The time window assignment vehicle routing problem with time-dependent travel times

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times

Remy Spliet, Said Dabia, Tom Van Woensel

To cite this article:
Remy Spliet, Said Dabia, Tom Van Woensel (2018) The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times. Transportation Science 52(2):261-276. https://doi.org/10.1287/trsc.2016.0705

http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Vol. 52, No. 2, March–April 2018, pp. 261–276
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

The Time Window Assignment Vehicle Routing Problem with Time-Dependent Travel Times

Remy Spliet,[a] **Said Dabia,**[b] **Tom Van Woensel**[c]

[a] Econometric Institute, Erasmus University, 3062 PA Rotterdam, Netherlands; [b] Department of Information, Logistics and Innovation, Vrije University, 1081 HV Amsterdam, Netherlands; [c] Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, 5600 MB Eindhoven, Netherlands

Contact: spliet@ese.eur.nl (RS); s.dabia@vu.nl (SD); t.v.woensel@tue.nl (TVW)

Received: January 29, 2016
Revised: May 27, 2016
Accepted: June 4, 2016
Published Online in Articles in Advance: March 21, 2017

https://doi.org/10.1287/trsc.2016.0705

Abstract. In this paper, we introduce the time window assignment vehicle routing problem (TWAVRP) with time-dependent travel times. It is the problem of assigning time windows to customers before their demand is known and creating vehicle routes adhering to these time windows after demand becomes known. The goal is to assign the time windows in such a way that the expected transportation costs are minimized. We develop a branch-price-and-cut algorithm to solve this problem to optimality. The pricing problem that has to be solved is a new variant of the shortest path problem, which includes a capacity constraint, time-dependent travel times, time window constraints on both the nodes and on the arcs, and linear node costs. For solving the pricing problem, we develop an exact labeling algorithm and a tabu search heuristic. Furthermore, we present new valid inequalities, which are specifically designed for the TWAVRP with time-dependent travel times. Finally, we present results of numerical experiments to illustrate the performance of the algorithm.

Keywords: vehicle routing • column generation • branch-price-and-cut

1. Introduction

Consider a distribution network consisting of a single depot and multiple customers. Before customer demand is learned, a time window is assigned to each customer during which the customer will receive its delivery. After demand of every customer is learned, vehicle routes are designed that satisfy the assigned time window constraints. This situation occurs frequently in practice, for instance, in retail chains where the customers are the stores that are resupplied on a regular basis. It is common practice to fix a time window for delivery for a long time period encompassing multiple deliveries. This is, for example, necessary for the scheduling of delivery-handling personnel or inventory management.

The time window assignment vehicle routing problem (TWAVRP), introduced by Spliet and Gabor (2015), is the problem of assigning time windows before demand is learned such that the expected transportation costs are minimized. In this problem, for each customer an endogenous time window of fixed width has to be selected from an exogenous time window (e.g., two hours within the opening hours of a store). It is a special case of the consistent vehicle routing problem introduced by Groër, Golden, and Wasil (2009) in which additionally customers have to be visited by the same driver at each delivery. Spliet and Desaulniers (2015) also studied the discrete variant of this problem, where a time window for each customer needs to be selected from a finite set of candidate time windows.

However, in these models the travel times are assumed to be constant, which is a significant shortcoming in many real-life applications. Typically, the travel time between two locations is not constant but varies during a day, affecting the efficiency and even feasibility of a delivery route. These varying travel times need to be taken into account when assigning time windows, in particular to assess whether efficient delivery routes can be made adhering to the assigned time windows. It makes little sense in this case to make very specific time window assignments if the travel time between customers is not properly incorporated. In this paper, we incorporate time-dependent travel times, resulting in the TWAVRP with time-dependent travel times.

Some variations in travel time are difficult to predict, like traffic accidents, which cause congestion of which both the occurrence and delay are difficult to foresee. Others are relatively easy to predict like congestion in morning and evening rush hours. We focus on predictable variations in travel time and assume the travel time between two locations at any time of the day is known in advance. Arguably, the predictable congestion is responsible for most variation in travel time in many urban areas, which is exactly what we capture in our model. We model the travel time using a time-dependent travel time function in a similar way as is

done for the traveling salesman problem by Cordeau, Ghiani, and Guerriero (2014), for the pollution-routing problem by Franceschetti et al. (2013), for the vehicle routing problem with time windows by Dabia et al. (2013), and for the vehicle dispatching problem by Ichoua, Gendreau, and Potvin (2003). See Gendreau, Ghiani, and Guerriero (2015) for a recent comprehensive review on time-dependent routing problems.

The TWAVRP with time-dependent travel times is a vehicle routing problem with consistency considerations. Indeed, for each realization of demand, or analogously for each day of delivery, the time of arrival needs to be consistent, i.e., within the same time window. Vehicle routing problems with consistency considerations have recently seen an increasing amount of attention in the scientific literature; see, for example, Kovacs et al. (2014) for an overview of vehicle routing problems with consistency considerations. Now that commercial routing software has become available to practitioners to reoptimize delivery routes on a daily basis, the managerial focus seems to be shifting toward consistency of deliveries. Kovacs et al. (2014) distinguish between three main pillars of consistency: arrival time, person oriented, and delivery consistency. The TWAVRP with time-dependent travel times obviously falls within the domain of arrival time consistency. However, to our knowledge, no research on vehicle routing problems with consistency considerations has been presented in the scientific literature that takes time-dependent travel times into account. In particular, for vehicle routing problems with arrival time consistency, incorporating time-dependent travel times seems to be an important next step.

Note that for a given time window assignment and demand realization, the TWAVRP with time-dependent travel times is a vehicle routing problem with time windows and time-dependent travel times as studied by Dabia et al. (2013). In their paper, they present a branch-and-price algorithm to solve this latter problem. In the paper by Spliet and Gabor (2015), a branch-price-and-cut algorithm is presented to solve the TWAVRP. Therefore, it is natural to build on these algorithms to develop a branch-price-and-cut algorithm for the TWAVRP with time-dependent travel times. However, as we will show in this paper, the former algorithms are not straightforwardly modified. Roughly stated, this is mainly due to additional limitations on which columns in the formulation may be combined to represent feasible routes. Hence, including time-dependent travel times in the TWAVRP requires a new formulation. In this paper, we present an exact and heuristic pricing algorithm and use this in an exact branch-price-and-cut algorithm. Next to presenting a state-of-the-art branch-price-and-cut algorithm, we also develop new valid inequalities specifically for the TWAVRP with time-dependent travel times.

The main contributions of this paper are the following. We introduce the TWAVRP with time-dependent travel times. Not only does this extend the TWAVRP with a crucial new feature, it also seems to be the first study on vehicle routing problems with arrival time consistency considerations in the scientific literature to include time-dependent travel times. Furthermore, we develop a branch-price-and-cut algorithm to solve this problem to optimality, employing newly introduced valid inequalities. In doing so, we encounter a pricing problem that has not been studied before. It is a shortest path problem with a capacity constraint, time-dependent travel times, time window constraints on both the nodes and on the arcs, and linear node costs. We develop an exact labeling algorithm based on the algorithm by Ioachim et al. (1998), as well as a tabu search heuristic. Finally, we present the results of numerical experiments to provide insight in the performance of the column generation and branch-price-and-cut algorithm.

The remainder of this paper is structured as follows. In Section 2, we provide a formal problem definition. In Sections 3–5 we present, respectively, a column generation algorithm, a tabu search pricing heuristic, and valid inequalities, which are used to find lower bounds for the TWAVRP with time-dependent travel times. Furthermore, Section 6 details the branch-price-and-cut algorithm used to find optimal solutions. In Section 7, the results of numerical experiments are shown. Finally, we end with some algorithmic insights and a conclusion in Sections 8 and 9, respectively.

2. Problem Definition

Consider a complete graph $G = (N, A)$, where $N = \{0, \dots, n+1\}$ is a set of locations such that 0 represents the starting depot, $n+1$ represents the ending depot, and $N' = \{1, \dots, n\}$ are the customers. Let $\tau_{ij}(t)$ denote the travel time from location i to j when i is departed from at time t. We assume this function is piecewise linear, continuous, and it satisfies the first in, first out (FIFO) property, that is, the arrival time function $A_{ij}(t) = t + \tau_{ij}(t)$ is strictly increasing. Let $c^f_{ij} \geq 0$ be the fixed costs to travel along arc (i, j) (for instance, fuel costs and tolls) and let $c^h \geq 0$ be the transportation costs per hour (for instance, driver salary). Furthermore, an unlimited number of vehicles of equal capacity Q is available. Note that we do not assume that the fixed travel costs satisfy the triangle inequality. Also, we do not assume that the travel time at any time t satisfies the triangle inequality.

Let Ω be a set of scenarios, where each scenario represents a realization of demand. The probability that scenario ω occurs is p_ω. Let demand at customer i in scenario $\omega \in \Omega$ be given by d^ω_i where $0 < d^\omega_i \leq Q$. For ease of notation, let $d^\omega_0 = d^\omega_{n+1} = 0$.

Associated with each location $i \in N$ is the exogenous time window $[s_i, e_i]$, which should not be confused with the endogenous time window. For ease of notation assume $0 \leq s_0 \leq s_i$ and $e_i \leq e_{n+1}$ for all $i \in N$.

In this paper, we use the term route to refer to a pair $(P, \bar{t})$ where P is a path in G starting at 0 and ending at $n+1$ and $\bar{t}$ is a vector containing the time of service at each location on the path. Furthermore, let t_r^i be the cumulative time of service of customer $i \in N'$ on route r, i.e., if location i is not visited $t_r^i = 0$, if it is visited once t_r^i is the time of service, and if customer i is visited multiple times t_r^i is the sum of the times of service. We refer to $t_r^{n+1} - t_r^0$ as the route duration. Denoting the arcs on path P corresponding to route r by $\{P_1, \ldots, P_k\}$, we assign to route r the costs $c_r = c^h(t_r^{n+1} - t_r^0) + \sum_{i=1}^{k} c_{P_i}^f$.

A route is considered feasible for scenario ω if (i) the capacity constraint in scenario ω is satisfied, (ii) the exogenous time window constraints are satisfied, and (iii) the time of service at location j is not before the time of service at location i plus the travel time if location j is visited directly after i. Note that waiting at a customer is allowed. Let $R(\omega)$ be the set of all feasible routes for scenario ω.

An endogenous time window of width w_i within the exogenous time window has to be assigned to each customer $i \in N'$ during which it will receive its delivery. The assignment is 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 TWAVRP with time-dependent travel times is to assign time windows before demand is known and selecting feasible routes in each scenario $\omega \in \Omega$ that satisfy these time windows. The objective is to minimize the expected transportation costs.

2.1. Travel Time Function

The function $\tau_{ij}(t)$ denotes the travel time from i to j when i is departed from at time t. We assume this function is piecewise linear and contains L_{ij} line pieces. We follow earlier work in modeling the travel time function; see, for example, Ichoua, Gendreau, and Potvin (2003). Denoting for the moment $L_{ij} = m$, it can be represented by line pieces $l_{ij1}, \ldots, l_{ijm}$ and we refer to the start and end of these line pieces as breakpoints $\beta_{ij1}, \ldots, \beta_{ijm+1}$. Line piece l_{ijk} is characterized by an intercept α_{ijk} and slope γ_{ijk} and is represented as follows:

$$l_{ijk} = \{\beta_{ijk}, \beta_{ijk+1}, \alpha_{ijk}, \gamma_{ijk}\}.$$

Furthermore, we assume $\tau_{ij}(t)$ is continuous, so in particular $\alpha_{ijk} + \gamma_{ijk}\beta_{ijk+1} = \alpha_{ijk+1} + \gamma_{ijk+1}\beta_{ijk+1}$. Moreover, we assume it satisfies the FIFO property, meaning that the arrival time function $A_{ij}(t) = t + \tau_{ij}(t)$ is strictly increasing. This implies $dA_{ij}(t)/dt > 0$ for all $t \in [b_{ij1}, b_{ijm+1}]$. Note that by extension $A_{ij}(t)$ is also piecewise linear, described by

$$A_{ij}(t) = \alpha_{ijk} + (1 + \gamma_{ijk})t \quad \forall k \in \{1, \ldots, m\}, \forall t \in [\beta_{ijk}, \beta_{ijk+1}].$$

That $A_{ij}(t)$ is strictly increasing thus yields for all $k \in \{1, \ldots, m\}$ and $t \in [\beta_{ijk}, \beta_{ijk+1}]$

$$\frac{dA_{ij}(t)}{dt} = 1 + \gamma_{ijk} > 0.$$

Therefore, we conclude that the FIFO property is satisfied if $\gamma_{ijk} > -1$ for all $k \in \{1, \ldots, m\}$. We assume that $\gamma_{ijk} > -1$ such that $A_{ij}(t)$ is strictly increasing and therefore also invertible.

Next, we describe the function $D_{ij}(T)$ denoting the departure time from i to j when arriving at j at time T, defined as the inverse of $A_{ij}(t)$. Observe that since A_{ij} is a piecewise linear function so is its inverse. The breakpoints of $D_{ij}(T)$ are $\beta_{ij1} + \tau_{ij}(\beta_{ij1}), \ldots, \beta_{ijm+1} + \tau_{ij}(\beta_{ijm+1})$. Note that because the travel time function is strictly increasing the order of the breakpoints does not change, i.e., $\beta_{ijk} + \tau_{ij}(\beta_{ijk}) \leq \beta_{ijk+1} + \tau_{ij}(\beta_{ijk+1})$ for all $k \in \{1, \ldots, m\}$.

For departure time $t \in [\beta_{ijk}, \beta_{ijk+1}]$, for some $k \in \{1, \ldots, m\}$, we have that the arrival time $T = A_{ij}(t) = \alpha_{ijk} + (1 + \gamma_{ijk})t \in [\beta_{ijk} + \tau_{ij}(\beta_{ijk}), \beta_{ijk+1} + \tau_{ij}(\beta_{ijk+1})]$. From this it follows, since we assume $\gamma_{ijk} > -1$, that the departure time t when arriving at $T \in [\beta_{ijk} + \tau_{ij}(\beta_{ijk}), \beta_{ijk+1} + \tau_{ij}(\beta_{ijk+1})]$ is given by

$$D_{ij}(T) = \frac{T - \alpha_{ijk}}{1 + \gamma_{ijk}}.$$

2.2. Mixed-Integer Linear Programming Formulation

We provide a mixed-integer linear programming formulation for the TWAVRP with time-dependent travel times. To do so, we introduce an auxiliary graph $\hat{G} = (N, \hat{A})$, where $\hat{A}$ contains a copy of each arc $a \in A$ for every line piece of the travel time function, $\hat{A} = \{(i, j, k) \mid (i, j) \in A, 1 \leq k \leq L_{ij}\}$. Travel along arc (i, j, k) corresponds to travel along arc $(i, j) \in A$ while the travel time is determined by line piece l_{ijk} of $\tau_{ij}(t)$. This means (i, j, k) can only be used when travel starts at $t \in [\beta_{ijk}, \beta_{ijk+1}]$ and travel time is given by $\tau_{ij}(t) = \alpha_{ijk} + \gamma_{ijk}t$. Observe that any feasible route (P, t) can be represented in $\hat{G}$, and similarly any route in $\hat{G}$ can be represented in G.

Furthermore, we introduce additional parameters. Let a_r^v be the number of times customer $i \in N'$ is visited by route r and let b_r^{ijk} be the number of times arc $(i, j, k) \in \hat{A}$ is used by route r.

Finally, let the time window variable y_i be the start time of the endogenous time window at location $i \in N'$. Note that $y_i \in [s_i, e_i - w_i]$ and we assume $s_i \leq e_i - w_i$. Let the binary route variable x_r^ω indicate whether route r is used for scenario ω. Note that as time is continuous, the number of variables x_r^ω is infinite, unless $w_i = e_i - s_i$

for all $i \in N$. Finally, let the binary arc flow variable θ^{ω}_{ijk} indicate whether arc $(i,j,k) \in \hat{A}$ is used in scenario ω. The TWAVRP with time-dependent travel times can be formulated using the following mixed-integer linear program:

$$\min \sum_{\omega \in \Omega} p_{\omega} \sum_{r \in R(\omega)} c_r x^{\omega}_r \tag{1}$$

$$\sum_{r \in R(\omega)} a^i_r x^{\omega}_r = 1, \quad \forall i \in N', \forall \omega \in \Omega, \tag{2}$$

$$\sum_{r \in R(\omega)} t^i_r x^{\omega}_r \geq y_i, \quad \forall i \in N', \forall \omega \in \Omega, \tag{3}$$

$$\sum_{r \in R(\omega)} t^i_r x^{\omega}_r \leq y_i + w_i, \quad \forall i \in N', \forall \omega \in \Omega, \tag{4}$$

$$\sum_{r \in R(\omega)} b^{ijk}_r x^{\omega}_r = \theta^{\omega}_{ijk}, \quad \forall (i,j,k) \in \hat{A}, \forall \omega \in \Omega, \tag{5}$$

$$x^{\omega}_r \in [0,1], \quad \forall \omega \in \Omega, \forall r \in R(\omega), \tag{6}$$

$$y_i \in [s_i, e_i - w_i], \quad \forall i \in N', \tag{7}$$

$$\theta^{\omega}_{ijk} \in \{0,1\}, \quad \forall (i,j,k) \in \hat{A}, \forall \omega \in \Omega. \tag{8}$$

The objective function (1) is 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 assigned time windows. The remaining Constraints (5)–(8) represent the integrality conditions on the arc flow in $\hat{A}$ represented in each scenario by the route variables x, and specify the domains of the decision variables.

Constraints (5) and (8) allow a single route in a solution to be represented by a convex combination of several routes, only if every arc in $\hat{A}$ is selected binary. This enables us to drop the integrality conditions on the route variables as is done in (6). More precisely, including (5) does not only ensure integer arc flow in A, corresponding to noncyclic routes satisfying the capacity and exogenous time window constraints, but also ensures that when traveling from i to j the travel time is determined by a single line piece l_{ijk} of $\tau_{ijk}(t)$. Indeed, when $\theta^{\omega}_{ijk} = 1$ the convex combination of routes yield $\sum_{r \in R(\omega)} b^{ijk}_r x^{\omega}_r = 1$. This means that for any fractionally selected route traversing (i,j) it holds that $t^i_r \in [\beta_{ijk}, \beta_{ijk+1}]$. The time of service of i on the route r^* corresponding to the convex combination of routes is given by $t^i_{r^*} = \sum_{r \in R(\omega)} t^i_r x^{\omega}_r$ and it holds that $t^i_{r^*} \in [\beta_{ijk}, \beta_{ijk+1}]$. Therefore, the travel time on (i,j) by the route r^* should be $\tau_{ij}(t^i_{r^*}) = \alpha_{ijk} + \gamma_{ijk} t^i_{r^*}$. We conclude by showing that this is indeed the case for this convex combination of routes

$$\begin{aligned}\sum_{r \in R(\omega)} \tau_{ij}(t^i_r) x^{\omega}_r &= \sum_{r \in R(\omega)} (\alpha_{ijk} + \gamma_{ijk} t^i_r) x^{\omega}_r \\ &= \alpha_{ijk} + \gamma_{ijk} \sum_{r \in R(\omega)} t^i_r x^{\omega}_r \\ &= \alpha_{ijk} + \gamma_{ijk} t^i_{r^*}.\end{aligned}$$

We emphasize that the formulation is incorrect when merely imposing integrality on the arcs in A instead of $\hat{A}$. In this case, convex combinations of routes with low travel times might be used to represent a route with low travel time at a moment when travel time is high, e.g., combining two routes before and after congestion to represent travel during congestion.

3. Column Generation

To find lower bounds to the TWAVRP with time-dependent travel times, we use a column generation algorithm to solve the linear programming (LP) relaxation of (1)–(8). Initially only a subset of all routing variables are included in the formulation, and new routing variables with negative reduced costs are identified by solving a pricing problem. In this case, we decompose the pricing problem into several problems, one for each scenario. For scenario ω, the pricing problem is to find a feasible route $(P, \bar{t})$, with minimum reduced cost. Let us denote the dual variables corresponding to (2)–(5) by λ, μ, ν, and ξ, respectively. For ease of notation, let $\pi = \nu - \mu$. Observe that λ, π, and ξ are all unrestricted. The reduced cost corresponding to route variable x^{ω}_r is given by

$$p_{\omega} c_r - \sum_{i \in N'} \lambda^{\omega}_i a^i_r - \sum_{i \in N'} \pi^{\omega}_i t^i_r - \sum_{(i,j,k) \in \hat{A}} \xi^{\omega}_{ijk} b^{ijk}_r. \tag{9}$$

We model the pricing problem for scenario ω using the auxiliary graph $\hat{G}$. With each node $i \in N'$ we associate demand d^{ω}_i, time window $[s_i, e_i]$, and the time of service cost coefficient $-\pi^{\omega}_i$. With node 0 we associate the cost coefficient $-p_{\omega} c^h$ and with node $n+1$ we associate the cost coefficient $p_{\omega} c^h$. Furthermore, with each arc $(i,j,k) \in \hat{A}$ we associate the time window $[\beta_{ijk}, \beta_{ijk+1}]$ corresponding to the breakpoints of line piece k of $\tau_{ij}(t)$. Arc (i,j,k) can only be used if the time of service t at i is such that $t \in [\beta_{ijk}, \beta_{ijk+1}]$. Furthermore, we associate with each arc (i,j,k) the time-dependent travel time function $\tau_{ij}(t) = \alpha_{ijk} + \gamma_{ijk} t$. Moreover, we associate with each arc the fixed costs $p_{\omega} c^f_{ij} - \lambda^{\omega}_j - \xi^{\omega}_{ijk}$ if $j \in N'$ and $p_{\omega} c^f_{ij} - \xi^{\omega}_{ijk}$ otherwise.

For each route $(P, \bar{t})$ we calculate the corresponding reduced cost in scenario ω as the sum of the costs of the arcs on path P and at each node the costs that are linear in the time of service, which includes the weighted hourly costs $p^{\omega} c^h$ times the route duration. The pricing problem is solved by finding a shortest path in $\hat{G}$ with a capacity constraint, time-dependent travel times, time window constraints (these are the exogenous time windows on the nodes and the breakpoint induced time windows on the arcs), and linear node costs. To our knowledge this problem has not been studied in the scientific literature, although it is quite similar to the shortest path problem with time windows and linear node costs in an acyclic graph, introduced by Ioachim et al. (1998).

3.1. Pricing Algorithm

To solve the pricing problem, we suggest using the following labeling algorithm. In this algorithm each label L corresponds to a partial path in $\hat{G}$ starting at the depot and ending at node $N(L)$, with time of service at $N(L)$ in $[s_L, e_L]$. With each label L we associate a load $q(L)$ defined as the sum of demand of all locations on the partial path. Similar to the algorithm by Feillet et al. (2004), we associate with each label L the binary parameters $f_j(L)$, $j \in N'$, 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 customer j as this would violate the capacity constraint. To this end, we define the function $F_j^\omega(L)$ that takes value 1 if $q(L) + d_j^\omega > Q$ indicating that L cannot be extended to j without violating the capacity constraint, and 0 otherwise.

Finally, associated with each label L is the reduced cost function $g_L(T)$ providing the minimal reduced cost of the partial path represented by L such that the time of service at $N(L)$ is in $[s_L, T]$, for $T \in [s_L, e_L]$. In our algorithm we only generate labels for which $g_L(T)$ is linear in T. Therefore, we can represent the function $g_L(T)$ by its intercept α_L and slope γ_L. To be more precise, $g_L(T)$ provides the minimal reduced cost of the partial path represented by L where the time of service at $N(L)$ is $T \in [s_L, e_L]$ if $\gamma_L < 0$, and where the time of service at $N(L)$ is precisely s_L if $\gamma_L = 0$. We do not generate labels where $\gamma_L > 0$. This is similar to the cost functions used in the algorithm proposed by Ioachim et al. (1998) to solve the shortest path problem with time windows and linear node costs in an acyclic graph.

3.2. Calculating Reduced Costs

Observe that the minimum costs of a partial path only containing the starting depot is (i) undefined for $t \le s_0$, (ii) $-p_\omega c^h t$ for $t \in [s_0, e_0]$, and (iii) $-p_\omega c^h e_0$ for $t \in [e_0, e_{n+1}]$ corresponding with service at time e_0 and waiting before departure. Note that we use the end of the time window of the ending depot, e_{n+1}, as the time horizon of the pricing problem. Hence, we can represent the cost function at the starting depot by at most two line pieces, in particular using two labels as described above. Note that if $e_0 \ge e_{n+1}$ only one label suffices. Next, we show that extending a label with linear costs to the next customer along an arc $(i, j, k) \in \hat{A}$ results in new labels with linear costs, which together represent the minimum costs of the extended partial path. This new cost function can be determined recursively from the label to be extended. By induction, this shows that the costs of every partial path can be represented by labels with linear costs.

Consider the label L corresponding to a partial path ending at location i. Furthermore, let the corresponding cost function be $g_L(T) = \alpha_L + \gamma_L T$ for all $T \in [s_L, e_L]$. Assume that line piece $l_{ijk} = \{\beta_{ijk}, \beta_{ijk+1}, \alpha_{ijk}, \gamma_{ijk}\}$ of $\tau_{ij}(t)$ is such that $[s, e] = [b_{ijk}, b_{ijk+1}] \cap [s_L, e_L]$ is not empty, i.e., time allows label L to be extended along arc $(i, j, k) \in \hat{A}$. We extend the label to customer j along arc $(i, j, k) \in \hat{A}$ and construct the corresponding cost function. Constructing the cost function when extending a label to the end depot $n + 1$ instead of to a customer is done in a similar way and is not presented for the sake of brevity.

First, let us consider the case where there exists a departure time in $[s, e]$ such that j can be reached from i within the time window of j, i.e., $[s', e'] = [s + \tau_{ij}(s), e + \tau_{ij}(e)] \cap [s_j, e_j] \neq \emptyset$. We will later return to the case where we can only arrive before the start of the time window or after the end. Denote the extended label on the interval $[s', e']$ by L'. The cost function $g_{L'}(T)$, for $T \in [s', e']$, is calculated recursively as follows:

$$s_{L'} = \max\{s_j, s + \tau_{ij}(s)\}, \tag{10}$$

$$e_{L'} = \begin{cases} e_{n+1} & \text{if } \gamma_L/(1+\gamma_{ijk}) - \pi_j^\omega \ge 0 \\ \min\{e_j, e + \tau_{ij}(e)\} & \text{otherwise,} \end{cases} \tag{11}$$

$$\alpha_{L'} = \begin{cases} \alpha_L + p_\omega c_{ij}^f - \lambda_j^\omega - \xi_{ijk}^\omega - \dfrac{\gamma_L \gamma_{ijk}}{1+\gamma_{ijk}} + \left(\dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi_j^\omega\right) s_{L'} \\ \qquad \text{if } \gamma_L/(1+\gamma_{ijk}) - \pi_j^\omega \ge 0 \\ \alpha_L + p_\omega c_{ij}^f - \lambda_j^\omega - \xi_{ijk}^\omega - \dfrac{\gamma_L \alpha_{ijk}}{1+\gamma_{ijk}} \\ \qquad \text{otherwise,} \end{cases} \tag{12}$$

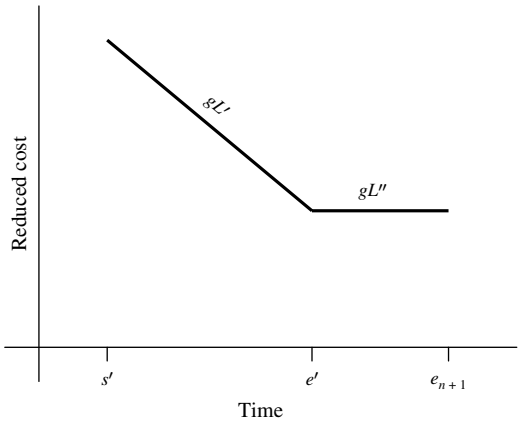
$$\gamma_{L'} = \begin{cases} 0 & \text{if } \gamma_L/(1+\gamma_{ijk}) - \pi_j^\omega \ge 0 \\ \dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi_j^\omega & \text{otherwise.} \end{cases} \tag{13}$$

Additionally, a label L'' is constructed such that $[s_{L''}, e_{L''}] = [e', e_{n+1}]$. The cost function of this label is constant and corresponds to the minimal costs of servicing j within $[s', e']$ and waiting with departure from j until $T \in [e', e_{n+1}]$. The cost function is given by $g_{L''}(T) = g_{L'}(e')$, for $T \in [e', e_{n+1}]$. Figure 1 shows an example of the reduced cost after extending label L as described above.

Now consider extending the label L while $[s', e'] = [s + \tau_{ij}(s), e + \tau_{ij}(e)] \cap [s_j, e_j] = \emptyset$. If $s + \tau_{ij}(s) > e_j$, the new customer cannot be reached in time and no new label is created. Otherwise $e + \tau_{ij}(e) < s_j$, that is, the latest arrival at j is before the start of the time window and the costs we compute next correspond to arriving before the time window and waiting with service until the start of the time window. We construct a new label L' for the interval $[s_j, e_j]$ with cost function $g_L(T)$, for $T \in [s_j, e_j]$, defined as follows:

$$\begin{aligned} g_{L'}(T) &= \min_{t \in [s_j, T]} \{g_L(e) + p_\omega c_{ij}^f - \lambda_j^\omega - \xi_{ijk}^\omega - \pi_j^\omega t\} \\ &= \begin{cases} g_L(e) + p_\omega c_{ij}^f - \lambda_j^\omega - \xi_{ijk}^\omega - \pi_j^\omega s_j & \text{if } \pi_j^\omega \le 0 \\ g_L(e) + p_\omega c_{ij}^f - \lambda_j^\omega - \xi_{ijk}^\omega - \pi_j^\omega T & \text{otherwise.} \end{cases} \end{aligned}$$

Figure 1. Example of the Reduced Cost After Extending Label L



Also, in this case, an additional label L'' is constructed such that $[s_{L''}, e_{L''}] = [e_j, e_{n+1}]$. The cost function of this label corresponds to the minimal costs of servicing j within $[s_j, e_j]$ and waiting with departure from j until $T \in [e_j, e_{n+1}]$. The cost function is given by $g_{L''}(T) = g_{L'}(e_j)$, for $T \in [e_j, e_{n+1}]$.

If $\tilde{s}$ is the earliest possible time of service at location j, observe that using the above described procedure to extend labels from i to j along every arc $(i, j, k) \in \hat{A}$, there exists by construction for every time $T \in [\tilde{s}, e_{n+1}]$ a label L' such that $T \in [s_{L'}, e_{L'}]$, which provides the minimal costs of servicing customer j before or at time T.

3.3. Dominance

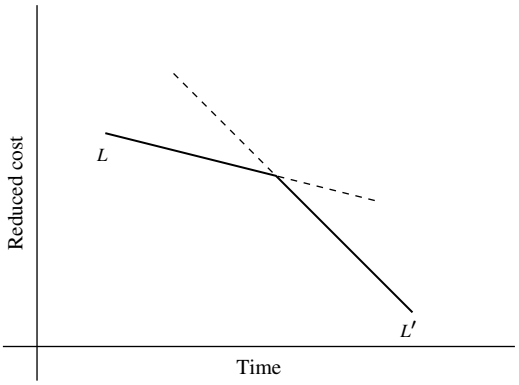
To limit the number of labels generated by the labeling algorithm, we use a dominance procedure to maintain only Pareto optimal labels. The procedure, which is similar to the dominance procedure proposed by Ioachim et al. (1998), is outlined as follows. When evaluating dominance of two labels L and L' such that $[S, E] = [s_L, e_L] \cap [s_{L'}, e_{L'}] = [\max\{s_L, s_{L'}\}, \min\{e_L, e_{L'}\}]$, we take the piecewise minimum of the two linear functions represented by L and L'. Figure 2 illustrates this, where the dashed lines indicate the parts of the labels that are dominated in terms of reduced cost. In particular, we compute $[s, e] \in [S, E]$ for which the costs of L is lower than the costs of L' and remove this interval from the label L'. Next, we formally define the dominance procedure.

The label L dominates label L' on $[s, e] \in [S, E]$ if

$$\begin{aligned} N(L) &= N(L'), \\ q(L) &\le q(L'), \\ f_j(L) &\le f_j(L') \quad \forall\, j \in N', \\ g_L(T) &\le g_{L'}(T) \quad \forall\, T \in [s, e]. \end{aligned}$$

Clearly, if L and L' satisfy the above conditions, any feasible extension of the partial path represented by L' is also feasible for the partial path represented by L,

Figure 2. Example of Dominance Between L and L'



while the total reduced cost is lower for every departure time $T \in [s, e]$.

To find the maximal interval $[s, e] \in [S, E]$ for which L has lower costs than L' we use the following procedure. Let $g_L(T) = \alpha_L + \gamma_L T$ and $g_{L'}(T) = \alpha_{L'} + \gamma_{L'} T$. If $\gamma_L = \gamma_{L'}$ then the costs of one label are always higher than the other. In particular, $[s, e] = [S, E]$ if $\alpha_L \le \alpha_{L'}$, and $[s, e] = \emptyset$ otherwise.

If $\gamma_L \neq \gamma_{L'}$ the linear functions $g_L(T)$ and $g_{L'}(T)$ intersect at time T^* given by

$$T^* = \frac{\alpha_{L'} - \alpha_L}{\gamma_L - \gamma_{L'}}.$$

Now we find $[s, e]$ as follows. If $\gamma_L > \gamma_{L'}$ then L has lower costs than L' on the interval $[s, e] = [S, \min\{E, T^*\}]$, otherwise on $[s, e] = [\max\{S, T^*\}, E]$. Furthermore, we conclude that L' has lower costs than L on the interval $[s', e'] = [S, E] \backslash [s, e]$.

Finally, if L dominates L' on $[s, e]$, we modify L' by removing $[s, e]$ from the interval $[s_{L'}, e_{L'}]$. In the case that the interval is split in two as a result of removing $[s, e]$, instead of modifying L', two new labels are constructed corresponding to either interval.

3.4. Remarks to Reduce the Number of Generated Labels

Next, we provide two remarks, which we use to reduce the number of generated labels. They highlight special cases of labels that will be dominated and therefore need not be generated.

Remark 1. When extending a label L from i to j along arc $(i, j, k) \in \hat{A}$ from the interval $[s, e] = [s_L, e_L] \cap [\beta_{ijk}, \beta_{ijk+1}]$ such that $e + \tau_{ij}(e) < s_j$, we create a label L' for the interval $[s_j, e_{n+1}]$, providing the costs of servicing i at some time in $[s_L, e_L]$, traveling to j and waiting with service at j until at least s_j.

Suppose breakpoint β_{ijk+1} of $\tau_{ij}(t)$ is such that $\beta_{ijk+1} \le e_L$ and therefore L can also be extended along arc $(i, j, k+1)$ from the interval $[\bar{s}, \bar{e}] = [s_L, e_L] \cap [b_{ijk+1}, b_{ijk+2}]$ to create $\bar{L}'$. In the case $\bar{e} + \tau_{ij}(\bar{e}) < s_j$ it is easily verified that L' is dominated by $\bar{L}'$ if $g_L(e) - \xi^{\omega}_{ijk} \ge$

$g_L(\bar{e}) - \xi^{\omega}_{ijk+1}$ in which case we do not generate L'. Otherwise, $s_j \in [\bar{s} + \tau_{ij}(\bar{s}), \bar{e} + \tau_{ij}(\bar{e})]$ in which case L' is also dominated by $\bar{L}'$ if $g_L(e) - \xi^{\omega}_{ijk} \geq g_L(D_{ij}(s_j)) - \xi^{\omega}_{ijk+1}$ and we similarly do not generate L'.

Remark 2. When extending a label L from i to j along arc $(i, j, k) \in \hat{A}$ we create a label L' for the interval $[s', e']$ and an additional label L'' for the interval $[e', e_{n+1}]$. In case the slope of $g_{L'}(T)$ is zero, just like the slope of $g_{L''}(T)$, we can merge labels L' and L'' since they have the same costs. To be more precise, we do not generate L'' but generate L' with as interval $[s', e_{n+1}]$.

3.5. Extension Procedure

Using Remarks 1 and 2, we provide an efficient extension procedure to extend a label L such that $N(L) = i$ to customer j, creating a collection of new labels. Represent $g_L(T)$ by its intercept α_L and slope γ_L. First, we consider all line pieces along which the next location is at the latest reached prior to the start of its time window. Let l_A be the first and l_B be the last line piece $l_{ijk} = \{\beta_{ijk}, \beta_{ijk+1}, \alpha_{ijk}, \gamma_{ijk}\}$ of $\tau_{ij}(t)$ with $[s, e] = [b_{ijk}, b_{ijk+1}] \cap [s_L, e_L]$ such that $e + \tau_{ij}(e) < s_j$. Select $l_K \in \arg\min_{A \leq k \leq B}\{g_L(e) - \xi^{\omega}_{ijk}\}$ as the line piece between l_A and l_B that yields the dominating extended label.

Let l_{B+1} be the first line piece along which location j can be reached at s_j. In correspondence with Remark 1, we extend label L only along line piece l_K provided that $g_L(e_K) - \xi^{\omega}_{ijK} < g_L(D_{ij}(s_j)) - \xi^{\omega}_{ijB+1}$. In this case, we use the following extension functions, where we consider Remark 2 to determine the label end time:

$$N(L') = j, \tag{14}$$

$$q(L') = q(L) + d^{\omega}_j, \tag{15}$$

$$f_v(L') = \begin{cases} 1 & \text{if } v = j \\ \max\{f_v(L), F^{\omega}_v(L')\} & \text{otherwise} \end{cases} \quad \forall v \in N', \tag{16}$$

$$s_{L'} = s_j, \tag{17}$$

$$e_{L'} = \begin{cases} e_{n+1} & \text{if } -\pi^{\omega}_j \geq 0 \\ e_j & \text{otherwise,} \end{cases} \tag{18}$$

$$\alpha_{L'} = \begin{cases} \alpha_L + \gamma_L e_K + p_{\omega} c^f_{ij} - \lambda^{\omega}_j - \xi^{\omega}_{ijK} - \pi^{\omega}_j s_j \\ \quad \text{if } -\pi^{\omega}_j \geq 0 \\ \alpha_L + \gamma_L e_K + p_{\omega} c^f_{ij} - \lambda^{\omega}_j - \xi^{\omega}_{ijK} \\ \quad \text{otherwise,} \end{cases} \tag{19}$$

$$\gamma_{L'} = \begin{cases} 0 & \text{if } -\pi^{\omega}_j \geq 0 \\ -\pi^{\omega}_j & \text{otherwise.} \end{cases} \tag{20}$$

Furthermore, if $e_{L'} < e_{n+1}$ also construct the additional label L'' with the extension functions (14)–(16), $\alpha_{L'} = g_L(e)$, $\gamma_{L''} = 0$, $s_{L''} = e_{L'}$, and $e_{L''} = e_{n+1}$.

Next, we extend label L along the remaining line pieces $l_{ijk} = \{\beta_{ijk}, \beta_{ijk+1}, \alpha_{ijk}, \gamma_{ijk}\}$ of $\tau_{ij}(t)$ for which arrival at j is possible after the start s_j of the time window and before the end e_j. That is, we consider the line pieces such that for $[s, e] = [\beta_{ijk}, \beta_{ijk+1}] \cap [s_L, e_L]$ it holds that (i) $[s, e]$ is not empty, (ii) $s_j \leq e + \tau_{ij}(e)$, and (iii) $s + \tau_{ij}(s) \leq e_j$. We construct a new label L' using the extension functions (14)–(16) and

$$s_{L'} = \max\{s_j, s + \tau_{ij}(s)\}, \tag{21}$$

$$e_{L'} = \begin{cases} e_{n+1} & \text{if } \dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi^{\omega}_j \geq 0 \\ \min\{e_j, e + \tau_{ij}(e)\} & \text{otherwise,} \end{cases} \tag{22}$$

$$\alpha_{L'} = \begin{cases} \alpha_L + p_{\omega} c^f_{ij} - \lambda^{\omega}_j - \xi^{\omega}_{ijk} - \dfrac{\gamma_L \gamma_{ijk}}{1+\gamma_{ijk}} \\ \quad + \left(\dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi^{\omega}_j\right) s_{L'} \\ \quad \text{if } \dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi^{\omega}_j \geq 0 \\ \alpha_L + p_{\omega} c^f_{ij} - \lambda^{\omega}_j - \xi^{\omega}_{ijk} - \dfrac{\gamma_L \alpha_{ijk}}{1+\gamma_{ijk}} \\ \quad \text{otherwise,} \end{cases} \tag{23}$$

$$\gamma_{L'} = \begin{cases} 0 & \text{if } \dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi^{\omega}_j \geq 0 \\ \dfrac{\gamma_L}{1+\gamma_{ijk}} - \pi^{\omega}_j & \text{otherwise.} \end{cases} \tag{24}$$

Furthermore, if $e_{L'} < e_{n+1}$ also construct the additional label L'' with the extension functions (14)–(16), $\alpha_{L'} = g_L(e)$, $\gamma_{L''} = 0$, $s_{L''} = e_{L'}$, and $e_{L''} = e_{n+1}$.

3.6. Labeling Algorithm

To summarize the labeling algorithm, let $\text{EXTEND}_j(L)$ denote the extension operator described in Section 3.5 providing a collection $\mathcal{L}$ of labels extended to j. Furthermore, let $\text{DOMINATE}(L, L')$ denote the dominance procedure described in Section 3.3 providing a collection $\mathcal{L}$ of dominating labels. We initialize the algorithm with two labels describing the reduced cost at the starting depot, as described in Section 3.2. We refer to these labels as L_1 and L_2. Recall that in case $e_0 \geq e_{n+1}$, actually only one label will be initialized. The labeling algorithm is summarized in Algorithm 1.

Algorithm 1 (Labeling algorithm)

```
Initialize the collection of labels Λ = {L_1, L_2}.
repeat
    Select an unprocessed label L ∈ Λ.
    for all j ∈ N such that f_j(L) = 0 do
        L = EXTEND_j(L).
        for all L' ∈ Λ such that N(L') = j do
            for all x ∈ L do
                D = DOMINATE(x, L').
                Replace x ∈ L with the labels in D
                    originally corresponding to x.
```

10: Replace $L' \in \Lambda$ with the labels in D originally corresponding to L'.
11: **end for**
12: **end for**
13: Add $\mathcal{L}$ to Λ.
14: **end for**
15: **until** No unprocessed labels remain.

3.7. *ng*-Path Relaxation

In the formulation, we might include routes $(P, \bar{t})$ for which P is cyclic, instead of only including routes for which P is elementary. These routes will never appear in an optimal integer solution as every customer is visited exactly once. However the LP bound becomes weaker. On the other hand, the pricing problem becomes easier to solve as elementarity is relaxed.

We apply *ng*-path relaxation, allowing the possibly cyclic *ng*-paths, which were introduced by Baldacci, Mingozzi, and Roberti (2011). To apply this route relaxation, a neighborhood $N_i \subseteq N'$ is introduced for each customer $i \in N'$. An *ng*-path is a path in which a customer i can be visited more than once only if between visits it also visits at least one other customer j such that $i \notin N_j$. That is, returning to the same customer is allowed if the cycle first leaves the neighborhood. Similar to Baldacci, Mingozzi, and Roberti (2011) and Ribeiro, Desaulniers, and Desrosiers (2012) all neighborhoods are of the same size Δ_{ng}. In each neighborhood we include the Δ_{ng} customers j for which the fixed travel costs c_{ij}^f are the lowest.

Our labeling algorithm is modified to find a shortest *ng*-path instead of an elementary path to solve the pricing problem. In particular, this is accomplished by replacing (16) used for updating the location resources of the label by

$$f_i(L') = \begin{cases} 1 & \text{if } i = j, \\ \max\{f_i(L), F_i^{\omega}(L')\} & \\ \quad \text{if } i \in N_j \backslash \{j\}, & \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in N'. \tag{25}$$

Note that now when extending label L to customer j to create label L', $f_i(L')$ is set to 0 even when $f_i(L) = 1$ in case i is not in the neighborhood N_j of customer j. Computational gains are achieved first because when checking dominance only the resources $f_i(L')$ have to be considered for $i \in N_j$, because all other values $f_k(L')$ are zero anyway. More importantly, significantly more labels are typically dominated. Observe that a low value of Δ_{ng} yields a fast labeling algorithm and a weaker LP bound while a high value of Δ_{ng} yields a slower labeling algorithm but a stronger LP bound.

3.8. Reusing Routes

Similar to Spliet and Gabor (2015), in each iteration of the column generation algorithm the pricing problems are solved iteratively per scenario. For each identified route with negative reduced cost that is added to the formulation for one scenario, we check whether this route is also feasible and has a negative reduced cost for the other remaining scenarios. If such a route is found, it is added to the formulation for that scenario as well, and the pricing algorithm is no longer used to solve the pricing problem for that scenario in the current iteration.

4. Tabu Search Pricing Heuristic

We propose a tabu search algorithm as a pricing heuristic used to find negative reduced cost columns at each iteration of the column generation algorithm. Applying the pricing heuristic to solve the pricing problem, and only using the exact labeling procedure when the heuristic fails to identify a column with negative reduced cost, potentially speeds up the column generation algorithm. Recall that we model the pricing problem as a shortest path problem in $\hat{G}$ with a capacity constraint, time-dependent travel times, time window constraints on the nodes and arcs, and linear node costs. To our knowledge, this problem has not yet been studied in the scientific literature, so no heuristic is readily available to us. Note that tabu search heuristics have been successfully applied to solve related shortest path problems in a column generation setting. We modify the tabu search heuristic implemented by Spliet and Desaulniers (2015) for a shortest path problem with capacity and time window constraints, to suit our needs. Similar to Spliet and Desaulniers (2015), we only consider elementary routes in the tabu search algorithm and do not maintain ng-routes.

4.1. Tabu Search Neighborhoods

The tabu search heuristic is initialized with an elementary route. Next, we iteratively replace this route by a route in its neighborhood. We define the neighborhood of each route as all other feasible routes that can be obtained by a single move. We consider two types of moves: adding a customer to the route and removing a customer from the route. Next, we define these moves more precisely.

A route is represented as a pair $(P, \bar{t})$, where P is a path in $\hat{G}$ and $\bar{t}$ are the corresponding times of service at each location on the path. Since $\hat{G}$ may contain multiple arcs connecting two nodes, inserting or removing a customer can be done in multiple ways. As an insertion move, we consider removing arc (i, j, k) from P and replacing this with two arcs (i, m, k') and (m, j, k''), followed by the (re)optimization of the times of service. Such a move corresponds with one way to insert customer m in between i and j. Similarly, as a deletion move, we consider removing arcs (i, m, k') and (m, j, k'') from P and inserting an arc (i, j, k), followed by the (re)optimization of the times of service. This

move corresponds to one way to remove customer m from the route. Next, we discuss the optimization of times of service for a given path in $\hat{G}$.

4.2. Optimizing Times of Service

Optimizing the time of service for a given path P in $\hat{G}$, requires choosing a time of service $\bar{t}_i$ for all locations i along path P. The time of service $\bar{t}_i$ should satisfy the exogenous time window $[s_i, e_i]$, as well as the time window $[b_{ijk}, b_{ijk+1}]$ imposed by each arc (i, j, k) on path P. Furthermore, the travel time along arc (i, j, k) is linear and given by $\alpha_{ijk} + \gamma_{ijk}\bar{t}_i$. Finally, costs are incurred at each location, which are linear in the time of service. For the moment, we can omit the remaining parts of the reduced cost pertaining to the choice of path P.

Next, we demonstrate that this problem is equivalent to the version without time windows on the arcs and constant travel times. This allows the use of the polynomial time algorithm of Dumas, Soumis, and Desrosiers (1990) to optimize the times of service. Their algorithm is designed for the latter problem, including convex cost functions of time of service at each node in general, and linear cost functions in particular.

First, observe that the two time windows on the time of service at node i along path P can be represented by a single time window. Simply impose $\max\{s_i, b_{ijk}\} \leq \bar{t}_i \leq \min\{e_i, b_{ijk+1}\}$ for all (i, j, k) on P, and $s_{n+1} \leq \bar{t}_{n+1} \leq e_{n+1}$.

Finally, transforming the time of service variables by an appropriate substitution allows us to reduce the linear travel times to constant travel times. To illustrate this, let us first simplify the notation. Let N_P be the number of locations along path P. For $i \in \{1, \ldots, N_P\}$ denote by $\tilde{t}_i$ the time of service at the ith location along path P. Similarly, we denote the corresponding single time window by $[\tilde{s}_i, \tilde{e}_i]$, the linear node costs by $\tilde{\pi}_i$, and the linear travel time function from the ith to the $(i+1)$th location by $\tilde{a}_i + \tilde{\gamma}_i\tilde{t}_i$.

Observe that the time of service $\tilde{t}_{i+1}$ should be later than the arrival time at the $(i+1)$th location, hence

$$\tilde{a}_i + (\tilde{\gamma}_i + 1)\tilde{t}_i \leq \tilde{t}_{i+1}.$$

Next, denote by $\Gamma_i = 1/(\prod_{j=1}^{i-1}(\tilde{\gamma}_j + 1))$, and note that by convention $\Gamma_1 = 1$. Consider the substitution $\tilde{t}'_i = \Gamma_i\tilde{t}_i$ for all $i \in \{1, \ldots, N_P\}$. By induction, it follows that

$$\Gamma_{i+1}\tilde{a}_i + \tilde{t}'_i \leq \tilde{t}'_{i+1}.$$

This shows that after substitution of the time of service variables, the travel time from the ith to the $(i+1)$th location is constant and given by $\Gamma_{i+1}\tilde{a}_i$.

We conclude that the problem of optimizing the time of service is equivalent to optimizing the time of service with constant travel times, single time windows $[\Gamma_i\tilde{s}_i, \Gamma_i\tilde{e}_i]$, and linear node costs $\Gamma_i\tilde{\pi}_i$ for all $i \in \{1, \ldots, N_P\}$. Therefore, the polynomial time algorithm by Dumas, Soumis, and Desrosiers (1990) can be used to optimize the time of service.

4.3. Tabu Search Algorithm

The tabu search algorithm initializes with a route that is selected with a positive value in the current solution to the LP relaxation. In each iteration of the algorithm, all neighbors are evaluated and the neighboring route with the lowest reduced cost is selected to replace the current route. The new route might have higher costs than the old route. To avoid cycling, we make the inverse move tabu for TS_{tabu} iterations. As inverse move corresponding with the insertion of customer m, we consider all moves that delete customer m. Similarly, as inverse move corresponding with the deletion of customer m, we consider all moves that insert customer m. Every time a route has been found with negative reduced cost, it is added to the relaxed master problem.

To diversify the search, after every TS_{div} iterations the algorithm reinitializes by selecting a new route for which the corresponding route variable has a positive value in the current solution to the LP relaxation. Both at initialization and at reinitialization, if the selected route is cyclic, we remove all visits to a customer except the first to obtain an elementary route. Removing a customer is done by removing the two incident arcs and adding a new arc corresponding to the earliest feasible departure time. Note that because the triangle inequality might not be satisfied for travel time, removing a customer might result in an infeasible route, although this is highly unlikely in the instances we consider. We check whether the obtained route has feasible times of service, if not we reinitialize with the next route.

The algorithm terminates when all positively selected routes in the current LP solution have been used to reinitialize the tabu search, or when a total of $TS_{\max}$ routes have been added to the relaxed master problem during this run of the tabu search algorithm. To limit computation time, we start by assessing all neighbors found by adding or removing a location at the end of the route, and iteratively continue to the start of the route. This way, the optimal times of service of the locations at the start of the route are maintained and only the times of service of later locations require optimization.

5. Valid Inequalities

To strengthen the LP bound, we suggest using valid inequalities. For the vehicle routing problem with time windows, many valid inequalities have been studied: for example, capacity, comb, hypotour, and multistar inequalities (Lysgaard, Letchford, and Eglese 2004), k-path inequalities (Kohl et al. 1999), and subset row inequalities (Jepsen et al. 2008). These inequalities are also applicable to each scenario in the TWAVRP with time-dependent travel times. In our implementation we use the capacity inequalities, which are presented next. Furthermore, we present a novel set of valid

inequalities specifically designed for the TWAVRP with time-dependent travel times, referred to as arc-synchronization inequalities.

5.1. Capacity Inequalities

Let $\theta_{ij}^{\omega} = \sum_{k=1}^{L_{ij}} \theta_{ijk}^{\omega}$ be the arc flow in G on arc (i,j) in scenario ω. Let $b(S)$ be the minimum number of vehicles needed to visit all customers in $S \subseteq V'$. The capacity inequalities are as follows:

$$\sum_{i \in S,\, j \notin S} \theta_{ij}^{\omega} \geq b(S) \quad \forall\, S \subseteq N',\, \forall\, \omega \in \Omega. \tag{26}$$

As is common, we replace $b(S)$ by the lower bound $\lceil \sum_{i \in S} d_i^{\omega} / Q \rceil$. We use the heuristic of Lysgaard, Letchford, and Eglese (2004) to separate them, more precisely, we use the implementation that can be found in the package by Lysgaard (2003). Note that including these inequalities does not affect the structure of the pricing problem.

5.2. Arc-Synchronization Inequalities

As highlighted in Section 2.2, a fractional solution might lead to biased representation of travel times, and hence a decreased value of the LP relaxation. To overcome this, we suggest the following arc-synchronization inequalities. They are based on the fact that if in scenario $\omega \in \Omega$ location $i \in N$ is departed from before time t, it is not possible in another scenario $\omega' \in \Omega$, $\omega \neq \omega'$, to depart from i later than $t + w_i$ without violating the endogenous time window constraints in at least one of the two scenarios. Although the time window constraints (3) and (4) prevent this for any integer solution, this observation can be used to strengthen the value of a fractional solution.

Let $L_{ij}^{-}(t) = \arg\max_{1 \leq k \leq L_{ij}} \{\beta_{ijk+1} \mid \beta_{ijk+1} \leq t\}$ correspond with the last line piece of $\tau_{ij}(t)$ that ends prior to time t. If for some $j \in N$ an arc $(i,j,k) \in \hat{A}$ is used in scenario ω for $k \leq L_{ij}^{-}(t)$, this implies that location i is departed from before time t. Similarly, let $L_{ji}^{+}(t) = \arg\min_{1 \leq k \leq L_{ji}} \{\beta_{jik} \mid A_{ji}(\beta_{jik}) > t + w_i\}$ correspond with the first line piece of $\tau_{ji}(t)$ for which the arrival at i is strictly later than $t + w_i$. If for some $j \in N$ an arc $(j,i,k) \in \hat{A}$ is used in scenario ω' for $k \geq L_{ji}^{+}(t)$, this implies that location i is arrived at later than $t + w_i$, which in turn implies that location i is departed from later than $t + w_i$. Finally, let B_i be a set of time points containing (1) all breakpoints β_{ijk} of τ_{ij} and (2) all arrival times at i when departing at a breakpoint of τ_{ji} minus the time window width, i.e., the times $A_{ji}(\beta_{jik}) - w_i$. The arc-synchronization constraints are the following:

$$\sum_{j \in N} \sum_{1 \leq k \leq L_{ij}^{-}(t)} \theta_{ijk}^{\omega} + \sum_{j \in N} \sum_{L_{ji}^{+}(t) \leq k \leq L_{ji}} \theta_{jik}^{\omega'} \leq 1 \quad \forall\, i \in N,\, \forall\, t \in B_i,\ \forall\, \omega, \omega' \in \Omega,\, \omega \neq \omega'. \tag{27}$$

Note that only the times $t \in B_i$ are of interest, as $L_{ij}^{-}(t)$ is constant for $t \in [\beta_{ijk}, \beta_{ijk+1})$ for all $(i,j) \in A$ and similarly $L_{ji}^{+}(t)$ is constant for $t \in (A_{ji}(\beta_{ji}) - w_i, A_{ji}(\beta_{jik+1}) - w_i]$. Therefore we need not consider all $t \in [s_i, e_i]$.

Observe that the number of arc-synchronization inequalities is $(\sum_{i,j \in N} L_{ij} + L_{ji})|\Omega|(|\Omega| - 1)$. All these valid inequalities might be added to the formulation directly, in which case no separation is required. Otherwise, checking for violated inequalities can be done in polynomial time by enumeration. Note that preliminary experiments have shown that adding all arc-synchronization inequalities to the formulation slows down the algorithm significantly. Therefore, in all experiments presented in this paper where these inequalities are used, none are added initially and violated inequalities are separated. Finally, note that including arc-synchronization inequalities does not affect the structure of the pricing problem, similar to the capacity inequalities.

6. Branch-Price-and-Cut

Next, we describe the branch-price-and-cut algorithm used to solve the TWAVRP with time-dependent travel times to optimality. Lower bounds are found by solving the LP relaxation of (1)–(8) using column generation and adding valid inequalities. When no new negative reduced cost routes are identified, violated valid inequalities are separated. We have experimented with a branch-price-and-cut algorithm in which only capacity inequalities are separated, and with an algorithm in which also violated arc-synchronization inequalities are separated when no violated capacity inequality is identified.

We apply two different branching rules. First, branching is performed on the arcs in G, that is, branching on the aggregate variables θ_{ij}^{ω}. If the arc flow on all arcs in G is integer, we branch on the arcs in $\hat{G}$ by branching on the variables θ_{ijk}^{ω}. Note that fixing $\theta_{ij}^{\omega} = 0$ during branching is achieved by fixing $\theta_{ijk}^{\omega} = 0$ for all $1 \leq k \leq L_{ij}$.

When branching on the arcs in G or $\hat{G}$, special ordered subset branching (SOS branching) is applied, similar to Spliet and Gabor (2015). We provide the details for branching on arcs in G here and omit the similar procedure for branching on arcs in $\hat{G}$ for the sake of brevity. For scenario ω and customer i, let $\delta_{\omega}^{-}(i)$ and $\delta_{\omega}^{+}(i)$ be the sets of in and out arcs of customer i, respectively. We select a customer i, a scenario ω, and an arc type $o \in \{-,+\}$ such that the number of arcs a in $\delta_{\omega}^{o}(i)$ for which $\theta_a^{\omega} > 0$ is the largest set. Let $\delta_{\omega}^{o}(i) = \{a_1, \ldots, a_k\}$ be ordered with respect to the arc flow in G, such that $\theta_{a_i}^{\omega} \geq \theta_{a_j}^{\omega}$ if $i < j$. The arcs are divided into two groups, S and its complement $\bar{S}$, where $S = \{a_1, \ldots, a_i\}$ is such that $\sum_{a \in S} \theta_a^{\omega} \geq 0.5$ and $\sum_{a \in S \setminus \{a_i\}} \theta_a^{\omega} < 0.5$. When branching, we disallow the use of the arcs in S in one branch and in the other we disallow the use of the arcs

in $\bar{S}$. Observe that the pricing problem is not altered by this branching procedure although the number of arcs to be considered decreases.

Finally, upper bounds are obtained only when the solution to the LP relaxation is integer and the node of the search tree with the lowest lower bound is selected at each iteration of the branch-price-and-cut algorithm.

7. Experiments

Next, we present the results of numerical experiments in which the presented algorithm is used to solve the TWAVRP with time-dependent travel times. We first describe how the test instances are generated. Then, the results are shown of experiments in which the column generation algorithm is used to solve the LP relaxation. Specifically, a comparison is made between the column generation with and without the new pricing heuristic. Finally, results are shown using the branch-price-and-cut algorithm to solve the TWAVRP with time-dependent travel times to optimality. Here, we emphasize the effect of adding the arc-synchronization inequalities.

All algorithms are coded in C++ and CPLEX 12.6.2 is used for solving linear programs. All experiments were performed on an Intel Core i5-2450M CPU, 2.5 GHz with 4 GB RAM. For each individual run, i.e., using either the column generation algorithm or the branch-price-and-cut algorithm on a single instance, a time limit of one hour is maintained.

7.1. Test Instances

The following procedure is used to generate a test instance, inspired by a Dutch retail chain. Customer locations are randomly generated using a uniform distribution over a square with sides of length 5. The depot is at the center of the square. The fixed travel costs c^f_{ij} between two locations $i \in N$ and $j \in N$ are set equal to the Euclidean distance. The hourly costs c^h is set to 1.

The depot has the exogenous time window $[6, 22]$. Each customer is given one of three exogenous time windows, each assigned with a fixed frequency. The exogenous time window $[10, 16]$ is given to 10% of the customers, $[8, 18]$ to 60%, and $[7, 21]$ to 30%. The endogenous time window width is set to 2 for all customers.

Three demand scenarios are generated to represent low, medium, and high demand. This is achieved by generating a realization of demand d_i for each customer $i \in N'$ from a normal distribution with mean 5 and variance 1.5. Next, this value is multiplied with a multiplier u^ω_i and rounded up, resulting in the demand of customer i in scenario ω of $d^\omega_i = \lceil u^\omega_i d_i \rceil$. For scenarios 1, 2, and 3, the customer specific multiplier is drawn from a uniform distribution on $[0.7, 0.8]$, $[0.95, 1.05]$, and $[1.2, 1.3]$, respectively. The vehicle capacity is 30.

Table 1. Speed Profile Assignment

	Small	Medium	Big
Small	Fast	Fast	Medium
Medium	Fast	Medium	Slow
Big	Medium	Slow	Slow

To create the time-dependent travel time functions, we randomly assign one of three speed profiles to each arc: slow, medium, or fast. To assign the speed profiles, we first randomly select one-third of the customers to represent customers located in big, medium, and small cities, respectively. Depending on the size of two cities, their connecting arc is assigned a speed profile. For instance, an arc between two big cities is assigned a slow profile. Table 1 shows how the speed profiles are assigned.

The speed profiles describe the speed on various times of the day. We use different constant speeds during five intervals of the day. The speed is chosen such that the speed profiles represent lower speeds during the morning and evening rush hours and higher speeds outside the rush hours. The speeds during each interval for the different speed profiles are provided in Table 2.

It is easily verified that the average speed during the exogenous time window of the depot, $[6, 22]$, is approximately 0.63. To determine the travel time, we first scale the Euclidean distance between customers by multiplying with 0.63 and rounding to two decimal places. Next, the time-dependent travel time functions can be constructed using this corrected distance and the corresponding speed profiles. The correction factor of 0.63 ensures that the average travel time corresponds roughly with the Euclidean distance.

We have generated 4 sets of 10 instances with 10, 15, 20, and 25 customers each, making a total of 40 instances.

7.2. Column Generation Experiments

Next, the results of applying the column generation algorithm to each of the 40 instances are presented. First, the results are presented of using the column generation algorithm without the tabu search pricing heuristic. We show the results in which only elementary paths are used in the formulation, in which the ng-path relaxation using the neighborhood size of $\Delta_{ng} = 5$ is applied, and all cyclic paths are allowed. Note

Table 2. Speed Profiles

Time	[0,7]	[7,10]	[10,15]	[15,19]	[19,24]
Slow	0.8	0.2	0.5	0.2	0.8
Medium	1	0.4	0.8	0.4	1
Fast	1	0.5	1	0.5	1

Table 3. Column Generation Experiment Results, Without Pricing Heuristic

		All cycles allowed				*ng*-paths with $\Delta_{ng} = 5$				Only elementary routes			
Inst.	$\|N'\|$	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP
1	10	2.59	2.40	19	23.91	4.06	3.95	19	27.98	14.63	14.49	17	27.98
2	10	0.94	0.81	15	27.36	0.98	0.87	15	31.10	3.59	3.48	14	31.10
3	10	2.23	2.03	22	33.74	1.89	1.75	18	36.52	8.00	7.86	19	36.52
4	10	1.01	0.86	18	38.69	1.23	1.03	22	41.24	4.32	4.15	20	41.24
5	10	0.41	0.28	15	37.60	0.64	0.48	18	40.55	1.31	1.15	17	40.55
6	10	0.55	0.41	17	34.66	0.81	0.67	15	39.36	1.87	1.79	18	39.36
7	10	1.17	1.04	19	28.41	1.06	0.97	15	32.20	2.85	2.70	19	32.20
8	10	0.98	0.83	16	29.20	2.67	2.47	22	30.70	9.80	9.59	19	30.89
9	10	1.37	1.26	20	23.97	2.90	2.75	19	26.01	11.00	10.89	14	26.01
10	10	0.73	0.64	15	31.31	2.23	2.03	24	34.96	5.46	5.29	20	34.97
11	15	5.87	5.50	21	44.25	15.30	14.95	20	47.25	1,285.02	1,284.52	26	47.26
12	15	7.16	6.68	27	29.98	13.62	13.25	23	32.57	1,519.66	1,519.28	24	32.65
13	15	1.48	1.22	18	33.91	5.43	4.99	26	40.12	155.30	154.89	25	40.12
14	15	4.71	4.26	29	35.83	7.80	7.42	22	40.69	338.65	338.18	26	40.71
15	15	2.32	1.98	21	40.47	4.93	4.51	26	42.43	131.28	130.81	23	42.43
16	15	3.53	3.19	22	38.16	9.42	8.91	27	41.96	258.23	257.87	21	42.03
17	15	13.21	12.76	24	43.13	15.35	14.94	24	46.71	332.74	332.35	24	46.81
18	15	2.90	2.56	20	40.94	11.76	11.26	26	45.16	275.78	275.31	25	45.17
19	15	3.68	3.29	24	47.67	7.63	7.15	28	51.43	123.18	122.78	24	51.46
20	15	3.31	2.89	23	45.26	10.87	10.17	40	48.40	330.94	330.35	34	48.40
21	20	12.99	12.11	30	43.85	31.66	30.82	30	48.32	3,600.00	—	—	—
22	20	15.07	14.10	34	56.35	41.45	40.09	39	60.72	3,600.00	—	—	—
23	20	21.58	20.59	34	45.19	115.27	113.88	44	50.57	3,600.00	—	—	—
24	20	7.32	6.58	26	55.57	28.10	27.05	34	62.00	3,600.00	—	—	—
25	20	17.57	16.55	35	50.38	43.38	42.11	42	54.70	3,600.00	—	—	—
26	20	9.08	8.33	26	46.56	24.31	23.21	36	51.79	3,600.00	—	—	—
27	20	7.52	6.78	27	54.30	17.22	16.28	33	60.70	1,665.43	1,664.59	31	60.71
28	20	14.59	13.38	38	53.13	33.04	31.87	38	58.79	3,600.00	—	—	—
29	20	14.77	13.96	28	43.78	43.32	41.92	44	46.81	3,600.00	—	—	—
30	20	12.48	11.70	26	48.97	31.17	30.14	33	52.09	3,600.00	—	—	—
31	25	47.02	44.74	47	60.16	117.28	114.85	52	65.23	3,600.00	—	—	—
32	25	46.01	43.84	46	51.95	107.53	105.28	47	56.77	3,600.00	—	—	—
33	25	17.86	16.41	34	46.40	45.65	43.94	39	50.74	3,600.00	—	—	—
34	25	30.97	29.02	42	59.04	70.72	68.55	43	63.50	3,600.00	—	—	—
35	25	74.83	72.53	50	67.40	129.40	126.83	49	69.39	3,600.00	—	—	—
36	25	68.19	66.28	42	50.12	165.99	163.21	56	55.12	3,600.00	—	—	—
37	25	55.32	53.18	46	49.89	127.80	125.33	49	52.86	3,600.00	—	—	—
38	25	45.90	44.07	41	51.16	169.90	167.56	49	56.42	3,600.00	—	—	—
39	25	44.63	42.37	47	60.99	117.22	115.11	45	66.86	3,600.00	—	—	—
40	25	35.38	33.35	44	63.51	96.75	94.08	53	67.90	3,600.00	—	—	—

that in all three cases the same implementation using *ng*-path relaxation is used, in which setting $\Delta_{ng} = |N'|$ corresponds to allowing elementary paths only and $\Delta_{ng} = 1$ corresponds to allowing all cyclic routes.

Table 3 shows the results of this experiment. The columns "Inst." and "$|N'|$" show the instance number and number of customers included in that instance, respectively. The columns labeled "T.Time" show the total time in seconds taken by the column generation algorithm, while "P.Time" shows the time in seconds taken by the pricing algorithm. Finally, the columns "Iter." show the number of column generation iterations that were performed per instance and the columns "LP" show the value of the LP relaxation using the various path relaxations.

First, it is clear that the algorithm spends most of its time on solving the pricing problems. Furthermore, the algorithm shows the typical behavior for the different route relaxations: allowing all cycles yields the fastest algorithm but lowest LP values, allowing elementary routes only yields the slowest algorithm but the strongest LP values, while the *ng*-path relaxation with $\Delta_{ng} = 5$ provides computation times that are relatively close to the fastest time while the LP values are close to the strongest values. Note that when the instance size increases so does the computation time. In particular, when allowing only elementary routes, the column generation algorithm cannot solve the LP relaxation within the time limit for all but one of the instances with 20 and 25 customers.

Table 4. Column Generation Experiment Results, Tabu Search

		All cycles allowed				ng-paths with $\Delta_{ng} = 5$				Only elementary routes			
Inst.	$\|N'\|$	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP	T.Time	P.Time	Iter.	LP
1	10	3.08	2.65	35	23.91	1.16	0.97	18	27.98	3.63	3.50	15	27.98
2	10	1.41	1.13	27	27.36	0.54	0.41	13	31.10	0.52	0.40	11	31.10
3	10	2.92	2.52	34	33.74	0.95	0.76	16	36.52	1.16	0.95	17	36.52
4	10	1.56	1.18	36	38.69	1.10	0.83	26	41.24	1.15	0.88	26	41.24
5	10	1.12	0.81	30	37.60	0.47	0.33	13	40.55	0.44	0.32	12	40.55
6	10	1.53	1.12	40	34.66	1.00	0.74	26	39.36	1.02	0.76	26	39.36
7	10	1.84	1.51	33	28.41	0.78	0.62	20	32.20	0.62	0.50	12	32.20
8	10	1.95	1.61	29	29.20	1.64	1.37	24	30.70	1.20	0.99	18	30.89
9	10	1.88	1.56	32	23.97	0.72	0.60	12	26.01	0.63	0.52	11	26.01
10	10	1.43	1.12	30	31.31	1.31	1.07	23	34.96	1.06	0.85	21	34.97
11	15	6.21	5.43	39	44.25	3.08	2.61	24	47.25	4.26	3.87	20	47.26
12	15	8.86	7.71	50	29.98	6.41	6.01	20	32.57	4.85	4.41	19	32.65
13	15	3.56	2.84	37	33.91	1.80	1.51	15	40.12	2.01	1.74	14	40.12
14	15	6.59	5.64	48	35.83	1.93	1.66	13	40.69	2.65	2.33	16	40.71
15	15	4.77	4.07	36	40.47	3.30	2.79	27	42.43	100.75	100.20	26	42.43
16	15	5.85	4.92	45	38.16	2.94	2.51	22	41.96	7.15	6.75	20	42.03
17	15	11.71	10.60	51	43.13	5.99	5.49	24	46.71	21.07	20.58	24	46.81
18	15	4.88	4.06	43	40.94	2.71	2.27	23	45.16	4.01	3.58	23	45.17
19	15	5.19	4.24	43	47.67	4.65	3.98	31	51.43	5.84	5.29	26	51.46
20	15	7.67	6.33	62	45.26	4.05	3.45	30	48.40	6.46	5.80	33	48.40
21	20	17.27	15.54	53	43.85	11.43	10.49	29	48.32	24.92	24.05	23	48.32
22	20	18.55	16.44	59	56.35	18.87	17.34	44	60.72	2,551.16	2,549.93	29	60.95
23	20	24.76	21.97	73	45.19	28.15	26.27	54	50.57	148.42	146.62	50	50.67
24	20	13.45	11.44	58	55.57	9.72	8.59	34	62.00	12.75	11.81	28	62.22
25	20	21.90	19.06	68	50.38	19.90	17.98	50	54.70	27.50	26.01	38	54.71
26	20	13.35	11.34	58	46.56	7.25	6.32	28	51.79	10.82	9.98	24	51.79
27	20	14.04	12.00	61	54.30	6.22	5.36	28	60.70	8.58	7.60	31	60.71
28	20	19.80	17.09	74	53.13	11.84	10.56	36	58.79	20.78	19.60	33	58.84
29	20	19.51	17.39	53	43.78	11.36	10.11	35	46.81	26.81	25.80	28	46.83
30	20	17.74	15.40	58	48.97	12.11	10.84	32	52.09	42.75	41.35	32	52.16
31	25	41.55	37.25	70	60.16	46.38	43.80	43	65.23	175.04	172.82	36	65.34
32	25	50.73	45.70	89	51.95	28.86	26.50	43	56.77	165.74	163.71	36	56.86
33	25	24.31	21.06	62	46.40	14.64	12.83	32	50.74	19.82	18.65	22	51.16
34	25	36.68	32.18	75	59.04	28.06	24.99	51	63.50	90.22	87.58	44	63.64
35	25	67.59	62.22	83	67.40	44.44	41.13	52	69.39	646.44	643.62	42	69.65
36	25	76.68	70.67	91	50.12	75.44	72.09	55	55.12	1,583.41	1,579.85	61	55.27
37	25	48.89	44.41	72	49.89	30.81	28.68	35	52.86	295.42	293.41	34	53.01
38	25	43.51	39.16	78	51.16	49.73	46.49	56	56.42	337.57	335.08	45	56.45
39	25	41.85	37.46	73	60.99	35.81	33.17	43	66.86	191.32	188.84	42	67.48
40	25	38.67	33.63	79	63.51	22.38	20.10	35	67.90	105.38	103.28	33	68.32

Table 4 shows the same experiment, now using the tabu search pricing heuristic, and only employing the exact labeling algorithm when the pricing heuristic fails to identify a negative reduced cost column. The following settings are used for the tabu search algorithm. The number of iterations that a move remains tabu is $TS_{\text{tabu}} = 5$, the number of iterations before the algorithm reinitializes with a new route is $TS_{\text{div}} = 15$, and the maximum number of routes added to the formulation by the tabu search algorithm during one iteration of the column generation algorithm is $TS_{\max} = 150$.

Observe that there is a steep decline in computation time from using the pricing heuristic when only elementary routes are allowed. Now the LP relaxation of all instances can be solved within the time limit by the column generation algorithm. The total time for solving all instances in this case is 6,655.33 seconds and it is even marginally faster than the ng-path relaxation with $\Delta_{ng} = 5$ in 7 of the smaller instances. The computation time of solving all instances with the ng-path relaxation with $\Delta_{ng} = 5$ has decreased from 1,677.44 seconds to 559.93 seconds. On the other hand, when all cycles are allowed, the computation time has actually increased from 659.23 seconds to 734.84 seconds. In this case using the pricing heuristic increases the computation time in 34 instances. This is caused by the fact that the tabu search heuristic only generates elementary paths. The optimal solution when all cyclic paths are allowed consists mostly of cyclic routes, and hence the column generation algorithm does not benefit much from generating good elementary routes quickly.

The results of the branch-price-and-cut experiments that are presented next are obtained using the column generation algorithm in which the tabu search pricing heuristic is employed and the *ng*-path relaxation with $\Delta_{ng} = 5$ is used. This setting exhibits the lowest computation times while the LP values are competitive with allowing only elementary paths.

7.3. Branch-Price-and-Cut Experiments

The experiments, of which the results are presented next, give insight into the performance of the branch-price-and-cut algorithm. Table 5 shows the results of applying the branch-price-and-cut algorithm without including any arc-synchronization inequalities. Like before, the columns "Inst.", "$|N'|$", and "Tot.Time" show, respectively, the instance number, the number of customers, and the total time in seconds taken by the algorithm. The column "Opt.Gap" shows the optimality gap remaining after termination of the algorithm. Furthermore, the column "LP Gap" shows the gap between the value of the best found solution and the value of the LP relaxation in percentages. Similarly, the column "Root Gap" shows this gap in percentages after adding valid inequalities to the LP relaxation. Finally, the column "Nodes" shows the number of nodes processed in the search tree and the column "CI" shows the number of added capacity inequalities. A dash indicates that no integer solution was found within the time limit.

The algorithm solves all instances with 10 customers and 9 out of 10 instances with 15 customers within the time limit. For the 20 and 25 customer instances the algorithm is in both cases only able to solve 1 out of 10 instances. For the unsolved instances, only for instance 11 is a feasible integer solution found within the time limit. Note that with different settings we were able to find the optimal solution of this instance, from which we conclude that the feasible integer solution is not optimal. In fact, the best found solution value is 0.55% higher than the optimum, and the LP gap and root gap with respect to the optimal solution value are 2.40% and 0.95%, respectively. Of all instances that are solved to optimality, the algorithm has a root gap of at most 0.96% and on three occasions the root gap is 0%.

Next, Table 6 shows the results of the same experiment with the branch-price-and-cut algorithm, with the difference being that now arc-synchronization inequalities are included. Violated arc-synchronization inequalities are separated whenever the column generation algorithm cannot find negative reduced cost routes and no violated capacity inequalities are identified. All arc-synchronization inequalities violated by more than 0.25 are added to the formulation. The column "ASI" shows the number of added arc-synchronization inequalities.

By including arc-synchronization inequalities, three more instances are solved to optimality, instances 11, 27, and 28, and for two more instances a feasible integer solution is found, instances 29 and 30. Using different settings for the branch-price-and-cut algorithm, we were also able to solve instance 30 to optimality; we do not present this experiment as the overall results were not as good. This allows us to conclude that the solution found above for instance 30 is the optimal solution. Furthermore, including arc-synchronization constraints closes the root gap to 0% for one additional instance, instance 20.

Next, consider the 21 instances that could already be solved without adding the arc-synchronization constraints. Adding the arc-synchronization inequalities yields lower computation times in 14 out of these 21 instances. In total, the computation time goes down with 507.25 seconds in these 14 instances, which is a decrease of 32.65%. In the other seven instances the

Table 5. Branch-Price-and-Cut Experiment Results

Inst.	$\|N'\|$	Tot.Time	Opt.Gap	LP Gap	Root Gap	Nodes	CI
1	10	6.69	0	0	0	11	2
2	10	2.12	0	0.51	0.01	7	1
3	10	37.44	0	3.11	0.95	143	10
4	10	26.58	0	1.05	0.47	191	6
5	10	6.60	0	0.44	0.44	46	4
6	10	45.74	0	0.95	0.95	449	23
7	10	2.26	0	1.20	0	12	1
8	10	26.25	0	1.78	0.46	89	25
9	10	32.09	0	1.81	0.96	100	4
10	10	32.84	0	1.09	0.47	162	16
11	15	3,600.00	0.59	2.94	1.50	4,363	63
12	15	24.32	0	1.51	0	8	20
13	15	19.78	0	0.14	0.08	23	7
14	15	131.53	0	2.62	0.41	171	33
15	15	40.65	0	1.97	0.16	46	26
16	15	31.31	0	1.19	0.02	19	15
17	15	386.21	0	3.76	0.69	262	44
18	15	170.43	0	1.07	0.28	268	20
19	15	129.72	0	2.47	0.41	327	50
20	15	10.34	0	1.71	0.03	3	21
21	20	505.55	0	3.14	0.53	189	34
22	20	3,600.00	—	—	—	1,591	185
23	20	3,600.00	—	—	—	960	97
24	20	3,600.00	—	—	—	2,031	140
25	20	3,600.00	—	—	—	1,225	113
26	20	3,600.00	—	—	—	2,293	112
27	20	3,600.00	—	—	—	2,504	66
28	20	3,600.00	—	—	—	1,796	109
29	20	3,600.00	—	—	—	1,788	134
30	20	3,600.00	—	—	—	1,486	126
31	25	3,600.00	—	—	—	409	161
32	25	3,600.00	—	—	—	716	128
33	25	579.72	0	2.92	0.04	143	78
34	25	3,600.00	—	—	—	319	208
35	25	3,600.00	—	—	—	333	227
36	25	3,600.00	—	—	—	471	121
37	25	3,600.00	—	—	—	487	160
38	25	3,600.00	—	—	—	520	95
39	25	3,600.00	—	—	—	340	160
40	25	3,600.00	—	—	—	298	271

Table 6. Branch-Price-and-Cut Experiment Results, Including Arc-Synchronization Inequalities

Inst.	$\|N'\|$	Tot.Time	Opt.Gap	LP Gap	Root Gap	Nodes	CI	ASI
1	10	7.61	0	0	0	11	2	0
2	10	2.37	0	0.51	0.01	7	1	2
3	10	36.64	0	3.11	0.90	103	9	28
4	10	12.45	0	1.05	0.13	61	5	6
5	10	7.46	0	0.44	0.44	46	5	3
6	10	41.25	0	0.95	0.91	263	7	27
7	10	1.03	0	1.20	0	2	1	1
8	10	24.15	0	1.78	0.33	73	23	14
9	10	26.71	0	1.81	0.57	77	5	10
10	10	14.29	0	1.09	0.32	76	15	15
11	15	2,574.84	0	2.40	0.75	3,020	53	53
12	15	24.76	0	1.51	0	8	20	0
13	15	21.19	0	0.14	0.08	23	7	0
14	15	48.21	0	2.62	0.26	36	21	15
15	15	33.29	0	1.97	0.14	31	18	6
16	15	28.94	0	1.19	0.02	20	23	2
17	15	240.26	0	3.76	0.57	114	34	13
18	15	74.10	0	1.07	0.26	80	16	12
19	15	132.12	0	2.47	0.39	165	52	24
20	15	6.69	0	1.71	0	1	21	3
21	20	525.21	0	3.14	0.53	200	38	5
22	20	3,600.00	—	—	—	1,431	160	55
23	20	3,600.00	—	—	—	864	88	53
24	20	3,600.00	—	—	—	1,370	115	84
25	20	3,600.00	—	—	—	1,133	114	81
26	20	3,600.00	—	—	—	2,359	112	19
27	20	2,680.08	0	1.64	0.53	1,677	56	47
28	20	2,086.61	0	2.80	0.45	858	77	62
29	20	3,600.00	0.46	2.51	1.58	1,761	121	36
30	20	3,600.00	0.03	1.56	0.61	1,249	116	58
31	25	3,600.00	—	—	—	285	153	36
32	25	3,600.00	—	—	—	488	116	31
33	25	458.13	0	2.92	0.04	104	84	1
34	25	3,600.00	—	—	—	358	200	22
35	25	3,600.00	—	—	—	286	223	38
36	25	3,600.00	—	—	—	370	121	57
37	25	3,600.00	—	—	—	394	153	18
38	25	3,600.00	—	—	—	370	86	34
39	25	3,600.00	—	—	—	183	156	40
40	25	3,600.00	—	—	—	244	249	31

increase in computation time is 25.94 seconds in total, which corresponds to an increase of 3.60%.

8. Algorithmic Insights

In this paper, a branch-price-and-cut algorithm is presented to solve the TWAVRP with time-dependent travel times to optimality. It uses a column generation algorithm with, at its core, a new exact pricing algorithm and a new heuristic pricing algorithm. In our experiments, the branch-price-and-cut algorithm is able to solve instances of modest size. To put this in perspective, please note the following. The algorithm by Spliet and Gabor (2015) to solve the TWAVRP was able to solve instances with up to 25 customers and 3 scenarios. Arguably such instances of the TWAVRP are comparable with a VRP instance with $25 \times 3 = 75$ customers, which is in line with the size of instances that current state-of-the-art VRP solvers can consistently solve. A similar argument can be made for the branch-price-and-cut algorithm in this paper. The instances with 15 and 20 customers and 3 scenarios might be thought of as comparable to the 50 customer instances of the VRPTW with time-dependent travel times solved in the paper by Dabia et al. (2013). With this in mind, we argue that the algorithm presented in this paper performs well.

Nonetheless, instances that need to be solved in real life are typically much larger than can be handled by our branch-price-and-cut algorithm. Still, the column generation algorithm could be of use when incorporated into a column generation-based primal heuristic as, for instance, described in Joncour et al. (2010). Such algorithms can be used to find good solutions for much larger instances. Obvious ways to do so are, for example, fixing the time window assignments found in the LP relaxation, only using the tabu search pricing heuristic and never using the exact pricing algorithm, or a rounding procedure in which iteratively the LP relaxation is solved and a fractional variable is fixed to 1. Such heuristics perform quite well for another variant of the time window assignment problem, the DTWAVRP, as demonstrated by Spliet and Desaulniers (2015). Because we believe that this behavior will also easily translate to our case of including time-dependent travel times, we decided not to include these experiments in this paper.

Furthermore, we would like to comment on our observation that the LP solutions are nonunique. When solving the LP relaxation of the TWAVRP with time-dependent travel times with different settings of our algorithm, we encountered different primal and dual LP solutions with the same solution value. In fact, many vertices of our feasible region are degenerate, not only the optimal one, which resulted in the plateau effect. This effect seems to be harmful in all nodes of the search tree. Often, the optimal objective value of a node in the search tree was found in the first iteration, but several iterations of the column generation algorithm were needed to prove optimality, adding columns at each iteration that do not change the objective value. This also hindered attempts to incorporate column management, as we often found that any columns that we removed had to be added again to prove optimality.

We implemented a dual stabilization procedure in an attempt to overcome the effects of degeneracy. In particular, we implemented the procedure of Ben Amor and Desrosiers (2006), which performed best in a comparison of the multidepot vehicle scheduling problem by Ben Amor, Frangioni, and Desrosiers (2009). However, like many stabilization procedures, the performance of this method is highly dependent on an initial estimate of the optimal dual values, which was

not at our disposal. Moreover, even in experiments in which we initialized the procedure with the optimal dual values, for some instances we found a marginal decrease in computation time, while for others we found a huge increase. Therefore, we did not include any experiments with the dual stabilization procedure in this paper.

To be able to solve larger instances to optimality, one way forward is to decrease the number of nodes in the search tree. Although this is an obvious statement, we wish to divulge a final interesting observation relating to this. The number of nodes in the search tree that need to be explored is much higher than when solving the TWAVRP without time-dependent travel times, even when using the branch-price-and-cut algorithm presented in this paper for both cases. At first glance, this might not seem surprising since now the more complex decision has to be made of which arcs in $\hat{G}$ have to be used instead of which arcs in G. However, interestingly, in our experiments with time-dependent travel times the branch-price-and-cut algorithm branched virtually always on the arcs in G and almost never on the arcs in $\hat{G}$. It seems that when the order in which the customers are visited is fixed, selecting the appropriate arcs in $\hat{G}$ is straightforward. Although the number of arcs in a TWAVRP solution with and without time-dependent travel times are roughly the same, deciding on which arcs should be chosen is apparently more difficult when dealing with time-dependent travel times. A possible explanation for this is that in the LP relaxation of our formulation, multiple arcs in $\hat{G}$ can be fractionally selected to misrepresent the travel times. As a result, the objective values in different search nodes might be closer together. To help overcome this, we introduced the arc-synchronization inequalities. Still future research is needed to improve on these bounds to further diminish the number of nodes in the branching tree that have to be explored.

9. Conclusion

In this paper, the TWAVRP with time-dependent travel times is introduced as well as a branch-price-and-cut algorithm to solve it to optimality. We consider the inclusion of time-dependent travel times to be important for vehicle routing problems with arrival time consistency considerations, in particular for the TWAVRP. The pricing problem in the branch-price-and-cut algorithm is a new version of the shortest path problem for which we developed an exact labeling algorithm and a tabu search heuristic. The labeling algorithm is based on the existing algorithm of Ioachim et al. (1998). Moreover, to find solutions for instances that are larger than those the branch-price-and-cut algorithm can currently solve, we wish to emphasize that the presented column generation algorithm is valuable in itself. It can be easily incorporated in various column generation-based primal heuristics. Finally, we present arc-synchronization inequalities. These valid inequalities strengthen the LP bound used by the branch-price-and-cut algorithm, allowing the algorithm to solve more instances in our experiments.

References

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

Ben Amor H, Desrosiers J (2006) A proximal trust-region algorithm for column generation stabilization. *Comput. Oper. Res.* 33(4): 910–927.

Ben Amor H, Frangioni A, Desrosiers J (2009) On the choice of explicit stabilizing terms in column generation. *Discrete Appl. Math.* 157(6):1167–1184.

Cordeau J, Ghiani G, Guerriero E (2014) Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Sci.* 48(1):46–58.

Dabia S, Ropke S, van Woensel T, de Kok T (2013) Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Sci.* 47(3):380–396.

Dumas Y, Soumis F, Desrosiers J (1990) Optimizing the schedule for a fixed vehicle path with convex inconvenience costs. *Transportation Sci.* 24(2):145–152.

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.

Franceschetti A, Honhon D, van Woensel T, Bektaş T, Laporte G (2013) The time-dependent pollution-routing problem. *Transportation Res. Part B: Methodological* 56(1):265–293.

Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems: A review. *Comput. Oper. Res.* 64(1):189–197.

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

Ichoua S, Gendreau M, Potvin J (2003) Vehicle dispatching with time-dependent travel times. *Eur. J. Oper. Res.* 144(2):379–396.

Ioachim I, Gélinas S, Soumis F, Desrosiers J (1998) A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks* 31(3):193–204.

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

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

Kohl N, Desrosiers J, Madsen OBG, Solomon MM, Soumis F (1999) 2-Path cuts for the vehicle routing problem with time windows. *Transportation Sci.* 33(1):101–116.

Kovacs AA, Golden BL, Hartl RF, Parragh SN (2014) Vehicle routing problems in which consistency considerations are important: A survey. *Networks* 64(3):192–213.

Lysgaard J (2003) CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper, Department of Management Science and Logistics, Aarhus School of Business, Aarhus, Denmark, 3–4.

Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Programming* 100(2):423–445.

Ribeiro GM, Desaulniers G, Desrosiers J (2012) A branch-price-and-cut algorithm for the workover rig routing problem. *Comput. Oper. Res.* 39(12):3305–3315.

Spliet R, Desaulniers G (2015) The discrete time window assignment vehicle routing problem. *Eur. J. Oper. Res.* 244(2):379–391.

Spliet R, Gabor AF (2015) The time window assignment vehicle routing problem. *Transportation Sci.* 49(4):721–731.