# Branch-and-Price-and-Cut for the Vehicle Routing and Truck Driver Scheduling Problem

Christian Tilk[*,a]

[a] *Chair of Logistics Management, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

**Abstract**

Many governments worldwide have imposed hours of service regulations for truck drivers to ensure that break and rest periods are regularly taken. Transport companies have to take these into account and plan the routes and schedules of their truck drivers simultaneously. This problem is called vehicle routing and truck driver scheduling problem (VRTDSP). With their paper "An exact method for vehicle routing and truck driver scheduling problems" [Technical Report No. 33, Jacobs University, School of Engineering and Science, Bremen, Germany] Goel and Irnich presented the first exact approach to the VRTDSP. They include hours of service regulations in a vehicle routing problem with time windows and use a branch-and-price algorithm to solve it. The main contribution of the paper at hand is to present a sophisticated branch-and-price-and-cut algorithm for the VRTDSP that is based on the parameter-free auxiliary network and the resource extension functions (REFs) defined in the work of Goel and Irnich. Their labeling algorithm is extended by means of defining backward REFs in order to build a bidirectional labeling. Feasible routes are constructed by a non-trivial merge procedure. Different acceleration techniques are used to speed up the solution process of the pricing problem. In addition, several classes of known valid inequalities are used to further strengthen the LP-relaxation of the master program. We present a detailed computational study to analyze the impact of the different techniques. The resulting algorithm is able to solve all VRTDSP benchmark instances with 25 customers and 44 out of 56 instances with 50 customers in two hours of computation time to proven optimality.

*Key words:* truck driver, routing and scheduling, branch-and-price-and-cut

## 1. Introduction

Many governments worldwide have imposed hours of service regulations for truck drivers to avoid fatigue-related accidents. These regulations ensure that break and rest periods are regularly taken, i.e. they define a minimum amount of break and rest times for truck drivers as well as a maximum driving time between two break or rest periods. Transport companies have to take these into account when planning the routes and schedules of their truck drivers. All kinds of such regulations strongly restrict the feasibility of vehicle routes that last several days and are assigned to a single driver. Two approaches can be used to construct feasible vehicle routes while minimizing overall costs: In a two-phase approach, routes are first built without considering hours of service regulations and later modified to take them into account. Solving the problem exactly makes it necessary to use an integrated approach: Vehicle routes and truck driver schedules have to be planned simultaneously, which is usually more complex but provides lower-cost solutions. This problem is called *vehicle routing and truck driver scheduling problem* (VRTDSP).

The paper at hand deals with the current hours of service regulations of the United States (U.S.) for property-carrying drivers, that entered into force in 2013 (Federal Motor Carrier Safety Administration,

[*]Corresponding author.
*Email address:* tilk@uni-mainz.de (Christian Tilk)

2011). According to these regulations, the day of a truck driver can be separated into four parts: Driving, working that does not include driving, e.g. loading and unloading the truck, break periods and rest periods. These parts are subject to the following restrictions:

(1) The accumulated driving time between two consecutive rest periods is at most eleven hours.

(2) The minimum duration of a break period is 30 minutes.

(3) The minimum duration of a rest period is ten hours.

(4) Driving can only be done until at most eight hours since the last break or rest period has elapsed.

(5) Driving can only be done until at most 14 hours since the last rest period has elapsed.

Note that these regulations only limit a truck driver in terms of driving, i.e. there are no limitations for working that does not include driving.

Recently, Goel and Irnich (2014) presented the first exact approach to the VRTDSP. They include hours of service regulations into a *vehicle routing problem with time windows* (VRPTW) and use a branch-and-price algorithm to solve it. The main contribution of the paper at hand is to present a sophisticated branch-and-price-and-cut algorithm for the VRTDSP that is based on the parameter-free auxiliary network and the resource extension functions (REFs, see Irnich (2008)) defined in (Goel and Irnich, 2014). We extend their labeling algorithm by means of defining backward REFs in order to build a bidirectional labeling. To obtain feasible routes a non-trivial merge procedure is presented. The concept of using a dynamic half-way point (Tilk *et al.*, 2016) and the *ng*-path relaxation (Baldacci *et al.*, 2011) are used to speed up the solution process of the pricing problem. In addition, different families of known valid inequalities are used to further strengthen the linear relaxation of the master program, namely subset-row (Jepsen *et al.*, 2008), two-path (Kohl *et al.*, 1999) and strong-degree inequalities (Contardo *et al.*, 2014) as well as a dynamic extension of the *ng*-neighborhood (Roberti and Mingozzi, 2014; Bode and Irnich, 2015). We present a detailed computational study to analyze the impact of the different techniques. The resulting algorithm is able to solve to proven optimality all VRTDSP benchmark instances with 25 customers and 44 out of 56 instances with 50 customers in two hours of computation time.

The remainder of this paper is as follows. Section 2 briefly reviews the literature on vehicle routing problems with hours of service regulations. In Section 3, an extensive formulation is given. Section 4 presents the pricing problem and its solution. Families of valid inequalities are discussed in Section 5. Section 6 presents a computational analysis of the algorithmic components. Concluding remarks are given in Section 7.

## 2. Literature

To the best of our knowledge, the first work considering hours of service regulations in a vehicle routing problem is (Savelsbergh and Sol, 1998). The break and rest rules considered in this work were similar to the regulations in the European Union (EU) that were in force at that time. A branch-and-price approach was used to solve the problem heuristically. Xu *et al.* (2003) investigated a pickup-and-delivery problem with several complicating side constraints, including the U.S. hours of service regulations that were in force at that time. Ceselli *et al.* (2009) solved a rich vehicle routing problem with a column-generation algorithm. Among other constraints, they impose an upper limit on the number of consecutive driving hours and enforce drivers' rest periods.

With the introduction of new hours of service regulations in the EU in 2007, several heuristic approaches for combined vehicle routing and truck driver scheduling have been proposed: Zäpfel and Bögl (2008) developed a two-phase algorithm for a VRPTW with certain driver rules. In the first phase, a metaheuristic is used to solve a vehicle routing problem, while in the second phase a personnel assignment problem is solved with a simple heuristic. Goel (2009) proposed a large neighborhood framework for a VRPTW including only a subset of the EU regulations. Bartodziej *et al.* (2009) designed a column-generation heuristic and three metaheuristics for solving a combined vehicle and crew scheduling problem with time windows and rest regulations. Drexl and Prescott-Gagnon (2010) investigated how European regulations can be formally modeled in an *elementary shortest path problem with resource constraints* (ESPPRC). Kok *et al.* (2010)

developed a heuristic dynamic-programming algorithm that can solve a VRPTW with all EU regulations. Prescott-Gagnon *et al.* (2010) also investigated the VRPTW with EU regulations and presented a large neighborhood search method that relies on a column-generation heuristic. Derigs *et al.* (2011) solved a real-world vehicle routing problem respecting the EU regulations.

The literature about the U.S. hours of service regulations is rather scarce. Rancourt *et al.* (2013) developed a tabu search approach to the VRTDSP with respect to the regulations that were in force until July 2013. Rancourt and Paquette (2014) designed a tabu search to solve a multi-objective VRTDSP with respect to the same regulations. Goel (2014) presented a simple heuristic to analyze the impact of the change in the regulations in 2013 in terms of cost and accident risks. Goel and Vidal (2014) presented a hybrid genetic search that has been used for different regulations worldwide, especially also with the new U.S. regulations. To the best of our knowledge, Goel and Irnich (2014) are the only authors presenting an exact solution approach to the VRTDSP. They developed a branch-and-price algorithm for the VRTDSP with the old and new U.S. regulations. Solutions of benchmark instances with up to 100 customers are reported.

## 3. Extensive Formulation

Using column-generation techniques to solve vehicle routing problems exactly is quite standard (Desaulniers *et al.*, 2010). In order to solve the VRTDSP with branch-and-price-and-cut, we use a set-partitioning formulation. In this setting, the master problem is quite easy, while capacity and time window constrains as well as hours of service regulations are tackled in the pricing problem (see Section 4).

The VRTDSP can be defined as follows: Let $C := \{1..n\}$ be the set of customer vertices, $o$ be the start depot and $d$ be the end depot. A time window $[a_i, b_i]$, a non-negative demand $q_i$ and a non-negative service time $s_i$ are associated with every vertex $i$. Service times and demands at the start and end depot are defined as zero, i.e. $q_o{=}q_d := 0$ and $s_o = s_d := 0$. Service at a vertex must start inside but may end outside of its time window. Let $A = \{(i,j) \in V \times V : i \neq j\}$ be the set of arcs. Each arc $(i,j)$ is associated with a non-negative travel time $t_{ij}$ (excluding break and rest times) and a non-negative travel cost $c_{ij}$. Note that we can remove all arcs that can not be feasibly traversed due to time window or capacity restrictions. Furthermore, let $K$ be a fleet of homogeneous vehicles with capacity $Q$. A vehicle route is defined as a path starting at $o$, ending at $d$ and visiting a subset of the customer vertices in between. A route is feasible if the capacity of the vehicle is not exceeded and if there exists a schedule complying with hours of service regulations and the time windows of the visited vertices. The cost of a route is defined as the sum of the travel cost of the traversed arcs. The goal of the VRTDSP is to find a set of feasible routes visiting each customer exactly once while minimizing the overall travel cost. Assuming that we know the set of all feasible routes $R$, the VRTDSP can be stated as follows:

$$\min \quad \sum_{r \in R} c_r \lambda_r \tag{1a}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir} \lambda_r = 1 \qquad \forall i \in C \tag{1b}$$

$$\sum_{r \in R} \lambda_r <= |K| \tag{1c}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in R \tag{1d}$$

The binary variables $\lambda_r$ indicate if route $r$ is used or not. The coefficients $c_r$ denote the travel cost of route $r$ and coefficients $a_{ir}$ denote the number of times route $r$ visits customer $i$. Hence, (1a) minimizes the overall travel cost. Constraints (1b) ensure that each customer is visited exactly once. The number of used vehicles is limited by (1c) and the variable domains are given in (1d).

In the following, the linear relaxation of Formulation (1) in which the set of all feasible routes is additionally replaced by a subset $\bar{R}$ is denoted as the *restricted master program* (RMP). For solving the RMP, a column-generation algorithm (Desaulniers *et al.*, 2005) is employed. The column-generation algorithm alternates between the LP re-optimization of the RMP and the column-generation pricing problem that adds additional variables (=columns) to the RMP. Let $\pi_i$ be the dual prices of the constraints (1b) of the RMP and let $\mu$ be the dual price of the convexity constraint (1c). The pricing problem asks for a route $r$ with negative reduced cost $\bar{c}_r := c_r - \mu - \sum_{i \in C} a_{ir}\pi_i$. The reduced cost of an arc $(i,j) \in A$ is defined as $\bar{c}_{ij} := c_{ij} - \frac{1}{2}\pi_i - \frac{1}{2}\pi_j$ with $\pi_o = \pi_d = \mu$. We initialize the set $\bar{R}$ with a set of dummy columns with infinite cost, each visiting one customer and having no impact on constraint (1c). Branching on arcs is required to finally ensure integer solutions of (1).

In order to stabilize the column-generation process, the partitioning constraints (1b) can be replaced by covering constraints if the triangle inequality holds for travel costs and times. This replacement is also possible if the triangle inequality does not hold. Then we have to add to the RMP the additional constraint that the number of visited vertices must not exceed $|C|$, i.e. $\sum_{r \in R} l_r \lambda_r \leq |C|$ , where $l_r$ is the number of times route $r$ visits a customer. The effect is that all dual prices $\pi_i$ for $i \in C$ are non-negative. To incorporate the added constraint in the pricing problem, its dual price has to be subtracted from the reduced cost of all arcs $(i,j)$ with $j \in C$.

## 4. Pricing Problem

In this section, we discuss the pricing problem of the branch-and-price-and-cut algorithm. First, in Subsection 4.1, we recapitulate the parameter-free model, the resources, and the REFs developed by Goel and Irnich (2014). This is the basis for the bidirectional labeling presented in Subsection 4.2. Subsection 4.3 deals with the *ng*-path relaxation (Baldacci *et al.*, 2011) for the VRTDSP. Finally, in Subsection 4.4, different techniques to accelerate the solution of the pricing problem are presented.

### *4.1. Forward Labeling*

The pricing problem is an ESPPRC that can be solved with a labeling algorithm. It tries to find a negative reduced-cost route that is feasible with respect to load, time windows and the hours of service regulations. According to the current U.S. hours of service regulations, a driver may take break or rest periods at any time and with any duration larger than 30 minutes and ten hours, respectively. However, driving periods can be scheduled only depending on the current state of the driver, and no limitations regarding service periods are specified. The relevant parameters are summarized in Table 1.

| **Symbol** | **Value** | **Description** |
|---|---|---|
| $t^{\text{drive}}$ | 11 hours | The maximum accumulated driving time between two consecutive rest periods |
| $t^{\text{break}}$ | $\frac{1}{2}$ hours | The minimum duration of a break period |
| $t^{\text{rest}}$ | 10 hours | The minimum duration of a rest period |
| $t^{\text{el}\|\text{B}}$ | 8 hours | The maximum time after the end of the last break or rest period until which a driver may drive |
| $t^{\text{el}\|\text{R}}$ | 14 hours | The maximum time after the end of the last rest period until which a driver may drive |

Table 1: Parameters imposed by the new U.S. hours of service regulations (Table 1 in Goel and Irnich, 2014)

Goel and Irnich (2014) defined an auxiliary network to tackle the difficulty of including hours of service regulations in a VRPTW. In this auxiliary network, a drivers' working day can be separated into a sequence of four different periods: working, service, break and rest. We assume that service times can not be interrupted by a break or a rest. Except for this assumption, a driver can take a rest or break at any point in time and space, especially between two customer location. To model this aspect an intermediate vertex $n_{ij}$ is created for each feasible arc $(i,j) \in A$. Now, a drivers' state can be modeled with nine resources and his possible activities with five REFs.

Beside the standard VRPTW resources cost, load and time, the other resources are needed to keep track of the drivers' current state. Namely, the accumulated driving time since the last rest, the remaining driving time to reach the next customer, the time elapsed since the last break, and the time elapsed since the last rest. Goel and Irnich (2014) have shown that it is beneficial in the labeling algorithm to keep rest and break periods as short as possible and to perform driving periods as long as possible. Obviously, this avoids time periods in which the driver is unproductive, however, this may cause unnecessary waiting times at subsequent customers due to time window restrictions. To overcome this issue, break and rest periods can be scheduled longer than required later on in the labeling algorithm. Therefore, two additional resources are needed to keep track of the latest possible points in time to which the end of the last rest and break period can be extended without violating any time window constraints. In the following, the nine resources needed to represent a drivers' state are listed:

- $T^{cost}$: the accumulated reduced cost of a route
- $T^{load}$: the accumulated load of a route
- $T^{time}$: the current time of day
- $T^{dist}$: the remaining driving time to reach the next customer
- $T^{drive}$: the accumulated driving time since the last rest
- $T^{el|B}$: the time elapsed since the last break
- $T^{el|R}$: the time elapsed since the last rest
- $T^{la|B}$: The latest possible point in time to which the end of the last break period can be extended without violating any resource constraints
- $T^{la|R}$: The latest possible point in time to which the end of the last rest period can be extended without violating any resource constraints

The possible activities of a driver are represented by the following REFs: First, the REF $f_{ij}^{start}$ models the starting point for going from vertex $i$ to vertex $j$ immediately after $i$ was served. Second, the REF $f_{ij}^{drive}$ is used to model driving from $i$ to $j$ for $\Delta_{ij}$ time units. The amount of driving time can be computed as $\Delta_{ij} = \min\{T^{dist}, t^{\text{drive}} - T^{drive}, t^{\text{el|B}} - T^{el|B}, t^{\text{el|R}} - T^{el|R}\}$. Note that $\Delta_{ij}$ is the maximal driving time that violates no resource constraints. Third, taking a 30-minute break or a ten-hour rest between the visit of customers $i$ and $j$ is modeled with the REFs $f_{ij}^{break}$ and $f_{ij}^{rest}$, respectively. Last, performing the service at customer $j$ is modeled with the help of the REF $f_{ij}^{visit}$. This REF also extends the length of the last rest and break period taken to avoid unnecessary waiting. The subnetwork defined by an arc $(i, j) \in A$ with the corresponding REFs represented as different arcs is depicted in Figure 1.

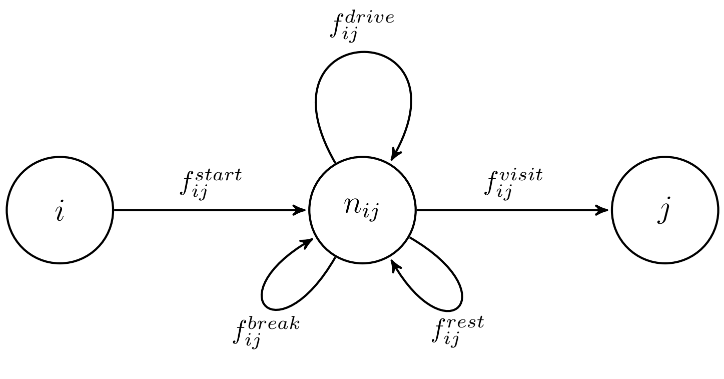

Figure 1: Subnetwork for arc $(i, j) \in A$

Now, a forward path $P = (o, .., u)$ in the auxiliary network defines a label $L_u = (u, S_u, T_u)$, where $u$ is the last visited vertex, $S_u$ the set of all visited vertices and $T_u := (T_u^{cost}, T_u^{load}, T_u^{time}, T_u^{dist}, T_u^{drive}, T_u^{el|B}, T_u^{el|R},$

$T_u^{la|B}, T_u^{la|R})$ the resource vector. The initial label representing a fully rested driver at the start depot $o$ is defined as $L_o = (o, \{o\}, T_o)$ with $T_o = (T_o^{cost}, T_o^{load}, T_o^{time}, T_o^{dist}, T_o^{drive}, T_o^{el|B}, T_o^{el|R}, T_o^{la|B}, T_o^{la|R}) := (0, 0, a_o, 0, 0, 0, 0, \infty, \infty)$.

When propagating a label $L_u = (u, S, T_u)$ forward over an arc in the auxiliary network corresponding to a REF $f_{ij}^{*}$, some preconditions must be fulfilled to create a feasible new label. The REF $f_{ij}^{start}$ produces a new feasible label if $j$ was not yet visited, the demand of $j$ fits in the vehicle and $j$ can be reached in its time window, i.e. if $j \notin S$, $T^{load} + q_j \leq Q$ and $T^{time} + t_{ij} \leq b_j$, respectively. Driving is only possible if the driver is fit, i.e. $f_{ij}^{drive}$ creates a feasible new label only if $\Delta_{ij} > 0$. Contrariwise, the REFs $f_{ij}^{break}$ and $f_{ij}^{rest}$ return a feasible label only if $\Delta_{ij} <= 0$. Finally, the service at vertex $j$ is possible only if the full distance between $i$ and $j$ was driven and $j$ is reached before the end of its time window, i.e. $f_{ij}^{visit}$ returns a feasible new label only if $T^{dist} = 0$ and $T^{time} \leq b_j$. Note that $f_{ij}^{visit}$ can be used independently of the value of $\Delta_{ij}$. Thus, the time elapsed since break or rest resources can exceeded $t^{\mathrm{el|B}}$ and $t^{\mathrm{el|R}}$, respectively. As a consequence, $\Delta_{ij}$ can have a negative value in the newly created labels.

The resources $T^{cost}$, $T^{load}$ and the set $S$ are only updated when the REF $f_{ij}^{start}$ is used while their resource values are kept by all other REFs. It holds: $T_{n_{ij}}^{cost} = f_{ij}^{start}(T_i^{cost}) = T_i^{cost} + \bar{c}_{ij}$, $T_{n_{ij}}^{load} = f_{ij}^{start}(T_i^{load}) = T_i^{load} + q_j$, and $S_{n_{ij}} = S_i \cup \{j\}$. All other resource updates of the REFs are summarized in Table 2. Blank entries indicate that the resource value is kept.

| REF | $T^{time}$ | $T^{dist}$ | $T^{drive}$ | $T^{el\|B}$ | $T^{el\|R}$ | $T^{la\|B}$ | $T^{la\|R}$ |
|---|---|---|---|---|---|---|---|
| $f_{ij}^{start}$ | | $t_{ij}$ | | | | | |
| $f_{ij}^{drive}$ | $T^{time} + \Delta_{ij}$ | $T^{dist} - \Delta_{ij}$ | $T^{drive} + \Delta_{ij}$ | $T^{el\|B} + \Delta_{ij}$ | $T^{el\|R} + \Delta_{ij}$ | $\min\{T^{la\|B}, T^{la\|R} + t^{\mathrm{el\|R}} - T^{el\|B} - \Delta_{ij}\}$ | |
| $f_{ij}^{break}$ | $T^{time} + \Delta_{ij}$ | | | $0$ | $T^{el\|R} + \Delta_{ij}$ | $\infty$ | |
| $f_{ij}^{rest}$ | $T^{time} + \Delta_{ij}$ | | $0$ | $0$ | $0$ | $\infty$ | $\infty$ |
| $f_{ij}^{visit}$ | $\max\{T^{time}, a_j\} + s_j$ | | | $\max\{T^{el\|B}, a_j - T^{la\|B}\} + s_j$ | $\max\{T^{el\|R}, a_j - T^{la\|R}\} + s_j$ | $\min\{T^{la\|B}, b_j - T^{el\|B}\}$ | $\min\{T^{la\|R}, b_j - T^{el\|R}\}$ |

Table 2: Resource extension functions

Note that all REFs are non-decreasing in the resources $T^{cost}, T^{load}, T^{time}, T^{dist}, T^{drive}, T^{el|B}$ and $T^{el|R}$ as well as non-increasing in the resources $T^{la|B}$ and $T^{la|R}$. Hence, a standard dominance rule can be defined (Irnich and Desaulniers, 2005). Additionally, Goel and Irnich (2014) have developed a second dominance rule to further reduce the number of labels.

**Rule 1.** *(Dominance 1) Let $L = (u, S, T)$ and $L' = (u, S', T')$ be two labels with identical last vertex $u$. Let $P = P(L)$ and $P' = P(L')$ be the respective partial paths. If $S \subseteq S'$, $T^{cost} \leq T'^{cost}, T^{load} \leq T'^{load}, T^{time} \leq T'^{time}, T^{dist} \leq T'^{dist}, T^{drive} \leq T'^{drive}, T^{el|B} \leq T'^{el|B}, T^{el|R} \leq T'^{el|R}, T^{la|B} \geq T'^{la|B}$ and $T^{la|R} \geq T'^{la|R}$, then any feasible extension of $P'$ towards $d$ is also a feasible extension of $P$ with non-smaller cost. Hence, $L'$ can be discarded.*

**Rule 2.** *(Dominance 2) Let $L = (u, S, T)$ and $L' = (u, S', T')$ be two labels with identical last vertex $u$. Let $P = P(L)$ and $P' = P(L')$ be the respective partial paths. If $S \subseteq S'$, $T^{cost} \leq T'^{cost}, T^{load} \leq T'^{load}, T^{dist} \leq T'^{dist}$ and $T^{time} + t^{\mathrm{el|R}} \leq T'^{time}$, then any feasible extension of $P'$ towards $d$ is also a feasible extension of $P$ with non-smaller cost. Hence, $L'$ can be discarded.*

### *4.2. Bidirectional Labeling*

This subsection presents the bidirectional labeling. First, a backward labeling is defined and the differences to forward labeling are emphasized. Then a non-trivial merge procedure for forward and backward labels is presented.

In order to use the same resources and similar REFs as in the forward labeling, the backward labeling is defined on an inverted network. To be precise, the time window of a vertex $i$ is defined as $[a_i^{bw}, b_i^{bw}] := [-b_i - s_i, -a_i - s_i]$ and the direction of all arcs in the auxiliary backward network is reversed. The time window is shifted by $s_i$ because service at a vertex $i$ must start inside but can end outside the time window. The change in the arc direction leads to a role reversal of the REFs: REF $f_{ij}^{visitBW}$ models the starting point for going backward from vertex $j$ to vertex $i$ and REF $f_{ij}^{startBW}$ models the service at customer $i$ when coming backward from customer $j$.

Additionally, we have to take care of another aspect of the problem: Since waiting and service times do not count as driving times, performing service is still possible although the resource $T^{el|R}$ would exceed the maximum value $t^{\mathrm{el|R}}$. In chronological order, this implies that a rest period must succeed the service period that cause this exceeding before any driving period can be executed. In the forward labeling, this is implicitly managed by allowing the use of REF $f_{ki}^{visit}$, even if the new resource value $T_i^{el|R} = f_{ki}^{visit}(T_{n_{ki}}^{el|R})$ exceeds $t^{\mathrm{el|R}}$ due to the service and waiting times at vertex $i$. In this case, a rest is implicitly enforced at the next intermediate vertex due to $\Delta \leq 0$ .

The same situation means in the reversed chronological order of the backward labeling that the resource $T^{el|R}$ can exceed $t^{\mathrm{el|R}}$ by the service and waiting times at vertex $i$ if and only if REF $f_{ij}^{restBW}$ was applied immediately before REF $f_{ij}^{startBW}$. To model this, two aspects of REF $f_{ij}^{startBW}$ have to be changed. First, when applying REF $f_{ij}^{startBW}$, the service and waiting times at vertex $i$ are not taken into account for computing the new resource values $T_i^{el|R}$ and $T_i^{la|R}$ if and only if REF $f_{ij}^{restBW}$ was applied immediately before. Note that the resources $T^{el|R}$ and $T^{la|R}$ are both affected since they are interdependent. Second, the resource $T^{el|R}$ may never exceed $t^{\mathrm{el|R}}$ since the service and waiting times that allow an exceeding are not taken into account for the computation. Therefore, applying REF $f_{ij}^{startBW}$ returns no feasible new label if the new resource value $T_j^{el|R} = f_{ij}^{startBW}(T_{n_{ij}}^{el|R})$ exceeds $t^{\mathrm{el|R}}$. A similar argumentation holds for the resource $T^{el|B}$ and the value $t^{\mathrm{el|B}}$.

Hence, the REFs in the backward labeling are defined as follows: The REFs $f_{ij}^{driveBW}$, $f_{ij}^{restBW}$, $f_{ij}^{breakBW}$ and $f_{ij}^{visitBW}$ use the same preconditions and resource updates as their counterparts $f_{ij}^{drive}$, $f_{ij}^{rest}$, $f_{ij}^{break}$ and $f_{ij}^{start}$ in the forward labeling. Besides the preconditions of its counterpart $f_{ij}^{visit}$, the REF $f_{ij}^{startBW}$ has two additional preconditions to create a feasible label:

$$f_{ij}^{startBW}(T_{n_{ij}}^{el|B}) \leq t^{\mathrm{el|B}} \quad \text{and} \quad f_{ij}^{startBW}(T_{n_{ij}}^{el|R}) \leq t^{\mathrm{el|R}}.$$

Furthermore, $f_{ij}^{startBW}$ updates the set $S$ and the resources $T^{cost}$, $T^{load}$, $T^{time}$, $T^{dist}$ and $T^{drive}$ as the REF $f_{ij}^{visit}$ updates them in the forward labeling. The updates of $T^{el|B}$, $T^{el|R}$, $T^{la|B}$ and $T^{la|R}$ change as stated below. Note that a value $T^{el|R} = 0$ implies that the last used REF was $f_{ij}^{restBW}$. Similarly, a value $T^{el|B} = 0$ implies that the last used REF was either $f_{ij}^{restBW}$ or $f_{ij}^{breakBW}$.

$$f_{ij}^{startBW}(T_{n_{ij}}^{el|B}) = \begin{cases} 0 & \text{if } T_{n_{ij}}^{el|B} = 0 \\ \max\{T_{n_{ij}}^{el|B}, a_i^{bw} - T^{la|B}\} + s_i & \text{otherwise} \end{cases}$$

$$f_{ij}^{startBW}(T_{n_{ij}}^{el|R}) = \begin{cases} 0 & \text{if } T_{n_{ij}}^{el|R} = 0 \\ \max\{T_{n_{ij}}^{el|R}, a_i^{bw} - T^{la|R}\} + s_i & \text{otherwise} \end{cases}$$

$$f_{ij}^{startBW}(T_{n_{ij}}^{la|B}) = \begin{cases} \infty & \text{if } T_{n_{ij}}^{el|B} = 0 \\ \min\{T_{n_{ij}}^{la|B}, b_i^{bw} - T^{el|B}\} & \text{otherwise} \end{cases}$$

$$f_{ij}^{startBW}(T_{n_{ij}}^{la|R}) = \begin{cases} \infty & \text{if } T_{n_{ij}}^{el|R} = 0 \\ \min\{T_{n_{ij}}^{la|R}, b_i^{bw} - T^{el|R}\} & \text{otherwise} \end{cases}$$

Now, a backward path $P = (u, .., d)$ in the auxiliary backward network defines a label $L_u = (u, S_u, T_u)$, where $u$ is the first visited vertex, $S_u$ the set of all visited vertices and $T_u := (T_u^{cost}, T_u^{load}, T_u^{time}, T_u^{dist},$

$T_u^{drive}, T_u^{el|B}, T_u^{el|R}, T_u^{la|B}, T_u^{la|R})$ the resource vector. The initial backward label representing a fully rested driver at the end depot $d$ is defined as $L_d = (d, \{d\}, T_d)$ with $T_d = (T_d^{cost}, T_d^{load}, T_d^{time}, T_d^{dist}, T_d^{drive}, T_d^{el|B}, T_d^{el|R}, T_d^{la|B}, T_d^{la|R}) := (0, 0, a_d^{bw}, 0, 0, 0, 0, \infty, \infty)$. Since all backward REFs are non-decreasing in the resources $T^{cost}, T^{load}, T^{time}$, $T^{dist}$, $T^{drive}, T^{el|B}$ and $T^{el|R}$ as well as non-increasing in the resources $T^{la|B}$ and $T^{la|R}$, the dominance rules 1 and 2 can be applied as in the forward labeling.

**Example 1.** *We give a concrete numerical example to emphasize the difference in forward and backward labeling. A feasible schedule in which $T^{el|B}$ exceeds $t^{\mathrm{el|B}}$ is depicted in Figure 2.*

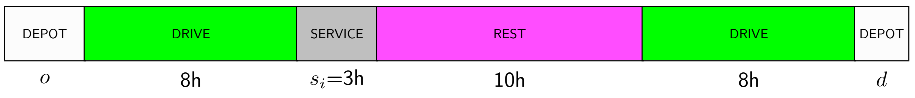

Figure 2: A feasible schedule in which $T^{el|B}$ exceeds $t^{\mathrm{el|B}}$ corresponding to a route $o$-$i$-$d$

*We assume that the demand of vertex $i$ fits in the vehicle and all time windows are not binding, e.g. $[a_j, b_j] = [0, 100]$ for $j = o, i, d$. Hence, no waiting time can occur making the resources $T^{la|B}$ and $T^{la|R}$ obsolete. For the sake of simplicity, we also omit the resources $T^{cost}$ and $T^{load}$. The forward path generated by applying the corresponding REF in the auxiliary network is as follows:*

$$\begin{pmatrix} T_o^{time} & = 0 \\ T_o^{dist} & = 0 \\ T_o^{drive} & = 0 \\ T_o^{el|B} & = 0 \\ T_o^{el|R} & = 0 \end{pmatrix} \xrightarrow[t_{oi}=8]{f_{oi}^{start}} \begin{pmatrix} 0 \\ 8 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[\Delta_{oi}=8]{f_{oi}^{drive}} \begin{pmatrix} 8 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix} \xrightarrow[s_i=3]{f_{oi}^{visit}} \begin{pmatrix} 11 \\ 0 \\ 8 \\ 11 \\ 11 \end{pmatrix} \xrightarrow[t_{id}=8]{f_{id}^{start}} \begin{pmatrix} 11 \\ 8 \\ 8 \\ 11 \\ 11 \end{pmatrix} \xrightarrow[\Delta_{id}=-3]{f_{id}^{rest}} \begin{pmatrix} 21 \\ 8 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[\Delta_{id}=8]{f_{id}^{drive}} \begin{pmatrix} 29 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix} \xrightarrow[s_d=0]{f_{id}^{visit}} \begin{pmatrix} 29 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix}$$

*The forward path starts from vertex $o$, then $t_{oi} = 8$ hours of driving are allowed due to $\Delta_{oi}$. When reaching vertex $i$, the service can still be performed although $\Delta$ gets a negative value. Afterward, a rest is enforced due to $\Delta \leq 0$ complying with the given schedule. Finally, realizing another driving period of eight hours is allowed to reach the depot.*

*The backward path generated by applying the corresponding REFs in the auxiliary backward network is given by:*

$$\begin{pmatrix} T_o^{time} & = -100 \\ T_o^{dist} & = 0 \\ T_o^{drive} & = 0 \\ T_o^{el|B} & = 0 \\ T_o^{el|R} & = 0 \end{pmatrix} \xrightarrow[t_{id}=8]{f_{id}^{visitBW}} \begin{pmatrix} -100 \\ 8 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[\Delta_{id}=8]{f_{id}^{driveBW}} \begin{pmatrix} -92 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix} \xrightarrow[\Delta_{id}=0]{f_{id}^{restBW}} \begin{pmatrix} -82 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[s_i=3]{f_{id}^{startBW}} \begin{pmatrix} -79 \\ 0 \\ 0 \\ \cancel{3}\ 0 \\ \cancel{3}\ 0 \end{pmatrix} \xrightarrow[t_{oi}=8]{f_{oi}^{visitBW}} \begin{pmatrix} -79 \\ 8 \\ 0 \\ \cancel{3}\ 0 \\ \cancel{3}\ 0 \end{pmatrix} \xrightarrow[\Delta_{oi}=\cancel{5}\ 8]{f_{oi}^{driveBW}} \begin{pmatrix} -71 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix} \xrightarrow[s_o=0]{f_{oi}^{startBW}} \begin{pmatrix} -71 \\ 0 \\ 8 \\ 8 \\ 8 \end{pmatrix}$$

*The backward path starts at vertex $d$ and initially $t_{id} = 8$ hours of driving are allowed. Subsequently, a rest can be performed due to $\Delta_{id} = 0$ complying with the given schedule. The driver can now serve vertex $i$ and here the first difference to the forward REFs occurs: The REF $f_{id}^{startBW}$ is used immediately after the REF $f_{id}^{restBW}$ was applied indicated by $T^{el|R} = 0$. Hence, the service time is not taken into account for the computation of $T^{el|B}$ and $T^{el|R}$. The crossed-out values show the resource values without that alternation. Afterward, the trip to the start depot $o$ is started, and a driving period of eight hours should be performed according to the schedule. As can be seen from the crossed-out values, it would not be possible to drive eight hours if the service time would have been added to $T^{el|B}$ and $T^{el|R}$. Hence, without the alternation of the REF $f_{id}^{startBW}$, the given feasible schedule could not be reproduced in the backward labeling. However, according to the values resulting from the correctly adjusted REF, eight hours of driving are allowed and the depot is reached complying with the given schedule.*

In bidirectional labeling algorithms, forward labels are not necessarily propagated until the end depot, and backward labels are not necessarily propagated until the start depot. Instead, labels are propagated only up to a so-called half-way point, thus limiting the overall number of created labels. Suitable forward and backward labels must then be merged to obtain complete $o$-$d$-paths. As described by Salani (2005), this is done using a half-way point test to avoid creating the same path from different pairs of forward and backward labels.

When using the half-way point $h := b_d/2$, we propagate forward labels at a vertex $i \in V$ in the auxiliary network only if $T_i^{time} \leq h$, and backward labels at a vertex $j \in V$ only if $-T_j^{time} > h$. Labels at an intermediate vertex are always propagated irrespective of their time value. When forward and backward labeling is finished, we merge forward and backward labels at all original vertices $i \in V$. A forward label $L_{fw} = (i, S_{fw}, T_{fw})$ at a vertex $i \in V$ is a candidate for merging if $i = d$ or $T_{fw}^{time} > h$. This condition prevents the creation of identical paths from different pairs of forward and backward labels. We check all backward labels at vertex $i$ whether or not they can be merged with the chosen forward label. A merge of a forward label $L_{fw} = (i, S_{fw}, T_{fw})$ and a backward label $L_{bw} = (i, S_{bw}, T_{bw})$ is feasible if the resulting path fulfills the following conditions:

- the path is elementary: $|S^{fw} \cap S^{bw}| = 1$
- the path respects the vehicle capacity: $T_{fw}^{load} + T_{bw}^{load} - q_i \leq Q$
- the path is feasible with respect to the time resource $T_{fw}^{time} - s_i \leq -T_{bw}^{time}$
- the accumulated driving time does not exceed the maximum time allowed: $T_{fw}^{drive} + T_{bw}^{drive} \leq t^{\text{drive}}$
- The time between the last break and rest period of the forward and the backward label do not exceed the maximum time allowed: $-T_{bw}^{la|B} - T_{fw}^{la|B} \leq t^{\text{el|B}}$ and $-T_{bw}^{la|R} - T_{fw}^{la|R} \leq t^{\text{el|R}}$
- Either the sum of the elapsed time since break resources respects the maximum time allowed or a rest or break was performed immediately before the visit in one of both labels:
  $T_{fw}^{el|B} + T_{bw}^{el|B} - s_i \leq t^{\text{el|B}}$ or $T_{fw}^{el|B} \leq s_i$ or $T_{bw}^{el|B} \leq s_i$
- Either the sum of the elapsed time since rest resources respects the maximum time allowed or a rest was performed immediately before the visit in one of both labels:
  $T_{fw}^{el|R} + T_{bw}^{el|R} - s_i \leq t^{\text{el|R}}$ or $T_{fw}^{el|R} \leq s_i$ or $T_{bw}^{el|R} \leq s_i$

The last two conditions model that exceeding the maximum elapsed time allowed at the merge vertex $i$ is possible if only service and waiting times immediately before a rest cause this exceeding. Note that all backward resources referring to a point in time are multiplied with -1 due to the inversion of the network. Moreover, the demand $q_i$ and the service time $s_i$ at vertex $i$ are subtracted in the capacity and time conditions because they are taken into account in both labels.

The reduced cost of a merged path can be obtained as $\bar{c}_r = T_{fw}^{cost} + T_{bw}^{cost}$. After the merge, we perform a final dominance test with dominance rules 1 and 2 for all routes resulting from merged labels. Finally, we add all undominated negative reduced-cost routes to the RMP.

## *4.3. Elementarity*

It is well known that the elementary shortest-path problem with resource constraints is NP-hard in the strong sense (Dror, 1994). Hence, in column-generation algorithms for VRPTWs, many authors solve non-elementary SPPRCs as pricing problems (Desaulniers *et al.*, 2014). This can be done in pseudo-polynomial time (Irnich and Desaulniers, 2005). Although this yields weaker lower bounds and routes with cycles must be removed in the branching process, solving only a relaxed pricing problem often pays off with respect to overall computation time. One recent approach that has been very successfully used for different types of vehicle routing problems is the $ng$-path relaxation introduced by Baldacci *et al.* (2011).

We use the $ng$-path relaxation in the VRTDSP as follows: A specific $ng$-path relaxation requires the definition of neighborhoods $N_i \subset C$ with $i \in N_i$ for all vertices $i \in C$. A forward $ng$-path $P = (o, .., i)$ is a non-necessarily elementary path starting at vertex $o$, ending at vertex $i$ and visiting all vertices within their time window. $P$ defines a forward $ng$-label $L = (i, S_i^{ng}, T_i)$, where $i$ is the last visited vertex and $S_i^{ng}$ is a (generally proper) subset of the visited vertices. The vector $T_i$ contains the same resources as in the elementary formulation. The key point of this relaxations is the update of the set $S_i^{ng}$ via REF $f_{ij}^{start}$, which differs from the update of the set $S_i$ in the elementary formulation: Forward propagation of a label $L = (i, S_i^{ng}, T_i)$ via $f_{ij}^{start}$, produces the new label $L = (n_{ij}, S_{n_{ij}}^{ng}, T_{n_{ij}})$, where $T_{n_{ij}} = f_{ij}^{start}(T_i)$ and $S_{n_{ij}}^{ng} = (S_i^{ng} \cup \{j\}) \cap N_j$. The interpretation is that the new label forgets the visited vertices that are not in the set $N_j$, so that cycles become possible. This change implies that more labels become comparable in the dominance rules, and hence, more labels can be dominated. The definition of a backward $ng$-label and its backward propagation are analogous. For the sake of simplicity, we skip the index $ng$ and write $S$ instead of $S^{ng}$ in the following.

The quality of the root lower bound computed by an $ng$-path relaxation strongly depends on the choice of the neighborhoods $N_i$. We limit the number of neighbors by a constant $\nu$ and test different values of $\nu$ in our computational experiments. For each vertex $i \in C$, we use an $ng$-neighborhood $N_i$ containing $i$ and the $\nu$ closest customers $j$ for which a cycle $(j, i, j)$ would be feasible with respect to time windows and travel and service times. Determining if cycles are possible and defining a good closeness criterion requires the computation of a lower bound for the driving time from $i$ to $j$ that takes the minimum number of rest and break periods between customers $i$ and $j$ into account. Goel and Irnich (2014) have computed the minimum number of mandatory rest and break periods ($k_{ij}^{rest}$ and $k_{ij}^{break}$) between two customers $i, j \in C$ as

$$k_{ij}^{rest} := \max\left\{0, \left\lceil \frac{t_{ij}}{t^{\text{drive}}} - 1 \right\rceil\right\} \text{ and } k_{ij}^{break} := \left\lceil \frac{\max\{0, t_{ij} - (k_{ij}^{rest} + 1)t^{\text{el|B}}\}}{t^{\text{drive}} - t^{\text{el|B}}} \right\rceil .$$

With the help of these two values, a lower bound on the duration of a trip along arc $(i, j) \in A$ can be easily computed as:

$$\hat{t}_{ij} = t_{ij} + k_{ij}^{rest} t^{\text{rest}} + k_{ij}^{break} t^{\text{break}}$$

Now, a cycle $(j, i, j)$ may be possible if $\max\{a_i + \hat{t}_{ij}, a_j\} + \hat{t}_{ji} \le b_i$ and closeness can be defined as $\hat{t}_{ij} + \hat{t}_{ji}$.

## *4.4. Acceleration techniques*

The labeling takes by far the most time in the overall branch-and-price-and-cut algorithm. To speed up the labeling four acceleration techniques are used. First, instead of using a static half-way point in bidirectional labeling, a dynamic half-way point is used to balance the time spent in forward and backward labeling (Tilk *et al.*, 2016). Second, an additional set of unreachable customers is defined to strengthen the dominance as proposed by Feillet *et al.* (2004). Third, the labeling is solved heuristically using a *limited discrepancy search* (LDS, see Feillet *et al.* (2007)). Fourth, a heuristic dominance rule is applied to further strengthen the dominance. Goel and Irnich (2014) used the last three techniques in a similar manner.

*Dynamic half-way point.* The forward and backward labeling can take a very different amount of time due to the asymmetry arising from the time windows and dual prices. This also affects the bidirectional labeling because the number of created and dominated labels in the forward and backward part may differ significantly resulting in a huge difference in the computation times of both parts. Tilk *et al.* (2016) have introduced a dynamic determination of the half-way point to balance the forward and backward parts of the bidirectional labeling.

We adapt this approach to the VRTDSP as follows: Since the time resource $T^{time}$ is non-decreasing, labels can be propagated in ascending order of the time resource. Thereby, the possibility is given to alternate between forward and backward extensions depending on the number of non-processed labels in both parts. Let $z^{fw}$ and $z^{bw}$ be the current time resource values of the next label that should be propagated in forward and backward labeling. Note that $z^{bw}$ is negative due to the inversion of the auxiliary network in backward labeling. Now, the idea is to use different dynamic half-way points $h_{fw}$ and $h_{bw}$ in forward and backward labeling that are initialized at the opposite ends of the time horizon and updated depending on $z^{bw}$ and $z^{fw}$, respectively. More precisely, initially the half-way points are set to $h_{fw} = b_d$ and $h_{bw} = a_0$. Every time a forward label is propagated the backward half-way point is updated to $h_{bw} = \min\{z^{fw}, h_{fw}\}$. Vice versa, when processing a backward label, the forward half-way point is updated to $h_{fw} = \max\{-z^{bw}, h_{bw}\}$. Obviously, as soon as $h_{fw}$ and $h_{bw}$ are equal, both half-way points are fixed and the rest of the labeling continues as the bidirectional labeling with a static half-way point value $h_{fw}$. The impact of this technique is analyzed in the computational results in Section 6.

*Unreachable Customers.* Feillet *et al.* (2004) proposed the use of a set of unreachable customers instead of the set $S$ to strengthen the dominance. Here, we use an additional resource set $U$ in each label to manage the unreachable customers because replacing $S$ by $U$ leads to a problem in the merge step: Given a forward and a backward label with unreachable set $U_{fw}$ and $U_{bw}$, respectively, a customer $i$ can be unreachable in both labels resulting in $|U_{fw} \cap U_{bw}| > 1$ which violates the merge condition although none of the labels has visited $i$ and the merge could be feasible. Therefore, the merge conditions are still tested with the set $S$ while REF preconditions and the dominance are realized with the set $U$.

The update of $U$ is done just as the update of $S$ in $f_{ij}^{visit}$ and $f_{ij}^{startBW}$, respectively. Afterwards, unreachable customers are identified and added to $U$. A customer can become unreachable due to capacity or time window restrictions. While capacity restrictions are easy to check, checking time window restrictions in the VRTDSP is rather complicated due to break and rest periods. However, Goel and Irnich (2014) have proposed a heuristic method to determine an appropriate subset of all unreachable customers using $\hat{t}_{ij}$. Given a label $L = (i, S, T)$ a customer $j$ is unreachable if $T^{load} + q_j > Q$ or $T^{time} + \hat{t}_{ij} > b_j$.

*Limited discrepancy search.* Using heuristic solvers for the pricing problem in branch-and-price algorithms to find negative reduced-cost columns quickly is quite standard. If all heuristic solvers fail in finding a negative reduced-cost column, the exact solver has to be invoked. Feillet *et al.* (2007) have proposed LDS as a heuristic solver. They divide the set of all arcs in good and bad arcs and limit the overall number of bad arcs used in a path by a parameter $\kappa$.

We adapt this method to the VRTDSP as follows: In each iteration of the pricing problem, the set of good arcs is newly determined with respect to the current reduced cost of the arcs. All arcs are sorted by increasing reduced cost and scanned in that order. An arc $(i, j)$ is then added to the set of good arcs, if the number of outgoing good arcs from $i$ and ingoing good arcs in $j$ do not exceed five. In addition, all outgoing arcs of the start depot $o$ and all ingoing arcs in the end depot $d$ are added to the set of good arcs. A binary resource $n^{bad}$ is added to each forward and backward label to count the number of bad arcs traversed in the label. $n^{bad}$ is initialized to zero. The REFs $f_{ij}^{start}$ and $f_{ij}^{visitBW}$ increase $n^{bad}$ by one if $(i, j)$ is a bad arc. Afterwards, the REFs check if $n^{bad}$ is greater than $\kappa$, and if so, the label extension is discarded.

We do not include $n^{bad}$ in the dominance and we also do not forbid merging two labels when the sum of their $n^{bad}$ resources exceed $\kappa$. The former is done to strengthen the (heuristic) dominance. The latter increases the number of generated paths and thereby raises the probability of finding a negative reduced-cost path while not increasing the computational effort. In our computational experiments, we use several solvers with increasing values of $\kappa$, see Section 6.

*Heuristic dominance.* In order to further speed up the solution process of the heuristic solvers, we use a heuristic dominance rule similar to the one proposed by Goel and Irnich (2014). The advantage of a heuristic dominance rule is that more labels are comparable and hence, more labels can be discarded. On the downside, a label may be discarded although it will lead to a pareto-optimal or even a minimum-cost solution. However, this path will be found by the exact pricing problem solver at the latest. In our dominance rule, two labels are only compared with respect to the set $S$ and the resources $T^{cost}$, $T^{load}$, $T^{time}$, $T^{dist}$ and $T^{drive}$. The other resources remain disregarded. We combine LDS with the heuristic dominance rule in our computational experiments.

## 5. Valid Inequalities

This section describes the four classes of valid inequalities that are used in our branch-and-price-and-cut algorithm to strengthen the linear relaxation of the master problem. In the computational results, we test different cutting strategies and the combination of these classes of valid inequalities (see Section 6).

### 5.1. 2-path inequalities

The k-path inequalities were introduced by Kohl *et al.* (1999) for the VRPTW. For the special case k=2, we adapt the 2-path inequalities for the VRTDSP as follows: Let $W \subset C$ be a subset of customers that can not be visited by one single vehicle due to capacity or time window restrictions. Moreover, let $\delta^{-}(W)$

be the set of all arcs $(i,j) \in A$ with $i \in W$ and $j \notin W$. The corresponding 2-path inequality is given by $\sum_{r\in R}\sum_{(i,j)\in\delta^-(W)} b^r_{ij}\lambda_r \geq 2$, where $b^r_{ij}$ is the number of times route $r$ traverses arc $(i,j) \in A$.

Let $\bar{\lambda}_r$ be the value of variable $\lambda_r$ in the current solution of the RMP. We use the heuristic proposed by Kohl *et al.* (1999) to generate candidate sets $W$ of maximal cardinality with $\sum_{r\in R}\sum_{(i,j)\in\delta^-(W)} b^r_{ij}\bar{\lambda}_r < 2$. Each candidate set is then tested whether or not it has to be served by at least two vehicles. The capacity test is done by simply summing up all demands of vertices in $W$ if it fails at least two vehicles are needed. If the capacity fits, we have to check the time window restrictions. This requires the solution of an ESPPRC in the auxiliary network induced by $W \cup \{0,d\}$ where all arcs have identical reduced cost $\bar{c}_{ij} = -1$. Now, the candidate set can be served by a single vehicle, if there exists a path with reduced cost $\bar{c}_r < -|W|$. This decision problem is very time-consuming for larger sets $W$, therefore, we limit the maximal cardinality of candidate sets $W$ by a constant parameter $w^{max}$.

The 2-path inequalities are robust, hence incorporating them needs no adjustments in the structure of the pricing problem. We only need to subtract the current dual value of each 2-path inequality present in the master problems from the reduced cost of the corresponding arcs: Let $\eta_k \geq 0$ be the dual price of the 2-path inequality corresponding to $W_k$, then $\eta_k$ is subtracted from the reduced cost of all arcs $(i,j) \in \delta^-(W_k)$.

### 5.2. *Subset-row inequalities*

Subset-row inequalities were first introduced by Jepsen *et al.* (2008) for the VRPTW. They are Chvatal-Gomory rank-1 cuts based on a subset of the constraints in the master program. For the VRTDSP, a subset-row inequality is defined on a subset of customers in the original network. We restrict ourselves to those inequalities defined on three customers as proposed by Jepsen *et al.* (2008) because they can be separated by straightforward enumeration. The inequality for a customer set $U_k \subset C$, in the following denoted by $SR(U_k)$, is given by $\sum_{r\in R}\lfloor\frac{h^r}{2}\rfloor\lambda_r \leq 1$, where $h^r$ is the number of times route $r$ visits a customer in $U_k$.

The addition of subset-row inequalities in the master problem requires the following adjustments to our pricing problem: Let $\sigma_k \leq 0$ be the dual price of the subset-row inequality $SR(U_k)$. The value $\sigma_k$ must be subtracted from the reduced cost for every second visit to vertices in $U_k$. Therefore, an additional binary resource $sr^k$, one for each inequality $SR(U_k)$, is necessary in the labeling algorithm for indicating the parity of the number of times a vertex in $U_k$ is visited. Note that the same vertex may be visited more than once in an $ng$-path relaxation.

Jepsen *et al.* (2008) have proposed a tailored dominance rule that avoids a point-wise comparison of all resources $sr^k$ and thereby reduces the number of incomparable labels significantly. Here, the dominance rules change in the condition regarding the resource $T^{cost}$ as follows: Let $L = (u,S,T)$ and $L' = (u,S',T')$ be two labels with identical last vertex $u$ and let $H := \{k : sr^k_{L'} = 0, sr^k_L = 1\}$ be the index set of the new resources on which $L$ is inferior compared to $L'$. The condition regarding the resource $T^{cost}$ for label $L$ to dominate label $L'$ changes to $T^{cost} - \sum_{k\in H}\sigma_k \leq T^{'cost}$.

The dominance rules can be strengthened further by taking *unreachable inequalities* into account. A subset-row inequality $SR(U_k)$ is unreachable for a label, if none of the vertices in $U_k$ can be reached due to capacity or time window restrictions. More precisely, $SR(U_k)$ is unreachable for a forward label $L = (u,S,T)$ if $T^{load} > Q - \min_{v\in U_k}\{q_v\}$ or $T^{time} > \max_{v\in U_k}\{b_v - \bar{t}_{uv}\}$. Each unreachable inequality is not taken into account for the dominance procedure, i.e. it is removed from $H$. A similar rule can be defined for backward labels. We can compute the values at the right-hand side of the two inequalities above in a preprocessing step. Note that the computation is not exact, i.e. an inequality could be unreachable although the computations disagree. An exact computation would require the consideration of break and rest times for each label independently which would be too time-consuming.

In addition, the computation of the reduced cost of a merged path in the bidirectional labeling algorithm slightly changes. Let $L_{fw} = (u,S_{fw},T_{fw})$ and $L_{bw} = (u,S_{bw},T_{bw})$ be a forward and a backward label, respectively. The computation of the reduced cost changes in two cases: First, if forward and backward label have both visited an odd number of customers in $U_k$, i.e. $sr^k_{fw} = sr^k_{bw} = 1$, and $u \notin U_k$, then the dual price $\sigma_k$ of the subset-row inequality $SR(U_k)$ has to be subtracted from the reduced cost. Second, if forward and backward label have both visited an even number of customers in $U_k$, i.e. $sr^k_{fw} = sr^k_{bw} = 0$, and $u \in U_k$, then the dual price $\sigma_k$ of the subset-row inequality $SR(U_k)$ has to be added to the reduced cost.

### 5.3. *Strong degree inequalities*

Strong degree inequalities were first introduced by Contardo *et al.* (2014) for the capacitated location-routing problem. A strong degree inequality $SD(i)$ in the VRTDSP can be defined for each customer vertex $i \in C$ as $\sum_{r\in R} g_{ir}\lambda_r \geq 1$, where $g_{ir} = 1$ if route $r$ visits $i$ at least once and $g_{ir} = 0$ otherwise. $SD(i)$ enforces partial elementary regarding vertex $i$. Separation is done by straightforward enumeration.

The addition of strong degree inequalities in the master problem requires the following adjustments to our pricing problem: Let $\psi_i \geq 0$ be the dual price of the strong degree inequality $SD(i)$. The value $\psi_i$ must be subtracted from the reduced cost of a label only when vertex $i$ is visited for the first time. Therefore, an additional binary resource $sd^i$, one for each inequality $SD(i)$, is necessary for the labeling algorithm to indicate if $i$ was visited or not. Recall that the same vertex may be visited more than once in an $ng$-path relaxation.

Similarly to the subset row inequalities, we can also avoid a point-wise comparison of all resources $sd^i$ to strengthen the dominance as proposed by Contardo *et al.* (2015). The dominance rules of the VRTDSP change the condition regarding resource $T^{cost}$ as follows: Let $L = (u, S, T)$ and $L' = (u, S', T')$ be two labels with identical last vertex $u$ and let $G := \{k : sd^k_{L'} = 0, sd^k_L = 1\}$ be the index set of the new resources on which $L$ is inferior compared to $L'$. The condition regarding the resource $T^{cost}$ for label $L$ to dominate label $L'$ changes to $T^{cost} + \sum_{i\in G} \psi_i \leq T^{'cost}$. We can again use unreachable inequalities to further strengthen the dominance. A strong degree inequality $SD(i)$ is unreachable for a forward label $L = (u, S, T)$, if $T^{load} > Q - u_i$ or $T^{time} > b_i - \bar{t}_{ui}$. Each unreachable inequality is removed from the set $G$. A similar rule can be defined for backward labels.

In bidirectional labeling, we have to slightly change the merge procedure. The dual price $\psi_i$ of a strong degree inequality $SD(i)$ must be added to the reduced cost of a merged path if forward and backward label have visited node $i$.

### 5.4. *Dynamic neighborhood extension*

As mentioned in Subsection 4.3, the quality of the root lower bound computed by an $ng$-path relaxation strongly depends on the choice of the neighborhoods $(N_i)_{i\in C}$, but it is not obvious what a good choice is. To overcome this issue, a dynamic neighborhood extension was proposed for several different vehicle and arc routing problems (Roberti and Mingozzi, 2014; Bode and Irnich, 2015; Tilk and Irnich, 2016). This procedure can be seen as adding valid inequalities to the RMP that forbid routes with certain cycles. Herein, the $ng$-neighborhood is determined as proposed in Subsection 4.3 and the linear relaxation is solved. Now, the solution is scanned for cycles. Let $D = (i, ..., i)$ be a cycle in a route in the current solution. The node $i$ is added to the $ng$-neighborhoods $N_j$ of all vertices $j \in D$ to forbid cycle $D$ in subsequent solutions of the pricing problem. Adding new vertices to a neighborhood strengthens the resulting relaxation so that a formerly feasible route can become infeasible. Routes that become infeasible are removed from the RMP. This can be done simply by inspection. Clearly, a larger neighborhood leads to a more difficult pricing problem. Therefore, we limit the maximal size of all neighborhoods $N_i$ by a constant $\nu^{max}$.

## 6. Computational Results

The results reported in this section were obtained using a standard PC with an Intel(R) Core(TM) i7-5930k 3.5 GHz processor and 64 GB of main memory. The algorithms were coded in C++ and compiled at 64 bit with MS-Visual Studio 2013. The callable library of CPLEX 12.6 was used for solving the linear relaxations of the restricted master program in the column-generation algorithm. We tested our algorithm on the 56 benchmark instances for the VRTDSP proposed by Goel (2009) which can be obtained at http://www.telematique.eu/research/downloads. These instances are derived from the VRPTW benchmark instances of Solomon (1987) that can be grouped in six different classes: Randomly distributed customers (R1 and R2), clustered customers (C1 and C2) and a mixed distribution (RC1 and RC2). Instance classes R1, C1 and RC1 have tight time windows and strict vehicle capacity, while C2, RC2, and R2 have wide time windows and loose vehicle capacity. Each instance contains 100 customers and the service time at every customer is set to 60 minutes. Like Goel and Irnich (2014), we create smaller instances by considering only the first 25 or 50 customers.

In the following, we compare different variants of the labeling algorithm, different sizes of the *ng*-neighborhoods and various cutting strategies. Finally, we give detailed results of the best setting found. We set a CPU time limit of two hours for all computations. Preliminary tests showed that applying LDS and heuristic dominance is beneficial. In total, we use six different heuristic solvers with LDS, each with five good arcs. The first three solvers use heuristic dominance and a bad arc limit $\kappa = 0, 1$ or 2. The last three solvers use the exact dominance and the same bad arc limits as above. Moreover, we stop each labeling procedure as soon as 500 or more negative reduced-cost routes have been found.

## 6.1. *Dynamic half-way point*

This subsection analyzes the impact of using a dynamic half-way point. We run two different tests, each with an *ng*-neighborhood size of $\nu = 10$: In a first test, we compare the solution of the linear relaxation of the RMP when either the dynamic or the static version of the half-way point is used in the pricing problem. Table 3 summarizes the results aggregated over the different instance sizes. The table contains the number of successfully solved linear relaxations, the average time in seconds and the average number of pricing problem iterations needed in the static and dynamic version, respectively. The last column indicates the average of the absolute values of the difference in the number of pricing problem iterations. All average values are computed only over the instances that were solved by both versions.

| $\lvert C\rvert$ | no. solved | | avg time | | avg no. pricing | | |
|---|---|---|---|---|---|---|---|
| | sta | dyn | sta | dyn | stat | dyn | \|stat-dyn\| |
| 25 | 56 | 56 | 96.80 | 39.11 | 35.71 | 34.68 | 10.39 |
| 50 | 51 | 54 | 702.08 | 388.32 | 105.55 | 102.20 | 48.20 |
| 100 | 19 | 26 | 2123.98 | 1023.38 | 220.84 | 234.74 | 35.47 |
| all | 126 | 136 | 646.60 | 327.92 | 91.68 | 92.02 | 29.18 |

Table 3: Comparison between using the dynamic and static half-way point

The results of the first test show that the labeling with a dynamic half-way point is superior to the static version. Ten more linear relaxations can be solved and the solution of a single instance takes on average nearly half the time. The average number of pricing problem iterations needed to solve a single instance is nearly identical for both versions but the absolute value of the difference fluctuates over all instances. This is due to the different columns added in the static and dynamic version during the algorithm when using pricing heuristics. Therefore a second test is necessary: We solve the linear relaxation of the RMP while the pricing problem is solved with the static as well as with the dynamic half-way point in each iteration. Thereby, a fair comparison of both labeling algorithms is given because they are solved with the same reduced cost in each iteration.

In the second test, we computed the following values per instance: the ratio dynamic to static of the values for added labels, extended labels, and time needed in pricing. The ratios are computed as the geometrical mean over all pricing iterations. Additionally, we compute for all instances the coefficient of variation (COV) of the dynamic half-way point as standard deviation divided by arithmetic mean. Whenever the linear relaxation is not completely solved within the time limit, the values are taken over the iterations solved up to this point. Table 4 contains the minimum, geometrical mean and maximum of these values, aggregated over the different instance sizes.

The results of the second test confirm the superiority of the dynamic version. The average ratio of the added and extended labels is for all instance sizes nearly identical around 0.87 and 0.74, respectively. This denotes that only 87 % of the labels needed in the static version, are needed in the dynamic version and that only 74 % of the labels extended in the static version are extended in the dynamic version. This results in an average computation time of 63 % for the dynamic version in relation to the static. Moreover, the

| $\lvert C \rvert$ | added labels | | | extended labels | | | time | | | COV | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | min | avg | max | min | avg | max | min | avg | max | min | avg | max |
| 25 | 0.48 | 0.86 | 1.07 | 0.45 | 0.73 | 0.90 | 0.11 | 0.64 | 1.68 | 0.02 | 0.10 | 0.72 |
| 50 | 0.75 | 0.87 | 1.02 | 0.59 | 0.75 | 0.95 | 0.32 | 0.67 | 1.02 | 0.02 | 0.14 | 0.50 |
| 100 | 0.65 | 0.87 | 0.96 | 0.56 | 0.75 | 0.87 | 0.21 | 0.59 | 0.86 | 0.02 | 0.11 | 0.55 |
| all | | 0.87 | | | 0.74 | | | 0.63 | | | 0.12 | |

Table 4: Ratios for using the dynamic and static half-way point with identical reduced cost

detailed results show that the overall problem can be solved faster with the static half-way point in only eleven out of 168 instances, where none of these instances take more than 20 seconds to solve. The coefficient of variation of the dynamic half-way point denotes that during the solution of a single instance, the value of the dynamic half-way point varied by twelve percent on average. This illustrates the positive impact of the dynamic half-way point, because the value of the computational optimal half-way point for a single instance is different for each iteration of the pricing problem due to the current reduced cost. Therefore, we use the labeling with the dynamic half-way point in the following computations.

## 6.2. Neighborhood size

In this subsection, we compare the results for different *ng*-path relaxations with sizes of $\nu = 6, 8, 10, 12, 14$ and 16 when only the linear relaxation is solved. Table 5 contains for all instance sizes the number of times the linear relaxation was successfully solved in the time limit and the average time it has taken. Table 6 contains for all instance sizes the average and maximum improvement of the linear relaxation value when the *ng*-neighborhood size $\nu$ is increased. In order to provide a fair comparison, the maximum and average times and linear relaxation values are taken only over those instances for which the linear relaxation was solved by all *ng*-path relaxations.

| \|C\| | no. of solved root nodes | | | | | | time [sec] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 6 | 8 | 10 | 12 | 14 | 16 | 6 | 8 | 10 | 12 | 14 | 16 |
| 25 | 56 | 56 | 56 | 56 | 56 | 56 | 46.01 | 39.28 | 28.58 | 68.45 | 138.61 | 320.67 |
| 50 | 52 | 52 | 54 | 54 | 51 | 49 | 691.32 | 637.75 | 393.38 | 576.42 | 739.07 | 846.01 |
| 100 | 25 | 27 | 26 | 26 | 26 | 27 | 1414.72 | 1617.15 | 1556.30 | 1387.65 | 1505.89 | 1694.18 |
| all | 133 | 135 | 136 | 136 | 133 | 132 | 515.15 | 524.53 | 416.38 | 477.06 | 590.38 | 744.75 |

Table 5: Comparison of the solution time for different *ng*-neighborhoods

| \|C\| | average increase of the root lower bound [%] | | | | | maximum increase of the root lower bound [%] | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 6->8 | 8->10 | 10->12 | 12->14 | 14->16 | 6->8 | 8->10 | 10->12 | 12->14 | 14->16 |
| 25 | 0.38 | 0.08 | 0.07 | 0.00 | 0.01 | 6.11 | 2.46 | 1.35 | 0.24 | 0.23 |
| 50 | 0.36 | 0.15 | 0.02 | 0.01 | 0.00 | 4.55 | 3.63 | 0.31 | 0.16 | 0.06 |
| 100 | 0.21 | 0.03 | 0.05 | 0.01 | 0.01 | 1.57 | 0.21 | 0.39 | 0.07 | 0.09 |
| all | 0.35 | 0.10 | 0.05 | 0.01 | 0.00 | 6.11 | 3.63 | 1.35 | 0.24 | 0.23 |

Table 6: Comparison of the root lower bound for different *ng*-neighborhoods

As can be seen from Table 5, the number of successfully solved linear relaxations takes the maximum value for $\nu = 10$ and 12. Moreover, the time to solve the linear relaxation takes its minimum for $\nu = 10$. This is surprising because one would expect that a smaller value of $\nu$ would lead to smaller computation times. We ascribe this result to the increasing number of feasible routes for smaller values of $\nu$. Table 6 shows that the increase in the value of the linear relaxation decreases as the value of $\nu$ gets larger, e.g. the average increase is only 0.05 and the maximum 1.35 percent when $\nu$ is increased from 10 to 12. Therefore, we take an $ng$-neighborhood size of $\nu = 10$ for all following computations.

## *6.3. Cutting strategy*

This subsection presents the results for solving Formulation (1) with different cutting strategies. After cutting is finished, we branch on the arcs of the original network to finally ensure integer solutions. Branching is implemented by deleting arcs from the auxiliary network. The following variable-selection strategy is used: Among all arcs $(i, j)$ with a fractional value in the current solution, we choose the one closest to 0.5. Ties are broken by preferring arcs with higher cost $c_{ij}$. Moreover, nodes in the branch-and-bound tree are processed according to the minimum solution value of their linear relaxation.

In a first test, we compare the results for using each class of valid inequalities on its own. A valid inequality is generated when it is violated by more than 0.05. We use the following parameters to define the further cutting strategy: The node level up to which the cuts can be added in the branch-and-bound tree, as well as the maximum number of cuts in total and per iteration. The level of a node in the branch-and-bound-tree is defined as the number of branching decisions taken up to this node. In addition, there are some class specific parameters: The maximum size $\nu^{max}$ up to which the $ng$-neighborhood can be dynamically extended, the maximum size $w^{max}$ of the candidate sets $W$ for the 2-path inequalities and the maximum number $s^{max}$ of subset row cuts a customer can be included in. Pre-tests have shown that it is important to carefully choose these parameters. The best settings found for each class of valid inequalities are summarized in Table 7.

| class | level | maximum number | | miscellaneous |
|---|---|---|---|---|
| | | in total | per iteration | |
| Strong Degree | 0 | 20 | 10 | - |
| Dyn. Ext. | 0 | - | 10 | $\nu^{max} = 20$ |
| 2-path | 0 | 100 | 50 | $w^{max} = 8$ |
| Subset Row | $\infty$ | 20 | 10 | $s^{max} = 2$ |

Table 7: Best settings for applying each class of valid inequalities individually

Table 8 compares the results of applying each class of valid inequalities on its own and the results when no valid inequalities are used. To ensure a fair comparison, we take only the instances into account for which the linear relaxation was solved in the time limit with a fractional solution. The table contains for each instance size the number of remaining instances, the number of solved instances and the average time the solution process has taken in seconds for each setting. The results show that applying strong degree inequalities or the dynamic neighborhood extension has only little impact on the number of solved instances and the solution time. However, the dynamic neighborhood extension is superior to the strong degree inequalities. Applying 2-path inequalities increases the number of solved instances by two and decreases the solution time on average by nearly five percent. Moreover, detailed results show that these two instances can not be solved by any other of the settings. However, the best results are obtained with applying subset row inequalities.

| \|C\| | no. | no. of solved instances | | | | | time [sec] | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | inst | none | SD | DynEx | 2Path | SR | none | SD | DynEx | 2Path | SR |
| 25 | 21 | 21 | 21 | 21 | 21 | 21 | 439.64 | 441.10 | 391.04 | 416.71 | 281.60 |
| 50 | 27 | 10 | 10 | 10 | 11 | 14 | 4754.41 | 4752.17 | 4730.83 | 4484.04 | 4526.05 |
| 100 | 15 | 3 | 3 | 4 | 4 | 4 | 5993.77 | 5993.77 | 5986.75 | 5815.30 | 5643.69 |
| all | 63 | 34 | 34 | 35 | 36 | 39 | 3650.06 | 3649.62 | 3622.45 | 3488.34 | 3419.78 |

Table 8: Results for applying each class of valid inequalities individually

Based on this results we decided to combine the subset row inequalities and the 2-path inequalities together with one or none of the other two classes. Note that we do not combine strong degree inequalities and dynamic neighborhood extension together since they are both used to ensure elementarity. Moreover, we separate them in order: First, 2-path-inequalities because they are robust. Second, subset-row-inequalities because they have the biggest influence. Third, dynamic neighborhood extension or strong degree inequalities. Further tests have confirmed that this order is superior to all other orders. Table 9 contains the number of solved instances and the average solution time per instance size for the three resulting settings. The results for all three settings are similar but all three are superior to applying a single class of valid inequalities on its own. However, applying 2-path inequalities, subset-row inequalities and dynamic neighborhood extensions lead to the best results. This confirms the superiority of the dynamic neighborhood extension compared to the strong degree inequalities.

| $\|C\|$ | no. of solved instances | | | time[sec] | | |
|---|---|---|---|---|---|---|
| | 2path+SR | +DynEx | +SD | 2path+SR | +DynEx | +SD |
| 25 | 21 | 21 | 21 | 186.68 | 183.63 | 182.54 |
| 50 | 16 | 17 | 13 | 4138.18 | 3937.09 | 4178.47 |
| 100 | 5 | 5 | 5 | 5344.13 | 5332.41 | 5342.31 |
| all | 42 | 43 | 39 | 3156.59 | 3067.05 | 3176.51 |

Table 9: Results for applying classes of valid inequalities together

## *6.4. Comparison with Goel and Irnich (2014)*

In this subsection, we compare our best setting found in the previous tests with the results of Goel and Irnich (2014). We also give detailed results of our best setting for each instance individually.

Table 10 shows the results of the comparison aggregated per instance-size. The table contains the number of successfully solved root nodes, the number of the instances solved to optimality, the time the solution of the root node takes on average, the time the overall solution process takes on average over all instances and the time the overall solution process takes on average only over those instances that were solved by both algorithms.

In total, we can solve 24 more instances and the solution time decreases around 21%. Moreover, the average solution time for the instances solved by both algorithms decreases in our setting about more than 60%. The results for solving only the root node are similar: Our algorithm is able to solve 21 more linear relaxations and the solution time decreases around 24 %.

| | Best Setting | | | | | Goel and Irnich (2014) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | no. solved | | time[Sec] | | | no. solved | | time[Sec] | | |
| \|C\| | root | tree | root | tree | solved | root | tree | root | tree | solved |
| 25 | 56 | 56 | 37.86 | 90.51 | 48.86 | 56 | 54 | 117.03 | 404.25 | 152.55 |
| 50 | 54 | 44 | 1052.13 | 2460.74 | 236.58 | 45 | 28 | 2069.04 | 3955.56 | 711.12 |
| 100 | 26 | 12 | 4951.06 | 6000.44 | 419.31 | 14 | 6 | 5733.51 | 6530.21 | 948.63 |
| all | 136 | 112 | 2013.69 | 2850.56 | 133.85 | 115 | 88 | 2639.86 | 3630.01 | 384.56 |

Table 10: Comparison of our best setting with the results in (Goel and Irnich, 2014)

Table 11 contains the detailed results of the best setting for all instances. The first column contains the name of the instance, followed by the solution time in seconds, the gap at the root node in percent, the percentage the root gap was closed at the end of the optimization and the value of the best known solution for instance sizes 25, 50 and 100. Values marked with an * are optimal solution values. The algorithm is able to solve all 25 customer instances in 90 seconds on average. Moreover, the computation time never exceeds 30 minutes. 44 out of 56 instances with 50 customers are solved to optimality and for only seven of them, the computation time exceeds 45 minutes. The 100 customer instances are hard to solve and only twelve optimal solutions can be found.

| inst | 25 customer | | | | 50 customer | | | | 100 customer | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Gap | | UB | Time | Gap | | UB | Time | Gap | | UB |
| | | Root | clo. | | | Root | clo. | | | Root | clo. | |
| C101 | 0.10 | 0.0 | - | 191.17* | 0.45 | 0.0 | - | 362.17* | 6.56 | 0.0 | - | 826.83* |
| C102 | 9.05 | 0.0 | - | 190.08* | 10.06 | 0.0 | - | 361.08* | 1683.34 | 0.0 | - | 826.83* |
| C103 | 52.23 | 0.0 | - | 189.42* | 354.91 | 0.0 | - | 360.42* | 7200.00 | 0.0 | - | 825.17* |
| C104 | 856.35 | 0.1 | 100.0 | 186.67* | 7200.00 | - | - | 358.17 | 7200.00 | - | - | 821.08 |
| C105 | 0.27 | 0.0 | - | 191.17* | 1.17 | 0.0 | - | 362.17* | 13.01 | 0.0 | - | 826.83* |
| C106 | 0.20 | 0.0 | - | 191.17* | 1.68 | 0.0 | - | 362.17* | 64.79 | 0.0 | - | 826.83* |
| C107 | 0.60 | 0.0 | - | 191.17* | 4.32 | 0.0 | - | 362.17* | 148.92 | 0.0 | - | 826.83* |
| C108 | 2.08 | 0.0 | - | 189.75* | 18.58 | 0.0 | - | 360.75* | 599.26 | 0.1 | 100.0 | 825.42* |
| C109 | 39.58 | 0.1 | 100.0 | 187.83* | 192.19 | 0.1 | 100.0 | 358.83* | 7200.00 | 1.0 | 9.7 | 823.50 |
| C201 | 2.58 | 8.7 | 100.0 | 248.00* | 272.85 | 11.7 | 100.0 | 434.75* | 3199.48 | 18.1 | 100.0 | 825.42* |
| C202 | 3.39 | 0.0 | - | 217.83* | 7200.00 | 4.9 | 46.9 | 390.50 | 7200.00 | 17.5 | 0.0 | 782.08 |
| C203 | 10.06 | 0.0 | - | 217.83* | 941.54 | 0.0 | - | 362.75* | 7200.00 | - | - | 703.42 |
| C204 | 32.79 | 0.0 | - | 214.17* | 7200.00 | - | - | 349.50 | 7200.00 | - | - | 652.58 |
| C205 | 1.07 | 0.0 | - | 214.42* | 81.01 | 0.0 | - | 359.33* | 7200.00 | 5.2 | 0.0 | 627.00 |
| C206 | 1.36 | 0.0 | - | 214.42* | 171.61 | 0.0 | - | 359.33* | 2121.84 | 0.0 | - | 585.33* |
| C207 | 38.17 | 0.0 | - | 214.17* | 754.42 | 0.0 | - | 358.58* | 7200.00 | - | - | 642.67 |
| C208 | 4.76 | 0.0 | - | 214.17* | 597.41 | 0.0 | - | 359.08* | 7200.00 | - | - | 639.00 |
| R101 | 0.29 | 1.3 | 100.0 | 502.25* | 1.89 | 0.0 | - | 847.83* | 1021.34 | 0.6 | 100.0 | 1280.50* |
| R102 | 0.33 | 0.0 | - | 446.25* | 33.71 | 0.5 | 100.0 | 753.92* | 7200.00 | 1.0 | 66.0 | 1152.08 |
| R103 | 1.04 | 0.0 | - | 401.08* | 1450.91 | 1.1 | 100.0 | 649.08* | 7200.00 | 12.7 | 0.0 | 1013.33 |
| R104 | 1.47 | 0.0 | - | 359.42* | 7200.00 | 0.6 | 51.6 | 536.42 | 7200.00 | - | - | 935.58 |
| R105 | 0.48 | 0.7 | 100.0 | 438.17* | 90.81 | 0.8 | 100.0 | 749.58* | 7200.00 | 2.4 | 49.1 | 1076.67 |
| R106 | 1.49 | 0.0 | - | 407.08* | 1429.34 | 1.4 | 100.0 | 687.67* | 7200.00 | 12.5 | 0.0 | 1059.08 |
| R107 | 8.64 | 1.4 | 100.0 | 391.83* | 4147.12 | 1.4 | 100.0 | 610.58* | 7200.00 | - | - | 907.00 |
| R108 | 359.27 | 1.6 | 100.0 | 349.42* | 7200.00 | 1.4 | 0.0 | 530.17 | 7200.00 | - | - | 932.75 |
| R109 | 4.66 | 2.3 | 100.0 | 385.08* | 6457.09 | 1.9 | 100.0 | 637.83* | 7200.00 | 7.7 | 3.8 | 964.25 |
| R110 | 99.12 | 1.1 | 100.0 | 354.42* | 2540.30 | 0.6 | 100.0 | 571.92* | 7200.00 | - | - | 926.58 |
| R111 | 5.96 | 0.8 | 100.0 | 387.67* | 7200.00 | 6.6 | 17.6 | 615.58 | 7200.00 | - | - | 972.67 |
| R112 | 700.96 | 1.7 | 100.0 | 337.33* | 7200.00 | 1.8 | 34.3 | 529.42 | 7200.00 | - | - | 851.08 |
| R201 | 0.44 | 0.7 | 100.0 | 463.58* | 17.21 | 0.2 | 100.0 | 798.92* | 2726.06 | 0.7 | 100.0 | 1157.00* |
| R202 | 0.61 | 0.0 | - | 410.75* | 255.73 | 0.4 | 100.0 | 714.33* | 7200.00 | 2.9 | 31.9 | 1063.25 |
| R203 | 5.54 | 1.4 | 100.0 | 391.83* | 2092.42 | 1.6 | 100.0 | 626.00* | 7200.00 | - | - | 952.33 |
| R204 | 34.27 | 1.6 | 100.0 | 355.17* | 4540.60 | 0.8 | 100.0 | 516.17* | 7200.00 | - | - | 905.33 |
| R205 | 1.42 | 0.3 | 100.0 | 404.08* | 173.77 | 0.6 | 100.0 | 695.25* | 7200.00 | 4.0 | 18.2 | 994.08 |
| R206 | 19.76 | 0.5 | 100.0 | 378.08* | 5325.31 | 1.4 | 100.0 | 642.17* | 7200.00 | - | - | 997.42 |
| R207 | 3.64 | 0.0 | - | 367.17* | 6325.04 | 1.5 | 100.0 | 578.42* | 7200.00 | - | - | 961.67 |
| R208 | 102.35 | 1.4 | 100.0 | 341.08* | 7200.00 | 7.0 | 0.0 | 527.33 | 7200.00 | - | - | 944.17 |
| R209 | 7.25 | 1.9 | 100.0 | 376.75* | 2037.77 | 0.8 | 100.0 | 615.58* | 7200.00 | 12.0 | 0.0 | 976.58 |
| R210 | 10.47 | 0.9 | 100.0 | 411.75* | 7200.00 | 5.2 | 27.6 | 681.58 | 7200.00 | 11.4 | 0.0 | 1015.33 |
| R211 | 23.10 | 0.2 | 100.0 | 351.17* | 7200.00 | 8.1 | 13.5 | 595.25 | 7200.00 | - | - | 858.17 |
| RC101 | 0.30 | 0.0 | - | 358.25* | 2.45 | 0.0 | - | 632.58* | 7200.00 | 5.0 | 15.6 | 1287.75 |
| RC102 | 3.78 | 0.0 | - | 335.92* | 41.59 | 0.0 | - | 604.42* | 7200.00 | 7.4 | 0.0 | 1177.08 |
| RC103 | 9.14 | 0.0 | - | 327.08* | 841.76 | 0.0 | - | 584.67* | 7200.00 | - | - | 1119.67 |
| RC104 | 91.48 | 0.0 | - | 299.75* | 3460.04 | 0.0 | - | 522.92* | 7200.00 | - | - | 996.67 |
| RC105 | 0.62 | 0.0 | - | 334.75* | 16.43 | 0.0 | - | 613.75* | 7200.00 | 7.7 | 0.0 | 1217.92 |
| RC106 | 3.18 | 0.0 | - | 310.83* | 92.33 | 0.0 | - | 564.92* | 7200.00 | 5.7 | 0.0 | 1111.67 |
| RC107 | 31.57 | 0.0 | - | 296.33* | 1152.58 | 0.0 | - | 522.67* | 7200.00 | - | - | 1066.17 |
| RC108 | 821.22 | 0.0 | - | 294.50* | 5131.88 | 0.0 | - | 517.67* | 7200.00 | - | - | 1008.08 |
| RC201 | 0.17 | 0.0 | - | 360.50* | 2.31 | 0.0 | - | 684.83* | 439.94 | 0.2 | 100.0 | 1294.58* |
| RC202 | 1.15 | 0.0 | - | 338.17* | 13.65 | 0.0 | - | 613.83* | 7200.00 | 2.5 | 0.0 | 1144.92 |
| RC203 | 4.34 | 0.0 | - | 327.08* | 63.32 | 0.0 | - | 594.92* | 7200.00 | - | - | 1068.25 |
| RC204 | 77.41 | 0.0 | - | 299.75* | 7200.00 | 1.5 | 0.0 | 491.58 | 7200.00 | - | - | 937.08 |
| RC205 | 0.28 | 0.0 | - | 338.08* | 11.65 | 0.0 | - | 631.83* | 7200.00 | 1.2 | 54.6 | 1184.25 |
| RC206 | 0.52 | 0.0 | - | 324.25* | 23.85 | 0.0 | - | 610.17* | 7200.00 | 3.8 | 31.8 | 1104.75 |
| RC207 | 2.67 | 0.0 | - | 298.33* | 226.49 | 0.0 | - | 560.00* | 7200.00 | - | - | 1041.75 |
| RC208 | 1573.67 | 1.4 | 100.0 | 294.50* | 7200.00 | 2.2 | 0.0 | 517.67 | 7200.00 | - | - | 879.08 |
| all | 90.51 | 0.5 | 100.0 | | 2460.74 | 1.2 | 70.1 | | 6000.44 | 4.6 | 32.5 | |

Table 11: Detailed results for best cutting strategy

## 7. Conclusion

This paper has investigated a branch-and-price-and-cut algorithm for *the vehicle routing and truck driver scheduling problem* (VRTDSP) with U.S. hours of service regulations. We presented backward resource extension functions in order to build a bidirectional labeling with a sophisticated merge procedure. The concept of using a dynamic half-way point and different other known techniques, e.g. the *ng*-path relaxation and limited discrepancy search, were used to speed up the solution process of the pricing problem. In addition, valid inequalities for the vehicle routing problem with time windows were adapted to strengthen the linear relaxation. In the computational experiments, we tested the impact of using a dynamic half-way point, different *ng*-neighborhood sizes and cutting strategies. Our findings are the following: First, using a dynamic half-way point reduces the number of extended labels significantly resulting in a huge gain in terms of computation time. Second, choosing a moderate *ng*-neighborhood size of ten provides an excellent tradeoff between the strength of the resulting bounds and the computational effort. Third, a careful choice of the parameters is necessary when applying valid inequalities. Using subset-row inequalities with the right parameters has by far the biggest impact when applying a single class of valid inequalities on its own. Fourth, using a dynamic neighborhood extension is superior to applying strong degree inequalities. Fifth, when combining different classes of valid inequalities the choice of the classes and the order of applying them are crucial. The best setting found was applying 2-path, subset-row inequalities, and the dynamic neighborhood extension in that order. The computational experiments with this setting show that our method delivers improved results for the VRTDSP benchmark instances introduced by Goel (2009). Our algorithm is able to solve all 25 customer instances in less than half an hour to optimality. In addition, nearly 80% of the 50 customer instances were solved to optimality in two hours of computation time. In total, 24 new optimal solutions could be found and the solution time could be significantly reduced compared to Goel and Irnich (2014).

## References

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

Bartodziej, P., Derigs, U., Malcherek, D., and Vogel, U. (2009). Models and algorithms for solving combined vehicle and crew scheduling problems with rest constraints: An application to road feeder service planning in air cargo transportation. *OR Spectrum*, **31**, 405–429.

Bode, C. and Irnich, S. (2015). In-depth analysis of pricing problem relaxations for the capacitated arc-routing problem. *Transportation Science*, **49**(2), 369–383.

Ceselli, A., Righini, G., and Salani, M. (2009). A column generation algorithm for a rich vehicle-routing problem. *Transportation Science*, **43**(1), 56–69.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, **26**(1), 88–102. (Forthcoming.).

Contardo, C., Desaulniers, G., and Lessard, F. (2015). Reaching the elementary lower bound in the vehicle routing problem with time windows. *Networks*, **65**(1), 88–99.

Derigs, U., Kurowsky, R., and Vogel, U. (2011). Solving a real-world vehicle routing problem with multiple use of tractors and trailers and EU-regulations for drivers arising in air cargo road feeder services. *European Journal of Operational Research*, **213**, 309–319.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Desaulniers, G., Desrosiers, J., and Spoorendonk, S. (2010). *The Vehicle Routing Problem with Time Windows: State-of-the-Art Exact Solution Methods*. John Wiley & Sons, Inc.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). *The Vehicle Routing Problem with Time Windows*, volume 18 of *MOS-SIAM Series on Optimization*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition.

Drexl, M. and Prescott-Gagnon, E. (2010). Labelling algorithms for the elementary shortest path problem with resource constraints considering eu drivers' rules. *Logistics Research*, **2**(2), 79–96.

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

Federal Motor Carrier Safety Administration (2011). Hours of service of drivers. Federal Register Vol. 76, No. 248, p. 81134 - 81188.

Feillet, D., Dejax, P., Gendreau, M., and Guéguen, 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.

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, **45**(4), 239 – 256.

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, **43**(1), 17–26.

Goel, A. (2014). Hours of service regulations in the United States and the 2013 rule change. *Transport Policy*, **33**, 48–55.

Goel, A. and Irnich, S. (2014). An exact method for vehicle routing and truck driver scheduling problems. Technical Report No. 33, Jacobs University, School of Engineering and Science, Bremen, Germany.

Goel, A. and Vidal, T. (2014). Hours of service regulations in road freight transport: An optimization-based international assessment. *Transportation Science*, **48**, 391–412.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, **30**(1), 113–148.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, New York, NY.

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

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

Kok, A., Hans, E., and Schutten, J.M.J.and Zijm, W. (2010). A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks. *Flexible Services and Manufacturing Journal*, **22**, 83–108.

Prescott-Gagnon, E., Desaulniers, G., Drexl, M., and Rousseau, L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science*, **44**(4), 455–473.

Rancourt, M.-E. and Paquette, J. (2014). Multicriteria optimization of a long-haul routing and scheduling problem. *Journal of Multi-Criteria Decision Analysis*, **21**(5-6), 239–255. MCDA-12-0037.R1.

Rancourt, M. E., Cordeau, J. F., and Laporte, G. (2013). Long-haul vehicle routing and scheduling with working hour rules. *Transportation Science*, **47**(1), 81–107.

Roberti, R. and Mingozzi, A. (2014). Dynamic ng-path relaxation for the delivery man problem. *Transportation Science*, **48**(3), 413–424.

Salani, M. (2005). *Branch-and-price algorithms for Vehicle Routing Problems*. Ph.D. dissertation, Università degli Studi di Milano, Facoltà die Scienze Matematiche, Fisiche e Naturali, Dipartimento di Tecnologie dell'Informatione, Milan, Italy.

Savelsbergh, M. W. P. and Sol, M. (1998). DRIVE: dynamic routing of independent vehicles. *Operations Research*, **46**, 474–490.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, **35**(2), 254–265.

Tilk, C. and Irnich, S. (2016). Dynamic programming for the minimum tour duration problem. *Transportation Science*. Forthcoming.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2016). Asymmetry helps: Dynamic half-way points for solving shortest path problems with resource constraints faster. Technical Report LM-2016-05, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Xu, H., Chen, Z., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, **37**(3), 347–364.

Zäpfel, G. and Bögl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, **113**, 980–996.