08	29.25	33.64	62.15	29.09	9.91
26	20	26.99	27.08	49.79	27.23	533.77	27.23	29.92	28.08	16.58	27.53	9.63
27	20	–	28.02	552.23	27.65	2989.60	27.93	32.66	28.84	496.24	30.06	12.59
28	20	26.53	26.65	108.63	26.58	542.25	26.53	27.88	26.83	79.20	29.41	12.67
29	20	29.49	29.79	315.37	29.85	549.26	29.75	28.72	31.67	100.89	30.81	9.77
30	20	23.55	23.98	184.84	23.59	232.30	23.65	39.32	24.61	35.29	27.60	17.19
31	25	35.47	35.60	849.26	35.47	601.19	35.84	44.99	36.88	93.98	36.76	18.30
32	25	–	32.66	362.23	32.80	530.27	33.17	54.78	36.17	133.68	34.75	25.94
33	25	–	31.91	3071.41	31.74	3279.53	31.82	65.17	32.86	431.08	32.19	26.35
34	25	–	34.20	255.60	34.14	534.75	34.19	45.59	35.02	92.28	34.79	14.21
35	25	–	30.29	266.38	30.29	1138.92	30.29	51.93	31.08	221.22	32.12	26.43
36	25	–	32.68	2062.42	33.00	1368.55	32.55	36.50	34.36	127.69	34.50	19.11
37	25	–	27.49	635.48	27.81	767.11	27.75	68.48	28.89	207.45	27.66	28.60
38	25	34.83	34.84	210.75	35.39	746.02	34.93	50.67	37.45	99.64	36.85	18.33
39	25	–	34.41	1108.48	34.39	807.19	34.67	56.16	35.87	156.53	36.32	23.34
40	25	30.73	30.76	231.39	31.30	656.45	30.98	69.26	33.19	133.76	31.27	24.32
41	30	–	36.57	3600.00	36.57	641.78	36.54	85.75	37.72	167.14	37.47	29.69
42	30	–	41.05	3600.00	41.13	869.45	41.11	88.22	43.77	145.94	42.26	33.92
43	30	–	37.59	3600.00	37.54	2013.42	37.48	100.49	39.86	176.64	37.82	26.21
44	30	–	38.02	572.11	38.09	647.73	38.40	96.47	39.68	101.68	38.33	31.68
45	30	–	37.58	3600.00	36.96	1003.62	37.08	96.00	38.69	127.13	38.48	25.37
46	30	–	35.25	3600.00	–	3600.00	35.01	105.84	35.82	587.70	35.62	37.46
47	30	–	42.49	3600.00	42.68	425.35	42.59	61.90	43.69	92.93	44.54	25.93
48	30	–	37.31	3600.00	–	3600.00	37.09	95.03	39.70	484.51	37.65	35.19
49	30	–	41.15	3600.00	40.99	1820.52	40.86	75.30	42.48	221.60	43.52	22.12
50	30	–	40.32	3600.00	39.90	1285.75	40.05	98.92	42.44	180.35	41.88	28.07
51	40	–	41.95	3600.00	41.54	3078.58	41.82	249.52	43.26	592.77	42.93	69.23
52	40	–	54.82	3600.00	–	3600.00	47.99	234.57	50.02	943.37	49.67	65.26
53	40	–	46.51	3600.00	–	3600.00	41.80	273.46	42.95	422.37	43.61	70.70
54	40	–	46.33	3600.00	–	3600.00	45.96	247.84	47.79	752.28	47.18	64.43
55	40	–	–	3600.00	–	3600.00	48.39	207.66	49.01	531.84	51.19	57.68
56	40	–	–	3600.00	–	3600.00	44.65	295.71	46.12	2706.25	45.55	88.90
57	40	–	–	3600.00	–	3600.00	44.28	230.39	45.69	896.51	46.00	57.80
58	40	–	43.28	3600.00	43.21	3234.82	43.25	193.02	43.77	584.46	43.64	59.48
59	40	–	49.18	3600.00	–	3600.00	48.76	195.76	49.69	634.14	51.77	67.10
60	40	–	47.76	3600.00	–	3600.00	47.62	200.10	47.58	395.48	50.34	46.93
61	50	–	–	3600.00	–	3600.00	52.18	439.03	53.28	1210.39	53.61	87.25
62	50	–	–	3600.00	–	3600.00	55.84	402.90	58.01	1026.38	58.07	96.25
63	50	–	–	3600.00	–	3600.00	50.38	517.14	51.79	1065.61	51.50	127.64
64	50	–	–	3600.00	–	3600.00	51.25	534.22	55.62	1823.49	53.72	121.79
65	50	–	–	3600.00	–	3600.00	54.40	446.72	56.37	1062.77	55.55	108.89
66	50	–	–	3600.00	–	3600.00	56.95	514.13	60.03	2003.95	59.05	99.68
67	50	–	58.29	3600.00	–	3600.00	57.67	404.41	59.27	679.43	58.55	99.97
68	50	–	–	3600.00	–	3600.00	55.79	510.41	58.89	1302.23	56.74	147.42
69	50	–	–	3600.00	–	3600.00	53.64	374.07	55.40	1478.90	56.65	104.94
70	50	–	–	3600.00	–	3600.00	56.80	432.60	57.71	1110.55	58.04	95.30

(*continued on next page*)

Table 5
(*continued*)

Inst.	$\|N'\|$	Opt.	Restricted master		TWDiving		TWDiving-Tabu		TWRounding		TWRounding-Tabu	
			Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
71	60	–	–	3600.00	–	3600.00	63.85	834.21	–	3600.00	65.93	168.69
72	60	–	–	3600.00	–	3600.00	62.18	765.56	–	3600.00	64.48	194.36
73	60	–	–	3600.00	–	3600.00	64.79	714.06	–	3600.00	66.12	161.60
74	60	–	–	3600.00	–	3600.00	68.84	703.17	70.91	3115.44	70.78	131.63
75	60	–	–	3600.00	–	3600.00	63.65	707.01	64.49	2115.73	65.25	138.44
76	60	–	–	3600.00	–	3600.00	64.45	854.16	–	3600.00	66.34	256.01
77	60	–	–	3600.00	–	3600.00	61.35	814.93	63.91	2765.95	63.37	223.91
78	60	–	–	3600.00	–	3600.00	64.02	772.99	66.03	2531.43	66.09	134.10
79	60	–	–	3600.00	–	3600.00	65.40	628.39	69.10	2460.19	66.94	164.99
80	60	–	–	3600.00	–	3600.00	64.20	700.72	66.23	1827.12	66.14	156.30

Table 6
Time window assignments evaluated using 25 scenarios.

Inst.	$\|N'\|$	BPC	TWDiving-Tabu			ADTWA	
		Three scenarios ETC	Three scenarios ETC	Five scenarios ETC	Three vs. five scenarios Diff. percent)	One scenario ETC	One vs. five scenarios Diff. percent)
81	20	**30.12**	**30.12**	30.17	−0.17	30.86	2.24
82	20	26.59	**26.57**	26.58	−0.04	27.94	4.87
83	20	26.40	26.48	**26.27**	0.79	27.84	5.64
84	20	28.44	28.58	**28.30**	0.98	28.98	2.35
85	20	28.07	27.74	**27.57**	0.61	29.17	5.49
86	20	33.07	33.27	**32.91**	1.08	34.21	3.80
87	20	31.38	31.38	**31.30**	0.25	32.43	3.48
88	20	27.06	**26.93**	26.97	−0.15	27.34	1.35
89	20	**28.95**	29.06	28.99	0.24	29.68	2.32
90	20	26.78	26.75	**26.57**	0.67	27.12	2.03
91	25		32.64	**32.21**	1.32	34.48	6.58
92	25		**33.89**	33.91	−0.06	35.79	5.25
93	25		37.40	**37.34**	0.16	37.62	0.74
94	25		32.05	**31.99**	0.19	33.23	3.73
95	25		30.88	**30.83**	0.16	31.78	2.99
96	25		34.11	**34.10**	0.03	35.03	2.65
97	25		**33.13**	33.19	−0.18	34.50	3.80
98	25		30.02	**29.88**	0.47	30.96	3.49
99	25		34.58	**34.57**	0.03	35.90	3.70
100	25		**30.30**	30.51	−0.69	32.58	6.35
101	30		**37.02**	37.04	−0.05	39.11	5.29
102	30		32.71	**32.53**	0.55	33.41	2.63
103	30		**35.93**	**35.93**	0.00	37.29	3.65
104	30		**36.48**	**36.48**	0.00	37.98	3.95
105	30		39.77	**39.67**	0.25	41.09	3.46
106	30		40.00	**39.86**	0.35	41.68	4.37
107	30		36.78	**36.30**	1.31	38.31	5.25
108	30		37.82	**37.62**	0.53	38.83	3.12
109	30		32.79	**32.78**	0.03	33.52	2.21
110	30		36.74	**36.53**	0.57	37.48	2.53

scenarios yields the best solution for most instances. In fact, the ETC of the ADTWA solutions are on average 3.64 percent higher than those of the solutions produced by the TWDiving-Tabu heuristic with five scenarios. For all instances except one, this difference is greater than 1.35 percent. Hence, these results suggest that the quality of the time window assignment benefits from considering multiple scenarios.

Secondly, the difference in the ETCs of the solutions computed by the TWDiving-Tabu heuristic with three scenarios and with five scenarios is on average only 0.31 percent in favor of the 5-scenario variant and never exceeds 1.32 percent. From these statistics, we deduce that the benefits of considering additional scenarios when assigning time windows diminish when the number of scenarios increase. This phenomenon is due to the high correlation between the demand scenarios, that is, all demands either increase or decrease at the same time. When no such correlation exists, one would expect that a large number of scenarios would be required to reach the best solutions.

Finally, the quality of the solutions obtained by the exact branch-price-and-cut algorithm and by the TWDiving-Tabu heuristic with 3 scenarios are similar for the 20-customer instances. Indeed, the heuristic yields the best solution for 4 of the 10 instances and an equivalent solution for two instances. The ETC produced by the exact algorithm is on average only 0.01 percent better. These results show that a "good" heuristic solution to an approximate model might be better than an optimal solution to this approximate model.

6. Conclusions

In this paper, we have introduced a new problem, the DTWAVRP. We have developed an exact branch-price-and-cut algorithm to solve it. The column generation algorithm exploits the fact that columns for one scenario can be reused in another scenario. Furthermore, we use an *ng*-route relaxation to speed up the pricing problem while limiting the decrease of the LP value and we also generate columns using a tabu search heuristic. Finally, the branch-price-and-cut algorithm incorporates valid inequalities that are known from vehicle routing, namely, capacity and subset row inequalities. We are able to solve to optimality instances of up to 25 customers and three scenarios within one hour of computation time.

Furthermore, we implemented five column generation heuristics, each incorporating our column generation algorithm and valid inequalities. They are a restricted master heuristic, two diving heuristics and two rounding heuristics. For the small instances (up to 25 customers), the restricted master heuristic produces the best solutions most often. The solution values are on average 0.29 percent above the optimal value for the instances solved to optimality. For the instances with more than 30 customers, the TWDiving-Tabu heuristic (that only uses the tabu search column generator) produces the best solutions most often and offers the best quality/time compromise.

Finally, we performed experiments to compare a multiple-scenario approach with a single-scenario approach that is typically used in practice. Our results show that considering five scenarios instead of one produces significantly better quality solutions, with an average reduction of 3.64 percent on the expected transportation cost.

In the future, various research directions ensuing from this work can be explored. One of them would be to consider customer preferences on the candidate time windows that can be assigned to them and to include in the DTWAVRP a secondary objective consisting of maximizing the customer preference satisfaction. Another line of research would be to enhance the proposed method or develop a new one to tackle instances involving a large number of scenarios.

Appendix A

Below we provide a proof of Proposition 1 that we restate here.

Proposition 1. *Let* (x, y) *be a solution to the LP relaxation of formulation* (7)–(11)*. When the corresponding arc flow in G is integer for every*

scenario, there exists an integer solution (x^*, y^*) *to the DTWAVRP of equal value.*

Proof. For each customer $i \in N'$, let $w(i,y) \in \arg\min\{\underline{w} \mid w \in W_i, y_{iw} > 0\}$ be the candidate time window with the earliest start time among the ones selected in solution (x, y).

Let F^s be the integer arc flow in G for scenario s, corresponding to solution (x, y). This arc flow can be represented as a set of $(0, n+1)-$paths in G, $F^s = \{P_1, \ldots, P_{k(s)}\}$. Furthermore, denote by F^s_a the flow on arc $a \in A$ in scenario s.

For any path $P \in F^s$ visiting the customers $\{i_1, \ldots, i_l\}$, consider the path $\hat{P}$ in $\hat{G}$ visiting the nodes $\{(i_1, w(i_1,y)), \ldots, (i_l, w(i_l,y))\}$, that is, the path using for each customer the selected time window with the earliest start time. Using path $\hat{P}$ for all $P \in F^s$, $s \in S$, and the time windows $w(i,y)$ for each $i \in N'$, yields a solution whose value is equal to that of (x, y). To complete the proof, we need to show that this solution is feasible. Because a path $\hat{P}$ visits the same customers as its parent path P, the capacity constraints are satisfied by the routes in the new solution. Hence, all that remains to be shown is that the time window constraints are also satisfied along those routes.

Consider the graph $\hat{G}(F^s, y) = (\hat{N}(y), \hat{A}(F^s, y))$, where $\hat{N}(y) = \{(i,w) \in \hat{N} \mid y_{iw} > 0\} \bigcup \{(0, w_0), (n+1, w_{n+1})\}$ contains the combinations of locations and time windows that are selected in solution (x, y), and $\hat{A}(F^s, y) = \{((i,w),(i',w')) \in \hat{A} \mid (i,w),(i',w') \in \hat{N}(y), F^s_{(i,i')} > 0\}$. Observe that all paths from $(0, w_0)$ to $(n+1, w_{n+1})$ in $\hat{G}(F^s, y)$ can be represented in $\hat{G}$. Moreover, any such path visits the same customers as some path $P \in F^s$ and in the same order.

Let $\underline{t}^s_{iw}$ be the earliest possible start of service time in node (i,w) by any path in $\hat{G}(F^s, y)$ starting at node $(0, w_0)$. Let $\underline{t}^s_{0w_0} = \underline{w}_0$. Observe that as $y_{iw} > 0$ for $(i,w) \in \hat{N}(y)$, constraints (9) ensure that there is a route $r \in R(s)$ such that $x_r > 0$ for all $s \in S$. Hence, $\underline{t}^s_{iw}$ exists for all $(i,w) \in \hat{N}(y)$ and all $s \in S$.

Let $W_i(y) = \{w | w \in W_i, y_{iw} > 0\}$. Next, let $\underline{t}^s_i = \min_{w \in W_i(y)}\{\underline{t}^s_{iw}\}$ be the earliest start of service time at customer i in $\hat{G}(F^s, y)$. Observe that $\underline{t}^s_0 = \underline{t}^s_{0w_0}$. For every pair (i,i') such that $F^s_{(i,i')} > 0$, it holds that

$$\underline{t}^s_{i'w(i',y)} = \max\{\underline{w(i',y)}, \underline{t}^s_i + t_{ii'}\} \leq \max\{\underline{w}, \underline{t}^s_i + t_{ii'}\}$$
$$= \underline{t}^s_{i'w} \qquad \forall w \in W_{i'}(y).$$

Therefore, $\underline{t}^s_i = \underline{t}^s_{iw(i,y)}$ and it follows that $\underline{t}^s_{i'w(i',y)} \geq \underline{t}^s_{iw(i,y)} + t_{ii'}$. This shows that using path $\hat{P}$ and the earliest start of service time in the time window with the earliest start time $\underline{t}^s_{iw(i,y)}$ for each node $(i, w(i,y))$ visited on this path provides a feasible route for each $P \in F^s$ and each scenario $s \in S$. □

References

Agatz, N., Campbell, A., Fleischmann, M., & Savelsbergh, M. (2011). Time slot management in attended home delivery. *Transportation Science*, *45*(3), 435–449.

Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, *59*(5), 1269–1283.

Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, *218*(1), 1–6.

Birge, J. R., & Louveaux, F. (1997). *Introduction to stochastic programming*. New York, NY: Springer-Verlag.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows. Part I. Route construction and local search algorithms. *Transportation Science*, *39*(1), 104–118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows. Part II. Metaheuristics. *Transportation Science*, *39*(1), 119–139.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, *8*, 101–111.

Desaulniers, G., Lessard, F., & Hadjar, A. (2008). Tabu search, partial elementarity, and generalized *k*-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, *42*(3), 387–404.

Desrochers, M. (1988). An algorithm for the shortest path problem with resource constraints, *(Technical report G-88-27, GERAD)*.

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, *40*(2), 342–354.

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, *44*(3), 216–229.

Ford, L. R., Jr., & Fulkerson, D. R. (1958). A suggested computation for maximal multi-commodity network flows. *Management Science*, *5*(1), 97–101.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, *11*(4), 630–643.

Irnich, S., & Villeneuve, D. (2006). The shortest path problem with resource constraints and *k*-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, *18*(3), 391–406.

Jabali, O., Leus, R., van Woensel, T., & de Kok, A. G. (2013). Self-imposed time windows in vehicle routing, *OR Spectrum* (Published online), 10.1007/s00291-013-0348-1.

Jepsen, M., Petersen, B., Spoorendonk, S., & Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, *56*(2), 497–511.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., & Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, *36*(1), 695–702.

Kallehauge, B., Larsen, J., Madsen, O. B. G., & Solomon, M. M. (2005). Vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, & M. M. Solomon (Eds.), *Column generation* (pp. 67–98). New York: Springer.

Kohl, N., Desrosiers, J., Madsen, O. B. G., Solomon, M. M., & Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, *33*(1), 101–116.

Lysgaard, J. (2003). *Cvrpsep: A package of separation routines for the capacitated vehicle routing problem*. Working paper 03-04. Aarhus, Denmark: Department of Management Science and Logistics, Aarhus School of Business.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming, 100*(2), 423–445.

Ribeiro, G. M., Desaulniers, G., & Desrosiers, J. (2012). A branch-price-and-cut algorithm for the workover rig routing problem. *Computers & Operations Research*, *39*(12), 3305–3315.

Spliet, R., & Gabor, A. F. (2014). The time window assignment vehicle routing problem, *Transportation Science* (Published online). 10.1287/trsc.2013.0510.