Branch-and-Cut for the Split Delivery Vehicle Routing Problem with Time Windows

Nicola Bianchessi*,[a], Stefan Irnich[a]

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

Abstract

The Split Delivery Vehicle Routing Problem with Time Windows (SDVRPTW) is a notoriously hard combinatorial optimization problem. First, it is hard to find a useful compact Mixed-Integer Programming (MIP) formulation for the SDVRPTW. Standard modeling approach either suffer from inherent symmetries (MIPs with a vehicle index) or cannot exactly capture all aspects of feasibility. Due to the possibility to visit customers more than once, the standard mechanisms to propagate load and time along the routes fail. Second, the lack of useful formulations has rendered any direct MIP-based approach impossible. Up to now, the most effective exact algorithms for the SDVRPTW are branch-and-price-and-cut approaches using a path-based formulation. In this paper, we propose a new and tailored branch-and-cut algorithm to solve the SDVRPTW. It is based on a new relaxed compact model, in which some integer solutions are infeasible to the SDVRPTW. We use known and introduce some new classes of valid inequalities in order to cut off such infeasible solutions. One new class is path-matching constraints that generalize infeasible-path constraints. However, even with the valid inequalities, some integer solutions to the new compact formulation remain to be tested for feasibility. For a given integer solution, we built a generally sparse subnetwork of the original instance. On this subnetwork, all time-window feasible routes can be enumerated and a path-based residual problem is then solved in order to decide on the selection of routes, the delivery quantities, and herewith the overall feasibility. All infeasible solutions need to be cut off. For this reason, we derive some strengthened feasibility cuts exploiting the fact that solutions often decompose into clusters. Computational experiments show that the new approach is able to prove optimality for several previously unsolved instances from the literature.

Key words: Vehicle routing problem, split delivery, time windows, valid inequalities

1. Introduction

In classical vehicle routing problems, it is usually assumed that each customer is served by exactly one vehicle in exactly one visit. When however multiple visits to customers are allowed in goods distribution, this option is called *split delivery*. Allowing split deliveries is somehow inevitable when the demand of some customers exceeds the capacity of every available vehicle. When there is no such a need, allowing split deliveries is nevertheless attractive because it can lead to significant cost reductions.

The *Split Delivery Vehicle Routing Problem* (SDVRP) is the counterpart of the classical *Vehicle Routing Problem* (VRP, Toth and Vigo, 2014), where in the latter problem multiple visits to a customer are forbidden. The SDVRP has been introduced in the literature by Dror and Trudeau (1989, 1990), who showed that very significant savings are possible when allowing split deliveries, both in terms of the total distance traveled and the number of vehicles employed. In particular, Archetti *et al.* (2006b) proved that savings up to 50 % are possible in distance traveled, and this bound is tight (assuming the validity of the triangle inequality). In the last decade, the interest towards the class of vehicle routing problems with split deliveries was rapidly increasing. The reader is referred to the recent surveys by Archetti and Speranza (2012) and Irnich *et al.* (2014) on the topic.

Despite of the increasing of interest over time, the *Split Delivery Vehicle Routing Problem with Time Windows* (SDVRPTW) has received limited attention. The SDVRPTW is the relaxation of the *Vehicle*

*Corresponding author.
Email address: nbianche@uni-mainz.de (Nicola Bianchessi)

Routing Problem with Time Windows (VRPTW, Desaulniers *et al.*, 2014) in which split deliveries are allowed. Frizzell and Giffin (1995), Mullaseril *et al.* (1997), and Sepúlveda *et al.* (2014) addressed the problem by means of constructive and improvement heuristics. In (Ho and Haugland, 2004), a tabu search algorithm is presented. Gendreau *et al.* (2006) introduced the first exact algorithm to solve the problem. Their branch-and-price-and-cut algorithm was able to solve instances with up to 50 customers. Later, Desaulniers (2010) proposed an alternative branch-and-price-and-cut algorithm. While Gendreau *et al.* (2006) decide on the quantities to deliver at the master problem level, Desaulniers (2010) handles the quantities to deliver at the subproblem level, avoiding the dynamic insertion of an exponential number of constraints in the master problem, that is, one capacity constraint for each generated route. The new branch-and-price-and-cut algorithm was able to solve 176 benchmark instances to optimality within one hour of computational time, including one 100-customer instance. Afterwards, Archetti *et al.* (2011b) proposed an enhanced version of the algorithm of Desaulniers (2010). The authors proposed a tabu search algorithm for accelerating the solution of the subproblem. To improve the value of the lower bounds computed in the search tree, they introduced extensions of several classes of valid inequalities together with a new heuristic separation algorithm for the k-path cuts, originally proposed by Kohl *et al.* (1999). Thanks to these enhancements, the number of benchmark instances solved to optimality within one hour of computational time increased from 176 to 262. A recent paper by Luo *et al.* (2016) considers a generalization of the SDVRPTW in which linear weight-related costs are considered. To test their branch-and-price-and-cut algorithm on the SDVRPTW benchmark, the authors disregard any weight-related costs so that their approach becomes very similar to the one of Archetti *et al.* (2011b), finally delivering 264 of 504 optimally solved instances.

In this paper, we propose a new and tailored branch-and-cut algorithm to solve the SDVRPTW. It is based on a new compact formulation, which in fact defines a relaxation of the problem. This means that some integer solutions to the relaxed formulation are infeasible to the SDVRPTW. We use known and introduce two new classes of valid inequalities in order to strengthen the relaxed compact formulation and possibly cut off solutions which are infeasible to the SDVRPTW. The first class of new valid inequalities is the extension to the SDVRPTW of the infeasible-path constraints proposed in Ascheuer *et al.* (2000, 2001) for the asymmetric *Traveling Salesman Problem with Time Windows* (TSPTW). The other new class is the path-matching constraints that generalize infeasible-path constraints. However, even with these valid inequalities, integer solutions to the new compact formulation remain to be tested for feasibility. Any given integer solution to the relaxed formulation induces a generally sparse subnetwork of the original instance. On this subnetwork, all time-window feasible routes can be enumerated. An extended set covering problem is then solved to decide on the selection of routes, the delivery quantities, and herewith the overall feasibility. All proved infeasible solutions are cut off from the feasible region of the relaxed problem. The solution approach extends and improves the branch-and-cut algorithm coined by Archetti *et al.* (2014) for the SDVRP. One important improvement is that we derive strengthened feasibility cuts exploiting the fact that solutions often decompose into clusters. Computational experiments show that our new solution approach is able to solve several previously unsolved benchmark instances, increasing overall the number of benchmark instances solved to optimality within one hour of computational time.

The remainder of the paper is organized as follows. In Section 2, we recall the definition of the SDVRPTW and summarize several properties that are known to hold for some optimal SDVRPTW solutions. In Section 3, we present the branch-and-cut algorithm for solving the SDVRPTW. Experimental results are presented in Section 4 before final conclusions are drawn in Section 5.

2. Problem Definition

The SDVRPTW can be defined on a directed graph $G = (V, A)$ with vertex set V and arc set A. The vertex set V consists of the set $N = \{1, \ldots, n\}$ that represents the n customers and vertices 0 and $n+1$ that both represent the depot where vehicle routes start and end, respectively. Each customer $i \in N$ has a positive demand d_i that has to be fulfilled by one or more visits starting within a given time window $[e_i, l_i]$. If a vehicle arrives at customer i prior to e_i, it must wait until e_i before starting the delivery. The planning horizon is modeled with the help of the time window $[e_0, l_0] = [e_{n+1}, l_{n+1}]$ of the depot. Each arc $(i, j) \in A$ represents a feasible movement of a vehicle from the location of i to the location of j characterized by a non-negative travel time t_{ij} and travel cost c_{ij}. As common practice, the additional arc $(0, n+1)$ is used to model idle vehicle. We assume that the travel time t_{ij} includes the service time (if any) at i. For each pair of vertices $i, j \in V, i \neq j$, there exists an arc $(i, j) \in A$ if $e_i + t_{ij} \leq l_j$. A fleet of K homogeneous vehicles each with a capacity of Q is available. The vehicles are initially housed at the depot 0 and have to return to the depot $n+1$ at the end.

A route is modeled as a path from 0 to $n+1$ in G. It is feasible if the total demand delivered at the visited customers does not exceed the vehicle capacity and the time window constraints are respected at the visited locations. The SDVRPTW consists of determining a set of least-cost feasible routes such that all customers' demands are met.

From now on, throughout the paper we will assume that the triangle inequality holds for travel times t_{ij} and costs c_{ij}, and that the service times at the customers are constant and, in particular, independent of the quantity delivered. Given these assumptions, it is possible to prove that there exists an optimal solution to the SDVRP(TW) in which:

Property 1. Two routes share at most one split customer (Dror and Trudeau, 1990);

Property 2. Each arc between two vertices representing customers is traversed at most once (Gendreau *et al.*, 2006);

Property 3. For each pair of reverse arcs between two customers at most one of them is traversed (Desaulniers, 2010);

Property 4. All routes are elementary (Desaulniers, 2010).

Moreover, we will assume that all customer time windows are reduced so that $e_i \geq e_0 + t_{0i}$ and $l_i \leq l_{n+1} - t_{i,n+1}$ holds for all customers $i \in N$.

If the vehicle capacity Q and all demands d_i for $i \in N$ are integer, then there exists an optimal solution to the SDVRPTW fulfilling Properties 1–4 and

Property 5. All delivery quantities are positive integers (Archetti *et al.*, 2006a, 2011a).

3. Branch-and-Cut Algorithm

In this section, we present the branch-and-cut algorithm we devised for solving the SDVRPTW. In Section 3.1, we define the relaxed compact formulation for the SDVRPTW and show how an optimal solution to this formulation may not be feasible to the original problem. In Section 3.2, we recall the old and introduce the new feasibility checking procedure and feasibility cuts. Finally, in Section 3.3, we present the valid inequalities used in order to strengthen the relaxed formulation and to cut off solutions which are infeasible to the SDVRPTW.

3.1. Relaxed Compact Formulation

The fundamental difficulty of developing a good compact formulation for the SDVRPTW comes from several sources. First, as we want to use such a formulation within a MIP solver, it should not have variables with vehicle indices (three-index formulation). Otherwise, the inherent symmetry makes any known branching scheme ineffective. Symmetry breaking constraints (see, e.g., Fischetti *et al.*, 1995) can only mitigate the negative effects of symmetry. Second, the fact that customers can be visited by several vehicles make it impossible to attach unique resource variables to the vertices, e.g., variables indicating the accumulated customer demand and the service time. Hence, MTZ-like formulations (see Miller *et al.*, 1960) are not directly applicable in the split-delivery context. Third, the formulation proposed by Maffioli and Sciomachen (1997) for the sequential ordering problem shows that resource variables may be associated with arcs. Even if we can exploit Property 2 and associate time variables with arcs between customers, there remains the problem that arcs between depot and customers (or vice versa) may be traversed by more than one vehicle. Hence, no unique time variables can be associated with these arcs.

Our relaxed compact formulation is a two-commodity flow formulation with additional variables and constraints. The first commodity represents the available vehicles and the second represents the service time imposed by the routes. The formulation uses

(i) integer variables z_i indicating the number of times vertex $i \in N$ is visited by the vehicles,
(ii) integer flow variables x_{ij} indicating the flow of vehicles along arc $(i,j) \in A$, and
(iii) non-negative continuous flow variables T_{ij} indicating the service start time at $i \in N$ if a vehicle directly travels from $i \in N$ to $j \in N$.

Note that the continuous flow variables are defined only for arcs in $N \times N$. In this sense, time flows originate and terminate at vertices in N. In the remainder, we will refer to T_{ij} as *service time flow variables*.

We use the following notation. Symbols $\Gamma^+(S)$ and $\Gamma^-(S)$ denote the forward and backward star of $S \subseteq N$, respectively. For the sake of simplicity, we write $\Gamma^+(i)$ and $\Gamma^-(i)$ for singleton sets $S = \{i\}$. We define $A(N) = \{(i,j) \in A : i \in N, j \in N\}$, $\Gamma_N^+(S) = \Gamma^+(S) \cap A(N)$, and $\Gamma_N^-(S) = \Gamma^-(S) \cap A(N)$. Again, we write $\Gamma_N^+(i)$ and $\Gamma_N^-(i)$ for singleton sets $S = \{i\}$. Finally, we define $K_S = \lceil \sum_{i \in S} d_i/Q \rceil$ as the minimum number of vehicles required to serve customers in set $S \subseteq N$.

The relaxed two-commodity flow formulation for the SDVRPTW is as follows:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1a}$$

$$\sum_{(h,i)\in\Gamma^-(i)} x_{hi} = \sum_{(i,j)\in\Gamma^+(i)} x_{ij} = z_i \qquad i \in N \tag{1b}$$

$$\sum_{(0,j)\in\Gamma^+(0)} x_{ij} = K \tag{1c}$$

$$\sum_{(i,j)\in\Gamma^+(S)} x_{ij} \geq K_S \qquad S \subseteq N,\ |S| \geq 2 \tag{1d}$$

$$e_i x_{0i} + \sum_{(h,i)\in\Gamma_N^-(i)} T_{hi} + \sum_{(h,i)\in\Gamma_N^-(i)} t_{hi}x_{hi} \leq \sum_{(i,j)\in\Gamma_N^+(i)} T_{ij} + l_i x_{in+1} \qquad i \in N \tag{1e}$$

$$e_i x_{ij} \leq T_{ij} \leq l_i x_{ij} \qquad (i,j) \in A(N) \tag{1f}$$

$$z_i \geq \lceil d_i/Q \rceil \text{ and integer} \qquad i \in N \tag{1g}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A(N) \tag{1h}$$

$$x_{ij} \geq 0 \text{ and integer} \qquad (i,j) \in A \setminus A(N) \tag{1i}$$

The objective function (1a) calls for the minimization of the total travel costs. Constraints (1b) impose flow conservation for the vehicle flow variables. The fleet size constraint is (1c). Constraints (1d) partially impose capacity constraints and prevent the generation of paths that are not connected to the depot; an example showing that (1d) is not sufficient is discussed below. Constraints (1e) and (1f) impose conservation for the service time flow, ensure consistency between the T_{ij} and x_{ij} variable values, and partially ensure time window prescriptions. Finally, constraints (1g) and (1i) define the domains of the integer variables. Note that the binary requirement in (1h) results from Property 2.

An optimal solution to (1) may not be feasible for the SDVRPTW as illustrated in Figure 1. The instance depicted in Figure 1(a) shows that time window constraints can be violated by an integer solution to (1). In this instance, the depicted arcs have cost and travel time equal to 1, while all other arcs (not shown) have cost and travel time equal to 2. The demand d_i and the time window $[e_i, l_i]$ of the $n = 5$ customers are presented close to each customer $i \in \{1, 2, \ldots, 5\}$. The depot time window is assumed to be non-constraining, i.e., $[e_0, l_0] = [e_{n+1}, l_{n+1}] = [0, 10]$. The capacity of the vehicles is $Q = 10$. The depicted arcs having flow 1 form the unique optimal solution to the relaxed model (1). With regard to demands and vehicle capacity, this solution can be converted into a feasible SDVRP solution, e.g., using the two routes $(0, 1, 3, 4, n+1)$ and $(0, 2, 3, 5, n+1)$. However, with regard to time-window constraints, there is no feasible SDVRPTW solution because neither the route $(0, 1, 3, 4, n+1)$ nor the route $(0, 1, 3, 5, n+1)$ is time-windows feasible so that customer 1 cannot be visited by a feasible route using exclusively arcs with positive flow. However, the following assignments $T_{13} = 3$, $T_{23} = 1$, and $T_{34} = T_{35} = 3$ to the service time flow variables are feasible for model (1).

In Figure 1(b), we present another example showing that integer solutions to (1) can violate the capacity constraints. We consider the same setting as in Figure 1(a) except that time windows are not binding and demands have changed according to the depicted values. Note first that the solution does not violate any capacity constraints (1d). However, neither route $(0, 1, 3, 4, n+1)$ nor route $(0, 2, 3, 4, n+1)$ is feasible, since the demand of the customers with only one visit, i.e., $d_1 + d_4 = d_2 + d_4 = 11$ exceeds the capacity $Q = 10$. Hence, customer 4 cannot be serviced by any feasible SDVRPTW route resulting from arc flows equal to 1 in the depicted solution.

3.2. Feasibility Checking

Recall that every time a feasible integer solution to the relaxed formulation (1) is found, a procedure must check if the solution is also feasible to the SDVRPTW. If not, a feasibility cut must be inserted to

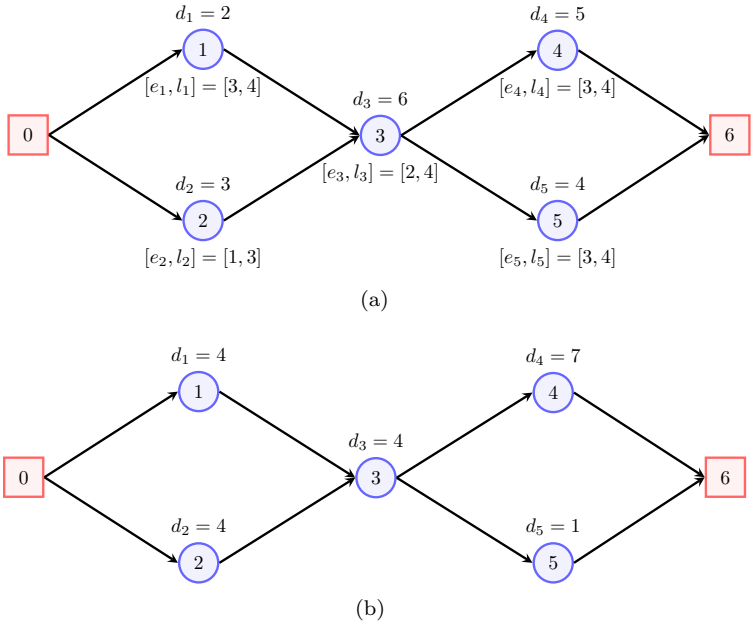

Figure 1: Optimal solutions to formulation (1) that are infeasible for the SDVRPTW.

cut off the proved infeasible solution from the feasible region of the relaxed problem. In Section 3.2.1, we first describe how the approach proposed by Archetti *et al.* (2014) for the SDVRP can be extended to the SDVRPTW. Then, in Section 3.2.2, we present improvements to this basic approach.

3.2.1. Basic Approach

Let $\bar{s} = (\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{T}})$ be an integer solution to the relaxed formulation (1), possibly augmented by branching and cutting constraints. Let $\bar{w} = \bar{\mathbf{c}}^\top \bar{\mathbf{x}}$ denote the cost of the solution.

For any subset $\bar{V} \subseteq V$, we define a residual network induced by the active vehicle flow variables. We will do this not only for $\bar{V} = V$ but also for partial solutions as explained in the next section. Moreover, let $H(\bar{V}, \bar{\mathbf{x}}) = (\bar{V}, \bar{A})$ be defined by $\bar{A} = \{(i, j) \in A \cap (\bar{V} \times \bar{V}) : \bar{x}_{ij} \geq 1\}$. Let $\bar{R}$ be the set of all elementary 0-$(n+1)$-paths (routes) in $H(\bar{V}, \bar{\mathbf{x}})$ satisfying all time-window constraints. We generate the route set $\bar{R}$ by exploring $H(\bar{V}, \bar{\mathbf{x}})$ in a depth-first way.

An instance of the SDVRPTW, defined on the basis of $\bar{V}$ and $\bar{\mathbf{x}}$ imposing the route set $\bar{R}$, can be modeled by a path-based formulation. Some additional notation is required. Let c^r be the cost of route $r \in \bar{R}$ and $\bar{N}(r) \subseteq \bar{N}$ be the subset of customers visited by route $r \in \bar{R}$ using the definition $\bar{N} = \bar{V} \setminus \{0, n+1\}$. We distinguish between routes $\bar{R}^s$ visiting a single customer, i.e., routes of the form $(0, v, n+1)$ for $v \in N$, and routes $\bar{R}^m$ visiting more than one customer. Obviously, $\bar{R} = \bar{R}^m \cup \bar{R}^s$ and $\bar{R}^m \cap \bar{R}^s = \varnothing$. Moreover, let b_{ij}^r be a binary arc indicator equal to 1 if arc $(i, j) \in \bar{A}(\bar{N})$ is used in route $r \in \bar{R}$, and 0 otherwise.

The path-based formulation for the SDVRPTW, defined relatively to $\bar{V}$ and $\bar{\mathbf{x}}$, uses

(i) nonnegative integer and binary variables λ^r indicating the number of vehicles assigned to route $r \in \bar{R}^s$ and $\bar{R}^m$, respectively, and

(ii) non-negative continuous variables δ_i^r indicating the quantity delivered to customer $i \in \bar{N}(r)$ by route $r \in \bar{R}$,

and it reads as follows:

$$w_{\bar{R}} = \min \sum_{r \in \bar{R}} c^r \lambda^r \tag{2a}$$

$$\text{s.t.} \sum_{r \in \bar{R}: i \in \bar{N}(r)} \delta_i^r \geq d_i \qquad i \in \bar{N} \tag{2b}$$

$$\sum_{i \in \bar{N}(r)} \delta_i^r \leq Q \lambda^r \qquad r \in \bar{R} \tag{2c}$$

$$\sum_{r \in \bar{R}} \lambda^r \leq K \tag{2d}$$

$$\sum_{r \in \bar{R}} (b^r_{ij} + b^r_{ji}) \lambda^r \leq 1 \qquad (i,j),(j,i) \in \bar{A}(\bar{N}), i < j \tag{2e}$$

$$\delta^r_i \geq 0 \qquad i \in \bar{N}, r \in \bar{R} \tag{2f}$$

$$\lambda^r \in \{0,1\} \qquad r \in \bar{R}^m \tag{2g}$$

$$\lambda^r \geq 0 \text{ and integer} \qquad r \in \bar{R}^s \tag{2h}$$

The objective function (2a) minimizes the cost of all routes in use. If the model (2) is infeasible, we set $\bar{w}_{\bar{R}} = \infty$. Constraints (2b) ensure that customer demands are met. Vehicle capacity constraints are imposed by (2c). Constraint (2d) guarantees that the fleet size is respected. Property 3 implies constraints (2e). Finally, constraints (2f)–(2h) define the domains of the δ^r_i and λ^r variables.

Note that constraints (2b)–(2h) do not impose that each arc $(i,j) \in \bar{A}$ is traversed exactly $\bar{x}_{ij}$ times by the selected routes. Hence, alternative SDVRPTW solutions are possible, and improving solutions are found whenever $\bar{w}_{\bar{R}} < \bar{w}$. Moreover, customer visits with zero deliveries are possible in (2), i.e., $\lambda^r > 0$ but $\delta^r_i = 0$ for some $i \in \bar{N}(r)$. Due to the validity of the triangle inequality, improving (or at least not worse) feasible solutions can be derived by removing customers with a delivery quantity of 0 from the routes in a solution to (2). Thus, we apply a greedy post-processing procedure in order to identify high quality solutions as early as possible in the course of the branch-and-cut. For the sake of exposition, we assume that $w_{\bar{R}}$ is updated to the value of such an improving solution whenever one is detected.

We strengthen formulation (2a)–(2h) by the following additional constraint:

$$\sum_{r \in \bar{R}} c^r \lambda^r \leq \bar{w}^* \tag{2i}$$

This constraint imposes an upper bound on the objective value $\bar{w}_{\bar{R}}$, where $\bar{w}^*$ is the upper bound to the SDVRPTW stored in the branch-and-cut algorithm.

In the basic approach, we restrict ourselves to residual networks $H(\bar{V}, \bar{\mathbf{x}})$ for the complete vertex set $\bar{V} = V$ and $\bar{\mathbf{x}}$ values that are the arc flow variables of a solution $\bar{s} = (\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{T}})$ to the relaxed model. We summarize what actions the possible outcomes of formulation (2) impose:

(i) $\bar{w}_{\bar{R}} \leq \bar{w}$: Since also $\bar{w} \leq \bar{w}^*$ holds, a new and globally improving feasible integer solution to the SDVRPTW has been found. The best known solution (value) can be updated by $\bar{w}^* := \bar{w}_{\bar{R}}$ and the branch-and-bound node can be terminated.

(ii) $\bar{w}_{\bar{R}} > \bar{w}$: The current integer solution $\bar{s}$ is infeasible. A feasibility cut must be added (see below). Moreover, the resulting branch-and-bound node must be further examined. It is worth noting that the upper bound $\bar{w}^*$ can however be updated by $\bar{w}^* := \bar{w}_{\bar{R}}$ if $\bar{w}_{\bar{R}} < \bar{w}^*$ holds.

As in the branch-and-cut of Archetti *et al.* (2014) for the SDVRP, the feasibility cut that excludes the current integer solution $\bar{s} = (\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{T}})$, here independent from the time schedule given by $\bar{\mathbf{T}}$, is

$$\sum_{(i,j) \in A \setminus \bar{A}} x_{ij} \geq 1. \tag{3}$$

Inequality (3) imposes that the set of active vehicle flow variables must be different from the one defining the solution $\bar{s}$. The inequality is globally valid for formulation (1).

3.2.2. Improvements

Three types of improvements compared to the basic approach are implemented in our branch-and-cut implementation. We present them now.

Extended Arc Set $\bar{A}$. Increasing the underlying arc set $\bar{A}$ defining the residual network $H(\bar{V}, \bar{\mathbf{x}}) = (\bar{V}, \bar{A})$ leads to a larger set of routes $\bar{R}$ and herewith to generally better feasible integer SDVRPTW solutions when solving the path-based formulation (2). At the downside, the size of the the path-based formulation (2) increases leading to generally longer computation times. However, we found that adding all depot arcs is often beneficial because the resulting formulation (2) remains solvable and often more and better improving integer solutions (w.r.t. the current objective value $\bar{w}$) are found. Hence, we enlarge $\bar{A}$ and define it as

$$\bar{A} = \{(i,j) \in A : \bar{x}_{ij} \geq 1\} \cup \{(0,j) : j \in \bar{N}\} \cup \{(i,n+1) : i \in \bar{N}\}.$$

The resulting larger set of routes $\bar{R}$ offers the possibility to use subroutes of the original routes generated from the residual network $H(\bar{V}, \bar{\mathbf{x}})$.

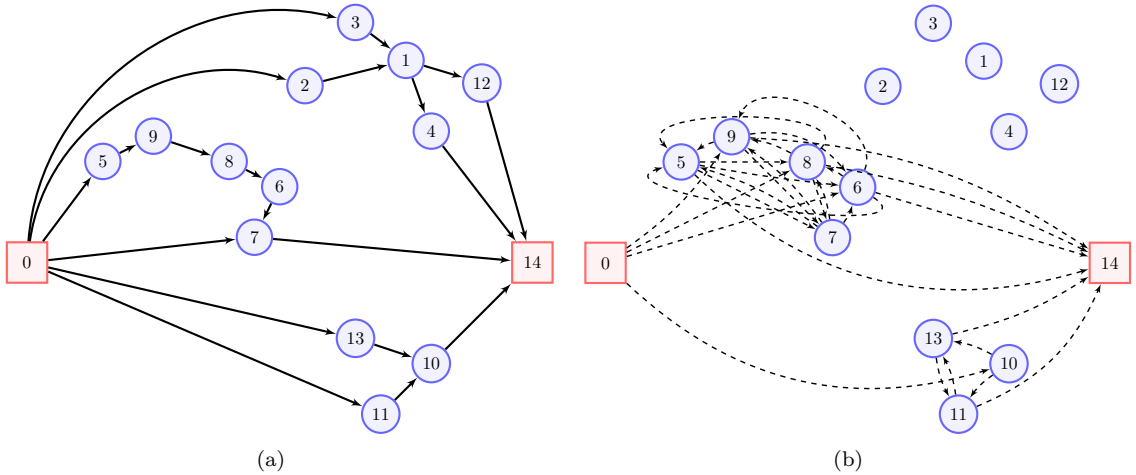

Figure 2: (a) An integer solution to the relaxed formulation which is infeasible for SDVRPTW and the associated residual network $H(V, \bar{\mathbf{x}})$, (b) Arcs that occur in the left-hand side of the standard feasibility cut (3) but not in the left-hand side of the lifted feasibility cut (5).

Reduced Path-Based Formulation. In order to accelerate the solution of (2) by the MIP solver, we can significantly reduce the number of continuous variables δ_i^r in this formulation. Let $\bar{S} = \{i \in \bar{N} : \bar{z}_i \geq 2\}$ be the set of customers receiving split deliveries (*split customers*) in solution $\bar{s}$. We can define variables δ_i^r only for split customers $i \in \bar{S}$ and routes $r \in \bar{R}$ with $i \in \bar{N}(r)$. For the *non-split customers* $i \in \bar{N} \setminus \bar{S}$, we know that the delivery quantity (before modeled by a variable δ_i^r) is identical to $d_i \lambda^r$. Hence, we can reformulate demand fulfillment and capacity constraints (2b) and (2c) and variable domains (2f) as

$$\sum_{r \in \bar{R} \,:\, i \in \bar{N}(r)} \delta_i^r \geq d_i \qquad i \in \bar{S} \tag{4a}$$

$$\sum_{r \in \bar{R} \,:\, i \in \bar{N}(r)} \lambda_i^r \geq 1 \qquad i \in \bar{N} \setminus \bar{S} \tag{4b}$$

$$\sum_{i \in \bar{S} \cap \bar{N}(r)} \delta_i^r + \sum_{i \in (\bar{N} \setminus \bar{S}) \cap \bar{N}(r)} d_i \lambda_i^r \leq Q \lambda^r \qquad r \in \bar{R} \tag{4c}$$

$$\delta_i^r \geq 0 \qquad r \in \bar{R}, i \in \bar{S} \cap \bar{N}(r) \tag{4d}$$

so that the improved formulation becomes (2a), (2e), (2g)–(2d), and (4). While (4a) is the pendant to (2b) for the split customers, constraints (4b) ensure that each non-split customer receives its entire demand when visited once (note that we assume that travel costs and times fulfill the triangle inequality). The new vehicle capacity constraints are given by (4c).

Lifting of Feasibility Cuts. We now show how the feasibility cuts (3) can be lifted. Integer solutions $\bar{s}$ to (1) often consist of independent clusters. Formally, let $\{\bar{N}^c : c \in \mathcal{C}\}$ be the set of weakly connected components of $H(V, \bar{\mathbf{x}})(N)$, i.e., of the vertex-induced subgraph of $H(V, \bar{\mathbf{x}})$ induced by the customers N. Smaller SDVRPTW instances can now be defined by $\bar{V}^c = \bar{N}^c \cup \{0, n+1\}$.

An example of an integer solution to the relaxed formulation which is infeasible for the SDVRPTW is displayed in Figure 2(a). Here, one can see $H(V, \bar{\mathbf{x}})$ and the three weakly connected component consist of $\bar{N}^1 = \{1, 2, 3, 4, 12\}$, $\bar{N}^2 = \{5, 6, 7, 8, 9\}$, and $\bar{N}^3 = \{10, 11, 13\}$.

The lifting procedure considers each weakly connected component: For each $c \in \mathcal{C}$, we define $\bar{x}_{ij}^c = \bar{x}_{ij}$ if $(i, j) \in \bar{V}^c \times \bar{V}^c$, and 0 otherwise. Then, we build $H(\bar{V}^c, \bar{\mathbf{x}}^c)$, generate the routes $\bar{R}$ over $H(\bar{V}^c, \bar{\mathbf{x}}^c)$, and solve the resulting formulation (2). Note that, in order to speed up the solution process, we do not consider an extended arc set here, and we impose using each arc $(i, j) \in \bar{V}^c \times \bar{V}^c$ exactly $\bar{x}_{ij}^c$ times. The additional constraints to insert into formulation (2) are of the form $\sum_{r \in \bar{R}} b_{ij}^r \lambda^r = \bar{x}_{ij}^c$, $(i, j) \in A$. The objective value $w_{\bar{R}}^c := w_{\bar{R}}$ must be compared against $\bar{w}^c := \bar{\mathbf{c}}^\top \bar{\mathbf{x}}^c$. If (2) is infeasible or $w_{\bar{R}}^c > \bar{w}^c$, then we

add the following lifted feasibility cut defined w.r.t. the weakly connected component $\bar{N}^c$:

$$\sum_{(i,j)\in\widehat{A}^c} x_{ij} \geq 1, \tag{5}$$

where the arc set $\widehat{A}^c$ defining the left-hand side is

$$\widehat{A}^c = \{(i,j) \in A \cap (\bar{V}^c \times \bar{V}^c) : \bar{x}_{ij} = 0\} \cup \Gamma_N^+(\bar{N}^c) \cup \Gamma_N^-(\bar{N}^c).$$

The lifted feasibility cut (5) imposes that either the set of active vehicle flow variables associated with the internal arcs of the component $\bar{N}^c$ must be different from the ones positive in the solution $\bar{s}$ or the component $\bar{N}^c$ itself must change. The inequality is globally valid. Thus, whenever $\bar{s}$ has been proved to be infeasible to the SDVRPTW, it can be cut off by imposing to change the current solution for at least one connected component $c' \in \mathcal{C}$. It happens regularly that lifted feasibility cuts for several components can be added at the same time.

Note that $\widehat{A}^c \subseteq A \setminus \bar{A}$ holds by definition of $\bar{A}$. Therefore, the left-hand side of (5) comprises less variables (in case of two or more components) as the original feasibility cut (3). In the example of the infeasible integer solution Figure 2(a), the first weakly connected component $\bar{N}^1 = \{1, 2, 3, 4, 12\}$ imposes a lifted feasibility cut. The relationship between the two arc sets is displayed in Figure 2(b), where the dashed arcs are those present in the left-hand side of the standard but not the lifted feasibility cut.

3.3. Valid Inequalities

In classical branch-and-cut algorithms the valid inequalities are used to strengthen the formulation of the problem addressed. Since (1) is a relaxed formulation, in our algorithm the valid inequalities are also used to cut off integer solutions to (1) that are infeasible to the SDVRPTW. While the inequalities presented in Sections 3.3.2–3.3.5 are known from the literature, the inequalities proposed in Sections 3.3.6 and 3.3.7 are new. The infeasible-path constraints proposed in Ascheuer *et al.* (2000, 2001) for the TSPTW are adapted to the SDVRPTW in Section 3.3.6. These inequalities are then generalized to so-called path-matching constraints in Section 3.3.7.

The inequalities presented in Section 3.3.2 are static in the sense that we insert them right from the beginning into (1). All the other inequalities are dynamically separated at each node of the branch-and-cut tree.

Our overall separation strategy can be summarized as follows: Only inequalities exceeding a violation of $\varepsilon = 0.05$ are inserted. The classes of valid inequalities are hierarchically considered according to the order with which they are presented in this section. The separation procedure stops as soon as violated inequalities are found in a given class. A maximum of 500 cuts is added in each call of the separation algorithm.

3.3.1. Preliminaries

Let $\mathcal{P}$ be the polyhedron formed by feasible solutions to the SDVRPTW fulfilling Properties 2–5. The polyhedron formed by solutions to the relaxed formulation (1) is denoted by $\mathcal{P}_R$ and fulfills $\mathcal{P}_R \supseteq \mathcal{P}$. While the inequalities presented in Sections 3.3.3 and 3.3.5 are valid for $\mathcal{P}$ and $\mathcal{P}_R$, all other presented inequalities are valid only for $\mathcal{P}$.

In order to introduce valid inequalities, some additional notation is required: A *path* $P = (v_0, v_1, \ldots, v_\ell)$ is any sequence of vertices with $(v_{i-1}, v_i) \in A$ for $i \in \{1, \ldots, \ell\}$. The *start vertex* of the path is $s(P) = v_0$ and the *end vertex* is $t(P) = v_\ell$. The *length* of the path is $\ell = \ell(P) \geq 1$. The arcs of P are denoted by $A(P)$, and we define $A_N(P) = A(P) \cap A(N)$. The vertices of P are $V(P) = \{v_0, \ldots, v_\ell\}$ and the *internal vertices* are $V^{int}(P) = \{v_1, \ldots, v_{\ell-1}\}$. Note that in the SDVRPTW the internal vertices of a feasible route are customers, i.e., $V^{int}(P) \subseteq N$. For the demand of the internal vertices we use the shorthand notation $d(V^{int}(P))$ for $\sum_{v\in V^{int}(P)} d_v$. Paths of length 1 have $V^{int}(P) = \varnothing$.

A path P with $|\{v_1, \ldots, v_\ell\}| = |\{v_0, v_1, \ldots, v_\ell - 1\}| = \ell$ is said to be *almost-elementary*. All the internal vertices of an almost-elementary path are distinct. An almost-elementary path $P = (v_0, v_1, \ldots, v_\ell)$ is *time-window infeasible* if there do not exist numbers $T_0, T_1, \ldots, T_\ell$ such that $e_{v_i} \leq T_i \leq l_{v_i}$ holds for all $i = 0, 1, \ldots, \ell$ and $T_{i-1} + t_{v_{i-1},v_i} \leq T_i$ holds for $i = 1, \ldots, \ell$. Given an almost-elementary path $P = (v_0, v_1, \ldots, v_\ell)$, we define the *minimum quantity* $\underline{d}(P)$ *to deliver* along the path as

$$\underline{d}(P) = \alpha\big(1 - \delta_{s(P),0}\big) + d(V^{int}(P)) + \alpha\big(1 - \delta_{t(P),n+1}\big), \tag{6}$$

where $\delta_{xy} \in \{0,1\}$ is the Kronecker delta which is equal to 1 if $x = y$ and 0 otherwise, and $\alpha \in \{0,1\}$ is equal to 1 if $d_i \in \mathbb{Z}_+$ for all $i \in N$, and $Q \in \mathbb{Z}_+$. The α-terms defining the minimum quantity to deliver exploit Property 5. An almost-elementary path P with $P \neq (0, i, n+1)$ for any $i \in N$ is *load infeasible* if $\underline{d}(P) > Q$. Note that all paths $P = (0, i, n+1)$ for $i \in N$ of length $\ell(P) = 2$ are feasible even if $\underline{d}(P) = d_i > Q$.

An almost-elementary path $P = (v_0, v_1, \ldots, v_\ell)$, is said to be *infeasible* (for the SDVRPTW) if it does not occur as a subpath in any route of a feasible solution to the SDVRPTW fulfilling Properties 2–5.

Definition 3.1. *An almost-elementary path $P = (v_0, v_1, \ldots, v_\ell)$, is infeasible if at least one of the following condition is satisfied:*
(i) P is time-window infeasible;
(ii) P is load infeasible;
(iii) P is a cycle, i.e. $s(P) = t(P)$.

Definition 3.2. *For a path $P = (v_0, v_1, \ldots, v_{\ell-1}, v_\ell)$ in G, the associated* depot-reduced path *is $dr(P)$ equal to*
(i) $P = (v_0, v_1, \ldots, v_{\ell-1}, v_\ell)$ if $v_0 \neq 0$ and $v_\ell \neq n+1$,
(ii) $(v_1, \ldots, v_{\ell-1}, v_\ell)$ if $v_0 = 0$ and $v_\ell \neq n+1$,
(iii) $(v_0, v_1, \ldots, v_{\ell-1})$ if $v_0 \neq 0$ and $v_\ell = n+1$, and
(iv) $(v_1, \ldots, v_{\ell-1})$ if $v_0 = 0$ and $v_\ell = n+1$.

Lemma 3.3. *Given an infeasible almost-elementary path $P = (v_0, v_1, \ldots, v_\ell)$ with*
(a) $v_0 = 0$ and $v_\ell \neq n+1$. Then, any almost-elementary path P' of the form $(S, dr(P))$ is infeasible for any path S, $\ell(S) \geq 1$.
(b) $v_0 \neq 0$ and $v_\ell = n+1$. Then, any almost-elementary path P' of the form $(dr(P), T)$ is infeasible for any path T, $\ell(T) \geq 1$.
(c) $v_0 = 0$ and $v_\ell = n+1$. Then, any almost-elementary path P' of the form $(S, dr(P), T)$ is infeasible for any pair of paths S, $\ell(S) \geq 1$, and T, $\ell(T) \geq 1$.

Proof. (a): P is time-window infeasible or load infeasible. Defining $S = (0, v_1)$ would result in $P' = P$. Any other 1-arc path S would lead to the definition of almost-elementary path P' which is time-window infeasible if P is time-window infeasible and such that $\underline{d}(P') \geq \underline{d}(P)$.
(b)+(c): Straightforward using similar arguments. □

For the presentation of separation procedures, we assume that the current (fractional) solution of (1) is given by $\bar{s} = (\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{T}})$. Moreover, for any customer $i \in N$ visited less than twice, i.e., $\bar{z}_i < 2$, $\pi(i)$ and $\sigma(i)$ denote a predecessor and a successor of i in the graph induced by $\bar{s}$, respectively. The different separation procedures use individual tie-breaker rules if predecessors or successors are not unique.

3.3.2. Static Inequalities

Due to Property 3, the inequalities

$$x_{ij} + x_{ji} \leq 1 \qquad (i,j),(j,i) \in A(N) : i < j \tag{7}$$

can be imposed.

3.3.3. Capacity Cuts

Capacity cuts (1d), i.e., inequalities $\sum_{(i,j)\in\Gamma^+(S)} x_{ij} \geq K_S$ for all $S \subseteq N$, $|S| \geq 2$, have been stated as a part of formulation (1). We separate violated capacity cuts by applying two shrinking heuristics presented in (Belenguer *et al.*, 2000) and (Ralphs *et al.*, 2003), namely the extended shrinking heuristic and the greedy shrinking heuristic. The reader is referred to the latter reference for details.

3.3.4. 2-Path Cuts

Kohl *et al.* (1999) introduced 2-path cuts in order to strengthen path-based formulations of the VRPTW. However, these inequalities solely refer to the vehicle flow on the arcs, and thus they can also be applied to arc-based formulations. Whenever a subset $S \subseteq N$ of the customers cannot be served with a single vehicle, the 2-path cuts

$$\sum_{(i,j)\in\Gamma^+(S)} x_{ij} \geq 2 \tag{8}$$

is valid. The precondition is fulfilled if $K_S > 1$, i.e., the demand of the customers S exceeds the vehicle capacity, or $S \cup \{0, n+1\}$ cannot be visited by a single vehicle due to time window restrictions. The latter means that the TSPTW induced by $S \cup \{0, n+1\}$ is infeasible. We separate violated 2-path cuts with the help of the greedy heuristic proposed in Kohl *et al.* (1999). Given the current arc-flow values $\bar{\mathbf{x}}$, the heuristic first identifies inclusion-maximal candidate sets S with $\sum_{(i,j)\in\Gamma^+(S)} \bar{x}_{ij} < 2$. Then, for each candidate set S, an exact dynamic programming algorithm for the associated TSPTW over $S \cup \{0, n+1\}$ is applied. If no feasible TSPTW solution exists, a violated 2-path cut is identified.

3.3.5. Connectivity Cuts

Already the capacity cuts ensure that any subset of customers is connected to the depot. A more general type of connectivity cuts has been used in three-index VRP formulations (Toth and Vigo, 2002, p. 15). In the SDVRPTW, connectivity cuts are of the form

$$\sum_{(i,j)\in\Gamma^+(S)} x_{ij} \geq z_u \qquad S \subseteq N, |S| \geq 2, u \in S. \tag{9}$$

We identify violated connectivity cuts by solving a maximum flow problem for each customer $i \in N$ using the software library devised by Boykov and Kolmogorov (2004). Only the violated connectivity cuts with $u^\star = argmax_{u\in S}\{z_u\}$ are inserted into (1).

3.3.6. Infeasible-Path Constraints

The generalization of infeasible-path constraints first introduced by Ascheuer *et al.* (2000, 2001) for the TSPTW is as follows:

Proposition 3.4. *For all infeasible almost-elementary paths P with $\ell(P) \geq 3$, the infeasible-path constraint*

$$\sum_{(i,j)\in A_N(P)} x_{ij} - \sum_{v\in V^{int}(P)} z_v \leq -\delta_{s(P),0} - \delta_{t(P),n+1} \tag{10}$$

is valid for the polyhedron $\mathcal{P}$.

Proof. Note first that if $s(P) = 0$, i.e., the path P starts at the depot, then the first arc of the path does not contribute to the left-hand side, since only arcs in $A_N(P)$ are considered. At the same time the right-hand side decreases by 1 due to the term $-\delta_{s(P),0}$. The respective statement is true if $t(P) = n+1$, i.e., when the path ends at the depot $n+1$.

In any case, a violation $\sum_{(i,j)\in A_N(P)} \bar{x}_{ij} - \sum_{v\in V^{int}(P)} \bar{z}_v > -\delta_{s(P),0} - \delta_{t(P),n+1}$ of the above inequality (10) by an integer solution is only possible if $\bar{x}_{ij} = 1$ for all $(i,j) \in A_N(P)$ and $\bar{z}_v = 1$ for all vertices $v \in V^{int}(P)$. This means that the vertices in $V^{int}(P)$ are visited only once and exactly in the sequence defined by path P. Hence, $dr(P)$ must be a subpath of a feasible SDVRPTW route, which is impossible due to the infeasibility of P and Lemma 3.3. □

Infeasible-path constraints (10) are separated as follows. The predecessors $\pi(i)$ and successors $\sigma(i)$ for $i \in N$ are undefined if $\bar{z}_i \geq 1.5$. We repeat the separation heuristic three times using one of the following rules to determine predecessors and successors for the other customers i with $\bar{z}_i < 1.5$:

$$\pi(i) = \underset{h\in N}{argmin}\{|1 - \bar{x}_{hi}|\}, \qquad \sigma(i) = \underset{j\in N}{argmin}\{|1 - \bar{x}_{ij}|\}; \tag{Rule 1}$$

$$\pi(i) = \underset{h\in N}{argmax}\{\max(e_i, e_h + t_{hi})\bar{x}_{hi}\}, \qquad \sigma(i) = \underset{j\in N}{argmax}\{\max(l_j, e_i + t_{ij})\bar{x}_{ij}\}; \tag{Rule 2}$$

$$\pi(i) = \underset{h\in N}{argmax}\{d_h\bar{x}_{hi}\}, \qquad \sigma(i) = \underset{j\in N}{argmax}\{d_j\bar{x}_{ij}\}. \tag{Rule 3}$$

After the initialization step, for each customer $i \in N$, we start with the almost-elementary path $P = (i)$ and extend it iteratively adding predecessors of $s(P)$ or successors of $t(P)$ to the respective endpoint. The extension stops when $s(P)$ has an undefined predecessor and $t(P)$ has an undefined successor. The resulting almost-elementary path P, if infeasible (see Definition 3.1), is then checked to violate the corresponding infeasible-path constraint (10) that is eventually added.

Each time a load infeasible almost-elementary path P is found, we also check if the corresponding capacity cut (1d) for $S = V^{int}(P)$ is violated. If so, we add the violated capacity cut.

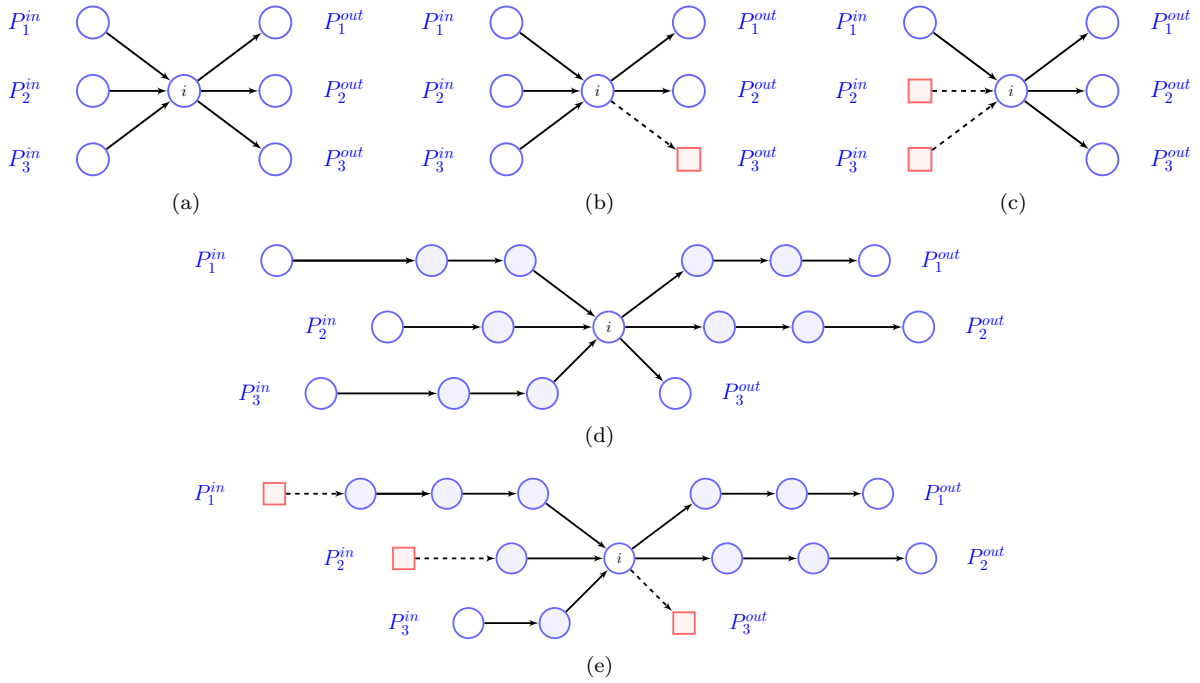

Figure 3: Examples of *stretched stars* $S(i, 3, P^{in}, P^{out})$.

3.3.7. Path-Matching Constraints

Path-matching constraints generalize infeasible-path constraints (10). We introduce them in order to cut off infeasible configurations such as those depicted in Figure 1.

Definition 3.5. *Let $i \in N$, $p \geq 1$, and $P_1^{in}, P_2^{in}, \ldots, P_p^{in}$ and $P_1^{out}, P_2^{out}, \ldots, P_p^{out}$ be paths with the following properties:*

*(i) All paths $P_1^{in}, P_2^{in}, \ldots, P_p^{in}$ have end vertex i (*in-paths*), i.e., $t(P_j^{in}) = i$ for $1 \leq j \leq p$.*
*All paths $P_1^{out}, P_2^{out}, \ldots, P_p^{out}$ have start vertex i (*out-paths*), i.e., $s(P_k^{out}) = i$ for $1 \leq k \leq p$.*

(ii) Internal vertices of all in- and out-paths are disjoint: $V^{int}(P_j^{in}) \cap V^{int}(P_{j'}^{in}) = \varnothing$ for $1 \leq j, j' \leq p$, $j \neq j'$; $V^{int}(P_k^{out}) \cap V^{int}(P_{k'}^{out}) = \varnothing$ for $1 \leq k, k' \leq p$, $k \neq k'$; $V^{int}(P_j^{in}) \cap V^{int}(P_k^{out}) = \varnothing$ for $1 \leq j, k \leq p$.

(iii) All concatenations of in-paths and out-paths, in the following denoted by (P_j^{in}, P_k^{out}), are almost-elementary for all $1 \leq j, k \leq p$.

(iv) If there is an in-path $P_j^{in} = (0, i)$, then (P_j^{in}, P_k^{out}) is not infeasible for all out-paths P_k^{out} for $1 \leq k \leq p$.
If there is an out-path $P_k^{in} = (i, n+1)$, then (P_j^{in}, P_k^{out}) is not infeasible for all in-paths P_j^{in} for $1 \leq j \leq p$.

(v) In-paths $P_j^{in} = (0, i)$ and out-paths $P_k^{in} = (i, n+1)$ do not occur together.

Such a set of in-paths $P^{in} = \{P_1^{in}, P_2^{in}, \ldots, P_p^{in}\}$ and out-paths $P^{out} = \{P_1^{out}, P_2^{out}, \ldots, P_p^{out}\}$ is called a stretched star *and denoted by $S(i, p, P^{in}, P^{out})$.*

Examples of five different stretched stars are depicted in Figure 3.

Given a stretched star $S(i, p, P^{in}, P^{out})$, any concatenated path (P_j^{in}, P_k^{out}) for $1 \leq j, k \leq p$ can be tested for infeasibility. While we use identical definitions of time-window infeasible paths and cycles as in Definition 3.1, a modified definition of load infeasible paths is required here. It is based on another definition of the *minimum quantity* $\underline{d}(P)$ *to deliver* along a path P, cf. (6), now defined as

$$\underline{d}(P_j^{in}, P_k^{out}) = \alpha(1 - \delta_{s(P_j^{in}),0}) + d(V^{int}(P_j^{in})) + \alpha + d(V^{int}(P_k^{out})) + \alpha(1 - \delta_{t(P_k^{out}),n+1}). \tag{11}$$

Thus, a path (P_j^{in}, P_k^{out}) is load infeasible if $\underline{d}(P_j^{in}, P_k^{out}) > Q$.

Definition 3.6. *Let $M \in \{0,1\}^{m \times n}$ be any binary matrix. We define the associated bipartite graph $B(M) = (P \cup Q, E_M)$ by vertices $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_n\}$ (the bi-partition), and edges $E_M = \{\{p_i, q_j\} : m_{ij} = 1$ for $1 \leq i \leq m, 1 \leq j \leq n\}$.*

Let $M(P^{in}, P^{out}) = (m_{jk})$ denote the *compatibility matrix* between the in-paths and out-paths, with $m_{jk} = 0$ if path (P_j^{in}, P_k^{out}) is infeasible, and $m_{jk} = 1$ otherwise. We define the *compatibility number* $n_M = n_M(P^{in}, P^{out})$ as the size of a maximum-cardinality matching in the bipartite graph $B(M(P^{in}, P^{out}))$.

Definition 3.7. *A stretched star $S(i, p, P^{in}, P^{out})$ is called infeasible if $n_M(P^{in}, P^{out}) < p$.*

Define the *number $n_D = n_D(P^{in}, P^{out})$ of paths with a depots* in the stretched star $S(i, p, P^{in}, P^{out})$ by $n_D = |\{j : 1 \le j \le p, s(P_j^{in}) = 0\}| + |\{k : 1 \le k \le p, t(P_k^{out}) = n + 1\}|$.

Theorem 3.8. *For all infeasible stretched stars $S(i, p, P^{in}, P^{out})$, the path-matching constraint*

$$\sum_{j=1}^{p} \left(\sum_{(g,h) \in A_N(P_j^{in})} x_{gh} - \sum_{v \in V^{int}(P_j^{in})} z_v \right) + \sum_{k=1}^{p} \left(\sum_{(g,h) \in A_N(P_k^{out})} x_{gh} - \sum_{v \in V^{int}(P_k^{out})} z_v \right) - z_i \le n_M - n_D \qquad (12)$$

with $n_M = n_M(P^{in}, P^{out})$ and $n_D = n_D(P^{in}, P^{out})$ is valid for the polyhedron $\mathcal{P}$.

Proof. For convenience, we define the number of short-depot paths (length 1) and long-depot paths (length greater than 1) as

$$\begin{aligned} n_D^{short} &:= |\{1 \le j \le p : P_j^{in} = (0, i)\}| + |\{1 \le k \le p : P_k^{out} = (i, n + 1)\}| \\ n_D^{long} &:= |\{1 \le j \le p : s(P_j^{in}) = 0, \ell(P_j^{in}) > 1\}| + |\{1 \le k \le p : t(P_k^{out}) = n + 1, \ell(P_k^{out}) > 1\}|. \end{aligned}$$

Then, $n_D = n_D^{short} + n_D^{long}$.

Let $\bar{s} = (\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{T}})$ be a feasible integer solution to the SDVRPTW. The multiset $A(\bar{\mathbf{x}})$ comprises exactly $\bar{x}_{ij}$ copies of each arc $(i, j) \in A$. We will show that $\bar{s}$ is not cut off by any path-matching constraint associated with an infeasible stretched star $S(i, p, P^{in}, P^{out})$. For the sake of exposition, we distinguish the following two cases for the infeasible stretched star:

(i) All in-paths and out-paths consist of single arcs;
(ii) Arbitrary in-paths and out-paths.

Case (i): All in-paths are of the form $P_j^{in} = (v_j, i)$ with $v_j \in V$ and all out-path are of the form $P_k^{out} = (i, v_k)$ with $v_k \in V$ as shown in Figure 3(a–c). With the definitions $A(N)^{in} = \{(v_j, i) : 1 \le j \le p, v_j \in N\}$ and $A(N)^{out} = \{(i, v_k) : 1 \le k \le p, v_k \in N\}$, the path-matching constraint (12) reduces to

$$\sum_{(h,i) \in A(N)^{in}} x_{hi} + \sum_{(i,h) \in A(N)^{out}} x_{ih} - z_i \le n_M - n_D.$$

Moreover, we know that in the given feasible integer solution the customer i is visited exactly $\bar{z}_i$ times. Consider the star $(i, \bar{z}_i, \Gamma^-(i) \cap A(\bar{\mathbf{x}}), \Gamma^+(i) \cap A(\bar{\mathbf{x}}))$ imposed by the integer feasible solution. It induces a compatibility matrix $\bar{M} = (\bar{m}_{jk})$ of dimension $\bar{z}_i \times \bar{z}_i$ and a maximum-cardinality matching of value $n_{\bar{M}} = \bar{z}_i = |\Gamma^-(i) \cap A(\bar{\mathbf{x}})| = |\Gamma^+(i) \cap A(\bar{\mathbf{x}})|$.

Since the value of the left-hand side of the path-matching constraint is $|A(N)^{in} \cap A(\bar{\mathbf{x}})| + |A(N)^{out} \cap A(\bar{\mathbf{x}})|$, we now consider the submatrix M' of $\bar{M}$ corresponding to the rows/arcs $A(N)^{in} \cap A(\bar{\mathbf{x}})$ and the columns/arcs $A(N)^{out} \cap A(\bar{\mathbf{x}})$. Note that M' results from $\bar{M}$ by the elimination of exactly $n^{in} = |\Gamma^-(i) \cap A(\bar{\mathbf{x}})| - |A(N)^{in} \cap A(\bar{\mathbf{x}})|$ rows and $n^{out} = |\Gamma^+(i) \cap A(\bar{\mathbf{x}})| - |A(N)^{out} \cap A(\bar{\mathbf{x}})|$ columns. This operation is equivalent to the removal of n^{in} vertices from the first and of n^{out} vertices from the second partition of $B(\bar{M})$. The maximum-cardinality matching in $B(\bar{M})$ of size $\bar{z}_i$ hence induces a matching in $B(M')$ of size not smaller than $\bar{z}_i - n^{in} - n^{out}$. (Note that in general, the elimination of exactly w vertices from a graph cannot remove more than w edges from any matching.)

Consider then the submatrix $\widehat{M}$ of $M = M(P^{in}, P^{out})$ resulting from the elimination of n_D^{short} rows or columns associated with the arcs $(0, i)$ or $(i, n + 1)$. (Note that condition (v) in the Definition 3.5 of a stretched star ensures that it is either rows or columns but not both.) According to condition (iv) of Definition 3.5, the n_D^{short} arcs are part of the maximum-cardinality matching in $B(M)$, since otherwise the matching would not have had maximum cardinality. Then, the size of a maximum-cardinality matching in $B(\widehat{M})$ cannot be greater than $n_M - n_D^{short}$.

Since M' is a submatrix of $\widehat{M}$, it follows

$$n_M - n_D^{short} \ge n_{\widehat{M}} \ge n_{M'} \ge \bar{z}_i - n^{in} - n^{out}. \qquad (13)$$

Now, we have

$$\begin{aligned}
&\sum_{(h,i)\in A(N)^{in}} \bar{x}_{hi} + \sum_{(i,h)\in A(N)^{out}} \bar{x}_{ih} \\
&= |A(N)^{in} \cap A(\bar{\mathbf{x}})| + |A(N)^{out} \cap A(\bar{\mathbf{x}})| \\
&= (\bar{z}_i - n^{in}) + (\bar{z}_i - n^{out}) && \text{(14a)} \\
&= \bar{z}_i + \bar{z}_i - n^{in} - n^{out} \\
&\leq \bar{z}_i + n_M - n_D^{short} && \text{(14b)} \\
&= \bar{z}_i + n_M - n_D && \text{(14c)}
\end{aligned}$$

where (14a) results from the definition of n^{in} and n^{out} using $\bar{z}_i = |\Gamma^-(i) \cap A(\bar{\mathbf{x}})| = |\Gamma^+(i) \cap A(\bar{\mathbf{x}})|$, (14b) uses (13), and (14c) is the assumption $n_D = n_D^{short}$ of Case (i). Subtracting $\bar{z}_i$ from (14) shows that the feasible integer solution satisfies the path-matching constraint in Case (i).

Case (ii): This is the case of in-paths and out-path of arbitrary length as shown in Figure 3(d–e). Consider the largest star $(i, \bar{z}_i, \bar{P}^{in}, \bar{P}^{out})$ fulfilling conditions (i)–(iv) of Definition 3.5 imposed by the integer feasible solution. Such a star is unique because conditions (ii) and (iii) impose that all internal vertices are non-split customers so that in a largest star all in-paths/out-paths either start/end at the depot $0/n+1$ or at split customers. Moreover note that condition (iv) is not restrictive for the definition of the largest star. Indeed, if condition (iv) would not be fulfilled, then at least one in-path or out-path would be infeasible, i.e., the integer solution $\bar{s}$ would be infeasible, which contradicts with our assumption of a feasible integer solution. The star $(i, \bar{z}_i, \bar{P}^{in}, \bar{P}^{out})$ induces a compatibility matrix $\bar{M} = (\bar{m}_{jk})$ of dimension $\bar{z}_i \times \bar{z}_i$ and a maximum-cardinality matching of value $n_{\bar{M}} = \bar{z}_i = |\bar{P}^{in}| = |\bar{P}^{out}|$.

We now consider the star $S(i, p, P_j^{in}, P_k^{out})$ defining the path-matching constraint (12). For each in-path $P_j^{in} \neq (0, i)$, its depot-reduced path $dr(P_j^{in})$ may occur as a subpath of the integer solution $\bar{s}$. We define the set of these in-paths by

$$P'^{in} := \{P_j^{in} : 1 \leq j \leq p, P_j^{in} \neq (0,i), \bar{P}_{j'}^{in} = (S_{j'}, dr(P_j^{in})) \text{ for some } j' \text{ and some path } S_{j'}, \ell(S_{j'}) \geq 1\}.$$

Similarly, for each out-path $P_k^{out} \neq (i, n+1)$, its depot-reduced path $dr(P_k^{out})$ may occur as a subpath of the integer solution, and we define the corresponding set

$$P'^{out} := \{P_k^{out} : 1 \leq k \leq p, P_k^{out} \neq (i, n+1), \bar{P}_{k'}^{out} = (dr(P_k^{out}), T_{k'}) \text{ for some } k' \text{ and some path } T_{k'}, \ell(T_{k'}) \geq 1\}.$$

Note that by definition both sets do not include short-depot paths. Each in-path $P_j^{in} \in P'^{in}$ is uniquely associated with an in-path $\bar{P}_{j'}^{in} \in \bar{P}'^{in} \subseteq \bar{P}^{in}$, and vice versa. Similarly, each out-path $P_k^{out} \in P'^{out}$ is uniquely associated with an out-path $\bar{P}_{k'}^{out} \in \bar{P}'^{out} \subseteq \bar{P}^{out}$, and vice versa. The following equalities hold:

$$\begin{aligned}
|P'^{in}| &= |\bar{P}'^{in}|, && \text{(15a)} \\
|P'^{out}| &= |\bar{P}'^{out}|. && \text{(15b)}
\end{aligned}$$

We will consider the submatrix M' of $\bar{M}$ corresponding to the rows inducing P'^{in} and the columns inducing P'^{out}. It results from $\bar{M}$ by the elimination of exactly $n^{in} = |\bar{P}^{in}| - |\bar{P}'^{in}|$ rows and $n^{out} = |\bar{P}^{out}| - |\bar{P}'^{out}|$ columns. There is also a submatrix $\widehat{\widehat{M}}$ of $M = M(P^{in}, P^{out})$ corresponding to the rows P'^{in} and the columns P'^{out}. Since compatibility is conserved on subpaths, the relation $M' \leq \widehat{\widehat{M}}$ holds (componentwise) so that we know $n_{M'} \leq n_{\widehat{\widehat{M}}}$. Similar to Case (i), we define a submatrix $\widehat{M}$ of M resulting from the elimination of n_D^{short} rows (or columns) associated with the short depot-paths $(0, i)$ and $(i, n+1)$. The matrix $\widehat{\widehat{M}}$ is a submatrix of $\widehat{M}$ so that $n_{\widehat{\widehat{M}}} \leq n_{\widehat{M}}$ holds. Also here the n_D^{short} arcs are part of the maximum-cardinality matching in $B(M)$ so that the size of a maximum-cardinality matching in $B(\widehat{M})$ cannot be greater than $n_M - n_D^{short}$. Putting all these results together, we get

$$n_M - n_D^{short} \geq n_{\widehat{M}} \geq n_{\widehat{\widehat{M}}} \geq n_{M'} \geq \bar{z}_i - n^{in} - n^{out}, \tag{16}$$

which is the analogue to (13) of Case (i).

For the jth in-path P_j^{in} with $P_j^{in} \neq (0,i)$, the term

$$\sum_{(g,h)\in A_N(P_j^{in})} x_{gh} - \sum_{v\in V^{int}(P_j^{in})} z_v + \delta_{s(P_j^{in}),0} \tag{17a}$$

is bounded by 1 (from above) and is 1 if $\bar{x}_{gh} = 1$ for all $(g,h) \in A_N(P_j^{in})$ and $\bar{z}_v = 1$ for all $v \in V^{int}(P_j^{in})$. This means that all internal vertices are customers that are visited exactly once and exactly in the sequence defined by P_j^{in}, which is equivalent to the condition $P_j^{in} \in P'^{in}$. Note the similarity of the arguments to those used in the proof of the infeasible-path constraints (10) (proof of Proposition 3.4).

The same can be said for the term

$$\sum_{(g,h)\in A_N(P_k^{out})} x_{gh} - \sum_{v\in V^{int}(P_k^{out})} z_v + \delta_{t(P_k^{out}),n+1} \tag{17b}$$

of the kth out-path P_k^{out}, $P_k^{out} \neq (i,n+1)$. A contribution of 1 occurs only if the internal vertices are non-split customers that are served exactly in the sequence defined by P_k^{out}, equivalent to $P_k^{out} \in P'^{out}$; otherwise the contribution is 0 or negative.

The following inequalities result:

$$\begin{aligned}
&\sum_{j=1}^{p}\left(\sum_{(g,h)\in A_N(P_j^{in})} \bar{x}_{gh} - \sum_{v\in V^{int}(P_j^{in})} \bar{z}_v\right) + \sum_{k=1}^{p}\left(\sum_{(g,h)\in A_N(P_k^{out})} \bar{x}_{gh} - \sum_{v\in V^{int}(P_k^{out})} \bar{z}_v\right) + n_D^{long} \\
&= \sum_{P_j^{in}\neq(0,i)} \underbrace{\left(\sum_{(g,h)\in A_N(P_j^{in})} \bar{x}_{gh} - \sum_{v\in V^{int}(P_j^{in})} \bar{z}_v + \delta_{s(P_j^{in}),0}\right)}_{=1,\text{ if } dr(P_j^{in}) \text{ is in the solution } \bar{s};\quad \leq 0,\text{ otherwise}} \qquad (18a) \\
&\qquad + \sum_{P_k^{out}\neq(i,n+1)}^{p} \underbrace{\left(\sum_{(g,h)\in A_N(P_k^{out})} \bar{x}_{gh} - \sum_{v\in V^{int}(P_k^{out})} \bar{z}_v + \delta_{t(P_k^{out}),n+1}\right)}_{=1,\text{ if } dr(P_k^{out}) \text{ is in the solution } \bar{s};\quad \leq 0,\text{ otherwise}} \\
&\leq |P'^{in}| + |P'^{out}| \qquad (18b) \\
&= |\bar{P}'^{in}| + |\bar{P}'^{out}| \qquad (18c) \\
&= (\bar{z}_i - n^{in}) + (\bar{z}_i - n^{out}) \qquad (18d) \\
&= \bar{z}_i + \bar{z}_i - n^{in} - n^{out} \\
&\leq \bar{z}_i + n_M - n_D^{short} \qquad (18e)
\end{aligned}$$

Equality (18a) holds because short-depot paths $P_j^{in} = (0,i)$ and $P_k^{out} = (i,n+1)$ contribute with 0 to the sum, and n_D^{long} is identical to the sum of the δ-values of the non-depot paths in the star. Inequality (18b) follows from (17), (18c) from (15), and (18d) from the definition of M' having dimension $(\bar{z}_i - n^{in}) \times (\bar{z}_i - n^{out})$. For the inequality (18e), we use (16), i.e., $\bar{z}_i - n^{in} - n^{out} \leq n_M - n_D^{short}$.

Subtracting n_D^{long} and $\bar{z}_i$ from (18) and using the equality $n_D = n_D^{short} + n_D^{long}$ shows that the path-matching constraint (12) does not cut off the feasible integer solution $\bar{s}$ in Case (ii). □

Example 3.9. *Consider the infeasible integer solution to the 5-customer SDVRPTW depicted in Figure 1(a). Defining the stretched star $(i = 3, p = 2, P^{in}, P^{out})$ with $P_1^{in} = (1,3)$, $P_2^{in} = (2,3)$, $P_1^{out} = (3,4)$, and $P_2^{out} = (3,5)$, we can immediately see that $(P_1^{in}, P_1^{out}) = (1,3,4)$ and $(P_1^{in}, P_2^{out}) = (1,3,5)$ are time-window infeasible, while $(P_2^{in}, P_1^{out}) = (2,3,4)$ and $(P_2^{in}, P_2^{out}) = (2,3,5)$ are time-window feasible. This leads to*

$$M = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \qquad \text{with} \qquad n_M = 1 < p = 2$$

and the associated path-matching constraint is

$$x_{13} + x_{23} + x_{34} + x_{35} - z_3 \leq 1 - 0 = 1,$$

which cuts off the infeasible integer solution.

Now consider the instance and solution given in Figure 1(b). Using the same stretched star, we compute $\underline{d}(P_1^{in}, P_1^{out}) = \underline{d}(P_1^{in}, P_2^{out}) = \underline{d}(P_2^{in}, P_1^{out}) = \underline{d}(P_2^{in}, P_2^{out}) = 1 + 1 + 1 = 3$. *With a capacity* $Q = 10$, *all in-paths are compatible with all out-paths leading to* $n_M = 2$. *Hence, the stretched star is* not *infeasible in this case. Indeed, the above constraint* $x_{13} + x_{23} + x_{34} + x_{35} - z_3 \leq 1$ *is not valid for the polyhedron* $\mathcal{P}$ *of the second instance. For example, the routes* $(0, 1, 3, 4, n + 1)$, $(0, 2, 3, 5, n + 1)$, *and* $(0, 1, 4, n + 1)$ *with appropriate delivery quantities form a feasible integer solution that does not fulfill the inequality.*

However, we can define the larger stretched star $(i = 3, p = 2, P^{in}, P^{out})$ *with* $P_1^{in} = (0, 1, 3)$, $P_2^{in} = (0, 2, 3)$, $P_1^{out} = (3, 4, n+1)$, *and* $P_2^{out} = (3, 5, n+1)$ *for the second instance. Then, the minimum quantities to deliver are*

$$\begin{aligned} \underline{d}(P_1^{in}, P_1^{out}) = \underline{d}(P_2^{in}, P_1^{out}) &= 0 + 4 + 1 + 7 + 0 = 11 > Q \\ \underline{d}(P_1^{in}, P_2^{out}) = \underline{d}(P_2^{in}, P_2^{out}) &= 0 + 4 + 1 + 1 + 0 = 6 \leq Q. \end{aligned}$$

We get

$$M = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \textit{with} \quad n_M = 1 < p = 2 \qquad \textit{and} \qquad n_D = 4,$$

and the associated path-matching constraint is

$$(x_{13} - z_1) + (x_{23} - z_2) + (x_{34} - z_4) + (x_{35} - z_5) - z_3 \leq 1 - 4 = -3,$$

which cuts off the infeasible integer solution ($-2 \not\leq -3$). *However, the feasible solution with routes* $(0, 1, 3, 4, n+1)$, $(0, 2, 3, 5, n + 1)$, *and* $(0, 1, 4, n + 1)$ *is not cut off because the left-hand side is -4 (note that* $\bar{z}_1 = \bar{z}_4 = 2$ *in this solution).*

For $p = 1$, properties (i)-(iii) and (v) of Definition 3.5 impose the stretched star $S(i, 1, P^{in}, P^{out})$ to be an almost-elementary path $P = (P_1^{in}, P_1^{out}) = (v_0, v_1, \ldots, v_\ell)$ such that $\ell(P) \geq 2$. Moreover, property (iv) ensures that P is not infeasible if $v_1 = i$ and $v_0 = 0$, or $v_{\ell-1} = i$ and $v_\ell = n + 1$. The set of infeasible almost-elementary paths induced by the stretched stars $S(i, 1, P^{in}, P^{out})$ is thus included in the set of all infeasible almost-elementary paths. For this reason, path-matching constraints (12) are separated only for infeasible stretched stars with $p \geq 2$.

Separation proceeds as follows: For each customer $i \in N$, we define $p = p(i) = \lfloor z_i + \frac{1}{2} \rfloor$, and if $p \geq 2$ we try to find violated inequalities for stretched stars of the form $S(i, p, P^{in}, P^{out})$. Tentative in-paths $P_1^{in}, \ldots, P_p^{in}$ and out-paths $P_1^{out}, \ldots, P_p^{out}$ are iteratively constructed. Initially, all in-paths and out-path consist of single arcs only (like in Figure 3(a–c)) resulting from the p arcs $(v, i) \in \Gamma^-(i)$ and the p arcs $(i, v) \in \Gamma^+(i)$ with maximum flow $\bar{x}_{vi}$ and $\bar{x}_{iv}$ (depot arcs with flow greater than 1 can lead to multiple copies of these arcs). In each iteration, it is first tested wether the current stretched star $S(i, p, P^{in}, P^{out})$ imposes a violated path-matching constraint. To do this, the difference between left-hand and right-hand side of (12) is computed. This requires to determine n_M for $M = M(P^{in}, P^{out})$ for which we compute the compatibility matrix M and then solve a (small) matching/assignment problem using a network flow solver. If $n_M = p$ the next steps for computing the possible violation of (12) can be skipped. Otherwise (the stretched star is infeasible in this case), the values of the left-hand side of (12) and of $n_D = n_D(P^{in}, P^{out})$ are computed. This latter computation is rather simple because from one iteration to the next we always add only a single arc to only one of the in-paths or out-paths. This next arc is one giving the highest contribution to the left-hand side of the path-matching constraint (12). More precisely, for arcs $(g, h) \in A$ that can extend an in-path P_j^{in}, i.e., $h = s(P_j^{in})$ for some $j \in \{1, 2, \ldots, p\}$, the contribution is $\bar{x}_{gh} - \bar{z}_h$, while arcs $(g, h) \in A$ that can extend an out-path P_k^{out}, i.e., $g = t(P_k^{out})$ for some $k \in \{1, 2, \ldots, p\}$, the contribution is $\bar{x}_{gh} - \bar{z}_g$. Moreover, we require $g = \pi(h)$ for in-paths and $\sigma(g) = h$ for out-paths to make the extensions unique, where predecessors and successors are defined as in Section 3.3.6 by Rule 1; this also includes that all internal vertices $v \in V^{int}(P_j^{in}) \cup V^{int}(P_k^{out})$ fulfill $\bar{z}_v < 1.5$. Iterations stop as soon as all in-paths have no predecessor $\pi(s(P_j^{in}))$ of their start vertex $s(P_j^{in})$ and all out-paths have no successor $\sigma(t(P_k^{out}))$ of their last vertex $t(P_k^{out})$.

4. Experimental Analysis

We test the branch-and-cut algorithm on the same benchmark instances also considered by Gendreau *et al.* (2006), Desaulniers (2010), and Archetti *et al.* (2011b). These instances have been derived from the

Table 1: Results obtained with the `Baseline` branch-and-cut algorithm not using the new classes of valid inequalities.

Instances			Results				
Class	Q	#	Solved	Time	$\underline{z}^*$	Nodes	Feas. checks
R1	30	12	3	1 617.9	15 459.9	13 553.5	71.3
	50	12	2	1 530.3	10 674.7	16 554.9	59.5
	100	12	2	1 500.8	7 640.6	18 400.3	23.0
C1	30	9	3	1 372.7	15 986.4	13 525.0	2 691.7
	50	9	6	687.2	10 131.8	12 793.8	1 090.3
	100	9	6	725.7	5 835.5	11 331.3	346.4
RC1	30	8	8	17.6	27 395.1	148.3	18.1
	50	8	8	64.7	18 151.3	2 696.4	140.0
	100	8	7	470.5	10 238.0	28 291.2	48.7
R2	30	11	0	1 800.1	15 281.0	14 989.9	87.6
	50	11	1	1 656.1	10 411.4	16 727.8	173.5
	100	11	1	1 643.3	6 931.4	17 415.0	66.4
C2	30	8	0	1 800.1	17 547.3	13 445.4	258.1
	50	8	1	1 614.2	11 450.6	17 653.9	173.4
	100	8	5	1 142.2	6 847.6	16 367.3	773.4
RC2	30	8	8	19.0	27 395.0	301.1	22.9
	50	8	8	87.3	16 996.3	6 634.6	148.0
	100	8	8	81.7	9 348.1	6 216.6	12.6
Total/Weighted Avg.		*168*	*77*	1 067.4	13 182.7	13 068.2	329.6

VRPTW benchmark of Solomon (1987) by allowing split deliveries. The 56 instances are divided into six classes R1, C1 , RC1, R2, C2, and RC2 with 100 customers each, where customers in the C instances are clustered in a 100 x 100 square, in the R instances they are randomly located, and in the RC instances the locations are mixed. The time window constraints of the R1, C1, and RC1 instances are more restrictive than those of the R2, C2, and RC2 instances. For each of the 100-customer instances, smaller instances have been constructed by considering the first 25 and 50 customers only. For defining SDVRPTW instances with different split characteristics, the vehicle capacity is varied by $Q = 25$, 50 and 100. The total number of benchmark instances for the SDVRPTW is thus $504 = 56 \times 3 \times 3$.

The branch-and-cut algorithm is implemented in C++ using CPLEX 12.6.0.1 with Concert Technology, compiled in release mode with MS Visual C++ 2013, experiments are carried out on a 64-bit Windows 10 PC with the Intel Xeon processor E5-1650v3, 3.50 GHz, and 64 GB of RAM allowing a single thread for each run. CPLEX built-in cuts have been used in all experiments. Due to numerical instability we set `IloCplex::NumericalEmphasis = CPX_ON` and `IloCplex::EpGap = 1.0e-5`. CPLEX's default values are kept for all the remaining parameters. At each run, we provide an initial feasible solution computed with a straightforward greedy constructive heuristic described in Section A of the Appendix.

4.1. Analysis of New Components of Branch-and-Cut

For the analysis of the branch-and-cut components, we restrict ourselves to the 168 instances with 50 customers because the other instances are generally either very easy or prohibitively hard to solve. We define `Baseline` as the version of the branch-and-cut algorithm in which *all the classical* valid inequalities, i.e., static inequalities (7), capacity cuts (1d), 2-path cuts (8), and connectivity cuts (9) are available, but no infeasible-path and no path-matching constraints are separated. Regarding feasibility, the improved feasibility cuts (5) are used. Here and in the following experiments, the run time for each SDVRPTW instance is limited to 1,800 seconds.

The results of the `Baseline` branch-and-cut algorithm are presented in Table 1. We report, for each group of instances, the number of instances solved to proven optimality (*Solved*), the average computation time (*Time*) in seconds, the average lower bound ($\underline{z}^*$), the average number of branch-and-bound nodes inspected (*Nodes*), and the average number of feasibility checks performed (*Feas. checks*). In total, 77 of the 168 instances are solved to optimality with the `Baseline` branch-and-cut algorithm.

In a first experiment, we compare `Baseline` against `ClassicalFeasCut`, that is, the branch-and-cut algorithm with classical feasibility cuts (3) instead of improved cuts (5). Table 2 summarizes values for computation time (*Time*), number of branch-and-bound nodes (*Nodes*), and number of feasibility checks (*Feas. checks*) as average ratios relative to `Baseline`. More precisely, the numbers presented under columns

Table 2: Results obtained without using the new classes of valid inequalities and by using feasibility cuts (3) instead of (5).

			Classical Feasibility Cuts (3)			
Instances				Ratio		
Class	Q	#	Solved	Time	Nodes	Feas. checks
R1	30	12	2	1.05	1.00	1.36
	50	12	2	1.07	1.13	1.37
	100	12	2	1.00	1.01	1.17
C1	30	9	2	1.05	0.80	1.04
	50	9	1	5.34	2.15	7.69
	100	9	5	1.19	0.91	1.61
RC1	30	8	7	6.31	5.96	13.14
	50	8	1	13.36	2.70	10.98
	100	8	5	6.00	3.96	22.07
R2	30	11	0	1.00	0.97	1.08
	50	11	1	1.02	1.02	1.26
	100	11	1	1.05	1.04	1.12
C2	30	8	0	1.00	1.59	2.16
	50	8	0	1.24	1.15	1.66
	100	8	5	1.28	1.10	1.98
RC2	30	8	8	1.04	0.79	4.44
	50	8	3	6.87	2.46	9.28
	100	8	7	4.20	4.14	10.50
Total/Geom. Mean		*168*	*52*	1.83	1.45	2.74

Ratio are geometric means of the ratios of *Time*, *Nodes*, and *Feas. checks* taken over the eight to twelve instances of each class. For example, the number 1.05 in the first row means that the average ratio $\frac{Time^{\texttt{ClassicalFeasCut}}}{Time^{\texttt{Baseline}}}$ is above 1, indicating that the use of strengthened feasibility cuts accelerates the branch-and-cut by this factor on average for the group R1 with 50 customers and with capacity $Q = 30$. The last row of Table 2 is the geometric mean over all 168 instances.

The most striking result is that only 52 of the 168 instances are solved to optimality with classical feasibility cuts compared to 77 instances solved with the `Baseline` algorithm. Moreover, computation times of the version with classical cuts are consistently longer, on average the factor is 1.83. The impact on run times however strongly depends on the group of instances. It is most pronounced for the groups RC1 and RC2. The effect on the number of feasibility checks is also substantial as for groups RC1 and group RC2 with $Q = 100$ the average number of feasibility checks is reduced by a factor of more than 10 when improved feasibility cuts (5) are applied. Summing up, closing the very last percentages of the optimality gap often requires a large number of feasibility tests because many integer solutions are then found close the optimum. When many of them are infeasible, feasibility cuts have to be applied. What distinguishes our branch-and-cut from previous branch-and-cut algorithms such as the one by Archetti *et al.* (2014) for the SDVRP is that the new feasibility cuts keep lower bounds improving, while with the classical feasibility cuts (3) the process is often stalling. The results shown in Table 2 are a very clear indication that strengthening feasibility cuts is crucial for branch-and-cut using relaxed formulations. All following results therefore compare with `Baseline`, which includes the improved feasibility cuts (5).

In the next series of experiments, we analyze how much the new classes of valid inequalities contribute to the performance of the branch-and-cut algorithms. We compare `Baseline` against branch-and-cut algorithms in which (a) only infeasible-path constraints (`InfPathOnly`), (b) only path-matching constraints (`PathMatchOnly`), and (c) both types of constraints (`Both`) are separated. Table 3 is composed as Table 2.

Overall, `Baseline` solves only 77 instances, while all variants using new valid inequalities solve 92, 87, and 94 instances to proven optimality. The combination of infeasible-path constraints (10) and path-matching constraints (12) in `Both` is not only superior regarding the number of optimal solutions, it is also faster on average, reducing the average runtime to a factor of 0.65. Only for groups R1 with $Q = 50$ and $Q = 100$, RC1 with $Q = 30$, and R2 with $Q = 30$, the average run time is superior with the setting `PathMatchOnly`. With `Both`, the number of feasibility checks is reduced to less than one forth on average compared to `Baseline`. Interestingly, the number of inspected branch-and-bound nodes is higher in `Both` than in `InfPathOnly`, which is however a positive result, as more nodes are processed in less time.

We originally planned to also include a comparison of the best lower bounds $\underline{z}^*$ into Table 3. However, it

Table 3: Effectiveness of the new classes of valid inequalities.

Instances			Infeasible-Path Constr. (10)				Path-Matching Constr. (12)				Both (10) and (12)			
				Ratio				Ratio				Ratio		
Class	Q	#	Sol.	Time	Nodes	Feas. ch.	Sol.	Time	Nodes	Feas. ch.	Sol.	Time	Nodes	Feas. ch.
R1	30	12	3	1.03	0.72	0.61	2	1.04	0.93	0.53	3	0.93	0.88	0.38
	50	12	3	1.07	0.77	0.61	2	0.96	0.90	0.39	3	1.14	1.13	0.23
	100	12	2	1.07	0.71	0.64	2	1.00	0.91	0.43	2	1.05	0.91	0.44
C1	30	9	9	0.35	0.18	0.02	8	0.58	0.53	0.43	8	0.27	0.16	0.05
	50	9	9	0.43	0.24	0.08	8	0.84	0.71	0.29	9	0.38	0.23	0.05
	100	9	9	0.62	0.39	0.15	7	0.62	0.60	0.16	9	0.44	0.37	0.07
RC1	30	8	8	1.47	0.43	0.40	8	1.23	1.08	1.18	8	1.57	0.68	0.58
	50	8	8	0.13	0.03	0.04	8	0.27	0.13	0.06	8	0.13	0.02	0.04
	100	8	7	0.73	0.40	0.45	7	0.63	0.51	0.33	8	0.34	0.25	0.28
R2	30	11	0	1.00	0.70	0.76	1	0.99	0.86	0.43	0	1.00	0.81	0.49
	50	11	1	1.07	0.71	1.01	2	0.99	0.88	0.37	2	0.95	0.80	0.52
	100	11	1	1.01	0.67	0.64	1	1.04	0.93	0.22	1	1.00	0.85	0.22
C2	30	8	1	0.97	0.59	0.39	0	1.00	1.05	1.01	1	0.96	1.41	0.29
	50	8	1	0.99	0.66	0.56	1	0.95	0.94	0.56	2	0.95	0.82	0.64
	100	8	6	0.98	0.61	0.56	6	0.78	0.83	0.18	6	0.70	0.67	0.21
RC2	30	8	8	0.98	0.29	0.56	8	1.27	0.73	1.01	8	0.76	0.15	0.42
	50	8	8	0.20	0.01	0.07	8	0.26	0.06	0.10	8	0.17	0.01	0.06
	100	8	8	1.21	0.57	0.78	8	0.97	0.82	0.62	8	0.74	0.49	0.55
Total/Geom. Mean		*168*	*92*	0.76	0.37	0.34	*87*	0.81	0.65	0.36	*94*	0.65	0.40	0.23

turned out that the best lower bounds $\underline{z}^*$ do not differ much between the `Baseline` setup and the variants `InfPathOnly`, `PathMatchOnly`, and `Both`. Indeed, all ratios are 1.00, possibly different in the following digits. In particular, for the instances that are not solved, the remaining optimality gap is less than 1 % in about one fifth of the cases (16 out of 74 instances).

We conclude our study of the different classes of valid inequalities on the behavior of the branch-and-cut with an overview on number of separated cuts and separation times. We analyze the overall separation strategy described in at the beginning of Section 3.3. In Table 4, we report for each group of instances and each class of valid inequalities (all classes are available), the average number of generated cuts (*#cuts*) and the percentage of time (*%time*) spent with separation. It is clearly shown in the table that, with a few exceptions, capacity cuts are most frequently separated, which does not seem unusual because they are on the first level of the separation hierarchy. However, the average time for capacity cut separation remains below 4.8 % never exceeding 11.1 % in the maximum. All other classes of cuts are less frequently separated and consume even less time. The average remaining computing time of 86.1 % is consumed by internal procedures of the CPLEX solver for LP re-optimization, internal cuts separation, and primal heuristics etc.

4.2. Comparison with Branch-and-Price-and-Cut Algorithms

Up to now, the predominant exact solution algorithms for the SDVRPTW are based on path-based formulations solved with branch-and-price (see Section 1). We compare our new branch-and-cut approach against the currently leading branch-and-price(-and-cut) implementations presented by Archetti *et al.* (2011b) and Luo *et al.* (2016). In line with their experimental setups, we extend the computation time and set the run time limit to 1 hour. All 504 instances with $n = 25$, 50, and 100 customers are considered. The results are summarized in Table 5, again with one entry for each group of instances, i.e., grouped by n, classes (R1, C1, RC1, R2, C2, RC2), and capacity $Q = 30$, 50, 100. In addition to the already introduced indicators, we report the average number vehicles/routes (*Veh.*), and the average number of split customers (*Splits*). Reported values are averages per group and solved instances.

The total number of 277 optimally solved instances compared to 262 and 264 optimal solutions in the respective branch-and-price algorithms clearly shows that our branch-and-cut approach is competitive. In summary, 22 instances are solved to proven optimality for the first time. Looking into the details, all three approached solve all 168 instances with 25 customers. Compared to Archetti *et al.* (2011b), our computation times on these small-sized instances are most of the time significantly smaller or at least comparable with the exception of group R1 with $Q = 100$. Here, outliers r102, r103, and r110 consume 758, 206, and 219

Table 4: Performance indicators: 50-customer instances.

Instances			Capacity Cuts (1d)		2-Path Cuts (8)		Connectivity Cuts (9)		Infeasible-Path Constr. (10)		Path-Match. Constr. (12)		Feasibility Cuts (5)	
Class	Q	#	#cuts	%time	#cuts	%time	#cuts	%time	#cuts	%time	#cuts	%time	#cuts	%time
R1	30	12	8 071.2	2.7	8.3	0.7	0.3	2.1	219.3	0.5	56.8	0.1	23.4	0.4
	50	12	5 469.9	4.4	68.5	2.4	4.8	3.4	293.9	0.8	115.4	0.1	7.9	0.1
	100	12	2 039.1	6.7	651.9	8.9	74.3	3.6	669.3	0.9	190.1	0.1	1.0	0.8
C1	30	9	6 076.8	2.5	0.2	0.5	0.0	1.7	218.4	0.4	11.8	0.0	35.9	1.5
	50	9	3 721.9	4.9	1.3	1.5	2.0	4.0	172.3	0.7	27.0	0.1	16.2	1.0
	100	9	1 486.0	5.4	18.3	8.5	18.3	3.6	197.9	0.8	53.7	0.1	9.4	0.3
RC1	30	8	2 624.6	5.4	0.0	0.3	0.0	2.1	26.1	0.3	2.3	0.0	12.0	16.0
	50	8	1 674.1	7.0	0.0	0.9	0.7	3.3	20.8	0.6	4.3	0.1	4.7	2.6
	100	8	996.4	11.1	18.6	8.2	23.3	7.7	112.9	1.7	53.1	0.3	0.2	0.5
R2	30	11	8 360.4	2.4	0.0	0.6	0.0	2.0	188.6	0.4	28.9	0.1	49.5	1.0
	50	11	6 136.3	3.1	0.0	1.8	3.0	2.5	191.3	0.7	59.5	0.1	32.4	0.1
	100	11	3 753.7	3.4	0.3	11.5	88.4	2.8	498.8	0.8	125.6	0.1	5.4	0.0
C2	30	8	8 633.0	2.1	0.0	0.4	0.6	1.8	545.0	0.3	22.9	0.1	94.5	1.9
	50	8	6 762.0	3.0	0.0	1.2	6.3	2.7	494.1	0.6	66.9	0.1	81.8	2.2
	100	8	3 222.9	4.3	0.0	5.8	19.5	3.9	317.4	0.9	66.3	0.1	67.0	0.7
RC2	30	8	2 496.5	4.8	0.0	0.2	0.0	1.4	25.1	0.2	1.4	0.1	4.8	3.3
	50	8	1 416.1	7.6	0.0	0.3	0.6	2.6	13.1	0.7	1.4	0.0	0.4	3.2
	100	8	815.5	8.7	0.1	5.7	14.6	5.0	40.0	1.0	15.9	0.0	0.5	1.4
Total/Weighted Avg.		*168*	4 276.2	4.8	54.0	3.4	15.9	3.1	249.5	0.7	56.0	0.1	24.0	1.8

seconds, respectively. Moreover, our solutions seem to tend towards less split customers, while the number of employed vehicles is most of the time identical to the results of Archetti *et al.* (2011b).

For the 50-customer instances, there is no clear picture regarding a comparison of computation times. However, our algorithm really outperforms the others on this subset (17 and 13 more instance solved). Moreover, for 8 (11) instances with 50 customers for which an optimal solution is unknown, the optimality gap is below 0.5 % (1 %). Detailed results for every 50-customer instance are reported in Section B of the Appendix.

Finally, the branch-and-price of Archetti *et al.* (2011b) wins with three more instances solved on the 100-customer instances. Nevertheless, we have been able to solve to proven optimality two new instances in this subset, i.e., C101 and C105 for $Q = 100$, with optimal values 13,911 and 13,893, respectively. This subset, with only 8 out of 168 solved instances, is a hard challenge for all algorithms.

5. Conclusions

In this paper, we presented a new branch-and-cut-based algorithm for the SDVRPTW. The proposed algorithm and its components were thoroughly tested and it was shown to be competitive with recent branch-and-price-and-cut algorithms. We computed 22 new optimal solutions in the standard SDVRPTW benchmark derived from Solomon's (1987) VRPTW instances. Moreover, several lower and upper bounds were improved.

While path-based formulations of Desaulniers (2010), Archetti *et al.* (2011b), and Luo *et al.* (2016) underlying the branch-and-price-and-cut approaches can easily ensure feasible routes, feasibility modeling is the fundamental problem of any two-index formulation for the SDVRPTW. The major complication is that customers can or must be visited several times so that time and load-related attributes cannot be directly attached to the vertices of the associated digraph. Our new two-index formulation exploits several properties known to be valid for at least some optimal solution to an SDVRP(TW) instance. In particular, we attach time-related attributes to arcs because one property guarantees that no arc is traversed more than once. However, the model we propose is still a relaxation of the SDVRPTW.

Although being an SDVRPTW relaxation, our new formulation is fairly compact, enabling short LP re-optimization times, and it is free of symmetries making branching more effective compared to three-index formulations. Overall, the success of the new branch-and-cut algorithm can be attributed to two major innovations: First, we found a new way to cut off infeasible integer solutions. Our strengthened feasibility cuts refer to individual clusters that are induced by the infeasible integer solution at hand. Former approaches for

Table 5: Comparison with branch-and-price-and-cut algorithms.

Instances				Archetti *et al.* (2011b)				Luo *et al.* (2016)	Our method			
n	Class	Q	#	Sol.	Veh.	Splits	Time	Sol.	Sol.	Veh.	Splits	Time
25	R1	30	12	12	12.0	3.8	48	12	12	12.0	2.8	2.3
		50	12	12	7.2	1.1	5	12	12	7.3	0.3	1.4
		100	12	12	4.9	0.1	2	12	12	5.1	0.1	108.7
	C1	30	9	9	16.0	5.2	9	9	9	16.0	4.0	4.1
		50	9	9	10.0	2.1	5	9	9	10.0	1.0	1.2
		100	9	9	5.0	0.0	15	9	9	5.0	0.0	0.6
	RC1	30	8	8	17.8	7.1	5	8	8	18.0	5.8	0.4
		50	8	8	10.6	1.8	5	8	8	11.0	1.3	0.7
		100	8	8	6.0	0.4	2	8	8	6.0	0.0	0.3
	R2	30	11	11	12.0	3.9	165	11	11	12.0	2.1	3.5
		50	11	11	7.0	1.1	15	11	11	7.0	0.1	0.4
		100	11	11	3.8	0.1	24	11	11	4.0	0.0	3.7
	C2	30	8	8	16.0	6.4	9	8	8	16.0	4.0	10.3
		50	8	8	10.0	2.5	11	8	8	10.0	1.0	5.6
		100	8	8	5.0	1.0	19	8	8	5.0	0.0	2.2
	RC2	30	8	8	18.0	6.6	9	8	8	18.0	6.1	0.5
		50	8	8	10.8	1.8	12	8	8	11.0	1.0	0.7
		100	8	8	6.0	0.4	4	8	8	6.0	0.0	0.2
50	R1	30	12	0	—	—	—	0	4	25.0	12.8	1 331.1
		50	12	2	15.0	4.0	533	1	5	15.0	4.0	1 638.3
		100	12	6	9.7	0.8	553	6	2	10.5	0.0	7.8
	C1	30	9	3	29.0	10.7	219	9	9	29.0	4.0	460.8
		50	9	9	18.0	4.3	114	9	9	18.0	2.8	128.0
		100	9	8	8.8	1.0	353	7	9	9.0	0.2	297.9
	RC1	30	8	8	33.0	8.9	50	8	8	33.0	6.6	24.4
		50	8	8	20.0	4.4	11	8	8	20.0	2.6	3.9
		100	8	8	10.0	1.0	22	8	8	10.0	0.5	123.0
	R2	30	11	0	—	—	—	0	2	25.0	13.0	3 004.1
		50	11	0	—	—	—	0	3	15.0	6.3	1 249.9
		100	11	1	8.0	1.0	134	2	2	8.0	0.5	928.7
	C2	30	8	0	—	—	—	0	1	29.0	6.0	1 160.5
		50	8	7	18.0	7.1	395	8	3	18.0	3.0	1 557.3
		100	8	2	9.0	3.0	1 314	1	7	9.0	0.3	840.9
	RC2	30	8	8	33.0	9.2	161	8	8	33.0	6.5	10.4
		50	8	8	20.0	4.8	30	8	8	20.0	2.8	3.3
		100	8	8	10.0	0.9	94	8	8	10.0	0.6	24.7
100	R1	100	12	1	20.0	0.0	5	1	1	20.0	0.0	2.1
	C1	100	9	5	19.0	2.4	1 667	4	4	19.0	0.8	441.5
	C2	100	8	2	19.0	5.5	1 407	0	0	—	—	—
Total				*262*				*264*	*277*			

the SDVRP considered the entire vertex set instead of a generally much smaller cluster to define a feasibility cut. Second, we introduced two new classes of valid inequalities for the SDVRPTW, namely infeasible-path constraints and path-matching constraints. They both have the purpose to strengthen the formulation so that fractional solutions as well as infeasible integer solutions are cut off from the solution space. While the generalization of infeasible-path constraints must exclude any interaction of the considered path with other routes, the path-matching constraints focus on the interdependency of routes that share a customer receiving split deliveries. Indeed, what path-matching constraints are cutting off is infeasible configurations formed by two or more routes. As far as we know, this is the first class of valid inequalities in the vehicle routing context that addresses infeasibilities resulting from violations of timing and capacity constraints provoked by more that one route. We think that such a technique may also be helpful for other variants of vehicle routing problems, in which certain vertices or arcs are traversed more than once, e.g., in VRPs with intermediate replenishment (Muter *et al.*, 2014) or for routing battery electric vehicles that can be recharged at recharging stations (Desaulniers *et al.*, 2016).

Acknowledgement

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grants no. IR 122/5-2 and DR 963/2-1.

References

Archetti, C. and Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, **19**, 3–22.

Archetti, C., Hertz, A., and Speranza, M. (2006a). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, **40**, 64–73.

Archetti, C., Savelsbergh, M., and Speranza, M. G. (2006b). Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, **40**, 226–234.

Archetti, C., Bianchessi, N., and Speranza, M. G. (2011a). A column generation approach for the split delivery vehicle routing problem. *Networks*, **58**, 241–254.

Archetti, C., Bouchard, M., and Desaulniers, G. (2011b). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, **45**(3), 285–298.

Archetti, C., Bianchessi, N., and Speranza, M. (2014). Branch-and-cut algorithms for the split delivery vehicle routing problem. *European Journal of Operational Research*, **238**(3), 685–698.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2000). A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks*, **36**(2), 69–79.

Ascheuer, N., Fischetti, M., and Grötschel, M. (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, **90**(3), 475–506.

Belenguer, J., Martinez, M., and Mota, E. (2000). A lower bound for the split delivery vehicle routing problem. *Operations Research*, **48**, 801–810.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split delivery vehicle routing problem with time windows. *Operations Research*, **58**, 179–192.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). *The Vehicle Routing Problem with Time Windows*, chapter 5, pages 119–159. In Toth and Vigo (2014).

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*. DOI: 10.1287/opre.2016.1535.

Dror, M. and Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, **23**, 141–145.

Dror, M. and Trudeau, P. (1990). Split delivery routing. *Naval Research Logistics*, **37**, 383–402.

Fischetti, M., Salazar-González, J.-J., and Toth, P. (1995). Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*.

Frizzell, P. and Giffin, J. W. (1995). The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers and Operations Research*, **22**, 655–667.

Gendreau, M., Dejax, P., Feillet, D., and Gueguen, C. (2006). Vehicle routing with time windows and split delivery. Technical Report 2006-851, Laboratoire Informatique d'Avignon, Avignon, France.

Ho, S. and Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers and Operations Research*, **31**, 1947–1964.

Irnich, S., Schneider, M., and Vigo, D. (2014). Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications, Second Edition, MOS-SIAM Series on Optimization*, pages 241–271. SIAM, Philadelphia.

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

Luo, Z., Qin, H., Zhu, W., and Lim, A. (2016). Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost. *Transportation Science*. DOI: 10.1287/trsc.2015.0666.

Maffioli, F. and Sciomachen, A. (1997). A mixed-integer model for solving ordering problems with side constraints. *Annals of Operations Research*, **69**, 277–297.

Miller, D., Tucker, A., and Zemlin, R. (1960). Integer programming formulations of traveling salesman problems. *J. Assoc. Comput. Mach.*, **7**, 326–329.

Mullaseril, P. A., Dror, M., and Trudeau, P. (1997). Split-delivery routing heuristics in livestock feed distribution. *Journal of the Operational Research Society*, **48**, 107–116.

Muter, I., J.-F., C., and Laporte, G. (2014). A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. *Transportation Science*, **48**(3), 425–441.

Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L. (2003). On the capacitated vehicle routing problem. *Mathematical Programmming*, **94**, 343–359.

Sepúlveda, J., Escobar, J., and Adarme-Jaimes, W. (2014). An algorithm for the routing problem with split deliveries and time windows (SDVRPTW) applied on retail SME distribution activities. *DYNA (Colombia)*, **81**(187), 223–231.

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

Toth, P. and Vigo, D., editors (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Appendix

A. Simple Construction Heuristic

The following greedy algorithm is used for computing initial feasible SDVRPTW solutions: Customers are considered in the sequence according to their identifiers (from the smallest to the largest). A set of routes defining a feasible solution is then built incrementally. When a new route is created, the first customer in the sequence not completely served is inserted into the route. When the route is non-empty, the customer remaining unfulfilled demand that can be feasibly visited along the route at the cheapest cost is selected and inserted into the route. Each time a customer is inserted into a route, the quantity delivered to the customer is set to the maximum of the residual demand of the customer and the residual capacity of the vehicle. These residual quantities are then updated accordingly. The construction of a route terminates when no further customer can be feasibly inserted. When all the customers are fully served, a feasible solution is available.

The solution computed by the constructive heuristic depends on the sequence in which customers are considered. Therefore, we execute the constructive heuristic n times, retaining the best among the n computed solutions. In the kth iteration, customers are cyclicly exchanged so that the sequences then begins with customer k instead of customer 1.

B. Detailed Computational Results

We report in Tables 6-11 the detailed results for the 50-customer instances. Each table shows the vehicle capacity (Q), the name of the instance (*Name*), the known optimal value (*Opt.*), the final upper bound ($\bar{z}^*$) and lower bound ($\underline{z}^*$) computed by the branch-and-cut algorithm, the percentage gap $100\,\%(\bar{z}^* - \underline{z}^*)/\underline{z}^*$ (*Gap (%)*), the number of vehicles/routes (*Veh.*) and split customers (*Splits*) in the upper bound solution, and, when the optimality gap is null, the computation time (*Time*) required to solve the instance to optimality (in seconds). The symbol † is reported close to each known optimal value if found by Luo *et al.* (2016) for the first time. All the other optimal values are taken from (Archetti *et al.*, 2011b). When our branch-and-cut algorithm is able to solve an instance to optimality for the first time, the corresponding upper and lower bounds as well as the optimality gap are highlighted in bold. Conversely, when our branch-and-cut algorithm is not able to solve an instance for which an optimal value is known, the optimal value is highlighted in bold.

Finally, whenever our values in columns $\underline{z}^*$ and $\bar{z}^*$ are inconsistent what was reported in previous papers, the symbol * is attached. This happens in 8 out of 92 cases. To explain this discrepancy, note that the triangle inequality is assumed to hold for travel times and costs. Therefore, in order to fulfill these assumptions, at pre-processing time we apply the Floyd-Warshall algorithm to travel times and costs independently. Archetti *et al.* (2011b) informed us that they assumed the validity of the triangle inequality, but they did however not pre-process the instances. Luo *et al.* (2016) informed us that they also assumed the validity of the triangle inequality, but they did a slightly different pre-processing. In their approach, they kept the distinction between travel and service times, and therefore they did not add service times to the travel times. Hence, travel times and costs are both identical to distances. To ensure the validity of the triangle inequality, they updated the distances by applying a shortest path algorithm. As a result, their times can be smaller than times produced with our pre-processing.

Table 6: Detailed results for class R1 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	r101		**16191**	**16191.0**	**0**	25	9	912
	r102		**15813**	**15813.0**	**0**	25	14	3059
	r103		15679	15511.5	1.08	25	15	
	r104		15522	15234.4	1.89	25	14	
	r105		**15803**	**15803.0**	**0**	25	14	545
	r106		**15596**	**15596.0**	**0**	25	14	809
	r107		15624	15316.8	2.01	25	13	
	r108		15855	15174.6	4.48	25	13	
	r109		15628	15523.2	0.68	25	11	
	r110		15877	15247.3	4.13	25	10	
	r111		15649	15371.2	1.81	25	14	
	r112		16119	15156.1	6.35	27	9	
50	r101	11911	11911	11911.0	0	15	3	25
	r102	**11142**	11142	11112.3	0.27	15	4	
	r103		10868	10615.8	2.38	15	10	
	r104		10787	10327.0	4.45	15	7	
	r105		**11325**	**11325.0**	**0**	15	5	1129
	r106		**10802**	**10802.0**	**0**	15	5	741
	r107		10846	10496.2	3.33	15	6	
	r108		10637	10254.3	3.73	16	5	
	r109		**10818**	**10818.0**	**0**	15	3	3577
	r110		12381	10335.5	19.79	17	5	
	r111		**10615**	**10615.0**	**0**	15	4	2719
	r112		10711	10290.5	4.09	16	4	
100	r101	10440	10440	10440.0	0	12	0	0
	r102	**9132**	9144	8404.7	8.8	11	0	
	r103	**8047**	8113	7419.4	9.35	10	0	
	r104		7268	6865.1	5.87	8	2	
	r105	9182	9182	9182.0	0	9	0	15
	r106	**8215**	8238	7731.1	6.56	9	1	
	r107		8132	7011.4	15.98	9	1	
	r108		7417	6766.4	9.62	8	2	
	r109	**8042**	8105	7664.4	5.75	9	0	
	r110		7697	6940.3	10.9	8	0	
	r111		7659	7097.7	7.91	8	1	
	r112		8205	6696.5	22.53	8	1	

Table 7: Detailed results for class C1 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	c101	15995†	15995	15995.0	0	29	4	187
	c102	15995†	15995	15995.0	0	29	4	296
	c103	15983	15983	15983.0	0	29	4	1911
	c104	15983	15983	15983.0	0	29	4	853
	c105	15995†	15995	15995.0	0	29	4	102
	c106	15995†	15995	15995.0	0	29	4	170
	c107	15995†	15995	15995.0	0	29	4	107
	c108	15983†	15983	15983.0	0	29	4	421
	c109	15983	15983	15983.0	0	29	4	100
50	c101	10158	10158	10158.0	0	18	3	44
	c102	10130	10130	10130.0	0	18	3	132
	c103	10123	10123	10123.0	0	18	3	322
	c104	10102	10102	10102.0	0	18	2	249
	c105	10158	10158	10158.0	0	18	3	18
	c106	10158	10158	10158.0	0	18	3	37
	c107	10158	10158	10158.0	0	18	3	54
	c108	10119	10119	10119.0	0	18	3	82
	c109	10101	10101	10101.0	0	18	2	215
100	c101	5876	5876	5876.0	0	9	0	8
	c102	5847	5847	5847.0	0	9	0	77
	c103	5821	5821	5821.0	0	9	0	814
	c104		**5788**	**5788.0**	**0**	9	0	1220
	c105	5876	5876	5876.0	0	9	0	4
	c106	5876	5876	5876.0	0	9	0	3
	c107	5876	5876	5876.0	0	9	0	12
	c108	5841	5841	5841.0	0	9	1	58
	c109	5798	5798	5798.0	0	9	1	485

Table 8: Detailed results for class RC1 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	rc101	27395	27395	27395.0	0	33	7	5
	rc102	27395	27395	27395.0	0	33	7	7
	rc103	27395	27395	27395.0	0	33	7	8
	rc104	27395	27395	27395.0	0	33	6	113
	rc105	27396	27396	27396.0	0	33	6	10
	rc106	27395	27395	27395.0	0	33	7	14
	rc107	27395	27395	27395.0	0	33	7	5
	rc108	27395	27395	27395.0	0	33	6	34
50	rc101	17083	17083	17083.0	0	20	2	2
	rc102	17005	17005	17005.0	0	20	1	8
	rc103	16968	16968	16968.0	0	20	3	3
	rc104	16967	16967	16967.0	0	20	3	3
	rc105	17001	17001	17001.0	0	20	1	4
	rc106	16990	16990	16990.0	0	20	3	6
	rc107	16986	16986	16986.0	0	20	4	3
	rc108	16967	16967	16967.0	0	20	4	4
100	rc101	9905	9905	9905.0	0	10	2	10
	rc102	9602	9602	9602.0	0	10	1	183
	rc103	9362	9362	9362.0	0	10	0	652
	rc104	9159	9159	9159.0	0	10	0	4
	rc105	9574	9574	9574.0	0	10	0	121
	rc106	9364	9364	9364.0	0	10	1	12
	rc107	9151	9151	9151.0	0	10	0	1
	rc108	9119	9119	9119.0	0	10	0	1

Table 9: Detailed results for class R2 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	r201		**15786**	**15786.0**	**0**	25	12	2989
	r202		**15596**	**15596.0**	**0**	25	14	3019
	r203		15661	15264.0	2.6	25	11	
	r204		16030	15196.5	5.48	25	10	
	r205		15666	15480.8	1.2	25	11	
	r206		15569	15493.3	0.49	25	13	
	r207		15553	15312.9	1.57	25	14	
	r208		15937	15188.3	4.93	26	10	
	r209		15587	15368.3	1.42	25	13	
	r210		15688	15422.3	1.72	25	14	
	r211		16180	15186.2	6.54	26	10	
50	r201		**11078**	**11078.0**	**0**	15	6	263
	r202		**10802**	**10802.0**	**0**	15	6	645
	r203		10644	10447.7	1.88	15	5	
	r204		10676	10312.5	3.52	15	7	
	r205		10859	10749.2	1.02	15	5	
	r206		**10715**	**10715.0**	**0**	15	7	2841
	r207		10629	10413.5	2.07	15	11	
	r208		10535	10264.3	2.64	15	5	
	r209		10545	10437.5	1.03	15	4	
	r210		10729	10646.9	0.77	15	4	
	r211		10909	10249.3	6.44	16	5	
100	r201	8430†	8432*	8432.0*	0	8	0	71
	r202		7716	7500.8	2.87	8	2	
	r203		7206	6983.7	3.18	8	2	
	r204		6919	6754.2	2.44	8	4	
	r205	7589	7588*	7588.0*	0	8	1	1786
	r206		7281	7247.5	0.46	8	2	
	r207		7086	6923.0	2.35	8	5	
	r208		7196	6700.4	7.4	8	3	
	r209		7204	7021.5	2.6	8	1	
	r210		7454	7227.1	3.14	8	1	
	r211		7024	6706.7	4.73	8	0	

Table 10: Detailed results for class C2 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	c201		**17790**	**17790.0**	**0**	29	6	1161
	c202		17788	17709.5	0.44	29	6	
	c203		17785	17639.0	0.83	29	6	
	c204		17788	17583.8	1.16	29	6	
	c205		17786	17731.4	0.31	29	6	
	c206		17784	17718.4	0.37	29	6	
	c207		17784	17730.0	0.3	29	6	
	c208		17784	17737.7	0.26	29	6	
50	c201	11598	11597*	11597.0*	0	18	3	204
	c202	11573	11572*	11572.0*	0	18	3	2893
	c203	**11571**	11576	11417.5	1.39	18	3	
	c204	**11569**†	11682	11384.3	2.62	18	4	
	c205	11571	11570*	11570.0*	0	18	3	1575
	c206	**11571**	11570*	11484.9	0.74	18	3	
	c207	**11571**	11570*	11464.0	0.92	18	3	
	c208	**11571**	11570*	11532.1	0.33	18	3	
100	c201		**6931**	**6931.0**	**0**	9	2	166
	c202	6862	6862	6862.0	0	9	0	803
	c203		6854	6827.7	0.38	9	0	
	c204		**6848**	**6848.0**	**0**	9	0	2703
	c205	6848	6848	6848.0	0	9	0	223
	c206		**6848**	**6848.0**	**0**	9	0	594
	c207		**6848**	**6848.0**	**0**	9	0	896
	c208		**6848**	**6848.0**	**0**	9	0	501

Table 11: Detailed results for class RC2 and $n = 50$ customers.

Q	Name	Opt.	$\bar{z}^*$	$\underline{z}^*$	Gap (%)	Veh.	Splits	Time
30	rc201	27395	27395	27395.0	0	33	7	7
	rc202	27395	27395	27395.0	0	33	7	7
	rc203	27395	27395	27395.0	0	33	6	5
	rc204	27395	27395	27395.0	0	33	6	14
	rc205	27395	27395	27395.0	0	33	7	14
	rc206	27395	27395	27395.0	0	33	6	9
	rc207	27395	27395	27395.0	0	33	6	15
	rc208	27395	27395	27395.0	0	33	7	13
50	rc201	17083	17083	17083.0	0	20	2	2
	rc202	17005	17005	17005.0	0	20	1	4
	rc203	16968	16968	16968.0	0	20	3	6
	rc204	16967	16967	16967.0	0	20	4	4
	rc205	17004	17004	17004.0	0	20	1	2
	rc206	16990	16990	16990.0	0	20	4	1
	rc207	16986	16986	16986.0	0	20	4	2
	rc208	16967	16967	16967.0	0	20	3	4
100	rc201	9662	9662	9662.0	0	10	2	1
	rc202	9465	9465	9465.0	0	10	1	151
	rc203	9264	9264	9264.0	0	10	1	33
	rc204	9159	9159	9159.0	0	10	0	1
	rc205	9467	9467	9467.0	0	10	1	3
	rc206	9408	9408	9408.0	0	10	0	2
	rc207	9241	9241	9241.0	0	10	0	5
	rc208	9119	9119	9119.0	0	10	0	1