Les Cahiers du GERAD ISSN: 0711–2440

Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models

G. Desaulniers,
T. Gschwind, S. Irnich

G–2019–48

July 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models

Guy Desaulniers [a,b]
Timo Gschwind [c]
Stefan Irnich [c]

[a] *GERAD, Montréal (Québec), Canada, H3T 2A7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, 55128 Mainz, Germany*

guy.desaulniers@gerad.ca
gschwind@uni-mainz.de
irnich@uni-mainz.de

July 2019
Les Cahiers du GERAD
G–2019–48

Abstract: Variable fixing by reduced costs is a popular technique for accelerating the solution process of mixed-integer linear programs. For vehicle routing problems solved by branch-price-and-cut algorithms, it is possible to fix to 0 the variables associated with all routes containing at least one arc from a subset of arcs determined according to the dual solution of a linear relaxation. This is equivalent to removing these arcs from the network used to generate the routes. In this paper, we extend this technique to routes containing sequences of two arcs. Such sequences or their arcs cannot be removed directly from the network because routes traversing only one arc of a sequence might still be allowed. For some of the most common vehicle routing problems, we show how this issue can be overcome by modifying the route generation labeling algorithm in order to remove indirectly these sequences from the network. The proposed acceleration strategy is tested on benchmark instances of the vehicle routing problem with time windows (VRPTW) and four variants of the electric VRPTW. The computational results show that it yields a significant speedup, especially for the most difficult instances.

Keywords: Integer programming, branch-price-and-cut, variable fixing, vehicle routing, labeling algorithm

1 Introduction

As highlighted by the recent survey of Costa et al. (2019), *branch-price-and-cut* (BPC) is the leading exact methodology for solving many *vehicle-routing problems* (VRPs) including the capacitated VRP (CVRP, Pecin et al. 2017b) and the *VRP with time windows* (VRPTW, Pecin et al. 2017a). A BPC algorithm is a branch-and-bound algorithm in which the lower bounds are computed by column generation and cuts are added dynamically to strengthen the linear relaxations. Column generation is iterative and solves, at each iteration, a *restricted master problem* (RMP) and one or several pricing problems. For most VRPs, the pricing problem is an elementary *shortest path problem with resource constraints* (SPPRC) which can be solved by a labeling algorithm (see Irnich and Desaulniers 2005). State-of-the-art BPC algorithms are complex and include various acceleration techniques such as bidirectional labeling, *ng*-route relaxation, limited-memory subset-row inequalities, variable fixing by reduced costs, and route enumeration. In this paper, we focus on variable fixing.

The procedure proposed by Irnich et al. (2010) allows to fix to 0 the variables associated with all routes containing at least one arc from a subset of arcs determined according to the dual solution of a linear relaxation. This is equivalent to removing the arcs in this subset from the network used to generate routes. As suggested by Pecin et al. (2017b), this fixing criterion can be refined to consider also the state of a resource (e.g., the load) before traversing an arc. With these techniques, both papers report fixing to 0 the flow on more than 90 % of the arcs for certain instances, yielding a significant speedup of the computation time.

In this paper, we extend the procedure of Irnich et al. (2010) by considering sequences of two arcs, i.e., under certain conditions, we can fix to 0 the variables associated with routes traversing a sequence of two arcs. This strategy is less straightforward than variable fixing for single arcs because it requires modifying the labeling algorithm used for pricing.

We test the new two-arc sequence fixing approach and compare it against single-arc fixing in comprehensive computational experiments on the well-known Solomon benchmark for the VRPTW. Moreover, we consider four variants of the *electric VRPTW* (EVRPTW, Desaulniers et al. 2016) in which a fleet of electric commercial vehicles is employed. These vehicles offer a limited driving range but have the possibility of visiting recharging stations during the course of a route. The variants differ in the recharging policy: On the one hand, either (S) at most a *single* recharge per route is allowed, or (M) *multiple* recharges per route are allowed. On the other hand, (F) batteries are always *fully* recharged on visit of a recharging station or (P) *partial* battery recharges are possible. At the end, this gives four policies and associated variants named EVRPTW-SF, EVRPTW-SP, EVRPTW-MF, and EVRPTW-MP. The experiments we conduct cover all four variants. Our computational results show, among others, that using two-arc sequence fixing can yield significant speedups, especially for the most difficult instances.

The remainder of this text is structured as follows: Section 2 sketches the basic ideas of variable fixing and introduces three types of properties that can be exploited for variable fixing in a column-generation context. Section 3 describes our BPC algorithms for the VRPTW and the four EVRPTW variants. In Section 4, the results of comprehensive computational experiments are presented and discussed. Final conclusions are drawn in Section 5.

2 Variable fixing

Variable fixing refers to a collection of techniques that allow the identification of variables of a mathematical optimization problem that must take unique values in every feasible or every optimal solution to the problem. These variables can obviously be replaced by unique constant values (this is *variable fixing*). In particular, for a variable that must be 0, it can be eliminated completely (*variable elimination*). The focus here is on the latter but we anyway speak of 'variable fixing'.

For this section, we assume that M is a (possibly mixed) integer linear minimization problem. Let the non-negative integer variables of M be $x_p \geq 0$ for $p \in P$, where P is the respective index set. Variable fixing by reduced cost is based on the following well-known result.

Proposition 1 *Let UB be an upper bound on the optimal value of problem M, and let ν be a dual solution to the linear relaxation of M providing a lower bound $LB(\nu)$. If an integer variable x_p for some $p \in P$ having reduced cost $\tilde{c}_p(\nu)$ with respect to ν fulfills $\tilde{c}_p(\nu) > UB - LB(\nu)$, then $x_p = 0$ holds true for every optimal solution to M.*

In a BPC algorithm, fixing to 0 a single variable, i.e., setting $x_p = 0$ for some $p \in P$, is not productive: One must not only set the variable to 0 in the RMP, but also forbid the re-generation of the same variable x_p by the pricing problem, which is typically difficult (for a detailed discussion we refer to Villeneuve and Desaulniers 2005, Lübbecke and Desrosiers 2005).

2.1 Properties and minimum reduced costs of subsets of paths

We are now interested in finding properties of variables, for which the precondition of Proposition 1 can be tested efficiently and for which the (re-)generation by the pricing problem of all variables having the same property can be prevented effectively. When the variables are associated with paths in a network, *containing a given arc* is an example of such a variable property. Others are listed below.

Definition 1 *Let Π be a* property *that can be tested for every $p \in P$. For the sake of simplicity, we assume $\Pi \subset P$ so that we can write $p \in \Pi$ if and only if p satisfies Π. We define:*

(i) A solution $\mathbf{x} = (x_p)_{p \in P}$ of problem M fulfills property Π if at least one variable x_p, $p \in \Pi$, is strictly positive.

(ii) Problem M fulfills property Π if at least one of its optimal solutions fulfills Π.

Obviously, for properties Π not satisfied by problem M, we can eliminate all variables x_p with $p \in \Pi$ from the problem and, in particular, forbid their generation by the pricing problem. To show that a property Π is not fulfilled by a problem M, Proposition 1 can be applied as follows:

$$\min_{p \in \Pi} \tilde{c}_p(\nu) > UB - LB(\nu) \quad \Rightarrow \quad \text{no optimal solution } \mathbf{x} = (x_p)_{p \in P} \text{ of } M \text{ fulfills } \Pi \tag{1a}$$

$$\Rightarrow \quad \text{problem } M \text{ does not fulfill } \Pi. \tag{1b}$$

This is the approach used by Irnich et al. (2010) for routing and scheduling applications, where the variables correspond to paths generated by solving SPPRCs. Indeed, they proposed to fix to 0 all variables x_p that represent feasible paths containing a specific arc (this is the property Π) if the first condition in (1a) holds. In this case, one can remove this arc from the underlying arc set and thus reduce the size of the pricing problem for all subsequent iterations.

We introduce some basic notation for describing the SPPRC and the dynamic-programming labeling algorithm. An instance of the SPPRC is defined over a directed graph $D = (V, A)$ where V is the vertex set and A the arc set. The digraph can be a multigraph with parallel arcs or loops. For an arc a, we denote by $t(a) = i$ the *tail vertex* and $h(a) = j$ the *head vertex*, i.e., $a = (i, j)$. Moreover, for each $i \in V$, the sets $\delta^+(i)$ and $\delta^-(i)$ describe all arcs that have tail and head vertex i, respectively. Typically, the paths that describe feasible routes or schedules are o-d-paths where o is the source vertex and d the sink vertex. *Resource extension functions* (REFs), one function f_a for each arc $a \in A$, describe how the attributes of partial paths are propagated through the network. We assume that such attributes of a partial path p are given by pairs (T_i, U_i) where T_i is a vector of resources/resource levels, including a resource for accumulated reduced costs, and U_i the set of *unreachable vertices* from the endpoint i of p. Given a pair (T_i, U_i) and an arc $a \in A$, the REF f_a interprets (T_i, U_i) as the resource level at the *tail vertex* $i = t(a)$ and computes the new resource level (T_j, U_j) for the *head vertex* $j = h(a)$. To simplify the description of labeling, we assume that the REF also checks the feasibility of the result (T_j, U_j),

and infeasible results are indicated by setting the reduced cost resource T_j^{rdc} to ∞. For input values $T_i^{rdc} = \infty$, REFs always return $T_j^{rdc} = \infty$ again.

We give some examples: In the VRPTW, at least three attributes are needed to describe the resource level at the end of a partial path: the accumulated reduced cost T_i^{rdc}, the cumulated demand T_i^{load} of the customers visited so far, and the earliest possible service start time T_i^{time} at the last vertex i of the partial path. Variants of the EVRPTW require more attributes and in particular more sophisticated REFs (Desaulniers et al. 2014) that we sketch in Section 3.2. Moreover, additional attributes must also be added when non-robust valid inequalities are used in the RMP in order to strengthen the linear relaxation (Jepsen et al. 2008), see Section 3.3.

In case of elementary SPPRCs, the set U_i comprises all already visited vertices of p and possibly additional *unreachable* vertices that cannot be feasibly reached due to the current resource level T_i (Feillet et al. 2004). When using the well-known *ng*-path SPPRC relaxation (Baldacci et al. 2011), which is defined by a neighborhood $N_i \subset V$ for all $i \in V$, the set U_i is reduced so that $U_i \subseteq N_i$ holds for the endpoint i of the partial path p.

The following properties are interesting in the context of column-generation algorithms that have an SPPRC subproblem:

1. The property $\Pi(a)$ that an arc $a \in A$ is traversed by a path. In case the precondition of (1a) holds, the arc a can be eliminated from the pricing network. This is exactly what was proposed by Irnich et al. (2010).
2. The property $\Pi(a_1, a_2)$ that two arcs a_1 and a_2 with $h(a_1) = t(a_2)$ are traversed consecutively by a path. In case the precondition of (1a) holds, it is not directly clear how to exploit this. Unlike the single-arc case, it is not possible to remove two-arc sequences directly from network G. The labeling algorithm must be modified with respect to label propagation as well as to dominance. This is the focus of the article at hand.

3a. The properties $\Pi(i, r, <, e)$ that a vertex i can be reached with a resource level smaller than e for one of the resources r. In case the precondition of (1a) holds, the lower bound of the resource window of resource r at vertex i can be increased to e.

3b. The properties $\Pi(i, r, >, l)$ that a vertex i can be reached with a resource level greater than l for one of the resources r. In case the precondition of (1a) holds, the upper bound of the resource window of resource r at vertex i can be decreased to l.

A combination of the properties $\Pi(a)$, $\Pi(i, r, <, e)$, and $\Pi(i, r, >, l)$ (1, 3a, and 3b) has been used in the work of Pecin et al. (2017b) for the CVRP, because they apply arc-fixing on a so-called *arc-load network* for pricing. In this arc-load network, vertices are pairs (i, q), where the combination indicates that the vehicle visiting customer i has an accumulated demand of q when leaving i. Arcs of the arc-load network are of the form $((i, q), (j, q+q_j))$ for $i, j \in V, i \neq j$ and they exist only for load values $q \geq q_i$ and $q+q_j \leq Q$. Arc fixing on this arc-load network allows fixations for original arcs $a = (i, j)$ for only some load values q, i.e., the fixing is more fine grained than a global arc fixing (with property $\Pi(a)$ alone). On the downside, Pecin et al. (2017b) show that dominance between two paths ending at a vertex i with different accumulated load is no longer possible. However, their computational experiments on CVRP benchmarks show that the smaller network resulting from the arc fixing overcompensates the less effective dominance.

Efficient reduced cost computation The efficient computation of the minimum reduced cost over paths that fulfill properties 1, 2, 3a, or 3b relies on bidirectional labeling as introduced for SPPRCs by Righini and Salani (2006). In contrast to standard bidirectional labeling algorithms, we have to perform a complete forward and a complete backward labeling without stopping the labeling process at a half-way point.

We assume that ν is the dual solution of the master program providing the lower bound $LB(\nu)$ to the overall problem. The result of forward labeling is the set $\mathcal{F}_i$ of non-dominated labels at each

vertex $i \in V$ generated by extending partial paths in forward direction, starting with the trivial path $p = (o)$ and using the forward REFs f_a for all arcs $a \in A$.

Backward labeling can either be described as labeling on the transposed (a.k.a. reversed or inverted) digraph to D (where, for each original arc, tail and head are swapped) or as labeling with backward REFs b_a for all original arcs $a \in A$. We follow the latter option as described in detail in (Irnich 2007, Tilk et al. 2017). Then, the result of backward labeling is the set $\mathcal{B}_i$ of non-dominated labels at each vertex $i \in V$ generated by extending partial paths in backward direction, starting with the trivial path $p = (d)$ using the backward REFs b_a for all arcs $a \in A$.

In bidirectional labeling, forward partial paths and backward partial paths are joined. For this purpose, a *merge operator* must be provided. It checks the feasibility of the concatenated path and computes the reduced cost of the latter. W.l.o.g. we assume that a merge is performed for labels of the same vertex (a merge over arcs is the alternative). For a forward label F and a backward label B associated with the same vertex, let p_f and p_b be the associated partial paths. We assume that our merge operator is a function $m(F, B)$ that returns the reduced cost of the path (p_f, p_b) if it is feasible, and $+\infty$ otherwise.

For the properties Π described above, we are now able to compute $\min_{p\in\Pi} \tilde{c}(p)$ with the help of the forward and backward labels and the merge operator.

First, for the property $\Pi(a)$ that refers to a single arc $a = (i, j) \in A$, we compute

$$\tilde{c}_a(\nu) := \min_{p\in\Pi(a)} \tilde{c}_p(\nu) = \min_{\substack{F\in\mathcal{F}_i,\\ B\in\mathcal{B}_j}} m(f_a(F), B). \tag{2a}$$

Second, for the property $\Pi(a_1, a_2)$ that refers to a two-arc sequence $a_1 = (i, j), a_2 = (j, k) \in A$, we compute

$$\tilde{c}_{a_1,a_2}(\nu) := \min_{p\in\Pi(a_1,a_2)} \tilde{c}_p(\nu) = \min_{\substack{F\in\mathcal{F}_i,\\ B\in\mathcal{B}_k}} m(f_{a_2}(f_{a_1}(F)), B). \tag{2b}$$

Third, for the properties $\Pi(i, r, <, e)$ and $\Pi(i, r, >, l)$ that refer to a vertex i, a resource r, and values e and l, respectively, we need some additional assumptions. If forward labeling uses a dominance that prefers smaller values T^r and backward labeling uses a dominance that prefers greater values T^r of the resource r, then we can compute

$$\tilde{c}_{i,r,<,e}(\nu) := \min_{p\in\Pi(i,r,<,e)} \tilde{c}_p(\nu) = \min_{\substack{F=(T,U)\in\mathcal{F}_i:\\ T^r<e,\\ B\in\mathcal{B}_i}} m(F, B), \tag{3a}$$

$$\tilde{c}_{i,r,>,l}(\nu) := \min_{p\in\Pi(i,r,>,l)} \tilde{c}_p(\nu) = \min_{\substack{F\in\mathcal{F}_i,\\ B=(T,U)\in\mathcal{B}_i:\\ T^r>l}} m(F, B). \tag{3b}$$

If forward and backward dominance rules prefer labels in the inverse sense, the role of ¡ and ¿ in the formulas (3a) and (3b) must be swapped.

In summary, if a value $\tilde{c}_a(\nu)$, $\tilde{c}_{a_1,a_2}(\nu)$, $\tilde{c}_{i,r,<,e}(\nu)$, or $\tilde{c}_{i,r,>,l}(\nu)$ exceeds $UB - LB(\nu)$, i.e., the *optimality gap*, then the arc a or the two-arc sequence (a_1, a_2) can be fixed to 0, or the resource window of resource r at vertex i can be tightened by intersecting it with $[e, \infty)$ or $(-\infty, l]$, respectively.

2.2 Labeling after fixing of two-arc sequences

We assume that the arcs A have undergone a preprocessing in which resource windows have been shrunk and infeasible arcs have been eliminated (see, e.g., Desrosiers et al. 1995, for how preprocessing works for time-window constraints). Let $\hat{A} \subset A$ be the set of feasible and potentially relevant arcs, i.e., those not yet fixed to 0. Since we typically apply arc fixing several times, the set $\hat{A}$ decreases accordingly. The set $Z = A \setminus \hat{A}$ describes the arcs fixed to 0.

Let ${}^2A$ be the set of two-arc sequences, i.e., ${}^2A = \{(a_1, a_2) \in A \times A : h(a_1) = t(a_2)\}$. We assume that ${}^2A$ contains only the pairs (a_1, a_2) for which preprocessing has approved feasibility, e.g., of the route $(o, t(a_1), h(a_1) = t(a_2), h(a_2), d)$ assuming that feasibility of routes is hereditary over subroutes. The new two-arc sequence fixing algorithm reduces ${}^2A$ to some subsets ${}^2\hat{A} \subseteq {}^2A$. Obviously, results from the single-arc fixing can be transferred to two-arc sequence fixing, and we assume that ${}^2\hat{A} \subset \hat{A} \times \hat{A}$ holds true.

Forward labeling In order to not create paths with two-arc sequences fixed to 0, the label definition must be extended to include two additional components denoted ℓ and E: Accordingly, a label is given by $F = (T, U, \ell, E)$, where ℓ is the last arc in the corresponding partial path and E is the subset of arcs along which this label can be extended. For the latter set we can assume $E \subseteq proj_2({}^2\hat{A} \cap (\{\ell\} \times \hat{A}))$, where $proj_2(\cdot)$ is the projection to the second component.

In forward labeling, the initial label at vertex o is $(T_o, \{o\}, \perp, \delta^+_{\hat{A}}(o))$, where the vector T_o is the problem-specific initial resource level, $\{o\}$ is the set of visited vertices, $\ell = \perp$ represents an undefined last arc, and $E = \delta^+_{\hat{A}}(o)$ is the set of all non-fixed arcs leaving the initial vertex o.

A label (T_i, U_i, ℓ_i, E_i) at vertex i can only be extended along an arc $a = (i, j) \in \delta^+(i)$, if $a \in E_i$ and $j = h(a) \notin U_i$. In this case, the new label at vertex j is of the form $(f_a(T_i, U_i), a, \delta^+_{{}^2\hat{A}}(a))$, where the REF f_a provides the new resource level and visited vertices $(T_j, U_j) = f_a(T_i, U_i)$, the new last arc is $\ell_j = a$, and further extensions of the new label can go along the arcs $E_j = \delta^+_{{}^2\hat{A}}(a) = proj_2({}^2\hat{A} \cap (\{a\} \times \hat{A}))$. Recall that feasibility of the new partial path ending at j is signaled by $T^{rdc}_j \neq \infty$.

Dominance As mentioned before, two-arc sequence fixing has an impact on dominance. It will turn out that two-arc sequence fixing is only applicable to labeling algorithms that base their dominance between two partial paths on identical extensions. To clarify applicability, we briefly recall the principles behind dominance:

Proposition 2 *(Dominance)*
Let p^1 and p^2 be two different partial paths starting at vertex o and ending at the same vertex $i \in V$. If, for each path q^2 that gives a feasible o-d-path $r^2 = (p^2, q^2)$ (this i-d-path q^2 is called an extension *of p^2),*

(i) the path $r^1 = (p^1, q^2)$ is also a feasible o-d-path,
(ii) or there exists another path q^1 such that $r^1 = (p^1, q^1)$ is also a feasible o-d-path (here the extension q^1 may or may not be identical to q^2),

and r^1 has a smaller or identical reduced cost compared to r^2, then the partial path p^2 is dominated.

Labels that belong to dominated partial paths are obviously not needed for finding Pareto-optimal o-d-paths and can therefore be discarded. (Only in case of a tie, i.e., bilateral dominance, one of the labels must be kept.) Many SPPRC labeling algorithms are solely based on Proposition 2(i), which we denote as the *identical extension dominance* (IED) principle. Examples are the CVRP, the VRPTW, and almost all of their extensions, where goods are either shipped from depot to customers and/or collected from customers and shipped to the depot. However, other VRPs like the *pickup-and-delivery problem* (PDP, Battarra et al. 2014) and the *dial-a-ride problem* (DARP, Doerner and Salazar-González 2014) that model point-to-point transportation typically rely on Proposition 2(ii). Recent BPC algorithms for these two problems are (Gschwind et al. 2018) and (Gschwind and Irnich 2015), respectively. Labeling algorithms that utilize the more general rule of Proposition 2(ii) are also used by Hintsch and Irnich (2018) when solving the *soft-clustered VRP* via BPC. In the following, we refer to this case as the *possibly different extension dominance* (PDED) principle.

We are now able to precisely describe the modified dominance for SPPRCs in which some two-arc sequences are fixed to 0. Let the (original) dominance relation between resources be given by an operator $\prec$, i.e., resource vector T^1 dominates resource vector T^2 if and only if $T^1 \prec T^2$ (in the VRPTW, the conditions are $T^{1,rdc} \leq T^{2,rdc}$, $T^{1,time} \leq T^{2,time}$, and $T^{1,load} \leq T^{2,load}$). After fixing some two-arc sequences, a straightforward dominance between two different labels is given by the following rule:

Rule 1 *(Straightforward dominance) Let $F^k = (T^k, U^k, \ell^k, E^k), k = 1, 2$ be two different labels referring to the same vertex. Moreover, assume that the (original) dominance relation $\prec$ between resource vectors is based on IED. Then F^1 dominates F^2 if $T^1 \prec T^2$, $U^1 \subseteq U^2$, and $E^1 \supseteq E^2$.*

Proof. Note that the preconditions $T^1 \prec T^2$ and $U^1 \subseteq U^2$ are valid for dominance in the standard case without two-arc sequence fixing. Hence, and because IED is assumed to hold, for any feasible extension of the second label along q^2, exactly this extension applied to the first label produces a feasible o-d-path with a reduced cost not greater than the reduced cost of the path resulting from the second extension. If the extension of the second label is feasible with respect to two-arc fixing, as it was assumed, then the additional condition $E^2 \subseteq E^1$ ensures that this extension is also feasible for the first label. □

This straightforward dominance given by Rule 1 is rather weak, i.e., it can be expected that only a small fraction of the labels can be discarded with this rule. Next, we describe a reduction rule that helps to discard additional labels.

Rule 2 *(Reduction) A label $F = (T, U, \ell, E)$ with $E \cap (V \times U) \neq \varnothing$ is equivalent to the label $F' = (T, U, \ell, E')$ with $E' = E \setminus (V \times U)$.*

Proof. The set of unreachable vertices U forbids the extension of label F to vertices in U, i.e., along arcs $\hat{A} \cap (V \times U)$. Thus, arcs $E \cap (V \times U)$ are irrelevant for extension, and can therefore be subtracted from the set E. □

One can use this rule immediately when a label is created. Note that the reduction rule does not negatively impact a label's potential to dominate other labels. Indeed, Rule 1 requires $U^1 \subseteq U^2$.

The following partial dominance rule is in the spirit of other partial dominance tests, e.g., for SPPRC with linear node costs (Ioachim et al. 1999), for SPPRC with 2-cycle and k-cycle elimination (Kohl et al. 1999, Irnich and Villeneuve 2006), and the SPPRC for a variant of the VRPTW with convex inconvenience costs (He et al. 2019).

Rule 3 *(Partial dominance) Let $F^k = (T^k, U^k, \ell^k, E^k), k = 1, 2$ be two different labels referring to the same vertex. Moreover, assume that the (original) dominance relation $\prec$ between resource vectors is based on IED. Then F^1 partially dominates F^2 if $T^1 \prec T^2$, $U^1 \subseteq U^2$, and $E^1 \cap E^2 \neq \varnothing$. If label F^1 partially dominates F^2, then the set E^2 can be replaced by $E^2 \setminus E^1$ which is a proper subset of the former.*

Proof. This is a direct consequence of the Rules 1 and 2. □

If at some point the set E^2 becomes empty, the label F^2 can be discarded.

Backward labeling Also for the backward labeling, the definition of a label must be adapted. A backward label describes a partial path from a vertex $j \in V$ to the sink d. It is given by $B_j = (T_j, U_j, \ell_j, E_j)$, where the latter two components are the first arc ℓ_j in the corresponding partial path and E_j is a subset of the arcs along which this label can be extended in backward direction, i.e., $E_j \subseteq proj_1(^2\hat{A} \cap (\hat{A} \times \{\ell_j\}))$.

The initial backward label at vertex d is $(T_d, \{d\}, \perp, \delta^-_{\hat{A}}(d))$, where the vector T_d is the problem-specific (upper bound on the) terminal resource level, $\{d\}$ is the set of visited vertices, $\ell = \perp$ represents an undefined first arc, and $E = \delta^-_{\hat{A}}(d)$ is the set of all non-fixed arcs leaving the sink vertex d.

A backward label (T_j, U_j, ℓ_j, E_j) at vertex j can only be extended along arc $a = (i, j) \in \delta^-(j)$, if $a \in E_j$ and $i = t(a) \notin U_j$. In this case, the new label at vertex i is of the form $(b_a(T_j, U_j), a, \delta^-_{^2\hat{A}}(a))$, where the backward REF b_a provides the new resource level and visited vertices $(T_i, U_i) = f_a(T_j, U_j)$,

the new first arc is $\ell_i = a$, and further extensions of the new label may use the arcs $E_i = \delta^-_{{}^2\hat{A}}(a) = proj_1({}^2\hat{A} \cap (\hat{A} \times \{a\}))$.

The same straightforward and partial dominance Rules 1 and 3 are valid for backward labeling. In the backward case, $\prec$ describes the dominance between the backward resources, the first arc ℓ_j of backward labeling takes the role of the last arc ℓ_i in forward labeling, and the set E_j of arcs along which an extension is possible takes the role of the forward arc set E_i. Moreover, the reduction Rule 2 has a straightforward equivalent in the backward labeling. For the sake of brevity, we omit the formal statement of the backward rules.

Merging forward and backward labels As in Section 2.1, we assume that a merge operator m is given. For a forward label $F_i = (T_i, U_i, \ell_i, E_i)$ and a backward label $B_i = (T'_i, U'_i, \ell'_i, E'_i)$ at the same vertex i, the result of the new merge is again the reduced cost given by $m((T_i, U_i), (T'_i, U'_i))$ of the resulting path, if this path is feasible, and ∞ in case of infeasibility. Using two-arc sequence fixing requires a modified feasibility check where the following additional condition

$$(\ell_i, \ell'_i) \in {}^2\hat{A} \tag{4}$$

must be fulfilled. This additional condition (4) must be also considered when computing minimum reduced costs using the formulas (2) and (3). This is particularly relevant for repeated use of SPPRCs with two-arc sequence fixing.

3 Branch-price-and-cut algorithms

We now formally but briefly define the VRPTW and the four variants of the EVRPTW. Moreover, we discuss our BPC implementation including the column-generation mechanism, the branching and cutting strategies, and the way in which we apply the single-arc and two-arc sequence fixing techniques.

3.1 VRPTW and four variants of the EVRPTW

The VRPTW is defined on a directed graph $D = (V, A)$, where the vertex set V comprises the customers N and two copies o and d of the depot. For each customer $i \in N$, let q_i be the demand, s_i the service duration, and $[e_i, l_i]$ the time window describing the possible service start time. For the depot, we assume $s_o = s_d = q_o = q_d = 0$. The arcs $(i, j) \in A$ represent the possible travel connections, each characterized by a travel time t_{ij} and a routing cost c_{ij}. A fleet of homogeneous vehicles each with capacity Q is hosted at the depot. A feasible route in the VRPTW is an elementary o-d-path satisfying the capacity and time-window constraints. More precisely, let an elementary o-d-path $(i_0, i_1, \ldots, i_p)$ for some $p > 1$ and $i_0 = o$ and $i_p = d$ be given. It fulfills the capacity constraint if $\sum_{j=0}^{p} q_{i_j} \leq Q$. The time-window constraints are satisfied if there exists a schedule $(T_0, T_1, \ldots, T_p) \in \mathbb{R}^{p+1}$ with $T_j \in [e_{i_j}, l_{i_j}]$ for all $j = 0, \ldots, p$ and $T_{j-1} + s_{i_{j-1}} + t_{i_{j-1}, i_j} \leq T_j$ for all $j = 1, \ldots, p$. The cost of a route is given by $\sum_{j=1}^{p} c_{i_{j-1}, i_j}$.

A feasible solution to the VRPTW is a set of feasible routes that together visit and serve all customers $i \in N$ exactly once. A feasible solution is optimal if it minimizes the total cost of the routes. Accordingly, the VRPTW can be formulated as a set-partitioning problem. Let Ω be the set of all feasible routes. For each route $r \in \Omega$, let c_r denote its cost and, for each customer $i \in N$, let $a_{ir} \in \{0, 1\}$ be the indicator whether route r visits customer i. With binary route variables λ_r, the set-partitioning formulation is

$$\min \sum_{r \in \Omega} c_r \lambda_r \tag{5a}$$

$$\text{subject to} \sum_{r \in \Omega} a_{ir} \lambda_r = 1 \qquad \forall i \in N \tag{5b}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \Omega \tag{5c}$$

where the objective (5a) minimizes the overall routing cost, (5b) are the partitioning constraints, and (5c) describes the domain of the route variables.

The EVRPTW is an extension of the VRPTW tailored to the limited driving range of the electric commercial vehicles and the possibility to recharge the vehicle's battery at dedicated recharging stations. Therefore, the vertex set V is enlarged by a set R of recharging stations. Arcs $a = (i, j) \in A$ represent the possible travel connections with a travel time t_{ij} and a routing cost c_{ij}. Desaulniers et al. (2016) have shown that the battery-capacity constraint can be modeled with the help of the time b_{ij} for each arc $(i, j) \in A$ required to recharge the consumed energy when traveling between locations i and j. Let B be the corresponding battery capacity (expressed in time units). For a route $r = (i_0, i_1, \ldots, i_p)$, one must decide on the amount to recharge at every visited recharging station. Moreover, the corresponding recharging time needs to be incorporated into the time-window constraints. A necessary condition for a route to be EVRPTW feasible is that there exist a schedule $(T_0, T_1, \ldots, T_p)$ and a (battery-)loading plan $(X_0, X_1, \ldots, X_p)$ that fulfill the following conditions: First, loading is only possible at recharging stations, i.e., $X_j = 0$ if $i_j \in N \cup \{o, d\}$ and $0 \leq X_j \leq B$ for $i_j \in R$. Second, the time-window constraints are fulfilled, i.e., $T_j \in [e_{i_j}, l_{i_j}]$ for all $j = 0, \ldots, p$ as well as $T_{j-1} + s_{i_{j-1}} + X_{i_{j-1}} + t_{i_{j-1},i_j} \leq T_j$ for all $j = 1, \ldots, p$ assuming that there are no service times at recharging stations, i.e., $s_{i_j} = 0$ for $i_j \in R$. Third, the loading plan must be feasible, i.e., $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q-1} X_{i_j} \geq 0$ and $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q} X_{i_j} \leq B$ for all $q = 1, \ldots, p$. Finally, the standard capacity constraint on cumulative demand must be respected.

For the EVRPTW-MP (multiple, partial recharges), the given conditions are sufficient. For the single recharge policy, at most one of the vertices $i_0, i_1, \ldots, i_p$ can be a recharging station. For the full recharge policy, the battery must always be completely recharged at recharging stations, i.e., $B - \sum_{j=1}^{q} b_{i_{j-1},i_j} + \sum_{j=1}^{q} X_{i_j} = B$ if $i_q \in R$. With the respective definition of Ω as the set of all feasible EVRPTW routes, all four EVRPTW variants can be modeled with the same set-partitioning formulation (5).

3.2 Column generation

The SPPRC pricing problems of the VRPTW and the EVRPTW variants are solved with dynamic-programming labeling algorithms as described in general in Section 2.1. For the sake of conciseness, we only summarize what is essential for the understanding of this paper and avoid details (such as providing the REFs) that can be found in the stated references.

Recall that forward as well as backward labels are given by pairs (T, U), where T is a vector of attributes or resources and U the set of unreachable customers. In the VRPTW, the three attributes T^{rdc}, T^{load}, and T^{time} are completely independent. The value T^{rdc} is the accumulated reduced cost along the partial path, T^{load} the accumulated load, and T^{time} the earliest feasible service start (in forward labeling) or latest feasible service end time (in backward labeling). For these resources, the forward and backward REFs are straightforward. Details can be found in (Irnich 2007, Desaulniers et al. 2014). Note that we use the *ng*-path SPPRC relaxation (Baldacci et al. 2011) instead of the strongly $\mathcal{NP}$-hard elementary problem.

In the EVRPTW, the resources T^{rdc}, T^{load}, and T^{time} of the VRPTW need to be complemented with additional resources (see Table 1) and their propagation depends on the EVRPTW variant. For the variants that follow the full recharging policy (SF, MF), resources also depend on the direction of label propagation. In forward direction, T^{rch} counts the number of recharges (visits to recharging stations) and T^{rt} is the cumulated required recharging time since the last recharge (if any) or since leaving the depot (otherwise). In backward direction, when reaching a recharging station, we do not know the actual amount that must be recharged, because it depends on the yet unknown subpath between two consecutive recharges. This subpath results from further backward extensions. Instead of a concrete amount, one can calculate slack times that can be used to recharge while maintaining overall time-window feasibility. Let us assume that the backward label (T, U) resides at a vertex j. The two associated resources are T^{sl} which is the cumulated slack time at j that can be used for recharging at the next recharging station without changing the latest service start time at j, and T^{avrt} which

is the maximum additionally available recharging time at the next recharging station that ensures time-window feasibility. Finally, an additional resource T^{nrch} counts the negative of the number of recharges.

Table 1: Resources in VRPTW and variants of EVRTPW

Problem		VRPTW	EVRPTW-SF/MF		EVRPTW-SP/MP
Attribute(s)	#	forward/backward	forward	backward	forward/backward
T^{rdc}, T^{load}	2	•	•	•	•
T^{time}	1	•	•	•	
T^{rch}	1		•		•
T^{rt}	1		•	•	
T^{nrch}, T^{sl}, T^{avrt}	3			•	
T^{tMin}, T^{tMax}, T^{rtMax}	3				•
Total number of attributes		3	5	7	6

For the EVRPTW variants that follow the partial recharge policy (SP, MP), forward and backward labeling can be based on the same type of attributes and REFs, because any time-window and recharging feasible forward *o*-*d*-path is, if reversed, a feasible *d*-*o*-path in the transposed network, and vice versa, i.e., there exists a possibly different but feasible schedule and battery-load plan for the reversed path if and only if one exists for the original path (see Desaulniers et al. 2016, p. 1398). Three additional attributes T^{tMin}, T^{tMax}, and T^{rtMax} are needed to describe the linear tradeoff between the maximum amount of energy that can be recharged (also expressed as a recharging time) and the earliest service time. The latter time is no longer fixed but can freely be chosen in the time interval $[T^{tMin}, T^{tMax}]$ over which the tradeoff-curve with slope -1 is described by the initial maximum amount of energy T^{rtMax}. Note that the tradeoff-curve is constructed so that it ensures that a feasible schedule and load plan exist. Complete details about the labeling algorithms can be found in the original work (Desaulniers et al. 2016).

For all EVRPTW variants, the interdependency between the attributes is high: for example, in the EVRPTW-SP/MP, the computation of the value of T^{rtMax} at a given node depends on the values of four resources at the previous node, namely, T^{rch}, T^{tMin}, T^{tMax}, and T^{rtMax}. One can therefore expect that labeling for an EVRPTW instance is more difficult compared to the respective VRPTW instance, because it creates more undominated labels and uses less effective dominance rules. The computational experiments of Section 4.2 show that the new two-arc sequence fixing reduces the number of created labels and dominance tests to be performed. Furthermore, they highlight that the more difficult to solve an instance is, the more pronounced this effect is.

Finally note that for both VRPTW and EVRPTW further resources are added to cope with the non-robust subset-row inequalities as explained in the next subsection.

Bidirectional labeling with dynamic half-way point Several works on different VRP variants have proven that bidirectional labeling algorithms outperform their monodirectional counterparts. As shown by the computational study of Desaulniers et al. (2016, Section 5.2), this remains true for the EVRPTW with the full recharge policy (SF, MF) even if the backward labeling requires seven attributes compared to only five attributes when labeling in forward direction.

To determine the half-way point when label extension stops in both directions, we use the monotone attributes T^{time} for the VRPTW and EVRPTW-SF/MF and T^{tMin} for the EVRPTW-SP/MP. As proposed by Tilk et al. (2017), we exploit the unbalance in forward and backward labeling by introducing a dynamic half-way point. In fact, we apply the following labeling strategy: All labels referring to the same time instant (T^{time} or T^{tMin}) are extended together, i.e., en block. For deciding between forward and backward label extension, we compare the total number of unprocessed forward and backward labels, i.e., labels created but not extended. If there are less unprocessed forward labels, then the forward direction is chosen, backward otherwise. The dynamic half-way point is then the time instant where forward and backward blocks meet.

Further acceleration strategies To speed up the column-generation process, heuristics can be used to identify negative reduced-cost columns fast. If the heuristics are unable to find additional columns, one has to resort to an exact method to solve the pricing problem. In our algorithm, we sequentially apply three pricing heuristics before calling the exact pricer. The heuristics rely on arc-reduced networks ensuring that a minimum of $k = 5$, 10, and 15 arcs, respectively, enter and exit each customer node (Desaulniers et al. 2008). If after performing arc-fixing the average out-degree $\sum_{i \in N} |\delta^+(i) \cap \hat{A}|/N$ of the customer nodes drops below k the respective pricing heuristic is skipped.

Finally, to avoid unnecessary dominance tests, we precompute for every pair of arcs ℓ_1 and ℓ_2 with joint head vertex $i = h(\ell_1) = h(\ell_2)$ whether $\delta^+_{2\hat{A}}(\ell_1) \cap \delta^+_{2\hat{A}}(\ell_2) \neq \varnothing$ holds true. Otherwise, a partial dominance between two labels (T_1, U_1, ℓ_1, E_1) and (T_2, U_2, ℓ_2, E_2) at vertex i cannot eliminate any feasible extension to either of the two labels in the partial dominance because $E_1 = E_1 \setminus E_2$ and $E_2 = E_2 \setminus E_1$. Therefore, dominance tests between any two such labels can be skipped. The same technique is used for backward labeling in an analog fashion.

3.3 Cutting

We use two types of valid inequalities: capacity cuts and limited memory subset-row inequalities described in the following. For any subset of customers (and recharging stations) $C \subseteq V \setminus \{o, d\}$, $C \neq \varnothing$, the associated *capacity cut* (CC) is

$$\sum_{r \in \Omega} \sum_{(i,j) \in \delta^-(C)} a_{ij,r} \lambda_r \geq \left\lceil \frac{\sum_{i \in C} q_i}{Q} \right\rceil, \tag{6}$$

where $\delta^-(C)$ is the set of arcs with tail vertex in $V \setminus C$ and head vertex in C, and the coefficient $a_{ij,r}$ describes how often a route $r \in \Omega$ traverses an arc $(i, j) \in A$. The dual price of a CC refers to the arcs $\delta^-(C)$ and can hence be directly incorporated into the reduced costs $\tilde{c}_{ij}$.

In order to identify violated inequalities (6), we use two variants of the shrinking heuristic (extended shrinking and greedy shrinking) as presented in (Ralphs et al. 2003).

Subset-row inequalities (SRIs) were first introduced for the VRPTW by Jepsen et al. (2008). They refer to subsets of the set-partitioning constraints (5b), i.e., SRIs are defined for subsets $S \subseteq N$ and weights for each vertex of S. As in several other works, we restrict ourselves to subsets of cardinality $|S| = 3$ and weights $1/2$ so that the associated SRI is

$$\sum_{r \in \Omega} \left\lfloor \frac{\sum_{i \in S} a_{ir}}{2} \right\rfloor \lambda_r \leq 1. \tag{7}$$

SRIs are non-robust, i.e., each SRI with a positive dual price requires an additional resource and a tailored but weaker dominance rule in the SPPRC labeling algorithm. To reduce the negative impact, Pecin et al. (2017b) introduced the limited memory variant of the SRIs. The idea is similar to the well-established *ng*-route relaxation (see Section 3.2), i.e., a vertex memory controls at which vertices the state of the SRI resource remains relevant. We use the same separation algorithm and vertex memory as described in the work of Pecin et al. (2017b). In particular, an inequality qualifies as violated only if the violation is at least $\varepsilon_{cut} = 0.05$. Note that we always try to add violated CCs first so that SRIs are only separated if no violated CCs have been identified in the current round of separation.

3.4 Branching

In our BPC algorithms, the branch-and-bound search tree is explored with a best-bound first node selection strategy. This strategy fosters high-quality lower bounds and small search trees at the cost of probably finding feasible integer solutions later.

Let $(\bar{\lambda}_r)_{r \in \Omega}$ be a fractional solution of the master program. The branching for VRPTW and EVRPTW differs but is in both cases based on a multi-level rule, i.e., decisions are evaluated in the

given order and the first type that can be imposed is selected. In case of choices, we always select a branching variable where the fractional value is closest to 0.5. Then, two branches are created, one in which the branching variable is bounded from above by the rounded-down value and one in which the branching variable is bounded from below by the rounded-up value.

For the VRPTW, we apply the following two-level branching strategy: branching on

(1) the total number of routes ($\sum_r \bar{\lambda}_r$) and
(2) the total flow ($\sum_r a_{ij,r}\bar{\lambda}_r$) on an arc $(i,j) \in A$.

In the RMP, all routes that do not respect the branching decision are temporarily set to 0. Moreover, an inequality is added to the RMP for the first-level branching, while the second-level branching is enforced by temporarily removing one or several arcs from the subproblem network.

For the EVRPTW, we apply the same four-level branching strategy as in (Desaulniers et al. 2016): branching on

(1) the total number of routes,
(2) the total number ($\sum_{i\in R}\sum_r a_{ir}\bar{\lambda}_r$) of recharges,
(3) the total number ($\sum_r a_{ir}\bar{\lambda}_r$) of recharges at a given recharging station $i \in R$, and
(4) the total flow on an arc.

Here, decisions on the first three levels are enforced by adding an inequality to the RMP. As all inequalities are robust, their dual prices only alter the reduced cost $\tilde{c}_{ij}$ of certain arcs (i,j) but do not change the structure of the SPPRC pricing problem.

3.5 Arc-fixing strategy

Arc fixing is possible whenever a feasible dual solution provides a lower bound and minimal reduced costs of subsets of routes can be computed. In particular, arc fixing can be performed at every branch-and-bound node (when the linear relaxation is solved) and after every round of adding valid inequalities. Such a fixing is then only locally valid in the search tree. For our later experiments however we want to keep the algorithmic setup as simple and reproducible as possible. Thus, we perform arc fixing only at the root node of the branch-and-bound tree allowing the addition of valid inequalities. In this case, all fixations are globally valid.

Computation of minimal reduced costs Now we explain when and how minimal reduced costs $\min_{p\in\Pi}\tilde{c}(p)$ for properties Π that refer to arcs and two-arc sequences are computed in our BPC algorithm (see also Equations (2)). When a linear relaxation is solved, complete monodirectional forward and backward labeling procedures have to be performed. At this point, the solution of the last bidirectional pricing with bounded forward and backward labeling using a half-way point (Righini and Salani 2006) can be exploited. All labels that have been generated but not extended due to the half-way condition can be used to restart the labeling. Without the half-way condition they can be further extended to complete the monodirectional runs. This prevents repeating an otherwise identical first phase of labeling.

Since forward labels can be used to bound backward labels, and vice versa, the next question is whether to start with the complete forward or the complete backward direction. The bidirectional labeling with the dynamic half-way point gives an indication which direction is probably less difficult: If the dynamic half-way point is left to the actual midpoint (we use time as the monotone resource and therefore the middle of the planning horizon), backward labeling is expected to be simpler. In this case, we start with the complete backward labeling, otherwise the complete forward labeling is performed first. The other direction is solved afterwards with the possibility of sharper bounding. Note that this bounding is crucial only for labels that are slightly beyond the half-way point, because bounding information is typically not stored for each possible resource state but bucketwise.

The information from the complete forward and backward labeling can then be used to compute lower bounds as detailed in Section 2.1.

Use of reduced costs and lower bounds Note first that the condition in (1a) can be rewritten as $LB(\nu) + \min_{p\in\Pi} \tilde{c}_p(\nu) > UB$ so that the left-hand side can be considered a *property-specific lower bound*. The simplest way to perform variable fixing is to adhoc compare a computed lower bound (such as the ones given by $LB(\nu)$ and (2) and (3)) with the best UB at hand. If the computed lower bounds that are currently below UB are stored, any improved UB found later in the branch-and-bound may allow additional fixings. For a very tight UB (if one is very confident that a solution is nearly optimal), there is hardly any value in storing lower bounds. Consequently, an exact computation of minima in (2) and (3) is not necessary. Instead, premature termination of the computation is possible if for a compatible combination (F, B) of forward and backward labels the value drops below $UB - LB(\nu)$. We use this shortcut to avoid extensive but irrelevant exact bound computations.

Actual variable fixation strategy Since two-arc sequence fixing benefits from a reduced arc set, we perform single-arc fixing first. Note also that the computational effort of computing the lower bounds for two-arc sequences is certainly higher because the number of sequences is generally cubic ($\mathcal{O}(n^3)$).

Finally, at almost no cost, the labeling that respects fixed two-arc sequences can naturally cope with two-cycle elimination. Indeed, each two-cycle is a two-arc sequence. We eliminate (fix to 0) all two-cycles independently of their lower bounds.

4 Computational results

In this section, we report our computational results on variable-fixing techniques in BPC algorithms for VRPTW and EVRPTW variants. All results were obtained using a standard PC with an Intel(R) Core(TM) i7-6900k processor clocked at 3.2 GHz, equipped with 64 GB RAM main memory running Windows 10. The BPC algorithms were implemented in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2013. The callable library of CPLEX 12.6.1 was used for solving the RMPs.

Note that we use an identical setup for our BPC algorithms independent of whether VRPTW or EVRPTW instances are solved. We would probably get better results with problem-tailored setups, but comparisons between VRPTW and EVRPTW would become more difficult to interpret.

VRPTW instances We use Solomon's instances for all experiments on the VRPTW. The original benchmark consists of 56 instances with 100 customers, where instances can be characterized according to the customer distribution as random (R), clustered (C), or mixed (RC) as well as by tight (series 1 with subsets R1, C1, RC1) or loose (series 2 with subsets R2, C2, RC2) constraints. Smaller instances result from considering only the first 25 or 50 customers, respectively. Optimal solutions have been computed for all $56 \times 3 = 168$ instances. While exact results are scattered over the literature (the only complete table with results that we are aware of can be found in the Online Supplement of He et al. 2019), today's most competitive metaheuristics (e.g., Vidal et al. 2013) routinely find these optimal solutions, too. In order to keep the computational setup as simple and reproducible as possible, we initially provide an upper bound of $UB = opt + 1$ for the BPC algorithms, where opt is the cost of an optimal solution. A similar initialization has been used in several other works (Pecin et al. 2017a,b, Pessoa et al. 2018).

EVRPTW instances Schneider et al. (2014) constructed the 100-customer EVRPTW benchmark instances from Solomon's benchmark in the following way: (i) 21 randomly generated recharging stations are added; (ii) the battery capacity is suitably set depending on the average route length of the corresponding VRPTW instance; and (iii) the time windows of some customers are enlarged to ensure feasibility. The energy consumption b_{ij} along an arc $(i, j) \in A$ is set equal to the arc cost c_{ij}, and the proportionality factor α is chosen such that a complete battery recharge requires three times the average customer service time of the considered instance. As before, smaller instances with 25

and 50 customers were created. With the four recharging variants (SF, SP, MF, MP) this leads to an overall benchmark of $56 \times 3 \times 4 = 672$ instances.

Reduction and grouping instances of the testbed We restrict the testbed to those (E)VRPTW instances that require cutting and/or branching to compute an integer optimal solution. In the other cases, the arc-fixing procedures are never performed so that the instances would otherwise bias the statistical analysis. Moreover, we restrict ourselves to those instances that the new BPC with single-arc and two-arc sequence fixing solves to integer optimality within 2 hours. For the VRPTW, 86 instances are already integer optimal for the linear relaxation and only instances `R208_100` and `R211_100` are not solved within 2 hours leading to 80 VRPTW instances for the experiments. For the EVRPTW, the remaining testbed comprises $101 + 90 + 101 + 87 = 379$ EVRPTW (SF, SP, MF, MP) instances ($40+50+39+51 = 180$ are dropped due to optimality of the linear relaxation and $27+28+28+30 = 113$ cannot be solved within the 2 hours). The appendix lists the instances we use and provides detailed instance-by-instance results for the EVRPTW.

Experiments with BPC and only single-arc fixing have been conducted with a time limit of 10 hours per instance. Computation times heavily depend on the (practical) difficulty of the instances. To explicitly point out this dependency, we group instances by their size (number of customers: 25, 50, and 100) as well as by computation time. For the latter grouping, we distinguish between instances where the BPC algorithm with single-arc fixing terminates with a proven integer optimum within less than 60 seconds (<60s) or runs longer (≥60s). Similarly, we use a 1-hour limit for two groups denoted by '<3600s' and '≥3600s'.

4.1 Arc fixing at the root node

In a first series of experiments, we compare *single-arc fixing* (SAF) with SAF that is complemented by *two-arc sequence fixing* (TASF). To be precise, we start by describing the two respective column-generation procedures in more detail. Both algorithms prematurely terminate after solving the linear relaxation of the master program and after having added valid inequalities (SRIs and CCs) according to the separation strategy described in Section 3.3.

The first column-generation algorithm applies SAF only. This happens the first time when the linear relaxation of the master program is solved (results are presented under head 'LP'). Iteratively, we call the cut separation procedure, add the computed violated inequalities, and re-apply the column-generation procedure on arc-reduced networks followed by SAF again. The algorithm terminates when no more violated inequalities are found (results for this point are indicated by 'LP+Cut'). Table 2 displays in its first five columns the results of the SAF grouped by the type of problem and instances. The number of instances in each group is shown in column '#Inst'. The average initial number of arcs and the average percentage of arcs fixed to 0 at time 'LP' and time 'LP+Cut' are displayed in the two last columns of the SAF section, i.e., $100 \cdot (|A| - |\hat{A}|)/|A| = 100 \cdot |Z|/|A|$.

Note that averages for absolute numbers (such as the number of arcs) are arithmetic means over the instances. In contrast, geometric means are computed for all ratios (here the percentage of fixed arcs).

The second column-generation algorithm uses SAF directly followed by TASF. The consequence is that every round of TASF uses a reduced set $^2\hat{A}$, where the reduction comes from SAF as well as the previous round(s) of TASF. The last five columns of Table 2 summarize what changes with the additional TASF. After solving the linear relaxation and applying SAF the first time ('after SAF LP'), the comparison of the columns $|^2\hat{A}|$ and $|^2A|$ (sixth and seventh columns) reveals that SAF already eliminates more than 90 % of the feasible two-arc sequences. Of those sequences that remain (i.e., $|^2\hat{A}|$), TASF then fixes to 0 approximately another half on average ('Fixed'). The interesting point here is that TASF for larger and more difficult instances (the reader may compare the respective groups 25, 50, 100 and groups <60s, ≥60s) yields a higher number of two-arc fixations. This is completely consistent over all five problem types VRPTW and EVRPTW (SF, SP, MF, MP).

Table 2: Comparison of Single-Arc Fixing (SAF) and Two-Arc Sequence Fixing (TASF) at the root node

		SAF Number of arcs			TASF Number of two-arc sequences				
			Fixed			after SAF LP		after LP+Cut	
Group	#Inst	$\lvert A\rvert$	LP	LP+Cut	$\lvert {}^2A\rvert$	$\lvert {}^2\hat{A}\rvert$	Fixed	$\lvert {}^2\hat{A}\rvert$	Fixed
VRPTW									
25	16	531	75.5 %	82.8 %	9 075	456	37.8 %	220	38.4 %
50	28	1 924	61.8 %	86.1 %	60 429	6 201	39.9 %	1 167	43.5 %
100	36	7 580	71.0 %	93.2 %	476 558	46 103	56.0 %	2 475	50.1 %
<60s	57	2 801	69.1 %	87.8 %	111 152	8 349	41.0 %	752	40.3 %
≥60s	23	7 634	66.9 %	90.5 %	550 332	59 337	61.1 %	3 583	60.1 %
All	80	4 190	68.4 %	88.5 %	237 416	23 008	46.0 %	1 566	45.2 %
EVRPTW-SF									
25	31	1 408	87.1 %	90.9 %	23 372	865	49.6 %	410	45.8 %
50	41	3 552	80.9 %	92.2 %	115 327	7 338	56.2 %	1 288	50.2 %
100	29	9 156	73.7 %	93.0 %	417 672	50 180	58.7 %	3 610	47.3 %
<60s	75	3 252	84.3 %	92.2 %	93 402	4 636	52.3 %	722	45.6 %
≥60s	26	8 112	70.8 %	91.4 %	406 165	55 201	62.4 %	4 463	55.8 %
All	101	4 503	80.6 %	92.0 %	173 915	17 653	54.7 %	1 685	48.0 %
EVRPTW-SP									
25	28	1 430	87.2 %	91.2 %	25 512	807	52.7 %	374	46.9 %
50	40	3 553	80.4 %	92.8 %	115 874	7 346	57.7 %	1157	48.8 %
100	22	9 333	78.5 %	94.3 %	489 049	32 268	59.7 %	2 553	45.6 %
<60s	68	3 209	84.3 %	92.8 %	94 027	4 561	54.4 %	582	44.6 %
≥60s	22	7 693	75.1 %	92.0 %	441 571	32 555	64.0 %	3 333	57.1 %
All	90	4 305	82.0 %	92.6 %	178,982	11 404	56.6 %	1 255	47.4 %
EVRPTW-MF									
25	30	1 473	85.8 %	91.0 %	30 470	1137	49.7 %	443	44.8 %
50	43	3 752	84.2 %	93.8 %	133 902	5662	53.0 %	781	46.9 %
100	28	9 880	74.5 %	94.3 %	485 832	51937	60.0 %	2 649	46.8 %
<60s	71	3 286	84.9 %	93.0 %	104 786	5563	51.0 %	563	43.7 %
≥60s	30	8 295	75.2 %	93.4 %	427 846	44562	60.9 %	2 704	53.0 %
All	101	4 774	81.9 %	93.1 %	200 744	17146	53.8 %	1 199	46.2 %
EVRPTW-MP									
25	26	1 495	85.3 %	91.0 %	32 614	1 252	51.4 %	456	46.7 %
50	38	3 775	82.5 %	93.2 %	140 066	6 143	55.4 %	904	48.7 %
100	23	9 737	73.0 %	94.6 %	507 074	49 571	60.5 %	2 015	49.2 %
<60s	58	3 262	85.2 %	92.7 %	108 873	3 521	52.4 %	646	46.1 %
≥60s	29	7 486	72.4 %	93.4 %	397 190	41 446	62.1 %	1 899	52.7 %
All	87	4 670	80.7 %	92.9 %	204 979	16 162	55.4 %	1 064	48.2 %

The two last columns of Table 2 give further details about TASF at the point when branching would start (denoted by 'after LP+Cut'). The additional valid inequalities added up to this point lead to a tighter lower bound that allows an even more effective variable fixing. Note that the entries '$\lvert {}^2A\rvert$' and 'Fixed' must be interpreted with care: We see that the absolute number of two-arc sequences to consider (i.e., '$\lvert {}^2\hat{A}\rvert$' at 'LP+Cut') is often reduced to less than 10 % compared to 'LP'. However, the entries in column 'Fixed' ('after LP+Cut') describe which percentage of two-arc sequences are fixed by TASF but has not been fixed by SAF up to this point. As the ground set changes, entries 'Fixed' (in the third last and last columns) cannot be directly compared.

Example 1 *We briefly explain the two entries 'Fixed' with the values in the first row of Table 2. To simplify the description, we argue as if these values were for a single 25-customer VRPTW instance.*

This fictive instance has 531 feasible arcs, from which one can construct 9075 feasible two-arc sequences. When the first linear relaxation is solved, SAF is applied for the first time, and 75.5 % of the arcs (401 arcs) can be fixed to 0 so that 130 arcs remain. From these 130 arcs one can construct 456 two-arc sequences. Of these 456 two-arc sequences, 37.8 % can be fixed so that 284 remain. Then, cut generation is followed by SAF which reduces the arc set again and also the set of two-arc sequences. Now TASF can even further reduce the latter set. After iteratively adding valid inequalities, applying SAF and TASF, and doing column generation, the process ends if no more valid inequalities and columns are added. The final set of two-arc sequences comprises 220 arc pairs in our example. This reduction by $236 = 456 - 220$ *two-arc sequences results from both SAF as well as TASF. 38.4 % of these 236 fixed two-arc sequences were fixed to 0 via TASF over the iterations of TASF.*

The differences of the percentage of fixed arcs and arc sequences among the different groups are small. In particular, differences between the four EVRPTW variants are not noteworthy. Therefore, the following experiments will aggregate over these subgroups. Significant is however that percentages are smaller for the groups <60s compared to ≥60s, respectively.

4.2 Comparison of number of labels and dominance tests

In the second series of experiments, we compare the behavior of the SPPRC labeling algorithms when using SAF only ('Standard': no impact on the labeling method) and using TASF ('with TASF': modified label extension and dominance). Two concurring effects are analyzed in more detail: On the one hand, the more restrictive label extension rule in TASF reduces the number of created labels (see Section 2.2). On the other hand, the weaker dominance compared to standard labeling (see Rules 1 and 3, we use the latter and more powerful partial dominance) increases the number of labels that survive a dominance test and may create more labels. It is therefore not clear in advance which type of SPPRC labeling creates more labels, performs more dominance tests, and is faster at the end. This is empirically analyzed now.

A fair comparison of two different SPPRC labeling algorithms is delicate, because already one different generated route from one of the algorithms probably yields a completely different overall trajectory. In order to exclude such effects that disguise the real behavior, we have implemented a special version of BPC in which the same dual solution is always passed to both SPPRC labeling algorithms, i.e., to the standard labeling algorithm and the labeling algorithm that incorporates the TASF. We do this only in column-generation iterations where an exact solution must be determined on the complete pricing network (note that pricing heuristics are not necessarily affected by TASF because they can, as heuristics, rely on the stricter label extension and on the stronger original dominance rule). In the exact pricing iterations, however, both labeling algorithms show their true nature. As we apply them to completely identical pricing networks, they can be compared well. Only the routes generated by the one that respects TASF are added to the RMP in the iteration.

Table 3 compares the number of created labels ('Labels Created') and the number of dominance tests that were performed ('Dominance Tests') for the two labeling algorithms. The statistic considers only SPPRCs solved after the first time that fixing has been performed. The absolute numbers (in the third to fifth columns) are average values over the SPPRCs solved and (E)VRPTW instances of the respective group. Again, averages are arithmetic means for absolute numbers and geometric means for ratios. For two EVRPTW-MF instances, no SRIs or CCs are violated at the root node. As a consequence, no column-generation iterations with TASF occur before branching so that their ratios are 0 making it unreasonable to consider them in the geometric mean. Therefore, only 377 (instead of 379) EVRPTW instances are considered in Table 3.

One result is very obvious: the absolute number of exact pricing iterations, the number of created labels, and the number of dominance tests increase strongly with the size of the instance.

Table 3: Number of labels and dominance tests

			Standard Number of ($\cdot 10^3$)		Labeling with TASF Ratio with TASF/Standard	
Group	#Inst	Number of Exact Iter.	Labels Created	Dominance Tests	Labels Created	Dominance Tests
VRPTW						
25	16	7.3	1.2	19.9	0.52	0.31
50	28	29.6	74.7	49 798.5	0.51	0.40
100	36	67.0	348.3	125 934.2	0.54	0.58
<60s	57	25.1	16.6	573.0	0.54	0.43
≥60s	23	83.7	595.7	256 332.4	0.48	0.49
<3600s	75	39.4	100.6	9 248.8	0.53	0.46
≥3600s	5	81.0	1 420.4	1 046 930.1	0.41	0.30
All	80	42.0	183.1	74 103.8	0.52	0.45
EVRPTW						
25	115	8.6	3.8	140.1	0.47	0.30
50	162	24.6	68.0	22 126.1	0.39	0.28
100	†100	61.1	287.5	68 560.8	0.46	0.45
<60s	272	15.8	14.1	616.6	0.44	0.31
≥60s	†105	64.8	346.4	97 989.7	0.41	0.37
<3600s	†359	26.7	77.0	15 614.3	0.43	0.32
≥3600s	18	82.8	697.6	269 506.0	0.50	0.51
All	†377	29.4	106.7	27 736.4	0.43	0.32

†: For the two EVRPTW-MF instances `C201_100` and `C202_100` no violated inequalities are added at the root node.

The most striking result, however, is the comparison on the basis of the ratios. Indeed, the ratio for the number of created labels is always between 0.39 and 0.54 showing that, for TASF, its weaker dominance is clearly overcompensated by the reduced number of feasible label extensions. While the comparison over subgroups (25, 50, 100 or <60s, ≥60s or <3600s, ≥3600s) is not really insightful, one can clearly see that ratios are consistently smaller for the EVRPTW compared to the VRPTW. The more difficult VRP variant benefits more.

Regarding the number of dominance tests, the ratios are even more in favor of the labeling with TASF, as they vary between 0.28 and 0.58 with smaller totals. Note that dominance tests are performed much more often than labels are created (the reader may compare the numbers in the fourth and fifth columns). Therefore, smaller ratios for dominance tests than ratios for created labels are pointing to a potentially high gain that may result from labeling with TASF. Nevertheless, the final proof of usefulness by analyzing overall BPC computation times remains to be done, see Section 4.4.

4.3 Impact on lower bound and search tree

In general, variable fixing in BPC algorithms may have a positive impact on the lower bounds that are computed all along the branch-and-bound tree. Already at the root node of this tree, the addition of valid inequalities in combination with variable fixing can lead to improved lower bounds. The reason is that an iterated application of a variable-fixing algorithm, each time with another dual solution and a possibly improved dual bound, can yield different fixations that are all valid. The use of an additional and improved fixation technique such as TASF therefore produces a superset of fixed variables and is hence never inferior to a standard fixation technique such as SAF.

If variables that could be fixed are nevertheless kept free, then subsequent linear relaxations of the master program can comprise some of these variables because they may become attractive again for the simplex algorithm. Therefore, these otherwise fixed variables may become part of the basis

deeper in the tree. Even branching on these variables may happen so that on average one can also expect slightly smaller branch-and-bound trees when applying (improved) variable-fixing techniques. Likewise, one can expect that less cuts are needed in comparison.

We analyze and quantify these effects for the VRPTW and EVRPTW. We have implemented specialized BPC algorithms with SAF and TASF. As in Section 4.2, also here the BPC algorithms are terminated before branching starts with the aim of producing better comparable results. For the same reason, we change the separation strategy for violated inequalities: No limit is imposed on the overall number of SRIs and CCs. Moreover, the minimum violation ε_{cut} is set to a tiny value of 0.0001.

Table 4 shows how much TASF improves the root lower bounds and reduces the number of separated valid inequalities compared to SAF. The columns '#Inst' and '#Non-Opt' show that the majority of the instances is solved without branching when we impose no limits on the cut generation. With this setup, only 17 of the 80 VRPTW instances require branching and, likewise, 90 of the 375 EVRPTW instances. TASF improve the SAF lower bound at 'LP+Cut*' (with the '*' we distinguish between the extensive use of cutting planes here and the end of the normal cut-and-column generation denoted by 'LP+Cut') for 5 of the 17 VRPTW instances (29 %) and for 28 of the 90 (31 %) EVRPTW instances (see column '#Better LB'). For these instances, we computed the percentage of the gap $(opt - LB)/opt$ that was closed by the improved bound resulting from TASF, i.e., $100 \cdot (LB_{TASF} - LB)/(opt - LB)$ where LB and LB_{TASF} are the lower bounds at the point LP+Cut* when only using SAF and when complementing it with TASF, respectively. We display the minimal, average, and maximal values in the section under head 'Gap closed (%)'. With a sample of only five elements, the respective gaps for VRPTW cannot reveal significant insights. However, for the EVRPTW we see that on average 17 % of the gap is closed by TASF. Finally, results under head '#Inst with more Cuts' are not really clear cut. On average, the TASF does not reduce the number of valid inequalities at the root for the 80 VRPTW instances, but it does it significantly for the EVRPTW.

Table 4: Impact of TASF on lower bounds, optimality gap, and number of separated SRIs

		after LP+Cut*						
			with TASF	Gap closed (%)			#Inst with more Cuts	
Group	#Inst	#Non-Opt	#Better LB	min	avg	max	SAF	TASF
VRPTW	80	17	5	0.17	3.21	100.00	31	26
EVRPTW	‡375	90	28	0.07	17.34	100.00	166	84

‡: For the two EVRPTW-SF instances `RC107_100` and `C205_100` and the two EVRPTW-MF instances `C201_100` and `C205_100` the column-generation process after adding violated inequalities did not terminate within the time limit of 10 hours.

4.4 Overall integer results

Next, we provide the final comparison of the two BPC algorithms, the first one using SAF only and the second one using it with subsequent TASF. Table 5 provides the average computation time ('Time [s]'), the average number of separated valid inequalities ('#Cuts'), and the average number of branch-and-bound nodes that are solved per group, where groups are again built by instance size (25, 50, 100), and computation time (<60s, ≥60s and <3600s, ≥3600s). The table section headed 'SAF' shows absolute numbers for the first BPC algorithm, while the section headed 'TASF' reports ratios of the respective numbers for TASF relative to SAF.

The BPC algorithm with SAF solves all of the $80 + 101 + 90 + 101 + 87 = 459$ instances except for four EVRPTW-MF instances (counted with the overall computation time limit of 36 000 seconds per instance), while the BPC with TASF solves all without exceptions. As one would expect, computation times increase fast with instance size, and the most difficult EVRPTW variants are those that allow multiple recharges (MF and MP). Even with the valid inequalities, these instances often require massive branching (e.g., the most difficult instances from groups ≥3600s can only be solved after evaluating a few hundred branch-and-bound nodes).

Table 5: Overall integer and BPC results comparing SAF only and SAF combined with TASF

		SAF Avg. Time and Number of			TASF/SAF Ratios with TASF/SAF		
Group	#Inst	Time [s]	#Cuts	#B&B	Time	#Cuts	#B&B
VRPTW							
25	16	0.3	20.4	1.0	1.00	0.94	1.00
50	28	606.8	83.0	5.6	0.74	1.01	0.96
100	36	1182.8	165.3	21.9	0.74	1.02	0.92
<60s	57	8.2	72.3	3.0	0.92	0.98	0.96
≥60s	23	2569.8	194.9	34.4	0.54	1.04	0.93
<3600s	75	151.8	104.9	12.7	0.85	1.00	0.95
≥3600s	5	9637.5	147.6	2.6	0.24	0.99	0.93
All	80	744.7	107.5	12.1	0.79	1.00	0.95
EVRPTW-SF							
25	31	0.8	22.0	1.2	1.02	0.98	0.97
50	41	263.0	79.1	14.4	0.82	0.92	0.85
100	29	1288.7	189.2	77.2	0.71	1.06	0.59
<60s	75	7.6	56.3	11.7	0.94	0.96	0.91
≥60s	26	1831.1	199.5	76.4	0.61	1.05	0.56
<3600s	97	235.6	82.7	25.9	0.88	0.98	0.83
≥3600s	4	6331.7	347.5	89.0	0.27	0.98	0.37
All	101	477.0	93.2	28.4	0.84	0.98	0.80
EVRPTW-SP							
25	28	1.1	24.3	1.0	1.02	1.00	1.00
50	40	203.1	76.6	9.8	0.85	0.92	0.93
100	22	951.2	199.8	95.3	0.78	0.96	1.03
<60s	68	7.1	60.3	2.4	0.95	0.95	0.99
≥60s	22	1299.9	183.6	107.0	0.69	0.97	0.91
<3600s	87	129.2	83.7	6.5	0.89	0.95	0.98
≥3600s	3	5948.3	284.3	650.3	0.67	1.04	0.87
All	90	323.1	90.4	27.9	0.88	0.96	0.97
EVRPTW-MF							
25	30	3.7	26.8	1.2	0.94	0.96	1.00
50	43	936.7	67.5	18.1	0.87	0.96	0.96
100	✠28	5044.1	♮185.5	85.6	0.51	♮0.86	0.72
<60s	71	11.0	48.7	11.3	0.94	0.95	0.95
≥60s	✠30	6028.1	♮181.3	80.4	0.47	♮0.88	0.79
<3600s	95	176.4	♮84.1	20.8	0.88	♮0.96	0.90
≥3600s	✠6	27477.3	118.8	205.8	0.09	0.61	0.83
All	✠101	1798.3	♮86.2	31.8	0.77	♮0.93	0.90
EVRPTW-MP							
25	26	2.5	30.5	1.1	0.96	0.93	1.00
50	38	268.1	77.3	48.5	0.76	0.99	0.79
100	23	1608.2	195.0	173.8	0.65	0.97	0.75
<60s	58	10.5	47.6	4.4	0.91	0.95	0.92
≥60s	29	1608.0	188.1	193.5	0.59	1.00	0.69
<3600s	81	158.8	85.0	26.9	0.83	0.97	0.84
≥3600s	6	5729.9	222.3	615.3	0.36	0.95	0.71
All	87	543.0	94.4	67.5	0.78	0.96	0.83

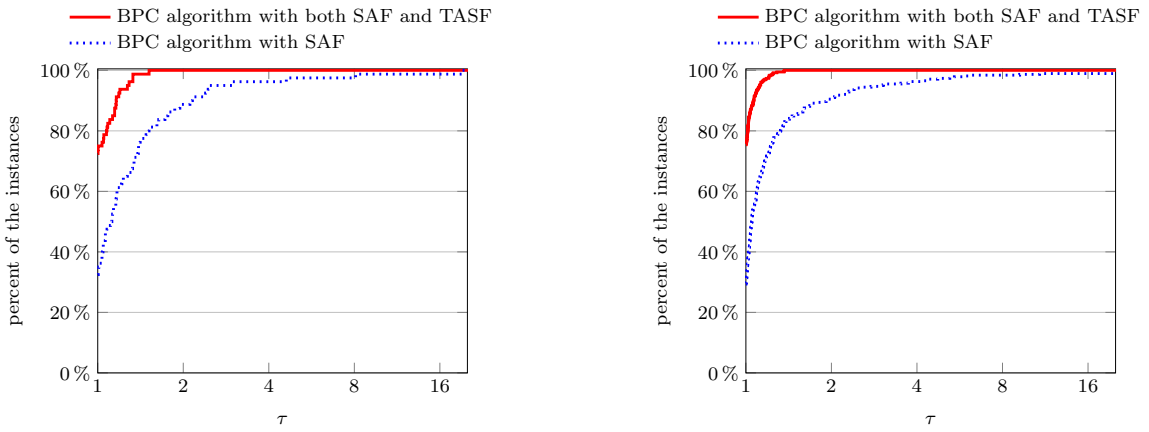
✠: Four EVRPTW-MF instances not solved within the time limit of 10 hours (computation time for each of them counted as 36,000 seconds).
♮: No violated inequalities for EVRPTW-MF instances C201_100 and C202_100. Averages are taken over the remaining 26, 28, 93, and 99 instances, respectively.

Over all instances, the use of TASF reduces the BPC computation time compared to SAF by approximately 19 %. This is a fair but certainly not striking result. Restricting the analysis to instances that require at least one minute of computation time (≥60s), the time ratio goes down to 0.57, i.e., a reduction by 43 %. Similarly, for the most difficult instances (≥3600s), the ratio further decreases

to 0.26, i.e., a reduction by 74 %. In other words, for the most difficult instances, the BPC algorithm with TASF is by approximately a factor 4 faster than the one using only SAF.

What is confirmed again is that TASF reduces the number of valid inequalities and the search tree size (all total ratios TASF/SAF for #Cuts and #B&B are less than or equal to 1.0 with a very few and small exceptions). This is in line with the results of the previous section.

We provide a final comparison with the performance profiles shown in Figure 1. Following Dolan and Moré (2002), given two algorithms $\mathcal{A} = \{\text{BPC with SAF}, \text{BPC with SAF and TASF}\}$ (or any set $\mathcal{A}$ of algorithms applied to the same set of instances), the curve $\rho_A(\tau)$ of the algorithm $A \in \mathcal{A}$ is the fraction of instances that algorithm A can solve within a factor τ of the fastest algorithm (unsolved instances are taken into account with infinite run time). In particular, $\rho_A(1)$ is the percentage of instances on which A is a fastest algorithm and the value $1 - \rho_A(\infty)$ is the percentage of instances not solved by A. Note that τ is displayed in logarithmic scale. One can see that for the VRPTW and the EVRPTW, the BPC algorithm with both SAF and TASF is the fastest in 72% and 75% of the cases and it never needs more than factor 1.52 and 1.36 longer compared to the BPC algorithm with SAF only, respectively. Hence, the BPC algorithm with TASF is clearly superior.



(a) VRPTW, 80 instances **(b) EVRPTW, 379 instances**

Figure 1: Performance profiles $\rho_A(\tau)$ for algorithms $A \in \mathcal{A} = \{$BPC with SAF, BPC with SAF and TASF$\}$

4.5 EVRPTW results

While all optimal solutions for Solomon's VRPTW benchmark are known, the EVRPTW benchmark contains several open problem instances. With our results we can now fill the gaps for 90 of the 194 unsolved instances. Table 6 provides a comparison with the results from the recent article on the EVRPTW by Desaulniers et al. (2016). The results are grouped by EVRPTW variant (SF, SP, MF, MP), instance size (25, 50, 100), and also by series (series 1: R1, C1, RC1 and series 2: R2, C2, RC2). The sections headed 'DEIS2016' refer to the results in (Desaulniers et al. 2016), and we show the number of instances solved to optimality ('#Opt') and the average computation time in seconds ('Time [s]', unsolved instances count with 0 seconds as done in the study of Desaulniers et al.). Similarly, we present the corresponding numbers for our BPC algorithm in the columns headed 'BPC with TASF'. Note that Desaulniers et al. used a time limit of 1 hour for their experiments. Therefore, differing from all our other tables, Table 6 reports results of the BPC algorithm with TASF obtained within a shorter time limit of 1 hour. In addition to the number of optima, we indicate the number of instances solved to optimality for the first time ('new') and the number of instances where Desaulniers et al. were able to compute an optimal solution while our new BPC algorithm with TASF fails to do ('fail').

In summary, for the 672 EVRPTW instances, our BPC algorithm with TASF provides 76 new optima within 1 hour of computation time and fails in comparison to the first BPC algorithm of Desaulniers et al. in only three cases. What is even more important is that computation times are

Table 6: Integer results for EVRPTW comparing the BPC algorithm of Desaulniers et al. (2016) and the new BPC algorithm with TASF

	R1, C1, RC1					R2, C2, RC2				
		DEIS2016		BPC with TASF			DEIS2016		BPC with TASF	
Group	#Inst	#Opt	Time [s]	#Opt/new/fail	Time [s]	#Inst	#Opt	Time [s]	#Opt/new/fail	Time [s]
EVRPTW-SF										
25	29	29	3.0	29/±0/±0	0.6	27	27	60.3	27/±0/±0	2.2
50	29	27	169.8	29/+2/±0	36.5	27	20	360.4	26/+6/±0	29.4
100	29	11	356.6	18/+7/±0	38.3	27	3	1630.7	11/+8/±0	56.3
All	87	67	128.2	76/+9/±0	21.2	81	50	274.6	64/+14/±0	16.4
EVRPTW-SP										
25	29	29	2.8	29/±0/±0	0.8	27	27	30.2	27/±0/±0	1.3
50	29	27	112.9	28/+1/±0	15.4	27	23	514.0	26/+3/±0	28.3
100	29	12	657.5	14/+2/±0	44.7	27	10	593.5	14/+5/−1	205.3
All	87	68	162.1	71/+3/±0	14.3	81	60	309.5	67/+8/−1	43.0
EVRPTW-MF										
25	29	29	27.5	29/±0/±0	1.6	27	27	164.2	27/±0/±0	9.1
50	29	27	60.5	28/+1/±0	14.0	27	19	541.0	26/+7/±0	71.4
100	29	11	595.2	17/+6/±0	42.7	27	2	1797.6	10/+8/±0	128.6
All	87	67	134.0	74/+7/±0	13.3	81	48	381.4	63/+15/±0	38.8
EVRPTW-MP										
25	29	29	7.1	29/±0/±0	1.8	27	26	48.0	27/+1/±0	2.6
50	29	26	380.6	28/+2/±0	62.3	27	19	651.2	26/+7/±0	79.4
100	29	10	419.6	15/+5/±0	50.8	27	8	840.0	11/+5/−2	149.3
All	87	65	220.0	72/+7/±0	33.5	81	53	383.8	64/+13/−2	48.5
All	348	267	160.6	293/+26/±0	20.5	324	211	336.3	258/+50/-3	37.0

significantly reduced (factor ≥ 8), where the different hardware may explain part of the improvement but certainly the refined methodology is responsible for the larger share.

Overall, optima are known for 568 of the 672 EVRPTW instances: 551 (=293+258) are solved by BPC with TASF in 1 hour, 3 additional instances have been solved by Desaulniers et al., 8 additional instances are solved by our BPC algorithm with TASF with the actual runtime of 2 hours that we used, and for another 6 instances proven optima have been found during our pretests. This means that 104 instances remain open (thereof 4 with 50 customers).

5 Conclusions

In this paper, we have extended the popular variable fixing technique used in branch-price-and-cut algorithms for solving vehicle routing problems to routes containing sequences of two arcs. For some of the most common vehicle routing problems, we have shown how the route generation labeling algorithm can be modified to prevent the generation of routes containing forbidden two-arc sequences. Our computational experiments on benchmark instances of the VRPTW and four variants of the EVRPTW have demonstrated the effectiveness of this extended technique: besides a few exceptions, it always reduces the overall computational times. More importantly, the speedup generally increases with the difficulty of solving the problem instance: average speedups of a factor of 4 have been achieved for groups of instances that were solved in more than one hour without this new variable fixing strategy.

Potential future research on this topic includes the following: First, investigating how variable fixing can be extended to two-arc sequences when the labeling algorithm is based on a PDED dominance rule would increase the set of vehicle routing problems that could benefit from this new technique. Second, it would be interesting to assess the impact of combining two-arc sequence variable fixing with route enumeration procedures that have been integrated into some recent branch-price-and-cut algorithms for vehicle routing (for details, see Costa et al. 2019). Finally, one could also explore variable fixing in branch-and-price algorithms where the pricing problems are not SPPRCs.

6 Appendix

6.1 Instances used in the main experimental study

Table 7 provides an overview of the instances used in our experimental study. For each instance it specifies whether it is included in our main experiments (•), it is excluded because the solution of the linear relaxation is already integer optimal (LP*), or it is excluded because our BPC algorithm with TASF could not solve it within two hours (TL).

Table 7: List of instances used in our experimental study

	VRPTW			EVRPTW-SF			EVRPTW-SP			EVRPTW-MF			EVRPTW-MP		
Inst	25	50	100	25	50	100	25	50	100	25	50	100	25	50	100
C101	LP*	LP*	LP*	•	•	LP*	LP*	LP*	•	•	•	LP*	LP*	LP*	LP*
C102	LP*	LP*	LP*	•	LP*	•	LP*	LP*	•	LP*	LP*	•	LP*	LP*	TL
C103	LP*	LP*	LP*	LP*	•	•	LP*	•	•	LP*	•	TL	•	•	TL
C104	LP*	LP*	LP*	LP*	•	TL	LP*	LP*	TL	•	•	TL	LP*	LP*	TL
C105	LP*	LP*	LP*	•	•	•	LP*	•	•	•	•	•	LP*	LP*	•
C106	LP*	LP*	LP*	•	•	•	•	•	•	•	•	•	•	•	•
C107	LP*	LP*	LP*	LP*	•	•	•	•	•	•	•	•	•	LP*	TL
C108	LP*	LP*	LP*	•	LP*	•	LP*	•	•	•	•	TL	LP*	•	TL
C109	LP*	LP*	LP*	•	•	TL	•	•	TL	•	•	TL	•	•	TL
R101	LP*	•	•	•	LP*	•	LP*	LP*	•	LP*	•	•	LP*	•	•
R102	•	LP*	LP*	•	•	•	•	•	•	•	•	•	LP*	•	•
R103	LP*	•	•	LP*	•	•	LP*	•	•	•	•	•	•	•	•
R104	LP*	•	•	•	•	TL	•	•	TL	•	•	TL	•	TL	TL
R105	LP*	•	•	•	•	•	•	•	•	•	•	•	•	•	•
R106	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
R107	LP*	•	•	•	•	•	•	•	TL	•	•	•	•	•	•
R108	•	•	•	•	•	•	•	•	TL	LP*	•	TL	LP*	•	TL
R109	LP*	•	•	•	•	•	•	•	TL	•	•	TL	•	•	•
R110	•	•	•	•	•	TL	LP*	•	TL	•	•	TL	•	•	TL
R111	•	•	•	•	•	TL	•	•	•	•	•	TL	•	•	TL
R112	•	•	•	•	•	TL	•	•	TL	•	•	•	•	•	TL
RC101	•	•	•	•	•	TL	•	•	TL	•	•	TL	•	•	•
RC102	LP*	•	•	•	•	TL	•	•	•	•	•	•	•	LP*	•
RC103	LP*	•	•	•	•	•	•	•	•	•	LP*	•	•	•	•
RC104	LP*	LP*	•	•	•	•	•	•	TL	•	•	TL	•	•	•
RC105	LP*	•	•	LP*	•	TL	LP*	•	TL	LP*	•	•	LP*	•	•
RC106	LP*	•	•	LP*	•	•	•	•	TL	LP*	•	•	LP*	•	•
RC107	LP*	•	•	LP*	•	•	LP*	•	TL	LP*	•	•	LP*	•	TL
RC108	LP*	•	•	•	•	TL	•	•	TL	•	•	TL	•	•	TL
C201	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	TL	LP*	LP*	•	LP*	LP*	LP*
C202	LP*	LP*	LP*	LP*	LP*	TL	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	TL
C203	LP*	LP*	LP*	LP*	LP*	TL	LP*	LP*	LP*	LP*	LP*	TL	LP*	LP*	LP*
C204	LP*	LP*	LP*	LP*	LP*	TL	LP*	LP*	TL	LP*	•	TL	LP*	•	TL
C205	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	TL
C206	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	LP*	LP*	LP*	•	LP*	LP*	LP*
C207	LP*	LP*	LP*	LP*	•	•	LP*	•	LP*	LP*	•	•	LP*	•	LP*
C208	LP*	LP*	LP*	•	LP*	•	•	LP*	LP*	LP*	LP*	•	LP*	LP*	LP*
R201	•	LP*	•	LP*	LP*	•	LP*	LP*	•	LP*	LP*	•	LP*	LP*	•
R202	LP*	LP*	•	LP*	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL
R203	LP*	•	•	•	•	TL	•	•	TL	•	•	TL	•	•	TL
R204	•	•	•	LP*	TL	TL	LP*	TL	TL	LP*	TL	TL	LP*	TL	TL
R205	•	•	•	LP*	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL
R206	•	•	•	•	•	TL	•	•	•	•	•	TL	•	•	•
R207	•	•	•	LP*	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL
R208	•	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL
R209	•	•	•	•	•	TL	•	•	TL	•	•	TL	•	•	TL
R210	•	•	•	LP*	•	TL	LP*	•	TL	LP*	•	TL	LP*	•	TL
R211	•	•	TL	•	•	TL	•	•	LP*	•	•	TL	•	•	TL
RC201	LP*	LP*	•	LP*	LP*	•	LP*	LP*	•	LP*	LP*	•	LP*	LP*	•
RC202	LP*	LP*	•	LP*	•	•	LP*	•	•	LP*	•	•	LP*	•	•
RC203	LP*	LP*	•	•	•	TL	•	•	•	•	•	TL	•	•	•
RC204	LP*	LP*	•	•	LP*	TL	•	LP*	TL	•	LP*	TL	•	LP*	TL
RC205	LP*	LP*	•	•	•	•	•	•	•	•	•	•	•	•	•
RC206	LP*	LP*	•	LP*	•	•	LP*	•	•	LP*	•	•	LP*	•	•
RC207	LP*	LP*	•	•	LP*	•	•	LP*	•	•	LP*	•	•	LP*	•
RC208	LP*	•	•	LP*	•	TL	•	•	TL	LP*	•	TL	LP*	•	TL

6.2 Detailed results for all EVRPTW instances

Tables 8–17 provide, on an instance basis, detailed results for our BPC algorithm with TASF and a comparison with the results of Desaulniers et al. (2016) denoted by DEIS2016. The columns of the tables have the following meaning:

- Inst: The name of the instance
- BKS: The best known solution; proven optimal solutions are marked with an asterisk *
- Opt 1h: An asterisk * indicates if the respective algorithm could solve the instance to proven optimality within 1 hour of computation time
- Time [s]: The computation time in seconds; TL1 (TL2) indicates the time limit of 1 hour (2 hours)
- LB_{LP}: The linear relaxation lower bound
- LB_{tree}: The lower bound when reaching the time limit of 2 hours
- % Gap: The percentage optimality gap at the time limit of 2 hours
- #B&B: The number of solved branch-and-bound nodes
- #CCs: The number of capacity cuts added
- #SRIs: The number of subset-row inequalities added

Table 8: Detailed results for 25-customer EVRPTW-SF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*642.58	*	0.2	*	640.34	642.58	0.1	0.00	1	0	5
C102	*543.78	*	0.4	*	542.26	543.78	0.3	0.00	1	0	14
C103	*345.54	*	0.9	*	345.54	345.54	1.0	0.00	1	0	0
C104	*459.74	*	1.5	*	459.74	459.74	0.8	0.00	1	0	0
C105	*541.49	*	0.3	*	541.11	541.49	0.1	0.00	1	0	4
C106	*564.60	*	0.4	*	556.16	564.60	0.2	0.00	1	0	3
C107	*505.86	*	0.1	*	505.86	505.86	0.1	0.00	1	0	0
C108	*531.64	*	1.5	*	524.48	531.64	0.5	0.00	1	0	62
C109	*503.87	*	3.0	*	494.09	503.87	0.6	0.00	1	0	26
R101	*672.41	*	0.5	*	671.80	672.41	0.1	0.00	1	0	3
R102	*455.58	*	2.9	*	454.26	455.58	0.1	0.00	1	0	7
R103	*499.23	*	0.6	*	499.23	499.23	0.2	0.00	1	0	0
R104	*352.16	*	6.6	*	348.76	352.16	1.5	0.00	1	0	27
R105	*621.89	*	2.8	*	617.38	621.89	0.2	0.00	1	0	7
R106	*480.22	*	1.7	*	474.23	480.22	0.4	0.00	1	0	3
R107	*416.51	*	1.4	*	415.63	416.51	0.5	0.00	1	0	6
R108	*456.50	*	19.4	*	446.81	456.50	1.4	0.00	1	0	105
R109	*467.68	*	7.9	*	461.87	467.68	0.2	0.00	1	0	10
R110	*424.38	*	3.3	*	414.17	424.38	1.3	0.00	3	0	38
R111	*383.04	*	6.7	*	380.27	383.04	1.8	0.00	1	0	20
R112	*397.42	*	13.4	*	385.02	397.42	1.2	0.00	1	0	47
RC101	*763.35	*	0.5	*	752.69	763.35	0.1	0.00	1	0	2
RC102	*675.66	*	1.1	*	669.03	675.66	0.3	0.00	1	0	9
RC103	*563.83	*	1.2	*	556.32	563.83	0.4	0.00	1	0	30
RC104	*519.20	*	2.0	*	514.02	519.20	1.5	0.00	1	0	6
RC105	*594.69	*	0.2	*	594.69	594.69	0.1	0.00	1	0	0
RC106	*557.23	*	0.8	*	557.23	557.23	0.1	0.00	1	0	0
RC107	*502.22	*	0.6	*	502.22	502.22	0.5	0.00	1	0	0
RC108	*479.73	*	4.0	*	474.90	479.73	1.2	0.00	1	0	18
C201	*416.37	*	0.5	*	416.37	416.37	0.4	0.00	1	0	0
C202	*404.57	*	1.1	*	404.57	404.57	0.6	0.00	1	0	0
C203	*391.72	*	8.3	*	391.72	391.72	1.9	0.00	1	0	0
C204	*362.93	*	3.9	*	362.93	362.93	1.0	0.00	1	0	0
C205	*409.05	*	0.7	*	409.05	409.05	0.4	0.00	1	0	0
C206	*389.19	*	1.0	*	389.19	389.19	0.5	0.00	1	0	0
C207	*400.37	*	1.2	*	400.37	400.37	0.6	0.00	1	0	0
C208	*419.19	*	31.2	*	415.13	419.19	1.1	0.00	1	0	22
R201	*374.28	*	5.2	*	374.28	374.28	0.1	0.00	1	0	0
R202	*375.30	*	17.9	*	375.30	375.30	0.6	0.00	1	0	0
R203	*382.77	*	32.6	*	380.47	382.77	0.9	0.00	1	0	22
R204	*315.52	*	90.7	*	315.52	315.52	2.3	0.00	1	0	0
R205	*374.00	*	6.5	*	374.00	374.00	0.2	0.00	1	0	0
R206	*424.51	*	32.8	*	423.22	424.51	0.5	0.00	1	0	12
R207	*330.96	*	194.3	*	330.96	330.96	3.0	0.00	1	0	0
R208	*315.59	*	889.7	*	315.59	315.59	32.4	0.00	1	0	0
R209	*422.39	*	10.9	*	417.87	422.39	0.6	0.00	1	0	51
R210	*315.27	*	53.0	*	315.27	315.27	1.0	0.00	1	0	0
R211	*383.81	*	90.5	*	383.27	383.81	2.9	0.00	1	0	24
RC201	*594.89	*	3.8	*	594.89	594.89	0.5	0.00	1	0	0
RC202	*540.87	*	7.3	*	540.87	540.87	0.2	0.00	1	0	0
RC203	*443.28	*	77.2	*	437.32	443.28	2.1	0.00	1	0	32
RC204	*445.51	*	16.8	*	444.66	445.51	1.5	0.00	1	0	6
RC205	*536.35	*	11.4	*	532.84	536.35	0.4	0.00	1	0	33
RC206	*589.19	*	2.5	*	589.19	589.19	0.3	0.00	1	0	0
RC207	*485.88	*	17.2	*	483.18	485.88	0.9	0.00	1	0	10
RC208	*440.24	*	19.7	*	440.24	440.24	2.9	0.00	1	0	0

Table 9: Detailed results for 50-customer EVRPTW-SF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*783.90	*	0.6	*	783.59	783.90	0.5	0.00	1	0	3
C102	*828.40	*	1.2	*	828.40	828.40	1.2	0.00	1	0	0
C103	*668.33	*	8.9	*	658.96	668.33	5.9	0.00	1	0	55
C104	*598.87		TL1	*	584.33	598.87	707.4	0.00	3	47	260
C105	*757.11	*	0.8	*	754.49	757.11	0.5	0.00	1	0	8
C106	*801.27	*	0.8	*	796.04	801.27	0.5	0.00	1	0	8
C107	*792.96	*	54.6	*	755.37	792.96	39.7	0.00	421	0	37
C108	*733.83	*	0.7	*	733.83	733.83	0.7	0.00	1	0	0
C109	*690.19	*	9.8	*	676.93	690.19	3.0	0.00	1	42	42
R101	*947.32	*	1.8	*	947.32	947.32	0.1	0.00	1	0	0
R102	*909.72	*	11.0	*	894.04	909.72	1.3	0.00	3	0	63
R103	*812.90	*	123.5	*	798.51	812.90	3.1	0.00	13	0	136
R104	*624.14		TL1	*	604.28	624.14	1074.7	0.00	3	0	231
R105	*842.75	*	42.9	*	835.85	842.75	0.5	0.00	1	0	29
R106	*811.47	*	84.6	*	793.77	811.47	2.8	0.00	25	0	32
R107	*739.15	*	134.1	*	723.91	739.15	6.7	0.00	1	0	136
R108	*543.83	*	68.1	*	532.15	543.83	30.1	0.00	1	0	64
R109	*833.80	*	78.6	*	818.21	833.80	3.7	0.00	1	0	182
R110	*734.48	*	84.3	*	722.57	734.48	6.1	0.00	3	0	113
R111	*786.59	*	31.5	*	779.59	786.59	3.8	0.00	1	0	20
R112	*658.69	*	3430.0	*	636.24	658.69	801.0	0.00	3	13	211
RC101	*1094.76	*	7.6	*	1079.71	1094.76	0.5	0.00	1	0	27
RC102	*913.05	*	4.5	*	907.50	913.05	1.0	0.00	1	0	22
RC103	*857.93	*	16.7	*	847.13	857.93	2.5	0.00	1	0	49
RC104	*694.20	*	14.2	*	685.61	694.20	15.9	0.00	1	0	31
RC105	*1009.44	*	5.1	*	1001.44	1009.44	0.8	0.00	1	0	10
RC106	*929.72	*	19.7	*	901.13	929.72	2.6	0.00	1	0	68
RC107	*791.52	*	64.2	*	772.79	791.52	12.8	0.00	1	12	90
RC108	*742.30	*	283.8	*	721.45	742.30	37.5	0.00	5	0	185
C201	*503.26	*	3.4	*	503.26	503.26	1.8	0.00	1	0	0
C202	*523.75	*	7.0	*	523.75	523.75	2.3	0.00	1	0	0
C203	*513.82	*	52.3	*	513.82	513.82	16.9	0.00	1	0	0
C204	*469.56	*	572.7	*	469.56	469.56	241.3	0.00	1	0	0
C205	*513.75	*	5.4	*	513.75	513.75	1.7	0.00	1	0	0
C206	*515.97	*	7.7	*	515.97	515.97	4.3	0.00	1	0	0
C207	*531.13	*	18.8	*	530.57	531.13	4.4	0.00	1	0	6
C208	*528.40	*	9.4	*	528.40	528.40	4.1	0.00	1	0	0
R201	*674.75	*	181.2	*	674.75	674.75	0.9	0.00	1	0	0
R202	*647.17	*	286.9	*	647.08	647.17	4.9	0.00	1	0	6
R203	*571.02	*	2395.8	*	569.98	571.02	13.2	0.00	1	0	22
R204	510.56		TL1		482.04	484.52	TL2	5.10	1	0	40
R205	*606.43	*	333.2	*	604.52	606.43	9.0	0.00	1	0	28
R206	*579.37		TL1	*	573.16	579.37	19.6	0.00	1	0	90
R207	*512.56		TL1	*	510.79	512.56	113.7	0.00	1	0	66
R208	*467.16		TL1	*	463.27	467.16	311.5	0.00	1	0	160
R209	*602.50		TL1	*	594.13	602.50	78.1	0.00	3	0	110
R210	*521.18		TL1	*	517.89	521.18	22.2	0.00	1	0	115
R211	*516.53		TL1	*	510.95	516.53	93.3	0.00	3	0	102
RC201	*812.23	*	21.8	*	812.23	812.23	0.5	0.00	1	0	0
RC202	*799.71	*	60.4	*	786.67	799.71	5.0	0.00	1	0	72
RC203	*670.03	*	497.2	*	666.94	670.03	21.6	0.00	1	0	19
RC204	*610.24	*	1372.8	*	610.24	610.24	84.2	0.00	1	0	0
RC205	*720.66	*	103.4	*	713.33	720.66	2.7	0.00	1	0	43
RC206	*795.20	*	77.4	*	794.38	795.20	2.6	0.00	1	0	10
RC207	*646.78	*	299.1	*	646.78	646.78	31.9	0.00	1	0	0
RC208	*577.33	*	901.3	*	575.41	577.33	135.7	0.00	1	0	10

Table 10: Detailed results for 100-customer EVRPTW-SF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*1059.65	*	2.0	*	1059.65	1059.65	2.7	0.00	1	0	0
C102	*1028.39	*	21.3	*	1025.57	1028.39	14.0	0.00	1	39	21
C103	*1014.77		TL1	*	997.02	1014.77	758.9	0.00	3	225	293
C104	985.47		TL1		925.85	940.64	TL2	4.55	1	4	44
C105	*1039.74	*	16.8	*	1033.72	1039.74	3.3	0.00	1	18	31
C106	*1033.08	*	12.0	*	1028.35	1033.08	6.5	0.00	1	53	54
C107	*1031.45	*	30.2	*	1017.86	1031.45	7.6	0.00	1	240	32
C108	*1024.69	*	150.2	*	1003.77	1024.69	24.8	0.00	3	90	182
C109	1010.46		TL1		959.52	998.68	TL2	1.17	262	114	267
R101	*1643.20	*	350.8	*	1624.80	1643.20	14.8	0.00	197	0	55
R102	*1475.68	*	222.6	*	1469.00	1475.68	5.1	0.00	19	0	119
R103	*1242.25		TL1	*	1217.68	1242.25	3001.5	0.00	645	0	273
R104	1095.49		TL1		1067.33	1072.44	TL2	2.10	1	64	51
R105	*1395.72	*	707.1	*	1369.92	1395.72	35.0	0.00	49	0	233
R106	*1261.96	*	448.4	*	1249.71	1261.96	18.7	0.00	1	0	197
R107	*1167.68	*	1960.6	*	1140.73	1167.68	288.5	0.00	19	0	269
R108	*1025.20		TL1	*	1012.90	1025.20	1500.8	0.00	1	0	194
R109	*1274.85		TL1	*	1249.89	1274.85	2893.6	0.00	637	0	290
R110	1120.48		TL1		1081.06	1104.51	TL2	1.43	61	0	320
R111	*1133.09		TL1		1110.37	1133.00	TL2	0.01	457	0	296
R112	1057.63		TL1		1027.25	1043.12	TL2	1.37	1	0	177
RC101	1738.72		TL1		1679.23	1734.04	TL2	0.27	1701	10	272
RC102	1601.24		TL1		1547.15	1596.05	TL2	0.32	930	62	224
RC103	*1347.77		TL1	*	1311.97	1347.77	269.0	0.00	1	72	264
RC104	*1219.98		TL1	*	1172.83	1219.98	2720.5	0.00	21	78	320
RC105	1546.65		TL1		1487.18	1533.39	TL2	0.86	434	12	230
RC106	*1408.89		TL1	*	1370.89	1408.89	527.0	0.00	131	21	320
RC107	*1258.41		TL1		1206.92	1258.41	3922.8	0.00	69	103	314
RC108	*1253.31		TL1		1192.12	1249.61	TL2	0.30	79	166	232
C201	*651.96		TL1	*	639.24	651.96	373.5	0.00	7	0	40
C202	660.91		TL1		638.64	640.88	TL2	3.03	1	0	52
C203	674.63		TL1		637.44	638.41	TL2	5.37	1	0	20
C204			TL1				TL2		0	0	0
C205	*641.53		TL1	*	634.26	641.53	728.3	0.00	39	0	10
C206	*638.56		TL1	*	633.12	638.56	155.9	0.00	1	0	52
C207	*638.56		TL1	*	633.12	638.56	211.5	0.00	1	0	35
C208	*638.56		TL1	*	633.12	638.56	167.7	0.00	1	0	50
R201	*1098.61	*	3006.9	*	1093.12	1098.61	26.8	0.00	1	0	187
R202	993.15		TL1		980.10	991.83	TL2	0.13	328	0	320
R203	896.11		TL1		853.82	856.23	TL2	4.45	1	0	40
R204	868.79		TL1		705.87	705.87	TL2	18.75	1	0	0
R205	*950.83		TL1		937.01	950.34	TL2	0.05	601	0	297
R206	946.05		TL1		878.69	884.23	TL2	6.54	1	0	51
R207	878.22		TL1		781.88	783.78	TL2	10.75	1	0	20
R208			TL1				TL2		0	0	0
R209	862.85		TL1		834.81	843.00	TL2	2.30	1	0	89
R210	894.96		TL1		814.99	819.07	TL2	8.48	1	0	40
R211	892.42		TL1		742.72	742.72	TL2	16.77	1	0	10
RC201	*1258.27	*	475.0	*	1252.61	1258.27	15.1	0.00	3	0	91
RC202	*1139.29		TL1	*	1129.07	1139.29	472.1	0.00	43	0	314
RC203	1032.46		TL1		948.67	953.83	TL2	7.62	1	0	63
RC204			TL1				TL2		0	0	0
RC205	*1072.08	*	1410.3	*	1069.34	1072.08	127.0	0.00	1	0	54
RC206	*1069.39		TL1	*	1053.23	1069.39	610.3	0.00	17	0	320
RC207	*925.72		TL1	*	918.56	925.72	2766.1	0.00	1	0	55
RC208			TL1				TL2		0	0	0

Table 11: Detailed results for 25-customer EVRPTW-SP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*622.89	*	0.1	*	622.89	622.89	0.1	0.00	1	0	0
C102	*541.30	*	0.4	*	541.30	541.30	0.2	0.00	1	0	0
C103	*345.54	*	1.2	*	345.54	345.54	1.0	0.00	1	0	0
C104	*459.68	*	1.4	*	459.68	459.68	1.3	0.00	1	0	0
C105	*537.06	*	0.2	*	537.06	537.06	0.1	0.00	1	0	0
C106	*557.65	*	0.9	*	549.21	557.65	0.3	0.00	1	0	15
C107	*505.15	*	0.5	*	501.06	505.15	0.2	0.00	1	0	10
C108	*509.15	*	0.3	*	509.15	509.15	0.2	0.00	1	0	0
C109	*493.85	*	7.5	*	485.44	493.85	0.9	0.00	1	47	43
R101	*662.30	*	0.3	*	662.30	662.30	0.1	0.00	1	0	0
R102	*453.50	*	3.3	*	450.84	453.50	0.2	0.00	1	0	8
R103	*498.58	*	0.7	*	498.58	498.58	0.2	0.00	1	0	0
R104	*352.16	*	6.9	*	347.30	352.16	2.9	0.00	1	0	35
R105	*616.38	*	1.8	*	611.94	616.38	0.2	0.00	1	0	9
R106	*480.22	*	2.7	*	473.59	480.22	0.4	0.00	1	0	16
R107	*416.51	*	3.0	*	412.08	416.51	0.9	0.00	1	18	27
R108	*456.50	*	19.8	*	446.81	456.50	1.8	0.00	1	0	97
R109	*462.03	*	4.9	*	460.29	462.03	0.2	0.00	1	0	10
R110	*410.04	*	0.7	*	410.04	410.04	0.6	0.00	1	0	0
R111	*383.04	*	9.1	*	377.74	383.04	3.3	0.00	1	0	33
R112	*385.51	*	3.4	*	380.95	385.51	0.8	0.00	1	0	18
RC101	*763.35	*	2.0	*	751.71	763.35	0.2	0.00	1	0	9
RC102	*675.66	*	1.7	*	668.16	675.66	0.2	0.00	1	0	9
RC103	*552.23	*	2.3	*	548.79	552.23	0.3	0.00	1	0	7
RC104	*519.20	*	2.0	*	514.02	519.20	2.8	0.00	1	0	6
RC105	*594.69	*	0.2	*	594.69	594.69	0.1	0.00	1	0	0
RC106	*557.23	*	1.2	*	556.30	557.23	0.2	0.00	1	0	5
RC107	*502.22	*	0.7	*	502.22	502.22	0.8	0.00	1	0	0
RC108	*471.08	*	2.5	*	469.56	471.08	1.4	0.00	1	0	8
C201	*416.37	*	0.9	*	416.37	416.37	0.4	0.00	1	0	0
C202	*404.57	*	1.3	*	404.57	404.57	0.6	0.00	1	0	0
C203	*391.72	*	6.6	*	391.72	391.72	1.6	0.00	1	0	0
C204	*362.93	*	3.1	*	362.93	362.93	1.0	0.00	1	0	0
C205	*409.05	*	0.8	*	409.05	409.05	0.5	0.00	1	0	0
C206	*389.19	*	0.8	*	389.19	389.19	0.6	0.00	1	0	0
C207	*400.37	*	1.1	*	400.37	400.37	0.5	0.00	1	0	0
C208	*419.19	*	22.2	*	414.18	419.19	1.0	0.00	1	0	34
R201	*374.28	*	5.7	*	374.28	374.28	0.1	0.00	1	0	0
R202	*375.30	*	9.8	*	375.30	375.30	0.4	0.00	1	0	0
R203	*382.77	*	35.3	*	380.47	382.77	1.1	0.00	1	0	19
R204	*315.52	*	52.8	*	315.52	315.52	5.6	0.00	1	0	0
R205	*374.00	*	6.5	*	374.00	374.00	0.2	0.00	1	0	0
R206	*424.51	*	26.3	*	423.22	424.51	0.6	0.00	1	0	12
R207	*330.96	*	72.9	*	330.96	330.96	3.0	0.00	1	0	0
R208	*315.59	*	316.9	*	315.59	315.59	7.0	0.00	1	0	0
R209	*422.39	*	11.0	*	417.87	422.39	0.6	0.00	1	0	53
R210	*315.27	*	23.8	*	315.27	315.27	1.5	0.00	1	0	0
R211	*383.81	*	57.3	*	383.27	383.81	2.4	0.00	1	0	19
RC201	*594.89	*	3.8	*	594.89	594.89	0.2	0.00	1	0	0
RC202	*540.87	*	5.3	*	540.87	540.87	0.2	0.00	1	0	0
RC203	*443.28	*	72.4	*	437.32	443.28	2.0	0.00	1	0	43
RC204	*445.51	*	13.6	*	444.66	445.51	1.6	0.00	1	0	8
RC205	*536.35	*	8.6	*	532.84	536.35	0.4	0.00	1	0	33
RC206	*589.19	*	2.4	*	589.19	589.19	0.3	0.00	1	0	0
RC207	*485.88	*	15.5	*	482.99	485.88	0.8	0.00	1	0	10
RC208	*440.24	*	39.1	*	439.76	440.24	1.8	0.00	1	0	10

Table 12: Detailed results for 50-customer EVRPTW-SP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*776.93	*	0.5	*	776.93	776.93	0.6	0.00	1	0	0
C102	*823.33	*	2.4	*	823.33	823.33	1.4	0.00	1	0	0
C103	*667.60	*	903.1	*	652.78	667.60	28.8	0.00	53	0	165
C104	*559.13	*	19.8	*	559.13	559.13	172.6	0.00	1	0	0
C105	*754.74	*	1.3	*	751.00	754.74	0.8	0.00	1	0	14
C106	*799.24	*	3.0	*	793.63	799.24	1.0	0.00	1	0	28
C107	*782.90	*	37.9	*	751.20	782.90	1.7	0.00	1	0	46
C108	*733.68	*	2.1	*	730.19	733.68	1.0	0.00	1	0	15
C109	*675.07	*	11.0	*	666.68	675.07	2.4	0.00	1	16	42
R101	*937.29	*	1.6	*	937.29	937.29	0.1	0.00	1	0	0
R102	*878.39	*	5.5	*	869.36	878.39	0.8	0.00	1	0	24
R103	*806.53	*	157.9	*	794.97	806.53	4.5	0.00	51	0	109
R104	*615.55		TL1	*	602.22	615.55	223.6	0.00	1	0	126
R105	*838.98	*	64.2	*	827.33	838.98	0.9	0.00	1	0	82
R106	*799.38	*	65.1	*	786.95	799.38	2.4	0.00	7	0	99
R107	*737.63	*	260.8	*	722.69	737.63	9.1	0.00	3	0	153
R108	*537.63	*	93.7	*	525.93	537.63	42.0	0.00	1	0	33
R109	*816.21	*	30.5	*	805.07	816.21	2.4	0.00	1	0	69
R110	*729.42	*	144.6	*	718.10	729.42	11.5	0.00	1	0	119
R111	*783.18	*	55.4	*	769.84	783.18	9.3	0.00	1	0	67
R112	*653.03		TL1		628.50	653.03	7005.5	0.00	177	16	227
RC101	*1066.48	*	6.8	*	1042.61	1066.48	0.6	0.00	1	0	26
RC102	*900.71	*	5.5	*	898.29	900.71	1.4	0.00	1	0	22
RC103	*847.00	*	8.1	*	841.51	847.00	4.6	0.00	1	0	16
RC104	*692.05	*	18.9	*	679.13	692.05	25.8	0.00	1	0	54
RC105	*997.59	*	5.5	*	986.52	997.59	1.3	0.00	1	0	40
RC106	*929.66	*	48.1	*	889.45	929.66	8.8	0.00	3	0	198
RC107	*788.67	*	963.9	*	757.07	788.67	54.5	0.00	9	0	248
RC108	*726.09	*	131.0	*	711.46	726.09	24.8	0.00	1	2	95
C201	*503.26	*	2.7	*	503.26	503.26	2.1	0.00	1	0	0
C202	*523.75	*	7.8	*	523.75	523.75	3.7	0.00	1	0	0
C203	*513.82	*	47.7	*	513.82	513.82	19.2	0.00	1	0	0
C204	*469.56	*	277.2	*	469.56	469.56	209.7	0.00	1	0	0
C205	*513.75	*	3.8	*	513.75	513.75	1.3	0.00	1	0	0
C206	*515.97	*	7.8	*	515.97	515.97	3.8	0.00	1	0	0
C207	*531.13	*	13.7	*	530.57	531.13	5.3	0.00	1	0	6
C208	*528.40	*	11.2	*	528.40	528.40	3.8	0.00	1	0	0
R201	*674.75	*	183.6	*	674.75	674.75	1.2	0.00	1	0	0
R202	*647.17	*	229.2	*	647.08	647.17	5.2	0.00	1	0	6
R203	*571.02	*	806.8	*	569.98	571.02	10.6	0.00	1	0	22
R204	508.03		TL1		482.04	485.28	TL2	4.48	1	0	39
R205	*606.43	*	223.2	*	604.52	606.43	5.1	0.00	1	0	21
R206	*579.37	*	1474.3	*	573.16	579.37	16.5	0.00	1	0	136
R207	*512.56	*	3339.6	*	510.79	512.56	110.6	0.00	1	0	34
R208	*467.16		TL1	*	463.14	467.16	288.1	0.00	1	0	130
R209	*602.50		TL1	*	594.13	602.50	70.7	0.00	3	0	127
R210	*519.78	*	1782.8	*	517.47	519.78	16.0	0.00	1	0	63
R211	*516.53		TL1	*	510.95	516.53	89.6	0.00	3	0	108
RC201	*812.23	*	19.9	*	812.23	812.23	0.5	0.00	1	0	0
RC202	*799.71	*	55.6	*	786.67	799.71	5.2	0.00	1	0	75
RC203	*670.03	*	400.2	*	666.94	670.03	18.9	0.00	1	0	19
RC204	*610.24	*	1772.3	*	610.24	610.24	92.8	0.00	1	0	0
RC205	*720.66	*	127.4	*	713.33	720.66	3.0	0.00	1	0	31
RC206	*795.20	*	59.3	*	794.38	795.20	1.9	0.00	1	0	10
RC207	*646.78	*	373.8	*	646.78	646.78	25.5	0.00	1	0	0
RC208	*577.33	*	601.0	*	575.10	577.33	89.0	0.00	1	0	20

Table 13: Detailed results for 100-customer EVRPTW-SP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*1025.41	*	2.2	*	1017.68	1025.41	29.4	0.00	26	0	156
C102	*1025.41	*	214.6	*	1017.68	1025.41	29.6	0.00	5	4	156
C103	*995.85		TL1	*	976.90	995.85	1708.9	0.00	5	26	320
C104	969.57		TL1		919.44	935.45	TL2	3.52	1	75	20
C105	*1031.24	*	11.8	*	1021.60	1031.24	7.3	0.00	1	176	53
C106	*1027.44	*	21.2	*	1024.02	1027.44	6.6	0.00	1	37	32
C107	*1030.40	*	60.4	*	1012.78	1030.40	16.3	0.00	3	204	216
C108	*1014.81	*	166.6	*	995.02	1014.81	27.8	0.00	1	180	119
C109	998.96		TL1		950.61	993.06	TL2	0.59	120	186	320
R101	*1603.06	*	39.8	*	1593.10	1603.06	1.2	0.00	1	0	35
R102	*1451.56	*	24.6	*	1449.51	1451.56	3.5	0.00	1	0	15
R103	*1210.90	*	2935.3	*	1192.70	1210.90	162.5	0.00	29	0	294
R104	1080.91		TL1		1045.74	1048.44	TL2	3.00	1	45	20
R105	*1372.94		TL1		1338.88	1372.94	5307.1	0.00	1489	0	295
R106	*1251.89	*	2131.2	*	1239.24	1251.89	85.1	0.00	71	0	252
R107	1159.73		TL1		1131.09	1155.92	TL2	0.33	277	0	316
R108	1018.76		TL1		999.73	1006.82	TL2	1.17	1	2	70
R109	1257.18		TL1		1222.47	1252.49	TL2	0.37	469	0	320
R110	1118.54		TL1		1063.09	1083.08	TL2	3.17	10	0	320
R111	*1111.65		TL1	*	1092.03	1111.65	567.9	0.00	13	0	290
R112	1055.69		TL1		1009.77	1022.06	TL2	3.19	1	0	71
RC101	*1713.86		TL1		1656.03	1713.00	TL2	0.05	2187	7	212
RC102	*1548.77	*	101.0	*	1521.31	1548.77	52.2	0.00	1	69	151
RC103	*1314.33	*	2181.2	*	1293.82	1314.33	115.2	0.00	1	101	26
RC104	1228.67		TL1		1159.57	1196.31	TL2	2.63	1	125	82
RC105	1538.68		TL1		1465.76	1507.56	TL2	2.02	329	42	234
RC106	1401.10		TL1		1340.30	1385.12	TL2	1.14	297	27	276
RC107	1258.41		TL1		1176.62	1226.25	TL2	2.56	1	50	263
RC108	1253.31		TL1		1177.71	1221.77	TL2	2.52	1	178	134
C201	*630.35	*	35.2				TL2		0	0	0
C202	*630.35	*	150.3	*	630.35	630.35	141.5	0.00	1	0	0
C203	*630.35	*	1324.4	*	630.35	630.35	1024.1	0.00	1	0	0
C204			TL1				TL2		0	0	0
C205	*630.35	*	100.4	*	630.35	630.35	22.5	0.00	1	0	0
C206	*630.35	*	179.3	*	630.35	630.35	99.1	0.00	1	0	0
C207	*630.35	*	491.1	*	630.35	630.35	221.2	0.00	1	0	0
C208	*630.35	*	417.2	*	630.35	630.35	154.6	0.00	1	0	0
R201	*1098.24	*	1745.2	*	1093.10	1098.24	24.4	0.00	1	0	104
R202	993.13		TL1		980.10	991.57	TL2	0.16	300	0	320
R203	890.23		TL1		853.82	859.09	TL2	3.50	1	0	82
R204	858.12		TL1				TL2		0	0	0
R205	*950.83		TL1		936.99	950.41	TL2	0.04	649	0	301
R206	*887.70		TL1	*	878.68	887.70	1034.9	0.00	1	0	187
R207	876.73		TL1		781.69	785.28	TL2	10.43	1	0	31
R208			TL1				TL2		0	0	0
R209	862.85		TL1		834.51	843.68	TL2	2.22	1	0	115
R210	894.96		TL1		814.63	819.60	TL2	8.42	1	0	50
R211	892.42		TL1		742.67	742.67	TL2	16.78	1	0	10
RC201	*1258.27	*	619.6	*	1252.61	1258.27	9.6	0.00	3	0	60
RC202	*1139.29		TL1	*	1129.07	1139.29	361.3	0.00	47	0	309
RC203	*957.22		TL1	*	948.46	957.22	1357.7	0.00	1	0	123
RC204			TL1				TL2		0	0	0
RC205	*1072.08	*	872.1	*	1069.28	1072.08	151.2	0.00	1	0	51
RC206	*1069.39		TL1	*	1053.19	1069.39	1277.5	0.00	197	0	320
RC207	*925.72		TL1	*	918.49	925.72	1797.4	0.00	1	0	54
RC208			TL1				TL2		0	0	0

Table 14: Detailed results for 25-customer EVRPTW-MF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*627.08	*	0.2	*	625.06	627.08	0.2	0.00	1	0	4
C102	*526.41	*	0.2	*	526.41	526.41	0.3	0.00	1	0	0
C103	*345.54	*	0.9	*	345.54	345.54	2.1	0.00	1	0	0
C104	*449.91	*	711.9	*	436.17	449.91	16.7	0.00	3	0	126
C105	*541.49	*	0.3	*	539.73	541.49	0.2	0.00	1	0	7
C106	*562.41	*	0.3	*	555.63	562.41	0.3	0.00	1	0	3
C107	*505.86	*	0.4	*	502.46	505.86	0.2	0.00	1	0	5
C108	*508.39	*	1.4	*	502.63	508.39	0.5	0.00	1	0	33
C109	*473.55	*	3.4	*	464.74	473.55	0.7	0.00	1	0	16
R101	*662.33	*	0.3	*	662.33	662.33	0.0	0.00	1	0	0
R102	*453.08	*	2.5	*	448.06	453.08	0.4	0.00	1	0	24
R103	*494.65	*	2.1	*	493.59	494.65	0.4	0.00	1	0	6
R104	*352.16	*	8.7	*	347.52	352.16	2.5	0.00	1	0	31
R105	*584.61	*	1.9	*	579.36	584.61	0.2	0.00	1	0	4
R106	*480.22	*	2.1	*	474.23	480.22	1.2	0.00	1	0	3
R107	*416.51	*	11.3	*	412.45	416.51	1.1	0.00	3	0	31
R108	*429.37	*	1.9	*	429.37	429.37	1.7	0.00	1	0	0
R109	*462.24	*	6.3	*	457.03	462.24	0.3	0.00	1	0	16
R110	*419.67	*	2.5	*	412.61	419.67	1.5	0.00	3	0	34
R111	*383.04	*	12.8	*	375.53	383.04	3.6	0.00	1	0	52
R112	*397.42	*	13.1	*	383.85	397.42	3.3	0.00	1	0	62
RC101	*738.17	*	1.1	*	730.25	738.17	0.1	0.00	1	0	2
RC102	*648.52	*	0.8	*	641.89	648.52	0.4	0.00	1	0	6
RC103	*560.91	*	1.1	*	552.64	560.91	0.6	0.00	1	0	38
RC104	*516.43	*	2.5	*	510.05	516.43	4.9	0.00	1	0	10
RC105	*589.86	*	0.3	*	589.86	589.86	0.2	0.00	1	0	0
RC106	*557.23	*	0.8	*	557.23	557.23	0.2	0.00	1	0	0
RC107	*497.65	*	0.6	*	497.65	497.65	0.8	0.00	1	0	0
RC108	*479.73	*	6.0	*	474.90	479.73	2.5	0.00	1	0	16
C201	*416.37	*	0.9	*	416.37	416.37	0.5	0.00	1	0	0
C202	*404.57	*	2.1	*	404.57	404.57	0.9	0.00	1	0	0
C203	*391.72	*	10.2	*	391.72	391.72	6.8	0.00	1	0	0
C204	*362.93	*	10.7	*	362.93	362.93	1.5	0.00	1	0	0
C205	*409.05	*	1.1	*	409.05	409.05	0.4	0.00	1	0	0
C206	*389.19	*	1.9	*	389.19	389.19	1.0	0.00	1	0	0
C207	*397.31	*	1.8	*	397.31	397.31	0.5	0.00	1	0	0
C208	*394.69	*	2.5	*	394.69	394.69	1.4	0.00	1	0	0
R201	*374.28	*	4.6	*	374.28	374.28	0.2	0.00	1	0	0
R202	*375.30	*	21.0	*	375.30	375.30	0.6	0.00	1	0	0
R203	*382.77	*	61.3	*	380.47	382.77	1.7	0.00	1	0	23
R204	*315.52	*	192.5	*	315.52	315.52	14.0	0.00	1	0	0
R205	*374.00	*	14.4	*	374.00	374.00	0.2	0.00	1	0	0
R206	*424.51	*	43.8	*	423.22	424.51	1.4	0.00	1	0	15
R207	*330.96	*	213.6	*	330.96	330.96	8.5	0.00	1	0	0
R208	*315.59	*	3405.1	*	315.59	315.59	177.0	0.00	1	0	0
R209	*422.39	*	18.8	*	417.87	422.39	0.7	0.00	1	0	62
R210	*315.27	*	117.0	*	315.27	315.27	3.0	0.00	1	0	0
R211	*383.81	*	116.9	*	383.27	383.81	8.1	0.00	1	0	22
RC201	*594.89	*	3.8	*	594.89	594.89	0.4	0.00	1	0	0
RC202	*540.87	*	5.3	*	540.87	540.87	0.2	0.00	1	0	0
RC203	*443.28	*	96.7	*	437.32	443.28	4.9	0.00	1	0	53
RC204	*445.51	*	20.1	*	444.66	445.51	3.3	0.00	1	0	8
RC205	*536.35	*	9.0	*	532.84	536.35	0.5	0.00	1	0	39
RC206	*589.19	*	3.6	*	589.19	589.19	0.4	0.00	1	0	0
RC207	*485.88	*	25.1	*	483.18	485.88	1.9	0.00	1	0	10
RC208	*438.39	*	30.6	*	438.39	438.39	6.8	0.00	1	0	0

Table 15: Detailed results for 50-customer EVRPTW-MF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*783.84	*	0.7	*	783.59	783.84	0.5	0.00	1	0	3
C102	*784.93	*	1.6	*	784.93	784.93	1.8	0.00	1	0	0
C103	*656.90	*	23.4	*	649.82	656.90	10.0	0.00	1	0	57
C104	*582.87	*	332.0	*	575.28	582.87	109.0	0.00	3	0	159
C105	*737.03	*	0.6	*	736.61	737.03	0.7	0.00	1	0	12
C106	*755.21	*	1.1	*	753.75	755.21	0.8	0.00	1	0	6
C107	*708.98	*	0.9	*	708.95	708.98	0.9	0.00	1	0	3
C108	*726.23	*	3.8	*	723.87	726.23	1.5	0.00	1	0	18
C109	*677.25	*	10.6	*	669.75	677.25	4.6	0.00	1	0	44
R101	*940.22	*	2.8	*	939.73	940.22	0.3	0.00	1	0	6
R102	*867.02	*	16.6	*	856.25	867.02	1.6	0.00	15	0	46
R103	*803.48	*	151.3	*	790.93	803.48	5.3	0.00	25	0	88
R104	*624.11		TL1		604.28	624.11	6440.8	0.00	11	0	292
R105	*842.73	*	75.5	*	830.93	842.73	1.0	0.00	1	0	65
R106	*794.31	*	29.1	*	782.14	794.31	5.4	0.00	1	0	119
R107	*691.70	*	21.4	*	685.73	691.70	5.2	0.00	1	0	41
R108	*543.83	*	66.4	*	532.15	543.83	62.5	0.00	1	0	79
R109	*789.72	*	134.7	*	776.37	789.72	4.2	0.00	1	0	94
R110	*713.74	*	35.5	*	705.27	713.74	6.9	0.00	1	0	44
R111	*745.44	*	70.6	*	733.08	745.44	12.9	0.00	3	0	123
R112	*603.11	*	168.4	*	600.09	603.11	36.5	0.00	1	0	33
RC101	*1074.42	*	7.2	*	1044.41	1074.42	1.0	0.00	1	0	42
RC102	*897.54	*	7.6	*	889.33	897.54	1.9	0.00	1	0	22
RC103	*829.68	*	2.9	*	829.68	829.68	4.8	0.00	1	0	0
RC104	*690.21	*	48.0	*	681.31	690.21	29.8	0.00	1	0	35
RC105	*984.24	*	339.4	*	970.61	984.24	42.2	0.00	579	0	46
RC106	*888.28	*	7.0	*	871.74	888.28	2.7	0.00	1	0	42
RC107	*786.43	*	73.4	*	764.01	786.43	22.8	0.00	1	11	85
RC108	*742.30		TL1	*	716.96	742.30	122.2	0.00	17	1	217
C201	*503.26	*	2.7	*	503.26	503.26	3.1	0.00	1	0	0
C202	*523.75	*	11.0	*	523.75	523.75	3.4	0.00	1	0	0
C203	*513.82	*	84.7	*	513.82	513.82	116.0	0.00	1	0	0
C204	*469.23		TL1	*	468.11	469.23	780.0	0.00	1	0	37
C205	*513.75	*	11.2	*	513.75	513.75	2.2	0.00	1	0	0
C206	*515.97	*	12.9	*	515.97	515.97	8.3	0.00	1	0	0
C207	*531.13	*	24.3	*	530.57	531.13	7.4	0.00	1	0	6
C208	*528.40	*	14.0	*	528.40	528.40	6.8	0.00	1	0	0
R201	*674.75	*	228.2	*	674.75	674.75	1.9	0.00	1	0	0
R202	*647.17	*	556.9	*	647.08	647.17	12.8	0.00	1	0	6
R203	*571.02	*	3595.4	*	569.98	571.02	43.7	0.00	1	0	22
R204	510.56		TL1				TL2		0	0	0
R205	*606.43	*	566.4	*	604.52	606.43	13.5	0.00	1	0	19
R206	*579.37		TL1	*	573.16	579.37	35.8	0.00	1	0	127
R207	*512.56		TL1	*	510.79	512.56	296.2	0.00	1	0	46
R208	*467.16		TL1	*	463.27	467.16	1081.4	0.00	1	0	188
R209	*602.50		TL1	*	594.13	602.50	183.2	0.00	3	0	101
R210	*521.18		TL1	*	517.89	521.18	51.9	0.00	1	0	73
R211	*516.53		TL1	*	510.95	516.53	235.2	0.00	15	0	112
RC201	*812.23	*	20.1	*	812.23	812.23	0.7	0.00	1	0	0
RC202	*799.71	*	97.6	*	786.67	799.71	8.2	0.00	1	0	73
RC203	*670.03	*	1095.2	*	666.94	670.03	40.0	0.00	1	0	19
RC204	*605.71	*	1829.7	*	605.71	605.71	623.9	0.00	1	0	0
RC205	*720.66	*	163.3	*	713.33	720.66	6.7	0.00	1	0	73
RC206	*795.20	*	82.6	*	794.38	795.20	5.8	0.00	1	0	10
RC207	*646.78	*	629.5	*	646.78	646.78	113.4	0.00	1	0	0
RC208	*577.33	*	1253.2	*	574.76	577.33	339.0	0.00	1	0	50

Table 16: Detailed results for 100-customer EVRPTW-MF instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*1054.19	*	2.3	*	1054.19	1054.19	2.9	0.00	1	0	0
C102	*1022.93	*	31.5	*	997.64	1022.93	52.1	0.00	1	61	58
C103	1014.77		TL1		966.98	992.97	TL2	2.15	1	220	108
C104	985.47		TL1		906.98	909.53	TL2	7.71	1	2	33
C105	*1034.28	*	48.8	*	1007.25	1034.28	15.8	0.00	1	277	64
C106	*1027.62	*	60.1	*	992.43	1027.62	28.1	0.00	1	62	135
C107	*1025.99	*	339.3	*	977.48	1025.99	100.2	0.00	27	248	165
C108	1016.06		TL1		955.35	1012.77	TL2	0.32	583	74	320
C109	1010.46		TL1		922.50	971.10	TL2	3.89	1	148	320
R101	*1602.41	*	23.7	*	1593.57	1602.41	1.5	0.00	1	0	37
R102	*1455.56	*	177.8	*	1448.01	1455.56	6.4	0.00	25	0	74
R103	*1195.85		TL1	*	1182.08	1195.85	563.1	0.00	345	0	230
R104	1077.60		TL1		1050.21	1056.31	TL2	1.98	1	1	43
R105	*1340.79	*	1273.4	*	1322.00	1340.79	75.0	0.00	199	12	153
R106	*1229.89	*	145.3	*	1222.99	1229.89	21.0	0.00	1	0	105
R107	*1148.35		TL1	*	1125.87	1148.35	1250.5	0.00	319	0	266
R108	1026.20		TL1		990.52	997.40	TL2	2.81	1	1	44
R109	1225.45		TL1		1188.84	1207.22	TL2	1.49	326	0	273
R110	1106.72		TL1		1071.18	1090.82	TL2	1.44	22	0	320
R111	1117.88		TL1		1080.58	1097.25	TL2	1.85	33	0	320
R112	*1017.31		TL1	*	1000.81	1017.31	1491.0	0.00	1	0	252
RC101	1694.94		TL1		1626.16	1687.97	TL2	0.41	1156	12	284
RC102	*1532.31	*	928.8	*	1489.14	1532.31	110.9	0.00	13	24	243
RC103	*1332.76		TL1	*	1295.99	1332.76	426.3	0.00	1	64	243
RC104	1228.67		TL1		1164.48	1191.30	TL2	3.04	1	64	99
RC105	*1482.70	*	3515.9	*	1459.07	1482.70	55.3	0.00	25	19	184
RC106	*1377.00		TL1	*	1334.52	1377.00	131.9	0.00	1	25	273
RC107	*1244.57		TL1	*	1187.49	1244.57	2653.6	0.00	9	72	320
RC108	1253.31		TL1		1184.60	1235.13	TL2	1.45	1	151	231
C201	*645.57		TL1	*	638.65	645.57	167.8	0.00	37	0	0
C202	*645.57		TL1	*	637.97	645.57	1529.2	0.00	17	0	0
C203	674.63		TL1		635.82	636.77	TL2	5.61	1	0	10
C204			TL1				TL2		0	0	0
C205	*641.53		TL1		634.26	641.53	3620.8	0.00	201	0	10
C206	*638.56		TL1	*	633.12	638.56	143.8	0.00	1	0	68
C207	*638.56		TL1	*	633.12	638.56	461.7	0.00	17	0	4
C208	*638.56		TL1	*	633.12	638.56	364.8	0.00	3	0	36
R201	*1098.61		TL1	*	1093.12	1098.61	42.2	0.00	1	0	144
R202	993.15		TL1		980.10	990.78	TL2	0.24	111	0	320
R203	896.11		TL1		853.82	854.91	TL2	4.60	1	0	30
R204	868.79		TL1				TL2		0	0	0
R205	951.00		TL1		937.01	949.80	TL2	0.13	380	0	320
R206	946.05		TL1		878.69	884.29	TL2	6.53	1	0	41
R207	874.35		TL1		781.84	783.51	TL2	10.39	1	0	10
R208			TL1				TL2		0	0	0
R209	862.85		TL1		834.79	841.67	TL2	2.45	1	0	64
R210	894.96		TL1		814.86	818.16	TL2	8.58	1	0	20
R211	892.42		TL1				TL2		0	0	0
RC201	*1258.27	*	1623.3	*	1252.61	1258.27	19.1	0.00	3	0	59
RC202	*1139.29		TL1	*	1129.01	1139.29	554.9	0.00	45	1	247
RC203	1032.46		TL1		948.67	954.85	TL2	7.52	1	0	52
RC204			TL1				TL2		0	0	0
RC205	*1072.08	*	1971.9	*	1069.19	1072.08	238.1	0.00	1	0	51
RC206	*1069.39		TL1	*	1053.23	1069.39	1393.6	0.00	139	0	320
RC207	*925.72		TL1		918.56	925.72	4296.0	0.00	1	0	53
RC208			TL1				TL2		0	0	0

Table 17: Detailed results for 25-customer EVRPTW-MP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*622.89	*	0.1	*	622.89	622.89	0.2	0.00	1	0	0
C102	*521.69	*	0.4	*	521.69	521.69	0.2	0.00	1	0	0
C103	*345.54	*	3.7	*	344.26	345.54	3.4	0.00	1	0	21
C104	*431.43	*	5.7	*	431.43	431.43	8.6	0.00	1	0	0
C105	*524.80	*	0.2	*	524.80	524.80	0.1	0.00	1	0	0
C106	*555.99	*	0.6	*	549.21	555.99	0.5	0.00	1	0	15
C107	*505.15	*	0.7	*	496.42	505.15	0.3	0.00	1	0	12
C108	*494.82	*	0.4	*	494.82	494.82	0.3	0.00	1	0	0
C109	*442.40	*	0.7	*	441.91	442.40	0.7	0.00	1	0	7
R101	*661.46	*	0.3	*	661.46	661.46	0.1	0.00	1	0	0
R102	*438.06	*	1.1	*	438.06	438.06	0.2	0.00	1	0	0
R103	*493.10	*	1.9	*	491.89	493.10	0.4	0.00	1	0	9
R104	*352.16	*	83.8	*	343.99	352.16	5.7	0.00	3	0	99
R105	*544.07	*	1.0	*	544.01	544.07	0.2	0.00	1	0	3
R106	*480.22	*	6.3	*	472.40	480.22	0.8	0.00	1	0	29
R107	*411.14	*	3.3	*	408.99	411.14	1.8	0.00	1	0	10
R108	*424.34	*	1.8	*	424.34	424.34	2.1	0.00	1	0	0
R109	*453.51	*	8.9	*	446.22	453.51	0.7	0.00	1	0	61
R110	*406.84	*	3.0	*	404.99	406.84	1.2	0.00	1	18	8
R111	*382.89	*	59.0	*	371.85	382.89	11.5	0.00	1	0	87
R112	*385.51	*	5.8	*	375.87	385.51	2.3	0.00	1	0	27
RC101	*728.35	*	1.7	*	717.63	728.35	0.1	0.00	1	0	5
RC102	*639.71	*	3.0	*	629.48	639.71	0.3	0.00	1	0	16
RC103	*552.23	*	1.5	*	545.30	552.23	0.7	0.00	1	0	13
RC104	*516.22	*	5.7	*	504.57	516.22	5.3	0.00	1	0	42
RC105	*589.86	*	0.2	*	589.86	589.86	0.2	0.00	1	0	0
RC106	*549.28	*	0.7	*	549.28	549.28	0.2	0.00	1	0	0
RC107	*497.65	*	0.7	*	497.65	497.65	1.0	0.00	1	0	0
RC108	*471.08	*	3.2	*	468.63	471.08	2.5	0.00	1	0	10
C201	*416.37	*	1.1	*	416.37	416.37	0.4	0.00	1	0	0
C202	*404.57	*	2.6	*	404.57	404.57	0.6	0.00	1	0	0
C203	*391.72	*	11.8	*	391.72	391.72	8.3	0.00	1	0	0
C204	*362.18	*	11.9	*	362.18	362.18	4.4	0.00	1	0	0
C205	*409.05	*	0.8	*	409.05	409.05	0.6	0.00	1	0	0
C206	*389.19	*	2.2	*	389.19	389.19	0.6	0.00	1	0	0
C207	*397.31	*	2.1	*	397.31	397.31	0.5	0.00	1	0	0
C208	*394.69	*	2.3	*	394.69	394.69	1.3	0.00	1	0	0
R201	*374.28	*	4.7	*	374.28	374.28	0.1	0.00	1	0	0
R202	*375.30	*	17.8	*	375.30	375.30	0.5	0.00	1	0	0
R203	*382.77	*	42.1	*	380.47	382.77	1.4	0.00	1	0	19
R204	*315.52	*	248.6	*	315.52	315.52	10.0	0.00	1	0	0
R205	*374.00	*	8.7	*	374.00	374.00	0.3	0.00	1	0	0
R206	*424.51	*	48.9	*	423.22	424.51	1.0	0.00	1	0	15
R207	*330.96	*	186.2	*	330.96	330.96	6.7	0.00	1	0	0
R208	*315.59		TL1	*	315.59	315.59	57.1	0.00	1	0	0
R209	*422.39	*	32.6	*	417.87	422.39	0.8	0.00	1	0	50
R210	*315.27	*	74.5	*	315.27	315.27	2.0	0.00	1	0	0
R211	*383.81	*	312.8	*	383.27	383.81	4.4	0.00	1	0	21
RC201	*594.89	*	5.6	*	594.89	594.89	0.4	0.00	1	0	0
RC202	*540.87	*	4.5	*	540.87	540.87	0.3	0.00	1	0	0
RC203	*443.28	*	90.0	*	437.32	443.28	2.9	0.00	1	0	43
RC204	*445.51	*	51.8	*	444.66	445.51	5.5	0.00	1	0	8
RC205	*536.35	*	10.3	*	532.84	536.35	0.6	0.00	1	0	39
RC206	*589.19	*	3.5	*	589.19	589.19	0.3	0.00	1	0	0
RC207	*485.88	*	22.6	*	482.99	485.88	1.9	0.00	1	0	10
RC208	*438.39	*	48.8	*	438.39	438.39	11.1	0.00	1	0	0

Table 18: Detailed results for 50-customer EVRPTW-MP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*771.50	*	0.6	*	771.50	771.50	0.5	0.00	1	0	0
C102	*777.19	*	2.4	*	777.19	777.19	1.9	0.00	1	0	0
C103	*631.94		TL1	*	617.00	631.94	507.6	0.00	159	0	169
C104	*550.99	*	44.9	*	550.99	550.99	164.0	0.00	1	0	0
C105	*729.80	*	0.9	*	729.80	729.80	0.8	0.00	1	0	0
C106	*735.54	*	1.2	*	735.10	735.54	1.1	0.00	1	0	3
C107	*704.12	*	1.4	*	704.12	704.12	1.0	0.00	1	0	0
C108	*718.91	*	2.2	*	715.22	718.91	1.5	0.00	1	0	10
C109	*668.97	*	317.7	*	659.16	668.97	4.7	0.00	3	0	94
R101	*930.19	*	8.2	*	928.04	930.19	0.4	0.00	3	0	12
R102	*848.77	*	16.8	*	837.35	848.77	1.9	0.00	3	0	93
R103	*778.01	*	17.0	*	771.68	778.01	2.8	0.00	1	0	55
R104	*615.19		TL1		599.82	612.00	TL2	0.52	1	0	171
R105	*829.44	*	21.7	*	818.03	829.44	0.9	0.00	1	0	54
R106	*789.34	*	487.3	*	773.88	789.34	1135.5	0.00	761	0	158
R107	*669.42	*	58.4	*	663.07	669.42	5.0	0.00	1	0	63
R108	*536.24	*	3584.7	*	523.99	536.24	58.6	0.00	1	0	96
R109	*763.92	*	776.2	*	751.63	763.92	10.6	0.00	19	0	168
R110	*704.54	*	2971.4	*	689.60	704.54	23.8	0.00	9	0	159
R111	*718.80	*	22.3	*	716.86	718.80	6.9	0.00	1	0	10
R112	*586.51	*	94.9	*	583.08	586.51	51.2	0.00	1	0	21
RC101	*1051.55	*	115.8	*	1012.54	1051.55	2.3	0.00	25	0	89
RC102	*875.43	*	3.4	*	875.43	875.43	1.5	0.00	1	0	0
RC103	*822.21	*	71.5	*	813.03	822.21	5.2	0.00	9	0	42
RC104	*686.97	*	56.7	*	676.11	686.97	33.2	0.00	1	0	57
RC105	*967.10	*	355.9	*	943.72	967.10	25.4	0.00	353	0	79
RC106	*858.53	*	553.5	*	840.61	858.53	3.9	0.00	1	0	80
RC107	*776.94		TL1	*	751.47	776.94	65.0	0.00	1	2	216
RC108	*720.81	*	307.2	*	698.40	720.81	75.2	0.00	1	3	167
C201	*503.26	*	7.3	*	503.26	503.26	1.5	0.00	1	0	0
C202	*523.75	*	12.1	*	523.75	523.75	3.3	0.00	1	0	0
C203	*513.82	*	171.1	*	513.82	513.82	34.7	0.00	1	0	0
C204	*469.23		TL1	*	468.11	469.23	614.1	0.00	1	0	35
C205	*513.75	*	7.6	*	513.75	513.75	1.6	0.00	1	0	0
C206	*515.97	*	15.6	*	515.97	515.97	5.4	0.00	1	0	0
C207	*531.13	*	28.9	*	529.72	531.13	8.3	0.00	1	0	13
C208	*528.40	*	20.3	*	528.40	528.40	5.5	0.00	1	0	0
R201	*674.75	*	207.5	*	674.75	674.75	0.9	0.00	1	0	0
R202	*647.17	*	438.4	*	647.08	647.17	9.9	0.00	1	0	6
R203	*571.02	*	1675.3	*	569.98	571.02	22.7	0.00	1	0	51
R204	504.45		TL1		482.04	482.04	TL2	4.44	1	0	10
R205	*606.43	*	325.5	*	604.52	606.43	8.5	0.00	1	0	30
R206	*579.37		TL1	*	573.16	579.37	37.6	0.00	1	0	129
R207	*512.56		TL1	*	510.79	512.56	211.9	0.00	1	0	59
R208	*467.16		TL1	*	463.14	467.16	835.2	0.00	1	0	151
R209	*602.50		TL1	*	594.13	602.50	201.3	0.00	3	0	144
R210	*519.78		TL1	*	517.47	519.78	37.5	0.00	1	0	55
R211	*516.53		TL1	*	510.95	516.53	208.2	0.00	3	0	130
RC201	*812.23	*	18.8	*	812.23	812.23	1.1	0.00	1	0	0
RC202	*799.71	*	92.1	*	786.67	799.71	10.0	0.00	1	0	72
RC203	*670.03	*	3231.9	*	666.94	670.03	30.4	0.00	1	0	19
RC204	*603.97	*	1903.5	*	603.97	603.97	614.1	0.00	1	0	0
RC205	*720.66	*	137.3	*	713.33	720.66	4.1	0.00	1	0	60
RC206	*795.20	*	97.3	*	794.38	795.20	3.5	0.00	1	0	10
RC207	*646.78	*	1089.4	*	646.78	646.78	106.8	0.00	1	0	0
RC208	*577.33	*	2891.9	*	574.36	577.33	636.9	0.00	1	0	20

Table 19: Detailed results for 100-customer EVRPTW-MP instances

		DEIS2016		BPC with TASF							
Inst	BKS	Opt 1h	Time [s]	Opt 1h	LB_{LP}	LB_{tree}	Time [s]	% Gap	#B&B	#CCs	#SRIs
C101	*1043.76	*	2.5	*	1043.76	1043.76	2.4	0.00	1	0	0
C102	1011.10		TL1		968.08	1008.10	TL2	0.30	509	28	195
C103	989.16		TL1		928.13	953.13	TL2	3.64	1	61	71
C104	913.98		TL1		873.85	877.18	TL2	4.03	1	42	20
C105	*1016.16	*	52.6	*	980.86	1016.16	60.5	0.00	19	298	147
C106	*1008.39	*	1061.6	*	977.54	1008.39	115.3	0.00	83	73	181
C107	1017.10		TL1		957.20	1003.86	TL2	1.30	302	284	260
C108	1013.23		TL1		928.92	985.10	TL2	2.78	55	161	295
C109	937.06		TL1		906.05	929.39	TL2	0.82	98	198	207
R101	*1573.83	*	11.7	*	1565.80	1573.83	1.3	0.00	1	0	22
R102	*1424.30	*	36.1	*	1419.69	1424.30	4.8	0.00	1	0	33
R103	*1162.61	*	1878.3	*	1152.89	1162.61	45.5	0.00	3	0	180
R104	1070.82		TL1		1025.00	1028.48	TL2	3.95	1	10	34
R105	*1303.07	*	68.4	*	1292.27	1303.07	7.9	0.00	1	0	89
R106	*1207.09		TL1	*	1202.21	1207.09	26.7	0.00	7	0	151
R107	*1105.13		TL1	*	1088.49	1105.13	534.1	0.00	91	0	320
R108	1008.71		TL1		968.46	974.25	TL2	3.42	1	3	30
R109	*1171.19		TL1	*	1149.81	1171.19	3010.7	0.00	619	0	320
R110	1087.64		TL1		1040.88	1059.50	TL2	2.59	1	0	320
R111	1076.63		TL1		1046.81	1065.70	TL2	1.02	63	0	320
R112	1014.68		TL1		971.22	982.90	TL2	3.13	1	0	71
RC101	*1615.91		TL1		1567.53	1615.91	5631.7	0.00	1597	16	269
RC102	*1483.07	*	237.0	*	1445.87	1483.07	90.0	0.00	1	53	186
RC103	*1291.52		TL1	*	1253.39	1291.52	677.6	0.00	1	128	138
RC104	*1152.60		TL1	*	1129.23	1152.60	1611.9	0.00	1	104	115
RC105	*1442.62	*	239.5	*	1421.99	1442.62	38.0	0.00	3	21	140
RC106	*1350.83	*	608.1	*	1302.69	1350.83	142.0	0.00	1	19	205
RC107	1244.57		TL1		1143.06	1187.26	TL2	4.60	1	66	153
RC108	1216.38		TL1		1151.31	1187.21	TL2	2.40	1	191	136
C201	*630.35	*	84.2				TL2		0	0	0
C202	630.35		TL1				TL2		0	0	0
C203	*630.35		TL1	*	630.35	630.35	1344.8	0.00	1	0	0
C204			TL1				TL2		0	0	0
C205	*630.35	*	159.8				TL2		0	0	0
C206	*630.35	*	306.6	*	630.35	630.35	256.2	0.00	1	0	0
C207	*630.35	*	933.3	*	630.35	630.35	212.9	0.00	1	0	0
C208	*630.35	*	595.1	*	630.35	630.35	149.0	0.00	1	0	0
R201	*1097.74	*	1984.1	*	1092.67	1097.74	35.2	0.00	1	0	148
R202	993.13		TL1		980.10	991.10	TL2	0.20	146	0	320
R203	890.23		TL1		853.82	858.56	TL2	3.56	1	0	70
R204	858.13		TL1				TL2		0	0	0
R205	950.83		TL1		936.99	949.57	TL2	0.13	513	0	267
R206	*887.70		TL1	*	878.68	887.70	1656.5	0.00	1	0	172
R207	874.36		TL1		781.69	783.52	TL2	10.39	1	0	21
R208			TL1				TL2		0	0	0
R209	862.85		TL1		834.50	844.02	TL2	2.18	1	0	84
R210	894.96		TL1		814.58	817.85	TL2	8.62	1	0	30
R211	892.42		TL1		742.67	742.67	TL2	16.78	1	0	0
RC201	*1258.27	*	940.0	*	1252.61	1258.27	17.0	0.00	3	0	74
RC202	*1139.29		TL1	*	1128.99	1139.29	304.7	0.00	21	1	278
RC203	*957.22		TL1	*	948.46	957.22	1666.5	0.00	1	0	110
RC204			TL1				TL2		0	0	0
RC205	*1072.08	*	1717.0	*	1069.06	1072.08	225.8	0.00	1	0	75
RC206	*1069.39		TL1	*	1053.19	1069.39	866.7	0.00	49	0	320
RC207	*925.72		TL1		918.49	925.72	4168.7	0.00	1	0	53
RC208			TL1				TL2		0	0	0

References

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. Operations Research, 59(5):1269–1283, October 2011. doi: 10.1287/opre.1110.0975.

Maria Battarra, Jean-François Cordeau, and Manuel Iori. Pickup-and-Delivery Problems for Goods Transportation, chapter 6, pages 161–191. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi: 10.1137/1.9781611973594.ch6.

Luciano Costa, Claudio Contardo, and Guy Desaulniers. Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science, 2019. Forthcoming.

Guy Desaulniers, François Lessard, and Ahmed Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. The vehicle routing problem with time windows. In Paolo Toth and Daniele Vigo, editors, Vehicle Routing, chapter 5, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi: 10.1137/1.9781611973594.ch5.

Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. Operations Research, 64(6):1388–1405, December 2016. doi: 10.1287/opre.2016.1535.

Jacques Desrosiers, Yves Dumas, Marius M. Solomon, and François Soumis. Time constrained routing and scheduling. In Michael O. Ball, Thomas L. Magnanti, B.L. Monma, and Georg L. Nemhauser, editors, Handbooks in Operations Research and Management Science, volume 8, chapter 2, pages 35–139. Elsevier, Amsterdam, 1995.

Karl F. Doerner and Juan-José Salazar-González. Pickup-and-Delivery Problems for People Transportation, chapter 7, pages 193–212. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014. doi: 10.1137/1.9781611973594.ch7.

Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2):201–213, January 2002. doi: 10.1007/s101070100263.

Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks, 44(3):216–229, 2004. doi: 10.1002/net.20033.

Timo Gschwind and Stefan Irnich. Effective handling of dynamic time windows and its application to solving the dial-a-ride problem. 49(2):335–354, 2015. doi: 10.1287/trsc.2014.0531.

Timo Gschwind, Stefan Irnich, Ann-Kathrin Rothenbächer, and Christian Tilk. Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research, 266(2):521–530, 2018. doi: 10.1016/j.ejor.2017.09.035.

Qie He, Stefan Irnich, and Yongjia Song. Branch-cut-and-price for the vehicle routing problem with time windows and convex node costs. Transportation Science, 2019. doi: 10.1287/trsc.2019.0891.

Timo Hintsch and Stefan Irnich. Exact solution of the soft-clustered vehicle-routing problem. Technical Report LM-2018-04, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany, 2018. https://logistik.bwl.uni-mainz.de/files/2018/12/LM-2018-04.pdf.

Irina Ioachim, Jacques Desrosiers, François Soumis, and Nicolas Bélanger. Fleet assignment and routing with schedule synchronization constraints. European Journal of Operational Research, 119(1):75–90, November 1999. doi: 10.1016/s0377-2217(98)00343-9.

Stefan Irnich. Resource extension functions: properties, inversion, and generalization to segments. OR Spectrum, 30(1):113–148, April 2007. doi: 10.1007/s00291-007-0083-6.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, Column Generation, pages 33–65. Springer-Verlag, 2005. doi: 10.1007/0-387-25486-2_2.

Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints andk-cycle elimination for $k \geq 3$. INFORMS Journal on Computing, 18(3):391–406, August 2006. doi: 10.1287/ijoc.1040.0117.

Stefan Irnich, Guy Desaulniers, Jacques Desrosiers, and Ahmed Hadjar. Path-reduced costs for eliminating arcs in routing and scheduling. INFORMS Journal on Computing, 22(2):297–313, May 2010. doi: 10.1287/ijoc.1090.0341.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research, 56(2):497–511, April 2008. doi: 10.1287/opre.1070.0449.

Niklas Kohl, Jacques Desrosiers, Oli B. G. Madsen, Marius M. Solomon, and François Soumis. 2-path cuts for the vehicle routing problem with time windows. Transportation Science, 33(1):101–116, February 1999. doi: 10.1287/trsc.33.1.101.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. Operations Research, 53 (6):1007–1023, December 2005. doi: 10.1287/opre.1050.0234.

Diego Pecin, Claudio Contardo, Guy Desaulniers, and Eduardo Uchoa. New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing, 29(3): 489–502, August 2017a. doi: 10.1287/ijoc.2016.0744.

Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. Mathematical Programming Computation, 9(1):61–100, March 2017b. doi: 10.1007/s12532-016-0108-8.

Artur Pessoa, Ruslan Sadykov, and Eduardo Uchoa. Enhanced branch-cut-and-price algorithm for heterogeneous fleet vehicle routing problems. European Journal of Operational Research, 270(2):530–543, October 2018. doi: 10.1016/j.ejor.2018.04.009.

Ted K. Ralphs, L. Kopman, William R. Pulleyblank, and L.E. Trotter. On the capacitated vehicle routing problem. Mathematical Programming, 94(2-3):343–359, January 2003. doi: 10.1007/s10107-002-0323-0.

Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Discrete Optimization, 3(3):255–273, September 2006. doi: 10.1016/j.disopt.2006.05.007.

Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. Transportation Science, 48(4):500–520, November 2014. doi: 10.1287/trsc.2013.0490.

Christian Tilk, Ann-Kathrin Rothenbächer, Timo Gschwind, and Stefan Irnich. Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. European Journal of Operational Research, 261(2):530–539, September 2017. doi: 10.1016/j.ejor.2017.03.017.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. 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, 2013. doi: 10.1016/j.cor.2012.07.018.

Daniel Villeneuve and Guy Desaulniers. The shortest path problem with forbidden paths. European Journal of Operational Research, 165(1):97–107, August 2005. doi: 10.1016/j.ejor.2004.01.032.