The Time Window Assignment Vehicle Routing Problem

Remy Spliet, Adriana F. Gabor

Econometric Institute
Erasumus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

EI2012_07

April 20, 2012

Abstract

In many distribution networks, it is vital that time windows in which deliveries are made are assigned to customers for the long term. However, at the moment of assigning time windows demand is not known. In this paper we introduce the time window assignment vehicle routing problem, the TWAVRP. In this problem time windows have to be assigned before demand is known. Next the realization of demand is revealed and an optimal vehicle routing schedule has to be made that satisfies the time window constraints. We assume that different scenarios of demand realizations are known, as well as their probability distribution. The TWAVRP is the problem of assigning time windows such that the expected traveling costs are minimized. We propose a formulation of the TWAVRP and develop two variants of a column generation algorithm to solve the LP relaxation of this formulation. Numerical experiments show that these algorithms provide us with very tight LP-bounds to instances of moderate size in reasonable computation time. We incorporate the column generation algorithm in a branch and price algorithm and find optimal integer solutions to small instances of the TWAVRP. In our numerical experiments, the branch and price algorithm typically finds the optimal solution very early in the branching procedure and spends most time on proving optimality.

1 Introduction

In many distribution networks deliveries are made at regular intervals. These deliveries are scheduled to occur within a time window. Typically, these time windows are endogenously imposed. The carrier and receiver might for instance agree on a specific time window for delivery. These endogenous time windows are long term decisions in certain industries. In retail it is very common that deliveries at a store are always made within a specific time interval on the same day of the week. This is crucial for many operational processes like inventory management and the scheduling of personnel. Note that distribution networks are often also faced with exogenous time windows. For example, an exogenous time window might be imposed by a local government which forces trucks to make deliveries in a populated area during day time only. Hence an endogenous time window can only be chosen within the exogenous time window.

When demand of the receivers becomes known, a vehicle routing schedule has to be made for making the deliveries within the provided time windows. This problem is known as the VRPTW, which is a well studied problem in the scientific literature, see for instance Desrochers et al. (1992) and Fischer et al. (1997). Many commercial software packages exist that are able to design such a routing schedule in a limited amount of time.

However, demand most often fluctuates per delivery and is unknown at the moment that time windows are assigned to the receivers. In practice these time windows are often determined

heuristically by solving a capacitated vehicle routing problem, CVRP, using average demand. The arrival times in the resulting schedule are used as a reference point for assigning the time windows. An overview of exact and heuristic methods for solving the CVRP can be found in Toth and Vigo (2002) and Laporte (2007) amongst others. The problem of assigning time windows before demand is known has largely been overlooked in the scientific literature.

In this paper a model is presented to assign time windows and design vehicle routes such that the expected costs incurred after vehicle routing are minimized. The problem of assigning time windows will be referred to as the time window assignment vehicle routing problem, TWAVRP. A finite number of scenarios is given, each scenario describing a realization of demand for each location. Furthermore, the probability with which each scenario occurs is assumed to be known at the moment of scheduling. The TWAVRP consists of finding a single time window assignment and a vehicle routing schedule for each scenario satisfying these time windows, such that the expected costs are minimized. The TWAVRP is NP-hard because it contains the VRPTW as an instance with one scenario.

In Jabali et al. (2010) a closely related problem is considered. In their paper demand is assumed to be known at the moment of assigning the time windows and travel time is stochastic. This problem is encountered by small package shipping companies. In another related problem, the consistent vehicle routing problem, ConVRP, examined by Groër et al. (2009), demand and travel time is deterministic and a schedule has to be made for multiple days. Not all receivers have to be visited on every day. For the days on which a location receives a delivery the arrival times should be roughly the same, within a prespecified amount of time apart. Moreover, a location should always receive its delivery from the same driver. In their paper, a heuristic is designed to find solutions to the ConVRP. The problem addressed in this paper, the TWAVRP, can be considered a variant of the ConVRP.

In this paper, we propose a branch and price algorithm to solve the TWAVRP. Branch and price has been successfully applied to solve instances of the classical CVRP and VRPTW, see for instance Chabrier (2006), Desrochers et al. (1992) and Desaulnier (2010). We specifically focus on a formulation of the TWAVRP that provides tight LP-bounds. We design a column generation algorithm to solve the LP relaxation of the TWAVRP. Using our formulation, the pricing problem decomposes into shortest path problems with two resource constraints and linear node costs, per scenario. These problems are solved to optimality using a modified version of the labeling algorithm proposed by Ioachim et al. (1998). In their paper they consider having only one resource constraint, we have extended their algorithm to cope with an additional resource constraint. We use this algorithm to find paths that may contain cycles, although no feasible integer solution to the TWAVRP will make use of such paths. We allow such paths to limit the computational complexity of the pricing problem at the expense of the value of the LP-bound. We have furthermore adapted the algorithm to eliminate 2-cycles, thereby significantly increasing the value of the LP-bound. In the column generation algorithm, we make use of the fact that routes generated by solving the decomposed pricing problem for a certain scenario, yielding a column with negative reduced costs, may also yield a column with negative reduced costs describing the same route for other scenarios.

This paper is organized as follows. A formal definition of the TWAVRP is given in section 2. In section 3 the column generation used to obtain lower bounds and the branching procedure is discussed. This is followed by numerical experiments in section 4, in which the solution approach is evaluated.

2 Problem Definition

Consider a complete graph $G = (V, E)$, where $V = \{0, ..., n+1\}$ is a set of locations such that 0 represents the starting depot, $n+1$ the ending depot and $V' = \{1, ..., n\}$ are the customers. Let $c_{ij} \geq 0$ be the cost to travel along edge (i, j) and let $t_{ij} \geq 0$ be the corresponding travel time.

Both the travel costs and travel time satisfy the triangle inequality. Furthermore, an unlimited amount of vehicles of equal capacity Q is available.

A set Ω of scenarios is given, where each scenario is characterized by a realization of demand. Let demand at location v in scenario $\omega \in \Omega$ be given by d_v^ω such that $0 < d_v^\omega \leq Q$. The probability that scenario ω occurs is p_ω.

For each location v, let s_v and e_v be the start and end time of the time interval during which the location has to be visited. The time window $[s_v, e_v]$ is exogenous and not to be confused with the endogenous time window.

In this paper we will use the term route to refer to a pair (P, t) where P is a path in G starting at 0 and ending at $n+1$ and t is a vector containing arrival times at each location on the path. An acyclic route is a route such that its path does not contain a cycle. To each acyclic route r we associate the following parameters: a_r^v indicates whether location v is visited on r and t_r^v is the time of arrival at location v on r. Let $t_r^v = 0$ whenever a location is not visited. If location j is visited directly after i on the acyclic route r then $t_r^i + t_{ij} \leq t_r^j$. Note that waiting at a location is allowed. Let $R(\omega)$ be the set of all feasible acyclic routes for scenario ω, i.e. all routes such that the capacity constraints and exogenous time window constraints are satisfied. To each acyclic route r with edges $\{r_1, ..., r_k\}$ we assign costs $c_r = \sum_{i=1}^k c_{r_i}$. A routing schedule is a collection of routes such that all customers are visited exactly once.

An endogenous time window will be assigned to each customer within which it will receive its delivery. The assignment will be made before the realization of demand is learned. Prior to the dispatching of the vehicles, demand becomes known and an optimal routing schedule will be designed to make the deliveries within the assigned time windows. The problem is to assign time windows of width w_v to each customer v such that the expected traveling costs are minimal.

Let the variable y_v be the start time of the endogenous time window at each location $v \in V'$. Note that $y_v \in [s_v,\ e_v - w_v]$. We will assume $s_v \leq e_v - w_v$. Let the binary variable x_r^ω indicate whether route r is used for scenario ω. Let $\mathbf{R} = [R(\omega)]_{\omega \in \Omega}$ be the vector containing all feasible acyclic routes for each scenario. The optimal solution to the TWAVRP is given by $OPT(\mathbf{R})$:

$$
\begin{aligned}
OPT(\mathbf{R}) = \min \sum_{\omega \in \Omega} p_\omega \sum_{r \in R(\omega)} c_r x_r^\omega & & (1)\\
\sum_{r \in R(\omega)} a_r^v x_r^\omega = 1 & \qquad \forall v \in V', \forall \omega \in \Omega & (2)\\
\sum_{r \in R(\omega)} t_r^v x_r^\omega \geq y_v & \qquad \forall v \in V', \forall \omega \in \Omega & (3)\\
\sum_{r \in R(\omega)} t_r^v x_r^\omega \leq y_v + w_v & \qquad \forall v \in V', \forall \omega \in \Omega & (4)\\
x^\omega \in \{0,1\}^{|R(\omega)|} & \qquad \forall \omega \in \Omega & \\
y \in [s, e - w] & &
\end{aligned}
$$

Here (1) are the expected total costs of a time window assignment. Constraints (2) ensure that every location is visited exactly once and constraints (3) and (4) ensure that all locations are visited within the time windows. Finally, as time is continuous in this model, observe that the number of routes in $\mathbf{R}$, and therefore also the number of variables, are infinite unless $w = 0$ and $s = e$.

As will become apparent in the next section, in our solution approach we will use an equivalent formulation of the TWAVRP in which we allow cyclic routes. A cyclic route is a route such that its path contains at least one cycle. It is feasible for scenario ω if the consecutive arrival times differ by at least the travel time and the capacity constraints are satisfied, i.e. for each cyclic route r visiting consecutively locations $r_1, ..., r_k$, it holds that $\sum_{i=1}^k d_{r_i}^\omega \leq Q$. For a cyclic route r define a_r^v as the number of times location v is visited and let t_r^v be any real number. In

Proposition 1 we state that including cyclic routes in the problem formulation in this way does not affect the optimal solution.

Let $C(\omega)$ be a collection of feasible cyclic routes and let $\mathbf{C} = [R(\omega) \cup C(\omega)]_{\omega \in \Omega}$, where $\mathbf{C}$ is a vector containing all feasible acyclic routes and a set of feasible cyclic routes for each scenario.

Proposition 1. $OPT(\mathbf{R}) = OPT(\mathbf{C})$.

Proof. Clearly $OPT(\mathbf{R}) \geq OPT(\mathbf{C})$. Observe that there does not exist an integer solution in which a cyclic route is selected as this would violate (2). Therefore equality holds. □

This Proposition shows that the formulations of the TWAVRP are equivalent for any choice of $\mathbf{C}$ and t_r^v. Clearly the value of the LP relaxation decreases when comparing a formulation using routes C and C' for $C \subset C'$.

In the next section we propose column generation algorithms to solve the LP relaxation of the TWAVRP for specific choices of $\mathbf{C}$ and t_r^v and use them in a branch and price algorithm to find the optimal integer solution.

3 Solution Approach

We propose to solve the TWAVRP using a branch and price algorithm. In the branch and price algorithm, lower bounds are obtained by solving the LP relaxation using a column generation algorithm, which is presented in section 3.1. The choice of $\mathbf{C}$ influences the quality of the lower bounds and the computational complexity of obtaining these lower bounds. We will in particular present algorithms for two choices of $\mathbf{C}$, including all cyclic routes visiting at most n locations and including all cyclic routes visiting at most n locations except routes containing $2-$cycles. Furthermore, we will define specific values of t_r^v for cyclic routes which are used in our algorithm. In section 3.2 the branching decisions are explained.

3.1 Solving the LP relaxation

To solve the LP relaxation of the TWAVRP for a formulation $OPT(\mathbf{C})$, a column generation algorithm is used. At each step of the algorithm, a solution is found to the LP relaxation including only a subset of the variables. New variables that exhibit negative reduced costs will be identified and added to the problem. Let us denote the dual variables by λ, μ and ν corresponding to (2)-(4) respectively. For ease of notation, let $\pi = \nu - \mu$. Observe that λ_v^ω and π_v^ω can both be positive and negative. The reduced costs corresponding to a non-basic variable x_r^ω are given by:

$$p_\omega c_r - \sum_{v \in V'} \lambda_v^\omega a_r^v - \sum_{v \in V'} \pi_v^\omega t_r^v$$

To find a non-basic variable with negative reduced costs, a route r has to be found in $R(\omega) \cup C(\omega)$ for some $\omega \in \Omega$ such that the above expression is negative. Note that when for all scenarios $\omega \in \Omega$ no such solution exists, the current solution to the LP relaxation is optimal. Therefore, we define the pricing problem as finding a route and scenario ω which minimize the reduced costs. Solving this problem either yields a variable with negative reduced costs or it proves optimality. The pricing problem can be decoupled into several problems, one for each scenario. Next, the decoupled pricing problem is described in more detail.

Recall that a route is a pair (P, t). Let the path P be represented by its edges, and let $\bar{P}$ be the corresponding vertex representation. Furthermore, recall that for a cyclic route r any value of the parameters t_r^v for location v provides us with a valid formulation of the TWAVRP. For the remainder of the paper, for each cyclic route r let us define t_r^v as the sum of the arrival times at location v provided by t. This enables us to formulate the decoupled pricing problem

as follows. For each $\omega \in \Omega$, the decoupled pricing problem is to find a pair (P, t) which is the optimal solution to the following problem:

$$\min_{(P,t)} \; p_\omega \sum_{(i,j)\in P} c_{ij} + \sum_{i\in\bar{P}} [\lambda_i^\omega + \pi_i^\omega t_i] \tag{5}$$

$$t_i + t_{ij} \leq t_j \qquad \forall (i,j) \in P \tag{6}$$

$$\sum_{i\in\bar{P}} d_i^\omega \leq Q \tag{7}$$

$$t_i \in [s_i, e_i] \qquad \forall i \in \bar{P}$$

$$P \in R(\omega) \cup C(\omega)$$

In the above formulation (5) are the reduced costs of the variable corresponding to the pair (P, t). Constraints (6) ensure that travel time is not violated by the arrival times. Recall that it is allowed to wait at a location. The capacity constraint is described by (7). This is a shortest path problem with two resources, time and capacity, with constant costs on the arcs and nodes as well as costs at the nodes that are linear in time. For simplicity we will refer to these decoupled problems as pricing problems as well.

The column generation algorithm

To initialize the column generation algorithm, a set of initial routes R_ω is constructed for each scenario $\omega \in \Omega$. In our implementation we have chosen to use return trips (routes r such that its path are of the form $\{(0, i), (i, n+1)\}$) to each customer. For each scenario, multiple return trips are generated per customer, with arrival times evenly spread over the exogenous time window, such that for every choice of endogenous time windows there is a feasible return trip.

In each iteration we solve the LP relaxation of $OPT([R_\omega]_{\omega\in\Omega})$ such that for each scenario ω only the routes included in R_ω are used. Next the pricing problem is solved to either find new routes to add to the model or prove optimality. The pricing problem can be solved for each individual scenario to find a feasible route corresponding to that scenario.

An algorithm for solving the pricing problem may yield several routes with negative reduced costs that are feasible for a single scenario ω, each route carrying a different amount of total demand. All of them are added to R_ω. In algorithm 1, pseudo code of the column generation algorithm is presented.

Algorithm 1 Column Generation Algorithm version 1 (CGA1)

```
Initialize R_ω for all ω
while Optimum not found do
  Solve LP using the routes R_ω for scenario ω
  Solve the Pricing Problems for all ω
  if There does not exist a route with negative reduced costs then
    Optimum found
  else
    Add all found routes with negative reduced costs to the corresponding R_ω
  end if
end while
```

The algorithm as described above will be referred to as CGA1. Next, another version of the column generation algorithm is described, using a different strategy for adding columns.

Observe that when solving the pricing problem for one scenario, the routes that are found might be used for other scenarios as well. In version 2 of the column generation algorithm,

CGA2, the pricing problems are solved iteratively. In this algorithm, all routes with negative reduced costs for scenario ω are added to $R_{\omega'}$ as well if they are feasible and have negative reduced costs for scenario ω'.

When a route found for scenario ω is added to $R_{\omega'}$, CGA2 does not solve the pricing problem for scenario ω' in the same iteration. This potentially reduces the number of pricing problems to be solved, and therefore also the computation time.

Moreover, including a route in multiple scenarios as is done in CGA2 also helps overcome the following problem. When a good route r is added to R_ω, for instance one that is part of the optimal solution, the solution of the LP relaxation might not improve much. The routes in $R_{\omega'}$, for $\omega' \neq \omega$, can not be used to construct an improved solution incorporating r when no endogenous time window satisfied by r is feasible.

In algorithm 2, CGA2 is described.

Algorithm 2 Column Generation Algorithm version 2 (CGA2)

```
Initialize R_ω for all ω
while Optimum not found do
  Solve LP using the routes R_ω for scenario ω
  Set Ω̂ = Ω
  while Ω̂ ≠ ∅ do
    Choose ω ∈ Ω̂ and remove it from Ω̂
    Solve the pricing problem for scenario ω, to find routes R
    Add all routes r ∈ R such that r has negative reduced costs for scenario ω to R_ω
    for All ω̂ ∈ Ω̂ do
      Let R̂ be all routes r ∈ R such that r is feasible and has negative reduced costs for
      scenario ω̂
      if R̂ ≠ ∅ then
        Add R̂ to R_ω̂ and remove ω̂ from Ω̂
      end if
    end for
  end while
  if No new routes are added to the formulation then
    Optimum found
  end if
end while
```

In the column generation algorithms, the computational difficulty lies in solving the pricing problems. Next, we describe how the choice of routes $\mathbf{C}$ affects the computational complexity of these pricing problems. We will suggest two specific choices of $\mathbf{C}$ for which we have designed an algorithm to solve the pricing problem.

Obtaining LP-bounds for different choices of C

As a customer is visited exactly once, the optimal solution to the TWAVRP will not contain any cyclic route for any formulation $OPT(\mathbf{C})$. Hence, a natural choice for allowed cyclic routes $C(\omega)$ is $C(\omega) = \emptyset$. This choice limits the paths in the pricing problem to $R(\omega)$, the elementary $(0, n+1)$ paths in G. However, this places excessive burden on the computational complexity of the pricing problem. Therefore, we suggest including cyclic routes in $\mathbf{C}$ to limit the computational complexity at the expense of a decreased LP-bound. Another option is to let $\mathbf{C}$ contain all acyclic and cyclic routes but eliminate routes that contain cycles of at most k nodes, generally referred to as $k-$cycles. As can be read in Irnich and Villeneuve (2003), $k-$cycle elimination has been successfully employed to find solutions to shortest path problems with resource constraints.

In the case of VRPTW, Kohl et al. (1999) have shown that $k-$cycle elimination can be used to reduce overall computation time.

There does not exist an integer solution to the TWAVRP in which routes that contain cycles are selected, hence no route that visits more than n locations will be selected. Therefore we will only consider routes that visit at most n locations. As will become apparent in the description of our algorithm for solving the pricing problem, this choice helps us in the design of the algorithm and it slightly increases the value of the LP-bound.

For the formulations of the TWAVRP in which all routes are allowed in which at most n locations are visited and that do not contain $k-$cycles, the value of the LP-bound increases with k, as does the computational complexity. In the paper by Kohl et al. (1999) it is illustrated for the VRPTW that $2-$cycle elimination provides a significantly better LP-bound than allowing all cyclic routes, while the increase in computational complexity is limited. In the remainder of the paper, we will consider two choices of $\mathbf{C}$; including all cyclic routes visiting at most n customers and including all cyclic routes visiting at most n customers except routes containing $2-$cycles.

Next, we will describe an algorithm for solving the pricing problem when all cyclic paths visiting at most n customers are allowed, referred to as no-cycle elimination. This is followed by a description of the algorithm for the case where $2-$cycles are eliminated, referred to as $2-$cycle elimination.

Solving the pricing problem with no-cycle elimination

When $\mathbf{C}$ includes all cyclic routes visiting at most n locations, the pricing problem is a shortest path problem with time window and capacity constraints, with constant costs on the arcs and nodes and linear costs in time defined on the following acyclic graph. Consider the graph G', containing the start and end depot and n copies of each customer, ordered in layers[1]. At each node in a specific layer, all outgoing arcs end at a node in the next layer or at the ending depot, see Figure 1 for an example with 4 locations. Observe that any elementary path in G corresponds to a path trough copies in G'. Moreover, G' also contains paths that visit multiple copies of the same customer, corresponding to cyclic paths in G visiting at most n locations. To properly define the pricing problem using G', let the values of the parameters associated to the copied nodes and arcs in G' be the same as for their originals in G.

In Ioachim et al. (1998) a labeling procedure is presented to solve the shortest path problem with time window constraints and linear costs in time on an acyclic graph. First we summarize their algorithm, followed by a description on how their algorithm is modified to incorporate capacity constraints in order to solve our pricing problem.

The algorithm of Ioachim et al. provides a shortest path for each possible arrival time at the end depot. One assigns a label to each node v in a graph describing both the costs of the shortest path to v as a function of the arrival time t with domain $[s_v, e_v]$, and the preceding node in the shortest path. The cost function of each label is piecewise linear with a finite number of line pieces. Associated to each line piece is a preceding node in the shortest path. The labels are propagated through the network using a basic dominance criterion common to labeling algorithms, which at each node compares the costs of the paths that have equal arrival time.

To modify the labeling algorithm such that capacity constraints are taken into account we have extended the label. In our modified labeling algorithm, a label for node v is a vector of cost functions, one function for each possible load upon arrival at v. Dominance of paths is evaluated for paths ending at v at the same time with the same load. The modified labeling algorithm provides a shortest path for each possible load and arrival time at the end depot.

[1]Let $G' = (V_{\bigcup}, E')$, let $V'_k = \{(v,k)|v \in V'\}$, then $V_{\bigcup} = \{0, \bigcup_{k=1}^{n} V'_k, n+1\}$ and $E' = \{((u,k),(v,k+1))|u,v \in V', u \neq v\} \bigcup \{(0,i)|i \in V'_1\} \bigcup \{(i,n+1)|i \in \bigcup_{k=1}^{n} V'_k\}$

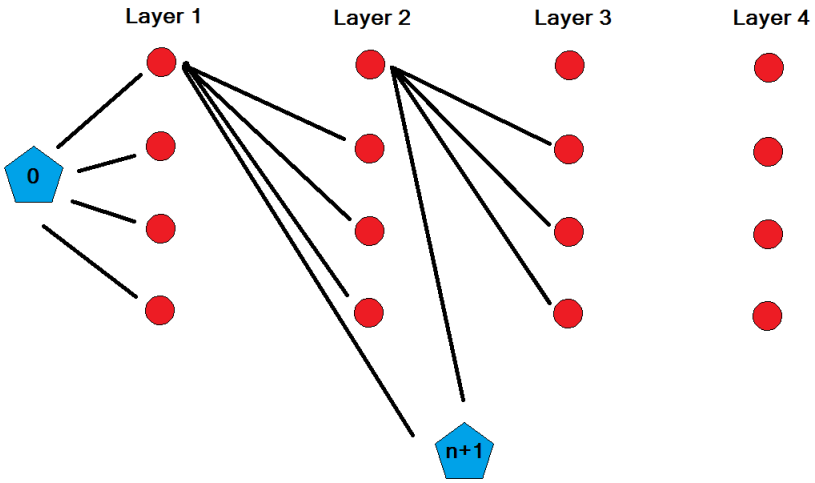


Figure 1: Example of G', not showing all arcs

For a detailed exposition of the labeling algorithm for the problem without capacity constraints on an acyclic graph we refer the reader to Ioachim et al. (1998). Pseudo code of the modified labeling algorithm is provided in Appendix 1.

Solving the pricing problem with 2−cycle elimination

Next we present an algorithm to solve the pricing problem for the TWAVRP with 2−cycle elimination. Note that $k-$cycles in G are represented in G' by paths visiting copies of the same location. This allows us to use G' to solve the pricing problem with 2−cycle elimination.

Irnich and Villeneuve (2003) propose a solution method for solving the shortest path problem with resource constraints and 2−cycle elimination. The above labeling algorithm is modified, to solve our pricing problem including linear node costs with 2−cycle elimination. Following the ideas of Irnich and Villeneuve (2003), two labels are assigned to each node. The first label describes the costs of the shortest path to a node arriving at time t with load q. The second label describes the costs of the shortest path with arrival time t and load q among the paths that have a different predecessor than the path in the first label.

Observe that by construction the first and second path ending at a node representing a copy of location i at time t with load q, have predecessors representing different locations j and k respectively. Therefore, when extending a path from i to a node representing location j, the label describing the costs of the first path is not used as this would in effect yield a path containing the 2−cycle $j-i-j$. Instead, the second path is now used for extending, yielding the shortest path not containing a 2−cycle. Pseudo code for the labeling algorithm with 2−cycle elimination is provided in Appendix 2.

Finally, note that Irnich and Villeneuve (2003) show that the required number of labels per node in a labeling algorithm that eliminates $k-$cycles quickly increases, as does the computation time of such an algorithm. For 2−cycle elimination, only 2 labels are required at each node. For 3−cycle elimination as much as 6 labels are necessary. Irnich and Villeneuve conjecture that a tight upper bound is $k!$ labels. Moreover, they have shown that at most $k(k-1)!^2$ labels are necessary per node for $k-$cycle elimination.

3.2 Branch and Price

To find the optimal solution to the TWAVRP we use a branch and price algorithm. Lower bounds are obtained by solving the LP relaxation of the TWAVRP with no-cycle elimination or with 2−cycle elimination using either CGA1 or CGA2. Upper bounds are obtained when an integer solution is found. Next, the branching rules are described.

Branching on arcs

Let (x^u, y^u) be the optimal solution to the LP relaxation of the TWAVRP at a specific node u in the branching tree. Based on this solution, one can determine the possibly fractional values z_e^ω with which each arc e in the graph G is used in the LP solution. We apply special ordered subset (SOS) branching on the arcs based on the values z.

More specifically, for branching node u, scenario ω and location i, let $\delta^o_{u\omega}(i)$ be the set of the in or out arcs indicated by o =in or o =out respectively. Next, a location i', a scenario ω' and arc type o' is selected such that the number of arcs e in $\delta^{o'}_{u\omega'}(i')$ for which $z_e^{\omega'} > 0$ is highest. Let $\delta^{o'}_{u\omega}(i') = \{e_1, ..., e_m\}$ be ordered with respect to z, such that $z_{e_k}^{\omega'} \geq z_{e_l}^{\omega'}$ if $k < l$. The arcs are divided into two groups, S and its complement, where $S = \{e_1, ..., e_k\}$ is such that $\sum_{e\in S} z_e^{\omega'} \geq 0.5$ and $\sum_{e\in S\setminus e_k} z_e^{\omega'} \leq 0.5$. In one branch we disallow the use of the arcs in S and in the other we disallow the use of the arcs in $\bar{S}$. This does not alter the pricing problem, in fact the number of arcs in the network decreases. In algorithm 3, the branching procedure is summarized.

Algorithm 3 Branch on arcs

- Calculate the arc use $z_e^\omega = \sum_{r\in R(d^\omega)|e\in r} x_r^{u\omega}$
- Choose $(i', \omega', o') \in \arg\max |\{e|e \in \delta^o_{u\omega}(i), z_e^\omega > 0\}|$
- For $\delta^{o'}_{u\omega'}(i') = \{e_1, ..., e_m\}$, let $S = \{e_1, ..., e_k\}$ such that $\sum_{e\in S} z_e^{\omega'} \geq 0.5$ and $\sum_{e\in S\setminus e_k} z_e^{\omega'} < 0.5$.
- Construct two new nodes for the branch and bound tree, in one node disallow the use of the arcs in S, in the other node disallow the use of the arcs in $\bar{S}$

Note that when no arcs are used fractionally, the current solution to the LP relaxation is either integer or could easily be transformed into an integer solution with equal value. In this case no branching is performed and the upper bound of the branch and bound tree is updated if necessary.

Branching on time windows and arcs

Next we investigate branching on the variable y, referred to as branching on time windows. Branching on time windows affects the routes that might be chosen in all scenarios simultaneously. This might have a greater impact on the value of the LP relaxation than branching on arcs, which only influences the routes that might be chosen in a single scenario.

We propose the following decision rule to branch on time windows. At each node of the branch and bound tree, first calculate the (fractional) amount of routes that are selected in the optimal solution of the LP relaxation, which fall outside the endogenous time window. Next choose the location for which this fraction is highest. Finally, create two new nodes for the branching tree by bounding the endogenous time window, or, equivalently, by altering the exogenous time window.

More precisely, let i be the location for which branching on time windows will be performed. Let L_i be the latest arrival time at location i among the fractionally selected routes and E_i the earliest arrival time. We create one new node in the branching tree by adding the constraint $y_i \leq \frac{E_i+L_i}{2} + \frac{w_i}{2}$ and another by adding the constraint $y_i \geq \frac{E_i+L_i}{2} - \frac{w_i}{2}$. The new time window constraints do not alter the structure of the pricing problem, as they can be added by modifying the exogenous time windows.

Note that branching solely on time windows does not guarantee that an integer solution is found. Therefore, algorithm 4 combines branching on arcs and time windows and is sufficient to obtain the optimal integer solution. A threshold is chosen which is used to decide whether branching on arcs or time windows should be performed. When the amount of routes selected outside the time window exceeds this threshold for all locations, branching on time windows is performed, otherwise branching on arcs is performed. In algorithm 4, this branching procedure is summarized.

Algorithm 4 Branch on time windows and arcs

Calculate the amount of routes selected outside the time window $h_i = \sum_{\omega \in \Omega} \sum_{r \in R(\omega) | \exists t_i^r \notin [y_i^*, y_i^* + w_i]} x_r^\omega$
Choose $j \in \arg\max\{h_i\}$
if $\frac{h_j}{|\Omega|}$ >Threshold **then**
  Construct two new nodes for the branch and bound tree, add constraint $y_j \leq \frac{E_j + L_j}{2} + \frac{w_j}{2}$ in one node and $y_j \geq \frac{E_j + L_j}{2} - \frac{w_j}{2}$ in the other
else
  Branch on arcs using Algorithm 3
end if

Note that the computation time of the labeling procedure is dependent on the number of arcs in the network and the width of the time windows. Therefore, the computation time necessary to solve the pricing problem will decrease in new nodes of the branch and bound tree when branching is performed as described in Algorithm 3 and 4.

4 Numerical experiments

In this section the results of numerical experiments are presented. In section 4.1 the performance of the column generation algorithm is discussed. In section 4.2 the performance of the branch and price algorithm to solve the TWAVRP to optimality is discussed. Moreover, we will comment on the relative gap between the value of the LP relaxation of our formulation and the optimal integer solution of the TWAVRP.

Unless stated otherwise, instances of the TWAVRP are randomly generated for different numbers of customers and different numbers of scenarios. The choice of parameter values is based on experiences with a Dutch retail company.

Locations:	Uniformly distributed in a square (5×5) area around the depot. The starting depot and ending depot are on the same location.
Travel Costs:	Equal to the Euclidean distance.
Travel Time:	Equal to the Euclidean distance.
Exogenous Time Windows:	Three versions $[10, 16], [9, 18]$ and $[7, 21]$, randomly assigned to customers at fixed frequency $\{0.1, 0.6, 0.3\}$ respectively. The depot has time window $[6, 22]$.
Time Window Width:	2.
Vehicle Capacity:	30.
Demand per scenario:	Randomly generated using a normal distribution with different randomly generated mean per customer and a variance of 1. The mean was generated using a normal distribution with mean 5 and variance 1 for each customer. All demand was rounded to the next integer. Demand below 1 is rounded up to 1 and demand over 30 is rounded down to 30.
Scenario Probability Distribution:	Equal probabilities.

In section 4.2 we also solve a modified version of the instances generated by Groër et al. (2009) for the ConVRP. Locations, travel costs, travel time, time window width, vehicle capacity and demand per scenario are the same as in their instances. We used the maximal driving time per driver as the exogenous time window width of the depot, as well as the exogenous time window of all other locations. We let the probability of a scenario occurring be equal for each scenario.

The algorithms are coded in C++ and CPLEX 12.3 is used for solving Linear Programs. All experiments were performed on Pentium(R) Dual-Core CPU, E5800, 3.2GHz with 4.00GB of RAM

4.1 Numerical experiments on the column generation algorithm

In this section the results of numerical experiments on solving the LP relaxation of instances of the TWAVRP are shown. The LP relaxations are solved to optimality using the proposed column generation algorithms using formulations with no-cycle elimination and with 2-cycle elimination. The instances vary in the number of customers and the number of scenarios. We will compare the computation time of both column generation algorithms. Furthermore, we will highlight the difference in LP value of the TWAVRP with no-cycle elimination and with 2-cycle elimination. In section 4.2 we will comment on the relative gap between the LP-Bound and the value of the optimal integer solution.

Table 1 contains average LP values of instances of the TWAVRP with 10 scenarios are shown, as well as the average computation time of CGA1 and CGA2. The instances are solved with no-cycle elimination and with 2-cycle elimination. For each choice of the number of customers, 25 instances were generated of which the average results are presented in table 1. The first column indicates the number of customers in the instances, the second shows the optimal LP value. Moreover, average total computation time in seconds is shown for both CGA1 and CGA2. The computation time is decomposed into time spent on solving LP problems and time spent on solving the pricing problems. There is a slight discrepancy between the total computation time and the cumulative time spent on solving the LP relaxations and the pricing problems. This is due to overhead such as initializing the algorithm. Finally, the number of generated routes are presented, including the initial routes. Routes that are used for multiple scenarios are counted multiple times.

First note that the LP value increases on average with 6.6% from solving the instances with no-cycle elimination to solving them with 2-cycle elimination. The computation time increases significantly for both CGA1 and CGA2. Secondly, it is apparent that the computation time grows more than linearly in the number of customers. Furthermore, CGA1 generates a higher number of routes than CGA2. Finally, observe that CGA2 spends less time on solving the instances than CGA1. It spends less time on solving LP's as well as on solving the pricing problems. Both CGA1 and CGA2 spend most of their time on solving the pricing problem.

Next, table 2 shows the computational results of numerical experiments where the LP relaxation of instances of the TWAVRP with 15 customers are solved. They are solved with no-cycle elimination and with 2-cycle elimination. In these experiments, 25 instances are repeatedly solved for different numbers of scenarios. The choices of the number of scenarios are indicated in the first column. The table shows average scores over the instances. In these experiments, the total computation of CGA1 seems to grow approximately linearly in the number of scenarios for both the no-cycle elimination as the 2-cycle elimination case. However, the time spent on the LP's grows more than linearly, significantly faster than the growth of the time spent on the pricing problems. The computation time of CGA2 grows more than linearly, although even for the instances with 100 scenarios, the computation time is less than half the computation time of CGA1.

Next, we will look at the amount of time spent per iteration on solving the LP's and on solving the pricing problems. Recall that the main idea behind CGA2 is to save time in the pricing problem phase, by solving less pricing problems. Moreover, the effect of using a route for

Table 1: Averaged computational results on CGA1 and CGA2 on instances with 10 scenarios

No-Cycle Elimination									
		CGA1				CGA2			
			Time				Time		
#Customers	LP Value	Total	LP	Pricing	#Routes	Total	LP	Pricing	#Routes
10	15.321	2.2090	0.232	1.971	1577.0	0.690	0.100	0.586	1246.4
15	20.011	7.041	0.361	6.668	2554.3	2.663	0.297	2.355	2031.6
25	29.436	33.530	1.089	32.395	4672.5	13.090	0.820	12.220	3508.2
35	38.272	104.779	3.348	101.317	6988.8	40.667	1.755	38.792	5208.0
50	49.975	351.356	9.593	341.429	10364.7	143.060	4.097	138.631	7437.9
2-Cycle Elimination									
		CGA1				CGA2			
			Time				Time		
#Customers	LP Value	Total	LP	Pricing	#Routes	Total	LP	Pricing	#Routes
10	16.857	6.462	0.148	6.311	1591.0	2.418	0.273	2.138	1258.3
15	21.465	23.394	0.380	23.001	2638.0	9.512	0.392	9.112	2075.3
25	31.434	128.808	1.978	126.789	4951.0	50.438	1.134	49.260	3833.5
35	40.082	396.004	6.844	389.043	7339.6	160.925	2.698	158.118	5579.6
50	52.228	1278.150	24.010	1253.810	10933.2	554.119	7.547	546.243	8323.3

Table 2: Averaged computational results on CGA1 and CGA2 on instances with 15 customers

No-Cycle Elimination								
	CGA1				CGA2			
		Time				Time		
#Scenarios	Total	LP	Pricing	#Routes	Total	LP	Pricing	#Routes
2	1.494	0.232	1.255	441.7	0.918	0.117	0.799	409.4
5	3.641	0.162	3.474	1255.8	1.672	0.195	1.471	1019.8
10	8.095	0.318	7.767	2663.9	3.229	0.382	2.835	2106.2
25	21.033	1.203	19.805	6830.3	7.734	0.991	6.715	5168.6
50	43.806	3.565	40.193	13829.8	15.880	3.070	12.758	10620.5
100	92.616	10.316	82.201	27793.1	41.150	9.687	31.363	21018.4
2-Cycle Elimination								
	CGA1				CGA2			
		Time				Time		
#Scenarios	Total	LP	Pricing	#Routes	Total	LP	Pricing	#Routes
2	4.643	0.117	4.520	478.3	2.992	0.119	2.870	418.4
5	12.224	0.153	12.065	1305.8	5.451	0.196	5.247	1064.7
10	26.132	0.412	25.708	2714.4	9.730	0.367	9.351	2140.7
25	68.393	2.006	66.360	6920.6	24.422	1.446	22.952	5405.0
50	140.090	6.287	133.754	13953.0	49.311	4.367	44.896	10640.0
100	296.201	16.405	279.699	27783.2	144.515	15.169	129.254	21169.7

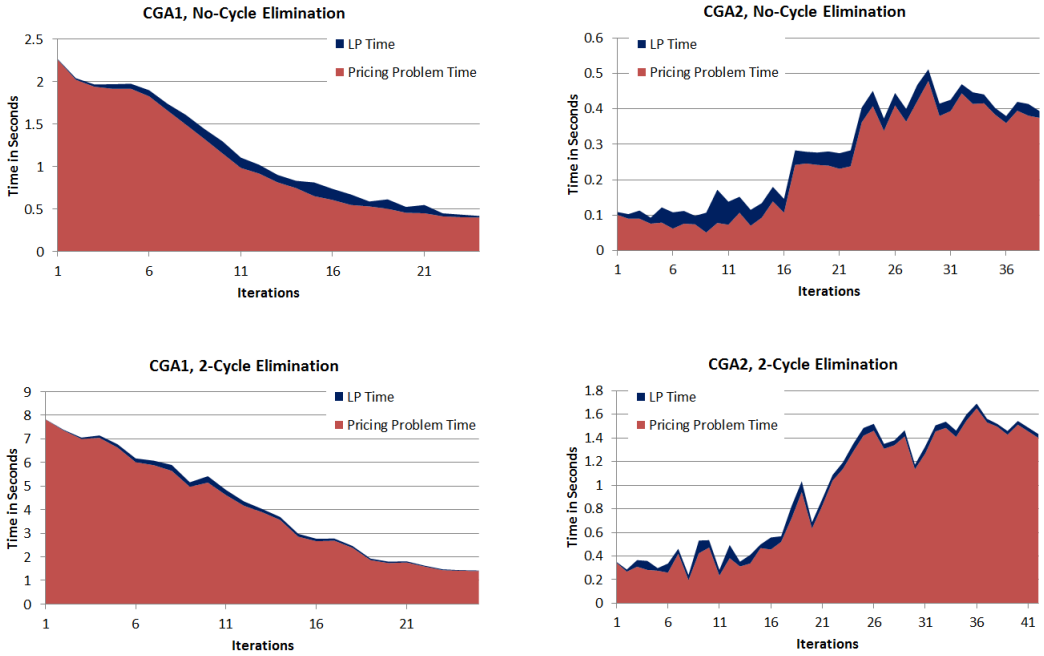
multiple scenarios will be greater for instances with a high number of scenarios. Figures 2 and 3 illustrate the allocation of time spent per iteration for a representative instance of the TWAVRP. They are generated by solving the LP relaxation of a single instance of the TWAVRP with 15 customers, with no-cycle elimination and with 2-cycle elimination, using CGA1 and CGA2, for 25 and 100 scenarios respectively.

In CGA1, the time spent per iteration decreases as the current solution value approaches optimality. However, in CGA2 the time spent per iteration increases. The time spent on the pricing problem in the initial iterations is lower for CGA2, since many routes that are found for one scenario are also suitable for inclusion in other scenarios. Also observe that the minimum time spent per iteration in CGA1 is close to the maximum time spent per iteration for CGA2.

Clearly, the number of iterations needed to find the optimum is higher for CGA2.

Furthermore, the time spent on solving the LP's is negligible in CGA1, as solving the pricing problem takes up most of the computation time per iteration. However, for CGA2, in particular for a larger number of scenarios, solving the LP relaxation takes up a significant portion of the computation time in each iteration. In the first iterations of the algorithm for the TWAVRP instance without 2-cycle elimination and with 100 scenarios as depicted in figure 3, the time spent on solving the LP relaxation is even higher than the time required to solve the pricing problems. This suggests that when solving instances with a high number of scenarios with CGA2, solving the pricing problem to optimality is not the main bottleneck with respect to computation time.

Figure 2: Time spent per iteration on an instance with 15 Customers and 25 Scenarios



4.2 Numerical experiments on the branch and price algorithm

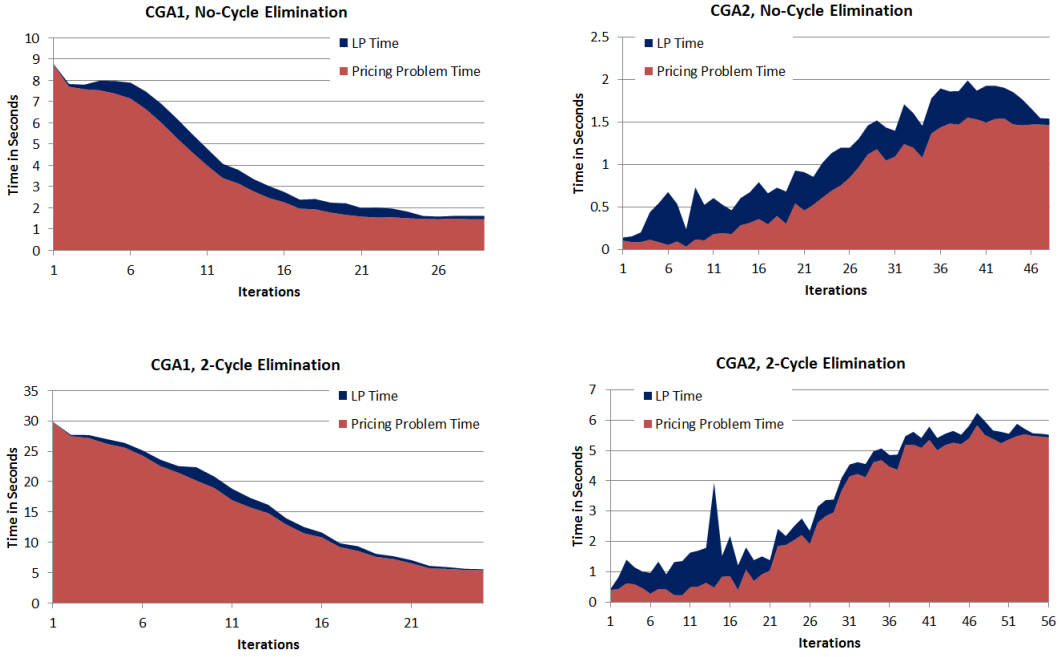
In this section, results of numerical experiments on finding integer optimal solutions to instances of the TWAVRP are presented. Furthermore we illustrate the relative gap between the value of the LP-bound and the optimal integer solution. As indicated in section 4.1, CGA2 outperformed CGA1 in all experiments. Therefore, we will only present findings of using CGA2 in the branch and price algorithm.

In table 3 running time in seconds can be found for applying the branch and price algorithm to 5 instances of the TWAVRP with 8 customers and 3 scenarios. The first column indicates the solved instance. The second column indicates whether no-cycle elimination or 2-cycle elimination was applied in finding lower bounds. The third column refers to the threshold for branching on time windows. We present findings on using a threshold of 0 and 0.5. Furthermore, a dash indicates that no branching on time windows was performed.

The column 'Opt Found' shows the time needed to find the optimal solution, if optimality was proven. The following columns show the time needed to close the optimality gap to 5%, 1% and 0% respectively. If optimality could not be proven within 5 days, the optimality gap at termination is shown in the last column.

Recall that solving the LP relaxation with no-cycle elimination is done significantly faster

Figure 3: Time spent per iteration on an instance with 15 Customers and 100 Scenarios



than with 2-cycle elimination using CGA2. Nevertheless, table 3 shows that on all 5 instances, the branch and price algorithm using 2-cycle elimination proves optimality more than nine times faster than when using no-cycle elimination. In one instance, the optimum could not even be found using no-cycle elimination.

When comparing branching on time windows to branching only on arcs, the computation time of the branch and price algorithm when branching on time windows seems less stable. In some instances branching on time windows helps finding the optimal solution very fast, in others the computation time explodes.

For the remainder of this section, we only present results obtained by using the branch and price algorithm with 2-cycle elimination and no branching on time windows. Table 4 shows the results of computational experiments on the TWAVRP with 9 locations and 3 scenariod. In three out of five instances, the calculation time is very high.

Next, we investigate the effect of the time window width on computation time. We provide results for instances 1 to 5 where the time window width is changed from 2 to 6. Table 5 shows that these modified instances are solved significantly faster. Obviously, when endogenous time windows are very large, the problem reverts to separate VRPTW's, one for each scenario.

Furthermore, we analyze instances of the TWAVRP which are modified versions of the ConVRP instances proposed by Groër et al. (2009). They solved these instances of the ConVRP using CPLEX 11.0 on a 1.4 GHz Intel processor and 512 MB of RAM. They did not specify exact computation times, but report that up to several days were needed to solve each instance.

As table 6 shows, only 4 out of 5 instances with 10 customers and 3 scenarios, and 3 out of 5 instances with 12 customers and 3 scenarios could be solved within 5 days. Observe that in 4 out of 7 solved instances more than 80% of the computation time is spent on proving optimality.

Finally, table 7 shows the relative gap between the value of the LP-bound and the value of the optimal integer solution for the instances for which an optimal integer solution was obtained. The relative gap when using no-cycle elimination is on average 10.2%. When using 2-Cycle elimination, the gap decreases to on average 3.6%. This illustrates that the LP-bound using our formulation, specifically when applying 2-cycle elimination, is very tight.

As the LP-bound is very tight and the branch and price algorithm typically finds the optimal

Table 3: Computation time in second

TWAVRP with 8 Locations, 3 Scenarios

Instance	Cycle Elimination	Threshold	Opt Found	5%	1%	Opt Proven	Final Gap
1	no	-	-	-	-	-	13.87
1	2	0	-	-	-	-	2.79
1	2	0.5	85.598	86.534	660.084	8884.4	-
1	2	-	34.554	26.177	162.069	547.577	-
2	no	-	133344	12819.8	183096	295902	-
2	2	0	3.120	0.795	4.836	5.335	-
2	2	0.5	1.950	1.077	9.859	10.171	-
2	2	-	15.585	6.381	33.665	43.805	-
3	no	-	9.267	22.558	335.26	1090.97	-
3	2	0	-	-	-	-	0.91
3	2	0.5	2.262	2.262	101.057	225.373	-
3	2	-	2.324	2.324	58.563	110.074	-
4	no	-	8.845	36.754	362.124	915.113	-
4	2	0	1.279	1.279	32.963	1913.83	-
4	2	0.5	1.233	1.233	155.236	215.608	-
4	2	-	26.161	1.825	26.348	31.153	-
5	no	-	1410.34	1738.92	1738.92	3266.38	-
5	2	0	0.920	0.920	1.950	1.950	-
5	2	0.5	0.827	0.827	1.872	1.872	-
5	2	-	39.905	22.136	40.576	64.194	-

Table 4: Computation time in second

TWAVRP with 9 Locations, 3 Scenarios

Instance	Opt Found	5%	1%	Opt Proven	Final Gap
6	33934.1	393.027	28768.8	45170.1	-
7	18.782	18.798	97.453	399.751	-
8	11699.7	7325.59	28573.2	313155	-
9	-	-	-	-	1.15
10	2.106	2.106	2.106	34.227	-

Table 5: Computation time in second

TWAVRP with 8 Locations, 3 Scenarios, and Triple Time-Window Width

Instance	Opt Found	5%	1%	Opt Proven	Final Gap
1	43.041	4.524	43.93	125.191	-
2	2.059	1.154	9.812	12.262	-
3	1.669	1.669	44.398	83.226	-
4	3.526	1.217	4.977	4.977	-
5	6.521	2.371	12.059	12.137	-

solution very early in the branching procedure, we conclude that the algorithm spends a lot of time on closing a small optimality gap.

Table 6: Computation time in second

Adapted Consistent Vehicle Routing Instances

Instance	Locations	Opt Found	5%	1%	Opt Proven	Final Gap
11	10	1.466	0.842	0.842	2.839	-
12	10	5969.51	2111.48	9848.06	31036.7	-
13	10	-	-	-	-	17.87
14	10	13.697	296933	298143	298144	-
15	10	5791.97	241.317	4278.51	6593.7	-
16	12	-	-	-	-	13.59
17	12	312.391	5.944	315.121	2844.88	-
18	12	26212.8	397.208	31454.1	166823	-
19	12	74315.3	47.455	6239.62	91851.6	-
20	12	-	-	-	-	1.75

Table 7: Relative gap in percentages

Instance	no-cycle Elimination	2-cycle Elimination
1	14.971	3.086
2	18.246	2.470
3	6.110	3.746
4	8.738	2.500
5	7.472	1.934
6	28.312	4.775
7	10.099	4.259
8	7.704	5.053
10	10.698	0.776
11	5.402	0.153
12	7.108	4.940
14	11.004	8.223
15	6.821	3.980
17	5.087	2.442
18	5.509	4.239
19	10.105	4.914

5 Conclusion

In many distribution networks it is very important to assign time windows to customers as a long term decision. In particular, time windows are often assigned before demand is known. The problem of assigning time windows has largely been overlooked in the scientific literature. In this paper we introduce the time window assignment vehicle routing problem, TWAVRP. It models the problem of assigning time windows to customers before demand is known. In this model, demand realizations occur according to a predefined set of scenarios with known probability distribution. After demand becomes known, an optimal vehicle routing schedule should be made adhering to the assigned time windows. The problem is to assign time windows such that the expected traveling costs are minimized.

To solve the TWAVRP, we suggest using a branch and price algorithm. We have designed two variants of a column generation algorithms to solve the LP relaxation of our formulation of the TWAVRP. Specifically CGA1 and CGA2 are presented, with as main difference that CGA2 uses routes found by solving the pricing problem for one scenario as solutions to the pricing problem of other scenarios. Numerical experiments show that this significantly speeds up the solution process. The technique of using solutions of one problem as solutions to others seems promising for other scenario based optimization problems as well.

We have formulated the TWAVRP in such a way that it allows the incorporation or elim-

ination of cyclic routes. We suggest a trade off between the quality of the LP-bound and its computation time, in the choice of cyclic routes included in the formulation of the TWAVRP. Specifically, numerical experiments show that the LP relaxation of the TWAVRP with no-cycle elimination can be solved faster than with 2-cycle elimination, using our column generation algorithms. However, numerical experiments on finding integer optimal solutions to the TWAVRP show that the increase in LP value for the formulation with 2-cycle elimination, is more than sufficient to offset the increased computation time on solving the LP relaxation. The total computation time of the branch and price algorithm is significantly lower when using 2-cycle elimination.

The proposed column generation algorithm can solve instances of up to 50 customers and up to 100 scenarios in reasonable computation time. However, the branch and price algorithm is only able to solve small sized instances to optimality. Numerical experiments suggest that the LP-bound obtained using our formulation of the TWAVRP with 2-cycle elimination is very tight. Moreover, the optimal solution is often found early in the branching procedure. The algorithm spends a long time on closing a very small optimality gap and further research is needed to speed this process up.

Appendix 1

In this appendix we present the labeling algorithm used for solving the pricing problem without $2-$cycle elimination. This algorithm provides the costs of the shortest $(0, n+1)$ paths for each possible load and arrival time at location $n+1$. Consider the network G' as described in section 3.1. For each node i we will construct a function $F_i(t, q)$ representing the costs of the shortest $(0, i)$ path when the total load of a vehicle arriving at location i is q and service at location i would start at time t. For fixed q, the function $F_i(t, q)$ is piecewise linear in t. For every t and q, the preceding node of i in the path yielding costs $F_i(t, q)$ is stored for backtracking.

The main observation in the design of this algorithm is that for a given path, finding the arrival times at each node, t, such that the total costs are minimized is computationally easy. As the costs at a node are linear in time, t will be as large as possible when the unit costs are negative and as small as possible otherwise. For details on the corectness of this algorithm we refer to Ioachim et al. (1998).

Algorithm 5 contains pseudo code for solving the the pricing problem with no-cycle elimination. Assume that the locations are numbered in topological order:

Algorithm 5 Labeling Algorithm: pricing problem with no-cycle elimination

Initialize $F_0(t, q) = 0$ and $F_i(t, q) = \infty$ for all $i > 0$, t and q.
for nodes $i < n + 1$ **do**
  for all out arcs (i, j) of node i **do**
    for all loads q at i such that there exists a time $t \leq e_j - t_{ij}$ for which $F_i(t, q) < \infty$ and $q + d_j \leq Q$ **do**
      Calculate the costs $f(t)$ of a shortest path starting at 0 and ending at j having predecessor i with a total load of $q + d_j^{\omega}$ and service commencing at time t.
      Let $F_j(t, q + d_j^{\omega})$ be the minimum of the piecewise linear functions $F_j(t, q + d_j^{\omega})$ and $f(t)$ at each t.
    end for
  end for
end for

Note that the costs $f(t)$ are ∞ when the arrival time t is before the start of the exogenous time window. The costs are constant for $t \geq e_i$, this corresponds to an arrival occuring at the end of the exogenous time window and waiting before departure. The optimal path for each possible load can be obtained trough backtracking.

Appendix 2

Next we present the algorithm for solving the pricing problem when 2-cycles are eliminated. We follow the same approach as in algorithm 5. The main difference is that at each node two labels (functions) need to be constructed in stead of one. The second function will be denoted $F_i'(t,q)$ and represents the costs of the shortest path in terms of costs, ending at i with total demand q and arriving at node i at time t, and with a different predecessor with respect to the path resulting in costs $F_i(t,q)$. For every t and q, the preceding node of i in the path yielding costs $F_i'(t,q)$ is stored for backtracking.

At each iteration, let P be the shortest path ending at location i at time t with load q and P' the second path. Consider extending the path P from i to j. When the predecessor of i in P is j, extending P to j would yield a $2-$cycle. By construction of the algorithm, the predecessors of i in P and P' are different. Hence extending P' to j does not yield a $2-$cycle and, if feasible, provides the shortest path ending at location j visiting predecessor node i at time t with load q. For more details on k-cycle elimination we refer to Irnich and Villeneuve (2003).

Algorithm 6 contains pseudo code of the algorithm for the pricing problem algorithm with 2-cycle elimination.

Algorithm 6 Labeling Algorithm: pricing problem with 2-cycle elimination

Initialize $F_0(t,q) = F_0'(t,q) = 0$ and $F_i(t,q) = F_i'(t,q) = \infty$ for all $i > 0$, t and q.
for nodes $i = 1, ..., n$ **do**
  for all out arcs (i,j) of node i **do**
    for all loads q at i such that there exists a time $t \leq e_j - t_{ij}$ for which $F_i(t,q) < \infty$ and $q + d_j \leq Q$ **do**
      Calculate the costs $f(t)$ of a shortest path starting at 0 and ending with nodes $k - i - j$, such that k and j are not a copy of the same location, with a total load of $q + d_j^\omega$ for all t.
      Replace $F_j(t, q + d_j^\omega)$ and $F_j'(t, q + d_j^\omega)$ by the minimum and second best respectively of the piecewise linear functions $F_j(t, q + d_j^\omega)$, $F_j'(t, q + d_j^\omega)$ and $f(t)$ at each t.
    end for
  end for
end for

References

- Chabrier, A. (2006), 'Vehicle Routing Problem with elementary shortest path based column generation', *Computers & Operations Research*, 33, pp. 2972-2990.
- Desaulnier, G. 2010, 'Branch-and-Price-and-Cut for the Split-Delivery Vehicle Routing Problem with Time Windows', *Operations Research*, Vol. 58, No. 1, pp. 179-192.
- Desrochers, M., Desrosiers, J. and Solomon, M. 1992, 'A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows', *Operations Research*, Vol. 40, No. 2, pp. 342-354.
- Desrochers, M., Soumis, F. 1988, 'A reoptimization algorithm for the shortest path problem with time windows', *European Journal of Operations Research*, 35, pp. 242-254.
- Fischer, M.L., Jörnsten, K.O. and Madsen, O.B.G. 1997, 'Vehicle Routing with Time Windows: Two Optimization Algorithms', *Operations Research*, Vol. 45, No. 3, pp. 488-492.
- Groër, C., Golden, B. and Wasil, E. 2009, 'The Consistent Vehicle Routing Problem', *Manufacturing & Service Operations Management*, Vol. 11, No. 4, pp. 630-643.
- Ioachim, I., Gélinas, S., Soumis, F. and Desrosiers, J. 1998, 'A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs', *Networks*, 31, pp. 193-204.

- Irnich, S. and Villeneuve, D. 2003,'The Shortest Path Problem with Resource Constraints and k-Cycle Elimination for $k \geq 3$', *Les Cahiers du GERAD*, 55.
- Jabali, O., Leus, R., van Woensel, T. and de Kok, A.G. 2010, 'Self-impose time windows in vehicle routing', *Working Paper* 1028, Katholieke Universiteit Leuven.
- Kohl, N., Desrosiers, J., Madsen, O.B.G., Solomon, M.M. and Soumis, F. 1999, '2-Path Cuts for the Vehicle Routing Problem with Time Windows', *Transportation Science*, Vol. 33, No. 1, pp. 101-116.
- Laporte, G. 2007, 'What you should know about the vehicle routing problem', *Naval research Logistics*, Vol. 54, No. 8, pp. 811-819.
- Toth, P. and Vigo, D. 2002, 'Models, relaxations and exact approaches for the capacitated vehicle routing problem', *Discrete Applied Mathematics*, 123, pp. 487-512.