Branch-and-Price-and-Cut for the Periodic Vehicle Routing Problem with Flexible Schedule Structures

Ann-Kathrin Rothenbächer*,[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Mainz D-55099, Germany.*

Abstract

This paper addresses the periodic vehicle routing problem with time windows (PVRPTW). Therein, customers require one or several visits during a planning horizon of several periods. The possible visiting patterns (schedules) per customer are limited. In the classical PVRPTW, it is common to assume that each customer requires a specific visit frequency and offers all corresponding schedules with regular intervals between the visits. In this paper, we permit all kinds of schedule structures and the choice of the service frequency. We present an exact branch-and-price-and-cut algorithm for the classical PVRPTW and its variant with flexible schedules. The pricing problems are elementary shortest path problems with resource constraints. They can be based on one of two new types of networks and solved with a labeling algorithm, which uses several known acceleration techniques such as the *ng*-path relaxation and dynamic half-way points within bidirectional labeling. For instances whose schedule sets fulfill a certain symmetry condition, we present specialized improvements of the algorithm such as constraint aggregation and symmetry breaking. Computational tests on benchmark instances for the PVRPTW show the effectiveness of our algorithm. Furthermore, we analyze the impact of different schedule structures on run times and objective function values.

Key words: Vehicle Routing, Multi Period, Branch-and-Price-and-Cut, Service Choice, Flexible Schedules

1. Introduction

The periodic vehicle routing problem (PVRP) extends the well-known vehicle routing problem to a planning horizon of multiple periods, which repeats continuously over time. For the sake of simplicity, we refer to a *day* instead of a period in the following. Customers need to be served one or several times during a planning horizon, following one of their offered visiting patterns. By respecting time windows of the customers during each day, the PVRP becomes the periodic vehicle routing problem with time windows (PVRPTW). The problem occurs in many real-world applications, for example, in product delivery, in waste collection and in some health-care services.

The PVRP can be described as follows. Starting from a given depot, a set of capacitated vehicles can perform round trips with a limited duration on every day. These round trips, called *routes*, serve a subset of the customers and may differ from day to day. Every customer has a daily demand and offers a set of convenient visiting patterns that are called *schedules*. A schedule contains a subset of the considered days whose combination is allowed for visiting the customer. Through the visiting days specified by a schedule, the delivery amount for every visit is also determined. For example, regarding a planning horizon of six days from Monday to Saturday, a customer may offer to be visited either on Monday and Thursday, or on Tuesday and Friday, or on Wednesday and Saturday. In every case, the demand for three days has to be delivered at each visit. The goal is to minimize the overall routing costs.

*Corresponding author.
Email address: anrothen@uni-mainz.de (Ann-Kathrin Rothenbächer)

In general, the PVRP combines an assignment and a vehicle routing decision. Exactly one schedule has to be selected for every customer. If this decision was fixed, a vehicle routing problem would need to be solved for every day to determine the set of customers served on the same route and the order in which they are served.

In the PVRP literature, it is common to assume a special periodic structure of the schedules. A schedule is called *regular* if the intervals between two visits are always equal (also with a cyclic repetition of the planning horizon). Two schedules are *overlapping* if both allow a visit on the same day. Typically, the schedules of a customer are assumed to be regular and non-overlapping. Furthermore, the number of required visits to a certain customer during the planning horizon, called the *visit frequency*, is usually specified. In addition, former publications usually presumed that every customer can be visited on every day. We call a set of schedules *standard* if all schedules are regular, pairwise non-overlapping, require the same visit frequency and if every day is contained in at least one schedule. The customers' schedule sets in existing PVRPTW benchmark instances are always standard. In this paper, we will relax these assumptions and consider additionally irregular and overlapping schedules as well as different visit frequencies per customer.

A small example of a PVRP with a planning horizon of four days $(0, 1, 2, 3)$ and its optimal solution for one available vehicle are depicted in Figure 1. For every customer (A, B, C, D, E), the offered schedules are indicated close to each node and the chosen schedule is underlined. Some customers offer standard schedule sets (i.e., customers A and E). In contrast, the schedule set of customer B fulfills none of the properties for standard schedule sets. The schedule $\{0, 2, 3\}$ is irregular because it induces visit intervals of 1 and 2 days. All schedules of B are overlapping because each allows a visit on day 0. The visit frequency varies between 2 and 3 visits during the planning horizon. Moreover, none of his schedules allows a visit on day 1. The optimal solution for the visits at customer D shows that it may be beneficial to not choose the schedule with the lowest visit frequency.

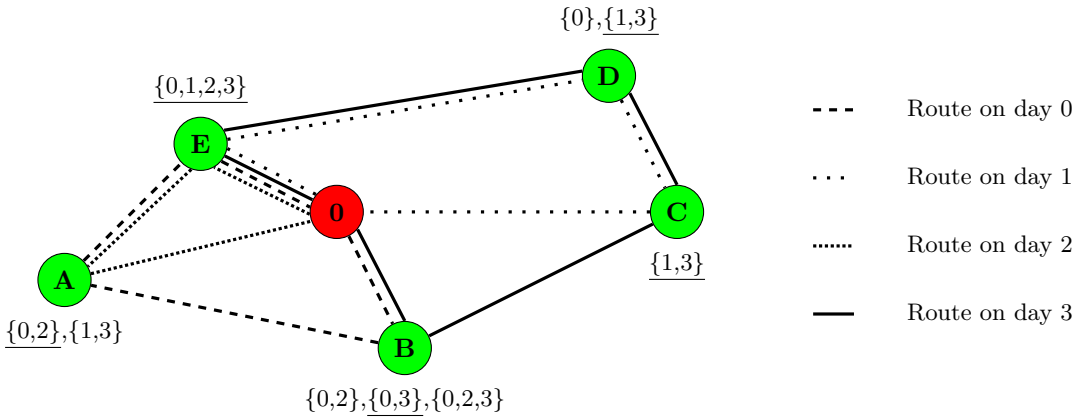

Figure 1: Example of a PVRP

The main contribution of the paper at hand is the presentation of the first exact algorithm for the classical PVRPTW and for variants with all possible schedule structures. We propose a branch-and-price-and-cut algorithm based on a new extended set-partitioning formulation. The pricing problem is an elementary shortest path problem with resource constraints (ESPPRC). It is solved with a bidirectional labeling algorithm, which can operate on two new types of auxiliary networks. Furthermore, we define a symmetry concept depending on certain properties of the customers' schedule sets. For instances that have this symmetry property, we present some techniques to reduce the number of pricing problems and the size of the branch-and-bound tree.

The remainder of this paper is structured as follows. Section 2 gives an overview of the related literature. In Section 3, we formally introduce the problem and present the master problem. Section 4 describes our exact branch-and-price-and-cut algorithm. It includes the definition of two new types of pricing networks, which are especially designed for non-standard schedule sets, and the labeling algorithm for the solution

of the arising ESPPRC. Moreover, it presents our branching strategy and the incorporation of subset-row inequalities (Jepsen *et al.*, 2008). Section 5 explains ideas for tackling certain instances that fulfill a symmetry property. In Section 6, we present the results of our computational study. Finally, we finish with a conclusion and a short outlook in Section 7.

2. Literature review

Research on the PVRP started with Beltrami and Bodin (1974) who tackled a problem of waste collection over several days through heuristics, treating the assignment and the routing successively. Russell and Igo (1979) and Christofides and Beasley (1984) formalized the problem, coined the name PVRP, and presented further heuristics. Since then a lot of applications and sophisticated heuristics have been described in the literature on the classical PVRP without time windows (Hemmelmayr *et al.*, 2009; Cacchiani *et al.*, 2014), and with time windows (Cordeau *et al.*, 2004; Pirkwieser and Raidl, 2009a; Vidal *et al.*, 2013). Surveys on the PVRP can be found in (Francis *et al.*, 2008), (Campbell and Wilson, 2014), and (Irnich *et al.*, 2014). Pirkwieser and Raidl (2009b) were the first to compute lower bounds for the PVRPTW by solving the linear relaxation of a set-covering formulation with column generation. The ESPPRC subproblems were solved with a labeling algorithm. The only exact approach for the pure PVRP so far was presented in (Baldacci *et al.*, 2011a). They used three relaxations based on a set-partitioning formulation to obtain lower bounds, generated all remaining routes with reduced costs smaller than the optimality gap, and finally solved the problem with an IP solver.

The literature on non-standard schedule structures is very sparse. Gaudioso and Paletta (1992) proposed a three-phase heuristic for a PVRP, in which schedules were allowed to be irregular and overlapping, but every customer still had a fixed visit frequency. More recently, Francis *et al.* (2006) studied selectable visit frequencies and irregular schedules in the so-called PVRP with Service Choice. With their exact approach based on Lagrangean relaxation and branch-and-bound, they showed possible savings through an increased visit frequency. However, they limited their study to non-overlapping schedules and used an estimation of customer demand per visit. The same authors published a Tabu Search heuristic, which is able to tackle the full range of possible schedules (Francis *et al.*, 2007). Moreover, they analyzed the impact of different dimensions of operational flexibility on different aspects of service consistency.

A summarizing comparison of the paper at hand and the most related papers, concerning the solution approach and the considered problem aspects, can be found in Table 1.

Table 1: Comparison of our paper with the related literature

	Our paper	(Pirkwieser and Raidl, 2009b)	(Baldacci *et al.*, 2011a)	Gaudioso and Paletta (1992)	Francis *et al.* (2006)	(Francis *et al.*, 2007)
Computes lower bounds	✓	✓	✓		✓	
Computes upper bounds	✓		✓	✓	✓	✓
Considers time windows	✓	✓				
Allows irregular schedules	✓			✓	✓	✓
Allows overlapping schedules	✓			✓		✓
Allows different visit frequencies	✓				✓	✓

Some other problems are closely related to the PVRP. The *Multi-Period VRP* (MPVRP) deals with a planning horizon of several days during which every customer offers a time span of several consecutive days in which he has to be served once. Bostel *et al.* (2008) solved an MPVRP with multiple depots, taking into account labor rules with a metaheuristic and an exact column-generation approach. A dynamic variant of

the MPVRP, in which customer demands are revealed over time, was tackled with a three-phase heuristic by Wen *et al.* (2010) and with an exact approach by Baldacci *et al.* (2011a) (who called it Tactical Planning VRP). A branch-and-price algorithm for the MPVRP with time windows was presented by Athanasopoulos and Minis (2013). They also described some techniques to reduce the number of pricing-problem solutions using the similarities of the pricing problems of the different days. Recently, Archetti *et al.* (2015) introduced a branch-and-cut algorithm for a MPVRP variant that considers inventory costs.

The *inventory routing problem* (IRP) also deals with a planning horizon of several days, but in this case the customers need to be served such that no stock-out occurs. Delivery quantities, visiting days, and visit frequencies are not predetermined but part of the decision problem. Recent surveys can be found in (Bertazzi *et al.*, 2008) and (Coelho *et al.*, 2014).

Closely related to the IRP is the *flexible PVRP*, which was recently introduced and solved by Archetti *et al.* (2017) with a load-based MIP formulation and some valid inequalities. Again, given a planning horizon of several days and a total demand for every customer for this time span, the visiting days and the delivery quantities can be chosen freely. Only the amount to deliver per customer and day is limited whereas the visiting day combinations are not limited to schedules offered by the customers.

3. Problem Formulation

The PVRPTW can be stated as follows. Let $P = \{0, \ldots, |P| - 1\}$ be a planning horizon of several days, which repeats continuously meaning that the last period $(|P| - 1)$ is again followed by the first period 0. A set of customers N and a single depot d represent all relevant locations $O = N \cup \{d\}$. On each day, a set of m homogeneous vehicles with capacity Q and a maximal route duration D is available to perform routes starting from and ending at the depot d and visiting a subset of the customers N in between. A route is defined by the ordered set of visited customers and the amounts delivered to each customer. Let R be the set of all feasible routes and $R^p \subset R$ be the subset feasible for day p. The costs c_r for a route r depend exclusively on the traveled distance. The objective is to minimize the sum of all route costs.

Every customer $n \in N$ has a specific demand per day q_n, a time window $[a_n, b_n]$ and a set of offered schedules $S_n \subseteq \mathcal{P}(P)$, with $\mathcal{P}$ indicating the power set. Both the customers' demands and time windows could be different for every day without affecting the applicability of our solution approach. However, the classical PVRPTW assumes identical demands and time windows per customer over the planning horizon such that we omit a day index for the ease of notation. The delivery of the demand of customer n for one day p is called the *task* n^p. The task n^p does not need to be fulfilled on day p. Instead, we assume that at every visit to a customer, his tasks for all days up to the next visit have to be fulfilled (taking into account the cyclic repetition of the planning horizon). Hence, in a planning horizon of four days $(0, 1, 2, 3)$, visits to a customer A on the days 1 and 2 require the fulfillment of the day-1 task (A^1) at the first visit and the fulfillment of the tasks of the days 2, 3 and 0 (A^2, A^3, A^0) at the second visit. Assuming a daily demand of one unit for this customer, one unit has to be delivered at the first visit and three units have to be delivered at the second visit.

As mentioned in the introduction, the offered schedule sets of a customer S_n are allowed to contain any number and structure of schedules. For a given schedule, we call every induced pair of visiting day and number of fulfilled tasks a *schedule part*. For example, in a planning horizon from Monday to Saturday, the schedule {Tuesday, Friday} induces the schedule parts (Tuesday, 3) and (Friday, 3), because it represents the delivery of the demand for three days on Tuesday and Friday, respectively. The schedule part of a customer n that represents the delivery of the demand of l days at a visit on day p is denoted by $n^{p:l}$. The number of covered demand days l of a schedule part $n^{p:l}$ is also called the *length* of the schedule part and is an element of the set $L = \{1, ..., |P|\}$. Let the set $S_n^{pl} \subseteq S_n$ identify the subset of schedules offered by customer n that include the schedule-part $n^{p:l}$.

Note that for regular schedules and a fixed visit frequency per customer, the amount to deliver per visit (and thus the length of each schedule part) is fixed and independent of the chosen schedule. Obviously, this is not the case for irregular schedules as different numbers of tasks can be fulfilled by one visit. Furthermore, by allowing overlapping schedules, the demand to deliver is not even determined when a visiting day is chosen.

Route-based formulation. Let the binary variable λ_r^p indicate whether the route $r \in R^p$ is performed on day $p \in P$. Moreover, let the variable z_n^s indicate whether the schedule $s \in S_n$ is chosen for customer $n \in N$. Then, a route-based set-partitioning model for the PVRPTW can be formulated as follows:

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

$$\text{s.t.} \quad \sum_{s \in S_n} z_n^s = 1 \qquad (\pi_n) \qquad \forall\, n \in N \tag{1b}$$

$$\sum_{r \in R^p} a_{rn}^{pl} \lambda_r^p = \sum_{s \in S_n} b_s^{pl} z_n^s \qquad (\rho_n^{pl}) \qquad \forall\, n \in N, p \in P, l \in L : S_n^{pl} \neq \varnothing \tag{1c}$$

$$\sum_{r \in R^p} \lambda_r^p \leq m \qquad (\mu_p) \qquad \forall\, p \in P \tag{1d}$$

$$z_n^s \in \{0,1\} \qquad \forall\, n \in N, s \in S_n \tag{1e}$$

$$\lambda_r^p \in \{0,1\} \qquad \forall\, p \in P, r \in R^p \tag{1f}$$

with the parameter a_{rn}^{pl} stating how often the route r includes the schedule part $n^{p:l}$ and the parameter b_s^{pl} stating whether schedule s induces the schedule part $(p{:}l)$.

The objective function (1a) minimizes the total routing costs. The first constraints (1b) ensure that for each customer exactly one of its offered schedules is chosen. Constraints (1c) link the route and schedule variables. A schedule variable may only be selected if all its induced schedule parts are included in a realized route. Note that the equality signs in the former constraints can be replaced by a $\geq$-sign to reduce the dual space whenever the triangle inequality for the costs holds. The constraints (1d) limit the number of available vehicles on every day. Finally, (1e) and (1f) restrict the variables to binary values.

Note that summing up constraints (1c) over all possible schedule part lengths $l \in L$ leads to the equalities

$$\sum_{r \in R^p} a_{rn} \lambda_r^p = \sum_{s \in S_n} b_s^p z_n^s \qquad \forall\, n \in N, p \in P \tag{2}$$

where the parameter a_{rn} states how often the route r visits the customer n and the parameter b_s^p states whether the schedule s requires a visit on day p. This aggregated variant is common in the literature (see Pirkwieser and Raidl, 2009b; Baldacci *et al.*, 2011a). However, it is not suitable for the case of overlapping and irregular schedules, since visiting a customer on a certain day does not specify how much demand is delivered but would be accepted for all schedules including this day. For example, consider $|P| = 4$ and a customer who offers the two overlapping schedules {0} and {0,2}, and a route performed on day 0 which delivers the customer's demand for two days. The constraints (2) would accept to set the schedule variable referring to the schedule {0} to 1, but this would lead to unfulfilled demand on the last two days. Nevertheless, for the case of standard schedule sets, both constraints (1c) and (2) are identical, because for every customer and visiting day there exists exactly one schedule part.

Note further that the other intuitive idea to use covering-of-tasks constraints instead of (1c), i.e.

$$\sum_{p \in P} \sum_{r \in R^p} a_{rn^{p'}} \lambda_r^p \geq 1 \qquad \forall\, n \in N, p' \in P$$

where the parameter $a_{rn^{p'}}$ states how often the route r fulfills the task $n^{p'}$, is also not valid in general because it would enable unallowed combinations of schedule parts. For example, consider $|P| = 3$ and a customer who offers the schedules {0,1}, {1,2}, and {0,2}. Then the combination of three routes fulfilling each one of the allowed schedule parts 0:1, 1:1 and 2:1 would lead to the schedule {0,1,2}, which is not offered by the customer.

4. Branch-and-Price-and-Cut Algorithm

The route-based formulation (1) is called the master problem and is solved with a branch-and-price-and-cut algorithm. The column-generation procedure manages the solution of the LP relaxation of this

master problem. Valid inequalities can be separated and added to the formulation to tighten the LP bound. Branching decisions force the variables to have integer values. In the following, we describe in Section 4.1 the solution of the pricing problem to generate columns, in Section 4.2 the incorporation of the subset-row inequalities introduced by Jepsen *et al.* (2008), and in Section 4.3 the used branching strategy for the PVRPTW.

4.1. Column Generation

Because the number of feasible routes is huge, promising routes should be added dynamically. Thus, we use column generation for the route variables (see Lübbecke and Desrosiers, 2005). The restricted master problem (RMP) is defined as model (1) modified by including only a subset of all possible routes $R' \subseteq R$ and by relaxing the integrality conditions for all variables. Let the dual variables ρ_n^{pl} and μ_p be associated with constraints (1c) and (1d), respectively. Then, the reduced costs $\tilde{c}_r$ of a route $r \in R^p$ performed on day p are given by

$$\tilde{c}_r = c_r - \sum_{n \in N} \sum_{l \in L} a_{rn}^{pl} \cdot \rho_n^{pl} - \mu_p.$$

In general, pricing problems have to find columns with negative reduced costs or need to detect that no more such columns exist. Here, an ESPPRC pricing problem needs to be solved for every day (see Irnich and Desaulniers, 2005). In the following, we introduce two different ways to model the underlying pricing networks that respect the challenges resulting from irregular and overlapping schedules, and the correct treatment of the demand. Afterward, we present the dynamic-programming labeling algorithm to solve the ESPPRC.

4.1.1. Pricing networks

For standard schedule sets, a route is uniquely determined by the ordered set of visited customers. The reason is that for every customer the number of fulfilled tasks by each visit is identical and thus the demand to deliver is known. For this case, a pricing network consisting of one vertex for every customer and the depot would suffice. However, for irregular schedules, a route has to specify how many tasks are fulfilled by each visit. Therefore, the vertices in the pricing network need to distinguish for every customer which of his tasks are fulfilled when the vertex is visited. Moreover, for non-standard schedule sets, the pricing networks differ per day. We propose two structurally different layouts for the pricing networks, namely the *schedule part network* and the *task network*.

Let $t_{o_1 o_2}$ and $c_{o_1 o_2}$ express the time and cost for traveling from location $o_1 \in O$ to location $o_2 \in O$, respectively (with $t_{o_1 o_2}$ including the service time at location o_1).

Schedule part network. We define the schedule part network for a day p as the graph $G_{SP}^p = (V_{SP}^p, A_{SP}^p)$ with vertices V_{SP}^p and arcs A_{SP}^p. The set of vertices V_{SP}^p consists of one vertex for every allowed schedule part of a customer with a visit on day p and two vertices d^+ and d^- modeling two copies of the depot. A schedule part $n^{p:l}$ is allowed, if at least one schedule $s \in S_n$ exists, that induces the schedule part $(p : l)$. Let $o(j) \in O$ be the location and $l(j)$ be the length of the schedule part represented by a vertex $j \in V_{SP}^p \setminus \{d^+, d^-\}$, respectively. The set of arcs A_{SP}^p contains all arcs $(i, j) \in (V_{SP}^p \setminus \{d^-\}) \times (V_{SP}^p \setminus \{d^+\})$ between two vertices of different locations, i.e. with $o(i) \neq o(j)$. Obviously, some arcs can be eliminated due to the customers' time windows.

An example of the schedule part networks of an instance with $|P| = 4$ and two customers A and B is depicted in Figure 2. For the sake of conciseness, we assume that the time windows require customer A to be visited before customer B. The figure shows that the networks of the different days are completely separated. It becomes clear that the number of vertices and the number of arcs strongly depends on the number and structure of offered schedules. For a customer with a standard schedule set as customer A, exactly one vertex in the network of each day is necessary. On the other hand, a customer who offers all possible schedules needs $|P|$ vertices per daily network. Because all vertices of one customer need an outgoing arc to all vertices of all (time-feasible) successive customers, the number of arcs per day can reach $\mathcal{O}(|N|^2|P|^2)$.

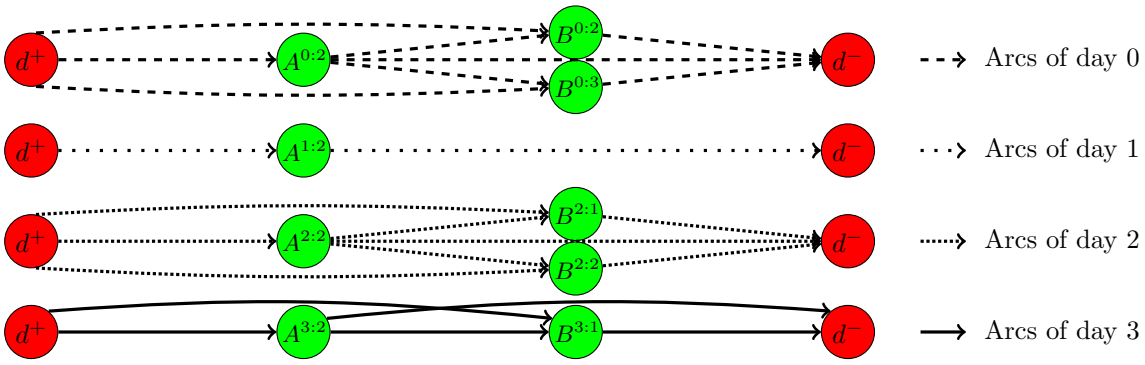

Figure 2: Example of schedule part networks with $S_A = \{\{0,2\},\{1,3\}\}$ and $S_B = \{\{0,2\},\{0,3\},\{0,2,3\}\}$

The cost and time for traveling along arc (i,j) are identical to the costs and time for traveling from location $o(i)$ to location $o(j)$. The dual price for the fleet constraint (1d) is subtracted from the costs of all arcs leaving the start depot whereas the dual prices for constraints (1c) are considered on the arcs leaving the associated schedule parts. The load collected at a vertex is the product of the customer's demand per day and the length of the associated schedule part. Thus, the arc weights for the reduced costs $\tilde{c}^p_{ij}$, the load d_{ij}, and the time t_{ij} in the graph G^p_{SP} are:

$$\tilde{c}^p_{ij} = \begin{cases} c_{o(i)o(j)} - \mu_p, & \text{if } o(i) = d^+ \\ c_{o(i)o(j)} - \rho^{pl(i)}_{o(i)}, & \text{otherwise} \end{cases}$$
$$d_{ij} = q_{o(j)} \cdot l(j)$$
$$t_{ij} = t_{o(i)o(j)}.$$

Task network. Another variant to model the pricing networks is based on the tasks that need to be fulfilled. We define the task network for day p as the graph $G^p_T = (V^p_T, A^p_T)$ with the vertex set V^p_T, which is a subset of the day-independent vertex set V_T, and the arc set A^p_T. The set of vertices V_T contains two depot vertices and one vertex for every customer $n \in N$ and day $p \in P$, which refers to the fulfillment of the associated task n^p. The day-dependent vertex set V^p_T contains the two depot vertices d^+ and d^- and all vertices referring to tasks that can be fulfilled during a route on day p, i.e., it contains a vertex for the task $n^{p'}$ whenever the customer offers a schedule that allows a visit on day p that includes the demand fulfillment of the day p'. Let again the location of a vertex $i \in V_T$ be given by $o(i)$ and let the associated day be determined by $p(i)$. Moreover, let $\Delta(p,p')$ indicate the number of days between the days p and p' including these two, i.e. we set

$$\Delta(p,p') \coloneqq (p'-p) \text{mod} |P| + 1.$$

An arc (i,j) between vertices of different locations exists in A^p_T, whenever a schedule part of the customer $o(i)$ starts on day p and fulfills all demands up to period $p(i)$, and a schedule part of the customer $o(j)$ starts on day p. More formally, an arc (i,j) with $o(i) \neq o(j)$ exists in A^p_T, if there exists a schedule part $o(i)^{p:l}$ with $(p+l-1) \bmod |P| = p(i)$ and $p(j) = p$. An arc (i,j) between vertices of the same customer exists in A^p_T, whenever day $p(i)$ is followed by day $p(j)$ and both days are covered by a schedule part of customer $o(i)$ starting on day p. Formally, an arc (i,j) with $o(i) = o(j)$ exists in A^p_T, if $p(i) + 1 \bmod |P| = p(j)$ and there exists a schedule part $o(i)^{p:l}$ with $\Delta(p,p(i)) \leq l$ and $\Delta(p,p(j)) \leq l$.

The four task networks for the example from above are illustrated in Figure 3. In contrast to the schedule part network, the task networks of the different days share the vertex set. However, every arc exists exclusively in the network of a certain day. Here, even for a customer with standard schedule set, the number of vertices in a daily network depends on the customer's visit frequency and can range from 1 to $|P|$. The main advantage of the task network is that a customer can only be entered through one single vertex in every network. Thus, when all customers offer all possible schedules, the number of arcs is in

$\mathcal{O}(|N|^2|P|)$ and thus smaller than in the schedule part network. Altogether, there is no clear preference between the two pricing network types as the network sizes depend very much on the offered schedule sets. Their performance for instances with varying schedule sets is evaluated in Section 6.2.

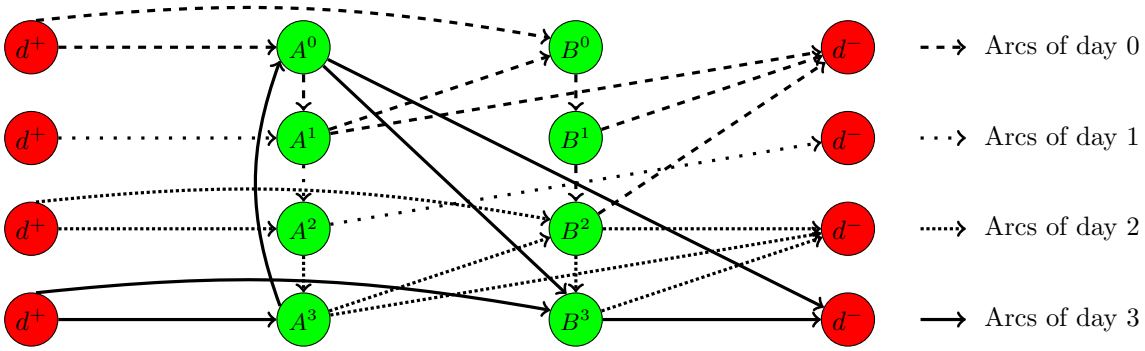

Figure 3: Example for task networks with $S_A = \{\{0,2\},\{1,3\}\}$ and $S_B = \{\{0,2\},\{0,3\},\{0,2,3\}\}$

The cost and time for traveling between two vertices belonging to different locations equal again the corresponding values for traveling between these locations. Accordingly, no cost and time consumption occurs on arcs between two vertices of the same location. All dual prices are again subtracted from the arcs leaving a location. Since every vertex represents the fulfillment of a single task, the demand per day is assigned to every vertex. Note that in this pricing network, the covered schedule part is determined only when a customer location is left because for overlapping schedules several schedule parts are possible when entering a customer location. Thus, the arc weights for the reduced costs $\tilde{c}^p_{ij}$, the load d_{ij}, and the time t_{ij} in the graph G^p_T are:

$$\tilde{c}^p_{ij} = \begin{cases} c_{o(i)o(j)} - \mu_p, & \text{if } o(i) = d^+ \\ c_{o(i)o(j)} - \rho^{p\,\Delta(p,p(i))}_{o(i)}, & \text{if } o(i) \neq d^+ \text{ and } o(i) \neq o(j) \\ 0, & \text{otherwise} \end{cases}$$

$$d_{ij} = q_{o(j)}$$

$$t_{ij} = \begin{cases} t_{o(i)o(j)}, & \text{if } o(i) \neq o(j) \\ 0, & \text{otherwise.} \end{cases}$$

4.1.2. Labeling algorithm

The dynamic-programming labeling algorithm for our PVRPTW pricing problems of the different days can operate on both pricing network types presented above to find feasible routes with negative reduced costs. Thus, let the graph $G^p = (V^p, A^p)$ for day p be either the schedule part network $G^p_{SP} = (V^p_{SP}, A^p_{SP})$ or the task network $G^p_T = (V^p_T, A^p_T)$. The feasibility of a route is given if both the capacity and the route duration limit of the vehicle are respected, if all customers are visited within their time window, and if the visiting day and the delivery amount fit to at least one offered schedule. While the schedule feasibility is ensured by the structure of the pricing networks, the other restrictions are guaranteed by the following labeling algorithm. To obtain a bidirectional labeling algorithm, which is the most successful version for ESPPRCs (Righini and Salani, 2006), we explain the forward labeling, the backward labeling, and finally the concatenation procedure subsequently.

A partial forward path r from the start depot d^+ to a vertex $i \in V$ is represented by a label $E_i = (E^{cost}_i, E^{load}_i, E^{time}_i, E^{dur}_i, E^{help}_i, (E^{cust_n}_i)_{n\in N})$, where the label components are:

E^{cost}_i: reduced cost of path r;
E^{load}_i: delivered demand on the path r;
E^{time}_i: earliest service start time at vertex i on the path r;

E_i^{dur}: minimum route duration of the path r;
E_i^{help}: negative of the latest possible departure time at the depot for the path r;
$E_i^{cust_n}$: number of times that customer $n \in N$ is visited along path r.

The resources E_i^{dur} and E_i^{help} are taken from (Tilk and Irnich, 2017) and are necessary for handling the route duration. In the initial label at vertex d^+, all components are set to 0 except $E_i^{time} = a_d$ and $E_i^{help} = -b_d$. The forward extension of a label E_i along an arc $(i,j) \in A^p$ is performed using the following resource extension functions (REFs):

$$
\begin{aligned}
E_j^{cost} &= E_i^{cost} + \tilde{c}_{ij}^p \\
E_j^{load} &= E_i^{load} + d_{ij} \\
E_j^{time} &= \max\{a_{o(j)}, E_i^{time} + t_{ij}\} \\
E_j^{dur} &= \max\{E_i^{dur} + t_{ij}, E_i^{help} + a_{o(j)}\} \\
E_j^{help} &= \max\{E_i^{dur} + t_{ij} - b_{o(j)}, E_i^{help}\} \\
E_j^{cust_n} &= \begin{cases} E_i^{cust_n} + 1, & o(j) \in N,\ o(j) = n, \text{ and } o(i) \neq o(j) \\ E_i^{cust_n}, & \text{otherwise.} \end{cases}
\end{aligned}
$$

A forward label E_j is feasible if and only if $E_j^{load} \leq Q$, $E_j^{time} \leq b_{o(j)}$, $E_j^{dur} \leq D$, and $E_j^{cust_n} \leq 1$ for all customers $n \in N$. Because of the non-decreasing REFs, a forward label E_j dominates another forward label E'_j belonging to the same vertex j, if $E_j \leq E'_j$ component-wise. Note that, when using the task network, it suffices to check for dominance between labels referring to the entrance vertex of each customer because the relations between the labels do not change while they are propagated at the same location.

A partial backward path r from a vertex $i \in V$ to the end depot d^- is represented by a label $\bar{E}_i = (\bar{E}_i^{cost}, \bar{E}_i^{load}, \bar{E}_i^{time}, \bar{E}_i^{dur}, \bar{E}_i^{help}, (\bar{E}_i^{cust_n})_{n \in N})$, where the three time-related resources have a slightly different meaning than in the forward labeling:

$\bar{E}_i^{time}$: latest service start time at vertex i;
$\bar{E}_i^{dur}$: difference between the maximal route duration D and the minimum route duration of the path r;
$\bar{E}_i^{help}$: difference between maximal route duration D and the earliest possible arrival time at the depot with route duration $\bar{E}_i^{dur}$;

The initial backward label at the end depot d^- is $\bar{E}_{d^-} = (0, 0, b_d, D, D - a_d, 0)$. The backward extension of a label $\bar{E}_j$ along an arc $(i,j) \in A^p$ is performed using the following REFs:

$$
\begin{aligned}
\bar{E}_i^{cost} &= \bar{E}_j^{cost} + \tilde{c}_{ij}^p \\
\bar{E}_i^{load} &= \bar{E}_j^{load} + d_{ij} \\
\bar{E}_i^{time} &= \min\{b_{o(i)}, \bar{E}_j^{time} - t_{ij}\} \\
\bar{E}_i^{dur} &= \min\{\bar{E}_j^{dur} - t_{ij}, \bar{E}_j^{help} - t_{ij} + b_{o(j)}\} \\
\bar{E}_i^{help} &= \min\{\bar{E}_j^{dur} - a_{o(j)}, \bar{E}_j^{help}\} \\
\bar{E}_i^{cust_n} &= \begin{cases} \bar{E}_j^{cust_n} + 1, & o(i) \in N,\ o(i) = n, \text{ and } o(i) \neq o(j) \\ \bar{E}_j^{cust_n}, & \text{otherwise.} \end{cases}
\end{aligned}
$$

A backward label $\bar{E}_i$ is feasible if and only if $\bar{E}_i^{load} \leq Q$, $\bar{E}_i^{time} \geq a_{o(i)}$, $\bar{E}_i^{dur} \leq D$, and $\bar{E}_i^{cust_n} \leq 1$ for all customers $n \in N$. A backward label $\bar{E}_i$ dominates another backward label $\bar{E}'_i$ belonging to the same vertex i, if the time-related resources of the former label are greater equal and all other resources are less equal than the corresponding resources of the second label, i.e., if $\bar{E}_i^{cost} \leq \bar{E}_i'^{cost}$, $\bar{E}_i^{load} \leq \bar{E}_i'^{load}$, $\bar{E}_i^{cust_n} \leq \bar{E}_i'^{cust_n}\ \forall n \in N$, $\bar{E}_i^{time} \geq \bar{E}_i'^{time}$, $\bar{E}_i^{dur} \geq \bar{E}_i'^{dur}$ and $\bar{E}_i^{help} \geq \bar{E}_i'^{help}$.

The concatenation of a forward partial path represented by the forward label E_j and a backward partial

path represented by the backward label $\bar{E}_j$ at the vertex j is feasible, if and only if

$$\begin{aligned} E_j^{load} + \bar{E}_j^{load} - q_{o(j)} &\leq Q \\ E_j^{time} &\leq \bar{E}_j^{time} \\ E_j^{dur} &\leq \bar{E}_j^{dur} \\ E_j^{help} &\leq \bar{E}_j^{help} \\ E_j^{cust_n} + \bar{E}_j^{cust_n} &\leq 1 \quad \forall\, n \in N \setminus \{o(j)\}. \end{aligned}$$

4.1.3. Further accelerations

To accelerate the solution of the pricing problem, we include several known techniques. First, we incorporate the *ng*-path relaxation to achieve more dominance while we need to accept some non-elementary routes (see Baldacci *et al.*, 2011b). Thereby, the size of the *ng*-neighborhoods is dynamically increased after the solution of each branch-and-bound-node, as we always search for cycles in the chosen routes and add the double visited vertex to the *ng*-neighborhoods of the vertices in between (Roberti and Mingozzi, 2014). Second, we accelerate the bidirectional labeling by a dynamic choice of the half-way point as described in (Tilk *et al.*, 2017). In this approach, the half-way point is not predetermined to the middle of a resource domain but is dynamically adjusted due to a heuristic criterion that aims to balance the forward and backward workload. Third, we use restricted networks as a heuristic pricing as long as this generates columns with negative reduced costs. Fourth, we use the special pricing problem order suggested by Pirkwieser and Raidl (2009b) to always complete the pricing problem of one day (with a certain network size) until it finds no more columns before changing to the next day. This corresponds to a special version of partial pricing (Gamache *et al.*, 1999).

4.2. Subset-row cuts

Valid inequalities are added to a linear program to cut off fractional solutions and thus strengthen the LP relaxation. For the PVRPTW, we add the non-robust subset-row (SR) inequalities introduced by Jepsen *et al.* (2008). These cuts are usually defined on a subset of customers and limit the number of routes serving a given number of these customers. However, a precondition is that every customer has to be served at most once, which is obviously not the case for the PVRPTW. We therefore need to adjust the SR cuts to our context and look at subsets of tasks or schedule parts, because both a single task and a single schedule part can be served at most once in the PVRP. The SR cut defined on a subset S of either tasks or schedule parts with $0 < k \leq |S|$ can be stated as

$$\sum_{p \in P} \sum_{r \in R^p} \left\lfloor \frac{\sum_{i \in S} a_{ri}^p}{k} \right\rfloor \lambda_r^p \leq \left\lfloor \frac{|S|}{k} \right\rfloor, \tag{3}$$

where the parameter a_{ri}^p indicates how often route r performed on day p contains the task/the schedule part i.

Because of pretests, we consider SR cuts defined for subset of tasks in the following. As common in the literature, we restrict ourselves to SR cuts with subsets of cardinality 3 and k equal to 2, because they can be separated through enumeration. However, the number of task subsets is $\mathcal{O}(|P|^3)$ times larger than the number of customer subsets. Therefore, we only separate subsets of tasks belonging to the same day, even though there can be violated SR cuts with subsets consisting of tasks of different days.

The necessary adaptations to respect these non-robust SR cuts in the labeling algorithm can be found in (Jepsen *et al.*, 2008). The only PVRP-specific point is that all task subsets need to be taken into account in all pricing networks and not just those with tasks belonging to the appropriate day.

4.3. Branching strategy

To achieve integrality of the decision variables, a branching strategy needs to be implemented. Here, a hierarchical four-level based strategy is used, which tries to fix decisions in decreasing order of influence on the solution space.

On the first level, we branch on the total number of routes whenever it is fractional. On the second level, we branch on the number of routes performed on one day. When an integer number of vehicles is used on all days, we branch on a visiting day of a certain customer on the third level. This is realized by forbidding the appropriate schedule columns. After no more fractional values on visiting days of individual customers can be detected, exactly one schedule is chosen for every customer. Thus, for every day remains the solution of a VRPTW with known customers and demands. Therefore, it suffices to finally branch on the decision to visit two customers successively on a certain day. This is realized by removing the appropriate arcs from the pricing networks. After these four branching levels there is either an integer solution or the problem is infeasible. A best-first search is executed to determine the next branch-and-bound node to process.

5. Symmetry

In this section, we assume that both demands and time windows of the customers do not vary from day to day, which is a common assumption in the PVRP literature. When the schedule sets of all customers are standard, the problem is symmetric in the way that the routes in every feasible solution may be shifted by any number of days without changing the objective function value or destroying the feasibility. In general, even non-standard instances may possess a kind of symmetry. For example, in a planning horizon of six days, the schedule set $S = \{\{0, 1\}, \{3, 4\}\}$ could be shifted by three days and would remain unchanged. Therefore, we give a general definition of symmetry in the PVRP context and show resulting application potential for standard and some types of non-standard instances.

We define the shift function $f_k : \mathcal{P}(P) \to \mathcal{P}(P)$ with $k \in P$ for a schedule $s \in S \subseteq \mathcal{P}(P)$ as:

$$f_k(s) = \{(p + k) \text{mod} |P| : p \in s\}.$$

A set of schedules S is called *k-shift-symmetric*, if and only if the mapping of all schedules $s \in S$ via $f_k(s)$ yields the original schedule set S again, i.e., if $S = \{f_k(s) : s \in S\}$. Accordingly, a problem instance is called k-shift-symmetric, if k is the minimal value for that all the schedule sets S_n of all customers $n \in N$ are shift-symmetric. If a problem instance is 1-shift-symmetric, we call it *completely symmetric*.

For the remainder of this section, we consider a k-shift-symmetric instance. For a given schedule $s \in S_n$, we define the set of equivalent schedules as $M_s = \{\bar{s} \in S_n : \exists m \in \{1, \dots, |P|/k\} : f_{mk}(s) = \bar{s}\}$. The number of elements in this set $m_s = |M_s|$ also represents the minimal number of k-shifts such that the schedule s is mapped to itself, i.e., $m_s = \text{argmin}\{m \in \{1, \dots, |P|/k\} : f_{mk}(s) = s\}$.

Two days $p, \bar{p} \in P$ are equivalent in a k-shift-symmetric instance, written $p \sim \bar{p}$, if there exists a multiplier $m \in \{0, \dots, |P|/k\}$, such that $|\bar{p} - p| = mk$. This means, that every route performed on day p could also be performed on day $\bar{p}$ while still fitting to offered schedule parts. The equivalence class $[p]$ of a day $p \in P$ is the set of all days $\bar{p} \in P$ that are equivalent to p, i.e., $[p] = \{\bar{p} \in P : p \sim \bar{p}\} = \{(p + mk) \text{mod} |P| : m \in \{0, \dots, |P|/k\}\}$. Let $U \subseteq P$ be a set of class representatives.

When several days are equivalent, the solution of a pricing problem for every single day creates unnecessary effort. Furthermore, symmetry causes multiple solutions with the same structure and objective function value at different nodes in the branch-and-bound tree. To avoid these drawbacks, this section introduces some techniques to deal with symmetric instances. In Section 5.1, we explain a valid aggregation of constraints to reduce the number of considered days and in Section 5.2, we describe two further techniques to exploit the symmetry.

5.1. Aggregation

Even in symmetric instances, the pricing problems of two equivalent days may generate different columns and the same route may have different reduced costs when assigned to these days. The reason is primal degeneracy leading to several optimal dual solutions. However, the equivalence of days can be used so that only one pricing problem per equivalence class needs to be considered. The idea is to aggregate all day-specific constraints in the linear relaxation of the PVRP model (1) over all days in the same equivalence class. The same aggregation approach was successfully applied by Rothenbächer *et al.* (2017) for the truck-and-trailer routing problem with a two-day planning horizon. In the following, we first present the general

aggregation for a k-shift-symmetric instance and then a further simplification for completely symmetric instances with only regular schedules.

5.1.1. General Aggregation

Aggregating (summing up) all day-equivalent constraints in the linear relaxation of the PVRP model (1) leads to the following *aggregated model*:

$$\min \quad \sum_{p\in U}\sum_{r\in R^p} c_r\lambda_r^p \tag{4a}$$

$$\text{s.t.} \quad \sum_{s\in S_n} z_n^s = 1 \qquad \forall\, n\in N \tag{4b}$$

$$\sum_{r\in R^p} a_{rn}^{pl}\lambda_r^p = \sum_{s\in S_n}\sum_{\bar p\in[p]} b_s^{\bar p l} z_n^s \qquad \forall\, n\in N, p\in U, l\in L : S_n^{pl}\neq\varnothing \tag{4c}$$

$$\sum_{r\in R^p}\lambda_r^p \le \sum_{\bar p\in[p]} m = |[p]|m \qquad \forall\, p\in U \tag{4d}$$

$$0\le z_n^s\le 1 \qquad \forall\, n\in N, s\in S_n \tag{4e}$$

$$0\le \lambda_r^p\le |[p]| \qquad \forall\, p\in U, r\in R^p \tag{4f}$$

Thereby, it suffices to include the route variables for the class representatives, because two identical routes $r\in R^p$ and $\bar r\in R^{\bar p}$ performed on equivalent days $p\sim\bar p$ have the same coefficients, i.e., $a_{rn}^{pl}=a_{\bar r n}^{\bar p l}$ for all $n\in N, l\in L$ and $c_r=c_{\bar r}$. The advantages of the aggregated model are a smaller problem size regarding the number of constraints and a reduced number of pricing problems that have to be solved.

The SR cuts can also be used in an aggregated version. Recall that we restrict ourselves to SR cuts based on subsets of tasks referring to the same day. The addition of a single SR cut for a subset S^p of tasks belonging to one certain day p would destroy the symmetry of the problem. Thus, the corresponding SR cuts with subsets $S^{\bar p}$ of tasks referring to the same customers, but to the other days $\bar p\in[p]\setminus p$ in the equivalence class, have to be generated too. Then, the associated cuts of the same equivalence class have to be aggregated as above to

$$\sum_{p\in U}\sum_{r\in R^p}\sum_{\bar p\in[p]}\left\lfloor\frac{\sum_{i\in S^{\bar p}} a_{ri}^p}{k}\right\rfloor\lambda_r^p \le |[p]|\left\lfloor\frac{|S^p|}{k}\right\rfloor.$$

The original model and the aggregated model are equivalent for k-shift-symmetric instances, because from a solution for one of them, a solution for the other with the same objective function value can be generated. The direction from the original model to the aggregated model is clear, because aggregation of constraints is always a relaxation. The other direction can be shown by stating a feasible transformation. For any solution $\bar\lambda_r^p$ with $p\in U, r\in R^p$ and $\bar z_n^s$ with $n\in N, s\in S_n$ of (4), a feasible solution of the linear relaxation of (1) with the same objective function value is given by setting

$$\hat\lambda_r^{\bar p}=\frac{1}{|[p]|}\bar\lambda_r^p \;\; \forall p\in U, r\in R^p, \bar p\in[p] \qquad \text{and} \qquad \hat z_n^s=\frac{1}{m_s}\sum_{\bar s\in M_s}\bar z_n^s \;\; \forall n\in N, s\in S_n.$$

The proof of the feasibility and the unchanged objective function value can be performed by recalculation.

As soon as the symmetry of the problem is broken, for example due to day-specific branching decisions, the constraints have to be disaggregated. Thus, formulation (1) has to be used again. To obtain the same objective function value, route variables referring to the days that were ignored, have to be added. More precisely, the route r of a variable λ_r^p with positive value in the solution of the aggregated model, has to be shifted to all other days in the equivalence class of the day p. The resulting route variables $\lambda_r^{\bar p}$ with $\bar p\in[p]\setminus p$ have to be added to the formulation.

Example. Consider an instance with a planning horizon of four days and two customers A and B with schedule sets $S_A = \{s_A^1 = \{0,1\}, s_A^2 = \{1,2\}, s_A^3 = \{2,3\}, s_A^4 = \{3,0\}\}$ and $S_B = \{s_B^1 = \{0,2\}\}$. Although the schedules of customer A are irregular, S_A is 1-shift-symmetric, while S_B is only 2-shift-symmetric even though customer B offers a regular schedule. Consequently the whole instance is only 2-shift-symmetric. Thus, the two days 0 and 1 constitute a set of representatives and it suffices to consider these two days in the algorithm until the symmetry is broken. An optimal solution of the aggregated model (4) is given by the following variable assignments:

$$\begin{aligned} &\bar{\lambda}^0_{r_1} = 1 \text{ with } r_1 = (d^+, A^{l=1}, B^{l=2}, d^-), && \bar{z}^1_A = 1, \quad \bar{z}^2_A = \bar{z}^3_A = \bar{z}^4_A = 0, \\ &\bar{\lambda}^0_{r_2} = 1 \text{ with } r_2 = (d^+, B^{l=2}, d^-), && \bar{z}^1_B = 1. \\ &\bar{\lambda}^1_{r_3} = 1 \text{ with } r_3 = (d^+, A^{l=3}, d^-), \end{aligned}$$

Note that setting $\bar{z}^1_A = 1$ is not the only possibility concerning the schedule variables of customer A in the aggregated model. All schedule variables with schedules in the same set M_s are treated equally, such that, because of $M_{s^1_A} = \{s^1_A, s^3_A\}$, every variable assignment fulfilling $\bar{z}^1_A + \bar{z}^3_A = 1, \bar{z}^2_A = \bar{z}^4_A = 0$ would be feasible for the given routes. An equivalent solution for the original, disaggregated model (1) is given by considering all four days and redistributing the variable values as described above:

$$\begin{aligned} &\hat{\lambda}^0_{r_1} = \hat{\lambda}^2_{r_1} = 0.5, && \hat{z}^1_A = \hat{z}^3_A = 0.5, \quad \hat{z}^2_A = \hat{z}^4_A = 0, \\ &\hat{\lambda}^0_{r_2} = \hat{\lambda}^2_{r_2} = 0.5, && \hat{z}^1_B = 1. \\ &\hat{\lambda}^1_{r_3} = \hat{\lambda}^3_{r_3} = 0.5, \end{aligned}$$

5.1.2. Aggregation for completely symmetric instances with only regular schedules

In the special case of completely symmetric instances with only regular schedules, the linking constraints (4c) can be further aggregated. Because of the complete symmetry, there is only one equivalence class, i.e., $U = \{0\}$. By multiplying both sides of constraints (4c) with the schedule part length and aggregating over the lengths, the following constraints result:

$$\sum_{r \in R^0} \sum_{l \in L} l\, a^{0l}_{rn} \lambda^0_r = \sum_{s \in S_n} \underbrace{\sum_{l \in L} l \sum_{\bar{p} \in [0]} b^{\bar{p}l}_s}_{\substack{\text{regularity of schedules} \\ = |P|}} \quad z^s_n \overset{(1b)}{=} |P| \quad \forall\, n \in N \tag{5}$$

because, for a fixed schedule, the unique schedule part length times the number of schedule parts in this schedule gives the total number of days. In consequence, route variables and schedule variables become independent. Since the schedule variables have no influence on the objective function, they can be ignored in the aggregated model for completely symmetric instances with only regular schedules. Thus, constraints (4b) and (4e) can be discarded in this case. All in all, the resulting *aggregated model for completely symmetric instances with only regular schedules* is given by:

$$\begin{aligned} \min \quad & \sum_{r \in R^0} c_r \lambda^0_r \\ \text{s.t.} \quad & (5), (4\text{d}) \text{ and } (4\text{f}) \end{aligned} \tag{6}$$

Again, the feasibility of this expanded aggregation can be proven by showing the necessary transformation to obtain a solution of the original model (1). For any solution $\bar{\lambda}^0_r$ with $r \in R^0$ of (6) of a completely symmetric instance with only regular schedules, a feasible solution of the linear relaxation of (1) with the same objective function value is given by setting

$$\hat{\lambda}^{\bar{p}}_r = \frac{1}{|P|} \bar{\lambda}^0_r \;\; \forall r \in R^0, \bar{p} \in [0] \qquad \text{and} \qquad \hat{z}^s_n = \frac{1}{|P|} \sum_{r \in R^0} a^{0l_s}_{rn} \bar{\lambda}^0_r \;\; \forall n \in N, s \in S_n,$$

where l_s is the unique schedule part length of all schedule parts of a regular schedule. Again, the proof can be done by recalculation. When the symmetry is broken, all schedule variables have to be included in the model and certain route variables have to be added as described in Section 5.1.1.

Example. Consider an instance with a planning horizon of four days and one customer A with the schedule set $S_A = \{s_A^1 = \{0,1,2,3\}, s_A^2 = \{0,2\}, s_A^3 = \{1,3\}\}$. Despite different visit frequencies of the offered schedules, all schedules are regular and the instance is completely symmetric. An optimal solution of the aggregated model for completely symmetric instances with only regular schedules (6) is given by the following variable assignments:

$$\bar{\lambda}^0_{r_1} = 2 \text{ with } r_1 = (d^+, A^{l=1}, d^-),$$
$$\bar{\lambda}^0_{r_2} = 1 \text{ with } r_2 = (d^+, A^{l=2}, d^-).$$

Thereby, route r_1 fulfills one task of customer A and is performed two times and route r_2 fulfills the remaining two tasks. An equivalent solution for the original model (1) is given by considering all four days, redistributing the route variable values equally to all days, and including the schedule columns as described above:

$$\hat{\lambda}^0_{r_1} = \hat{\lambda}^1_{r_1} = \hat{\lambda}^2_{r_1} = \hat{\lambda}^3_{r_1} = 0.5, \qquad \hat{z}^1_A = 0.5, \quad \hat{z}^2_A = \hat{z}^3_A = 0.25.$$
$$\hat{\lambda}^0_{r_2} = \hat{\lambda}^1_{r_2} = \hat{\lambda}^2_{r_2} = \hat{\lambda}^3_{r_2} = 0.25,$$

5.2. Further techniques handling symmetry

For k-shift-symmetric instances, the model in its original formulation (1) offers more degrees of freedom than necessary. As mentioned, every solution can be shifted by any number of days to which the instance is symmetric. Thus, an optimal solution remains reachable if for one single customer the decision to visit him on one certain day is fixed. Therefore, we choose a customer with the highest number of possible schedules and among them one with the highest demand, and fix him to be visited on a certain day that is allowed by at least one of his offered schedules. This breaks the symmetry of the problem and can significantly reduce the size of the branch-and-bound tree.

Furthermore, as long as no day-specific branching is applied (i.e., including the first branching level), all generated routes that do not contain the fixed customer are valid for all days in the same equivalence class. Hence, we shift every route generated in the pricing problem of a specific day to all other days in the same equivalence class and compute the reduced costs of the corresponding column. If its reduced costs are also negative, we directly add it to the RMP. This can reduce the number of necessary pricing problem iterations.

6. Computational Study

The branch-and-price-and-cut algorithm was implemented in C++ and uses CPLEX 12.6.0. All computations were performed in single thread mode on a PC with an Intel Core i7-4790K CPU clocked at 3.60 GHz, with 8 GB RAM, running Windows 7 Enterprise. Apart from the single-thread mode, default settings were used for the CPLEX solver. Preliminary tests revealed that the best results can be obtained with a size of the *ng*-neighborhoods of 12 which may be increased dynamically up to 18, and reduced pricing networks with 5, 10, 20, and 40 neighbors per vertex. The overall number of SR cuts was limited to 100, whereby at most four SR cuts per day were allowed to be added per pricing iteration.

We start with a description of the used instances in Section 6.1. In Section 6.2, the success of the proposed PVRP-specific settings is evaluated and the two introduced pricing network types are compared. Section 6.3 shows the effects of different schedule structures and presents the results for PVRPTW benchmark instances.

6.1. Instances

The basis for the computational results are the 20 PVRPTW benchmark instances published by Cordeau *et al.* (2001), called *PR* in the following. Herein, the number of customers ranges from 48 to 288 customers and planning horizons of four and six days are considered. The time windows in the first 10 instances are narrow and larger in the remaining 10 instances. All customers offer standard schedule sets, such that the

instances are completely symmetric. For the instances with four days, there exist three groups of customers requiring the different possible visit frequencies 1, 2, and 4. For the instances with six days, there exist four groups of customers requiring the different possible visit frequencies 1, 2, 3, and 6. Up to now, these instances were not tackled with exact approaches. Concluding from the successes on VRPTW instances with 100 customers and more, most of the instances in PR are too large to be solved to optimality in reasonable times (see Desaulniers *et al.*, 2014).

Thus, we created 20 smaller instances by selecting a subset of 36 customers for the instances with a planning horizon of four days and a subset of 24 customers for the instances with a planning horizon of six days. Thereby, we took the same number of customers of each of the aforementioned groups with different visit frequencies to keep the ratios between these groups unchanged. Moreover, we limited the number of vehicles in these smaller instances to three. We refer to these smaller instances as PR_s.

To test the effects of different schedule structures, we created two more variants of the instance set PR_s by modifying the offered schedule sets. In the instance set PR_s^+, we added for each customer all regular schedules with a higher visit frequency than originally required. The intention is the assumption that customers need a certain minimal visit frequency but are willing to be visited more often. Indeed, even a bonus for more frequent visits seems realistic and was presumed for example by Francis *et al.* (2006). In the instance set PR_s^{++}, we extended PR_s by adding for each customer all irregular schedules with the same number of visits as originally required. Thereby, we assume that customers do not require exactly the same intervals between each two visits, but need only a certain number of visits during the planning horizon. Obviously, both modified instance sets are harder because of the increased number of allowed schedules. On the other hand, the solution values can only be decreased since all solutions that are feasible for the original instances in PR_s stay feasible for the appropriate modified instances. Furthermore, to the best of our knowledge, these two sets could not be tackled by exact approaches from the literature because of the overlapping (and in the second case irregular) schedules.

The most important properties of the regarded instance sets are summarized in Table 2.

Table 2: Properties of the regarded instance sets

Instance set	#Days	#Customers	Avg. $\lvert S_n \rvert$	Irregular schedules	Overlapping schedules	Varying visit frequency
PR	4	48–288	2.75			
	6	72–288	3.00			
PR_s	4	36	2.75			
	6	24	3.00			
PR_s^+	4	36	4.50		x	x
	6	24	5.50		x	x
PR_s^{++}	4	36	3.75	x	x	
	6	24	10.50	x	x	

6.2. PVRP specific settings evaluation

In this paper, several options to tackle the symmetry and to model the pricing networks were discussed that are specific to PVRPs. The effectiveness of these settings is measured through extensive tests that are presented in the following. A run time limit of one hour was used for all tests in this section.

Table 3 shows the results of all combinations of the following settings: (i) aggregation of constraints (5.1), (ii) symmetry breaking through fixation of one customer (5.2), (iii) addition of columns shifted to other days (5.2), and (iv) the pricing problem order that continues with one day until no more columns for this day are found (4.1.3). Each entry shows the average run time and the number of solved instances from 80 test runs, consisting of the 40 instances of the sets PR_s and PR_s^+, tested with both the task network and the schedule part network. Note that we did not include the instance set PR_s^{++} in this test, because we wanted

to apply a consistent form of aggregation for the comparison (and the aggregation (6) is not applicable for PR_s^{++} because of the irregular schedules).

Table 3: Comparison of run times for different solving strategies

		Symmetry breaking					
		true	true	false	false		
Aggregation	true	976.4 (68)	1027.7 (65)	1370.1 (60)	1417.6 (59)	true	Add shifted columns
	true	982.7 (67)	1045.7 (66)	1371.4 (60)	1446.4 (58)	false	
	false	1035.7 (67)	1116.6 (66)	1652.5 (55)	1668.1 (55)	true	
	false	1051.1 (67)	1058.9 (66)	1704.5 (51)	1605.1 (53)	false	
		true	false	true	false		
		Special pricing problem order					

The values show the average run times over forty test instances solved with both the task network and the schedule part network with the PVRP specific settings indicated at the borders. Instances that were not solved to optimality in the time limit of 1 hour were counted with the time limit. The number in brackets gives the number of instances solved to optimality.

The best results regarding both the run times and the number of solved instances were obtained with the setting using all the options listed above. Switching off all these options leads to an increase of the average run time by more than 60 %. Obviously, the symmetry breaking has the biggest influence, but also the use of the aggregated model produces an improvement independent from the other settings. The addition of shifted columns is especially beneficial when the special pricing problem order is used. All in all, we concluded to use all the options (i) to (iv) in all following tests.

The next test evaluates the performance of the two different pricing networks introduced in this paper. All small instance sets were run with the schedule part network and the task network. The results of this comparison can be found in Table 4. Hereby, the average run times were calculated only over the instances solved with both network types.

Table 4: Task network versus schedule part network on different instance sets

		SP network		Task network	
Instance set	TW	#Opt	Avg. time[s]	#Opt	Avg. time[s]
PR_s	small	10 / 10	25.6	10/ 10	26.0
	large	9 / 10	1059.5*	9/ 10	732.4
PR_s^+	small	10 / 10	377.7	10/ 10	292.6
	large	4 / 10	2057.2	6/ 10	1505.1
PR_s^{++}	small	6 / 10	821.3	6/ 10	382.7
	large	4 / 10	302.7	4/ 10	121.9

*Ignoring one outlier instance: 776.9s
(this instance needed nearly 8 times as many B&B nodes with the SP network as with the task network)

Overall, the algorithm performed better using the task network in the pricing problem. When one extreme outlier of the tests with the schedule part network is ignored (which is due to an unfavorable branching process), the run times were similar for the small instance set with the standard schedule sets. However, with an increasing number of offered schedules (in instance sets PR_s^+ and PR_s^{++}), the difference between the applications of the two pricing networks becomes larger. In the instance set with the highest

average number of offered schedules per customer, PR_s^{++}, the algorithm with the task network is more than twice as fast as the algorithm with the schedule part network. Therefore, we decided to use the task network for all further computations.

6.3. Results

This section presents the computational results for the small instance sets and the original instances. Table 5 shows the aggregated results for the small instances. The columns state the name of the instance set, the number of instances solved to optimality during one hour, the average run time (unsolved instances are counted with the time limit), and the average optimality gap over all instances. For the instance sets $PR_s^+$and PR_s^{++}, the table gives additionally the information about the improvement compared to the basis instance set PR_s. We present the number of instances with improved objective value and for these instances the average percental improvement. In cases where instances were not solved to optimality, we compared the lower bound of an instance in PR_s with the best found upper bound of the corresponding instance in $PR_s^+$and PR_s^{++}, respectively.

Table 5: Comparison of the results for different schedule structures

instance set	#Opt	Time[s]	Gap[%]	#Improved solutions	Avg. improvement[%]
PR_s	19 / 20	522.6	0.03		
PR_s^+	16 / 20	1317.8	0.52	4	1.2
PR_s^{++}	10 / 20	1939.2	2.23	18	1.4

As expected, increasing flexibility of the customers' schedule sets leads to higher run times and less instances solved to optimality. On the other hand, several instances produced better solution values when more flexible schedule sets were allowed. Even in the instance set PR_s^+, where only schedules with higher visit frequencies were added, four improved solutions could be detected. In the instance set PR_s^{++}, where the visit frequency stayed unchanged, nearly all instances could find better solutions. This is especially surprising because our algorithm does not focus on the search for good upper bounds and half of the instances were not solved to optimality. This could also be a reason for the rather small percental improvements. Detailed results for the individual instances can be found in Table 7 in the Appendix.

In Table 6, we present new lower bounds and two optimal solutions for the benchmark set *PR*. Beside the instance characteristics, name, number of customers, number of days, number of vehicles, and the size of the time windows, we list the root lower bounds computed by Pirkwieser and Raidl (2009b) (if available), the lower bounds computed with our algorithm after two hours of runtime, the best known upper bounds that were published by Vidal *et al.* (2013), and the optimality gap between these upper and our lower bounds.

Considering the large numbers of customers in these instances, the results are quite good. For seven instances, the first reasonable lower bounds could be computed. The optimality of two known solutions could be proven for the first time. The small gaps together with the two optimality proofs indicate that both the heuristic of Vidal *et al.* (2013) and our lower bounds are satisfactory. However, it becomes apparent that larger time windows complicate the solution a lot because the number of instances whose root node could be solved decreased to less than the half by enlarging the time windows.

7. Conclusions

In this paper, the classical PVRPTW and its extension to fully flexible schedule sets was studied. Therefor, we presented a slightly modified master problem and a new exact branch-and-price-and-cut algorithm. For the pricing problem, two new underlying networks were described on which a bidirectional labeling operates to find new columns. Moreover, we presented efficient methods to deal with symmetric instances using constraint aggregation and a symmetry breaking variable fixing.

Table 6: Results for PVRPTW benchmark instances *PR*

Instance	#Cust.	#Days	#Veh.	TW	LB[PR09]	Our LB	UB[V13]	Gap[%]
pr01	48	4	3	small	2882.01	2909.02	2909.02	0
pr02	96	4	6	small	4993.48	5010.85	5026.57	0.3
pr03	144	4	9	small	6841.44	6853.10	7023.90	2.5
pr04	192	4	12	small		7536.01	7755.77	2.9
pr05	240	4	15	small		7991.78	8311.17	4.0
pr06	288	4	18	small			10473.24	
pr07	72	6	5	small	6641.39	6724.24	6783.23	0.9
pr08	144	6	10	small	9153.79	9159.92	9574.80	4.5
pr09	216	6	15	small		12439.90	13201.06	6.1
pr10	288	6	20	small		15893.90	16920.96	6.5
pr11	48	4	3	large	2258.85	2277.44	2277.44	0
pr12	96	4	6	large		4034.77	4121.50	2.1
pr13	144	4	9	large		5335.27	5489.33	2.9
pr14	192	4	12	large			6347.77	
pr15	240	4	15	large			6777.54	
pr16	288	4	18	large			8582.72	
pr17	72	6	4	large		5323.57	5481.61	3.0
pr18	144	6	8	large			7599.01	
pr19	216	6	12	large			10532.51	
pr20	288	6	16	large			13406.89	

PR09 = Pirkwieser and Raidl (2009b), V13 = Vidal *et al.* (2013)

An extensive computational study was performed with PVRPTW benchmark instances and smaller instances with varying schedule structures. The effectiveness of the proposed PVRP-specific methods could be confirmed and benefits of the task-based network for increasing schedule sets per customer could be shown. While raising flexibility through more offered schedules per customer increases the difficulty and thus the solution times, we could show improvements in terms of costs in many instances. For the large benchmark instances, several lower bounds that are close to known upper bounds could be found and the optimality of two solutions could be proven.

However, the results do not show the full potential of the presented algorithm. The algorithm is able to deal with more general schedule structures than the ones tested here for the sake of comparability. With our algorithm, more realistic problems with customers that have more specific requirements than just regular visits and a certain visit frequency can be tackled for the first time. Since these problems had to be solved heuristically so far, the usage of an exact algorithm may lead to notable savings.

Future research could address the branching and, for symmetric instances, the transition from the aggregated model to the original model. During preliminary tests, we observed that slightly different tie breakers in the branching decisions can have a huge influence on the run times. Therefore, a detailed study of branching decisions and the usage of strong branching could lead to significant improvements. Concerning the transition from the aggregated to the original model, the current approach usually produces a very fractional solution of the original model, even when the solution of the aggregated model was integer. A good heuristic could help to reduce the branching effort of finding an integer solution after the transition.

References

Archetti, C., Jabali, O., and Speranza, M. G. (2015). Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, **61**, 122–134.

Archetti, C., Fernández, E., and Huerta-Muñoz, D. L. (2017). The flexible periodic vehicle routing problem. *Computers & Operations Research*, **85**, 58–70.

Athanasopoulos, T. and Minis, I. (2013). Efficient techniques for the multi-period vehicle routing problem with time windows within a branch and price framework. *Annals of Operations Research*, **206**(1), 1–22.

Baldacci, R., Bartolini, E., Mingozzi, A., and Valletta, A. (2011a). An exact algorithm for the period routing problem. *Operations Research*, **59**(1), 228–241.

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

Beltrami, E. J. and Bodin, L. D. (1974). Networks and vehicle routing for municipal waste collection. *Networks*, **4**(1), 65–94.

Bertazzi, L., Savelsbergh, M., and Speranza, M. G. (2008). Inventory routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 49–72. Springer US, Boston, MA.

Bostel, N., Dejax, P., Guez, P., and Tricoire, F. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 503–525. Springer US, Boston, MA.

Cacchiani, V., Hemmelmayr, V., and Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, **163**, 53–64. Matheuristics 2010.

Campbell, A. M. and Wilson, J. H. (2014). Forty years of periodic vehicle routing. *Networks*, **63**(1), 2–15.

Christofides, N. and Beasley, J. E. (1984). The period routing problem. *Networks*, **14**(2), 237–256.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, **48**(1), 1–19.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **52**(8), 928–936.

Cordeau, J. F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *The Journal of the Operational Research Society*, **55**(5), 542–546.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). The vehicle routing problem with time windows. In D. Vigo and P. Toth, editors, *Vehicle Routing*, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Francis, P., Smilowitz, K., and Tzur, M. (2006). The period vehicle routing problem with service choice. *Transportation Science*, **40**(4), 439–454.

Francis, P., Smilowitz, K., and Tzur, M. (2007). Flexibility and complexity in periodic distribution problems. *Naval Research Logistics (NRL)*, **54**(2), 136–150.

Francis, P. M., Smilowitz, K. R., and Tzur, M. (2008). The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 73–102. Springer US, Boston, MA.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large scale aircrew rostering problems. *Operations Research*, **47**, 247–262.

Gaudioso, M. and Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, **26**(2), 86–92.

Hemmelmayr, V. C., Doerner, K. F., and Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, **195**(3), 791–802.

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

Irnich, S., Schneider, M., and Vigo, D. (2014). Four variants of the vehicle routing problem. In P. Toth and D. Vigo, editors, *Vehicle Routing*, chapter 9, pages 241–271. Society for Industrial and Applied Mathematics, Philadelphia, PA.

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

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Pirkwieser, S. and Raidl, G. R. (2009a). Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ILP techniques. In S. Voss and M. Caserta, editors, *Proceedings of the 8th Metaheuristic International Conference*.

Pirkwieser, S. and Raidl, G. R. (2009b). A column generation approach for the periodic vehicle routing problem with time windows. In M. S. et al, editor, *Proceedings of the International Network Optimization Conferencee*.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

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

Rothenbächer, A.-K., Drexl, M., and Irnich, S. (2017). Branch-and-price-and-cut for the truck-and-trailer routing problem with time windows. *Transportation Science*, page forthcoming.

Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, **9**(1), 1–17.

Tilk, C. and Irnich, S. (2017). Dynamic programming for the minimum tour duration problem. *Transportation Science*, **51**(2), 549–565.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidi-

rectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, **40**(1), 475–489.

Wen, M., Cordeau, J.-F., Laporte, G., and Larsen, J. (2010). The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, **37**(9), 1615–1623.

Appendix

Table 7 lists the lower and upper bounds as well as the solution times for the small instances PR_s, PR_s^+, and PR_s^{++} for a run time limit of one hour. Upper bounds for instances in PR_s^+ and PR_s^{++} that were smaller than the appropriate solution in PR_s are indicated in bold.

Table 7: Detailed results for different schedule structures

	PR_s			PR_s^+			PR_s^{++}		
Instance	LB	UB	Time[s]	LB	UB	Time[s]	LB	UB	Time[s]
$pr01_s$	2510.27	2510.27	3.9	2510.27	2510.27	29.0	2506.60	**2509.06**	TL
$pr02_s$	3324.68	3324.68	43.8	3324.68	3324.68	573.5	3306.73	**3306.73**	241.3
$pr03_s$	2802.02	2802.02	6.3	2802.02	2802.02	35.5	2797.23	**2797.23**	313.6
$pr04_s$	3012.19	3012.19	25.0	3012.19	3012.19	270,0	2996.28	**2996.28**	793.3
$pr05_s$	2387.90	2387.90	35.6	2387.90	2387.90	559.7	2353.54	**2353.54**	9.6
$pr06_s$	2891.15	2891.15	3.9	2891.15	2891.15	23.8	2881.00	**2881.00**	936.6
$pr07_s$	4268.41	4268.41	0.6	4251.06	**4251.06**	1.8	4239.22	**4239.22**	1.8
$pr08_s$	3065.62	3065.62	32.3	3056.07	**3056.07**	275.6	3001.15	**3007.07**	TL
$pr09_s$	3227.32	3227.32	85.7	3227.32	3227.32	956.4	3132.44	**3134.56**	TL
$pr10_s$	3775.87	3775.87	22.9	3775.87	3775.87	200.6	3665.55	**3672.33**	TL
$pr11_s$	2042.37	2042.37	38.9	2042.37	2042.37	642.9	2034.53	**2034.53**	269.5
$pr12_s$	2716.89	2733.35	TL	2705.63	2766.34	TL	2703.27	3492.08	TL
$pr13_s$	2419.54	2419.54	100.0	2419.54	2419.54	1730.6	2400.41	**2400.41**	54.5
$pr14_s$	2404.58	2404.58	110.2	2404.58	2404.58	2450.8	2394.43	**2396.22**	TL
$pr15_s$	1812.11	1812.11	18.5	1812.11	1812.11	565.3	1803.34	**1808.62**	TL
$pr16_s$	2474.41	2474.41	3534.0	2466.35	2483.04	TL	2454.40	**2458.48**	TL
$pr17_s$	3774.49	3774.49	37.2	3766.64	**3766.64**	698.9	3724.02	**3724.02**	79.5
$pr18_s$	2707.06	2707.06	562.7	2646.14	2713.36	TL	2634.74	3011.30	TL
$pr19_s$	2955.68	2955.68	840.7	2842.27	**2842.27**	2942.2	2782.91	**2784.24**	TL
$pr20_s$	3198.93	3198.93	1349.8	3081.18	3233.57	TL	3069.71	**3069.71**	84.0