Cut-First Branch-and-Price-Second for the Capacitated Arc-Routing Problem

Bode Claudia, Stefan Irnich

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

Abstract

This paper presents the first full-fledged branch-and-price (bap) algorithm for the capacitated arc-routing problem (CARP). Prior exact solution techniques either rely on cutting planes or the transformation of the CARP into a node-routing problem. The drawbacks are either models with inherent symmetry, dense underlying networks, or a formulation where edge flows in a potential solution do not allow the reconstruction of unique CARP tours. The proposed algorithm circumvents all these drawbacks by taking the beneficial ingredients from existing CARP methods and combining them in a new way. The first step is the solution of the one-index formulation of the CARP in order to produce strong cuts and an excellent lower bound. It is known that this bound is typically stronger than relaxations of a pure set-partitioning CARP model. Such a set-partitioning master program results from a Dantzig-Wolfe decomposition. In the second phase, the master program is initialized with the strong cuts, CARP tours are iteratively generated by a pricing procedure, and branching is required to produce integer solutions. This is a cut-first bap-second algorithm and its main function is, in fact, the splitting of edge flows into unique CARP tours.

Key words: transportation: vehicle routing, integer programming: cutting-plane and branch-and-price algorithm.

1. Introduction

The capacitated arc-routing problem (CARP) is the basic multiple-vehicle arc-routing problem and has applications in waste collection, postal delivery, winter services such as snow plowing and salt gritting, meter reading, school bus routing and more. It was first introduced by Golden and Wong (1981) and has received a lot of attention since then; see for instance the edited book by Dror (2000) and the annotated bibliography by Corberán and Prins (2010).

This paper presents the first full-fledged branch-and-price (bap) algorithm for the CARP. Prior exact solution techniques either rely on cutting planes or the transformation of the CARP into a node-routing problem. As already pointed out by Letchford and Oukil (2009), the drawbacks are either models with inherent symmetry, dense underlying networks, or a formulation where edge flows in a potential solution do not allow the reconstruction of unique CARP tours. The proposed algorithm circumvents all these drawbacks by taking the beneficial ingredients from existing CARP methods and combining them in a new way. The first step is the solution of the one-index formulation of the CARP in order to produce strong cuts and an excellent lower bound. It is known that this bound is typically stronger than relaxations of a pure set-partitioning CARP model. Such a set-partitioning master program results from Dantzig-Wolfe decomposition. In the second phase, the master program is initialized with the strong cuts, CARP tours are iteratively generated by a pricing procedure, and branching is required to produce integer solutions. This is a cut-first bap-second algorithm and its main function is, in fact, the splitting of edge flows into unique CARP tours.

Email addresses: claudia.bode@uni-mainz.de (Bode Claudia), irnich@uni-mainz.de (Stefan Irnich)

The novelty of our approach comprises the following aspects: First, pricing of CARP tours is fast because it can be performed on the original network that is sparse for real-world instances. A key property for not being forced to use a transformed network is that deadheading variables can be guaranteed to have non-negative reduced cost. The addition of dual-optimal inequalities (Ben Amor et al., 2006) to the column generation master program is the device to ensure non-negativity. Second, in a CARP solution edges might be traversed more than once. This creates the problem that not all solutions with integer flows on edges impose integer path variables in the master problem. Therefore, a new hierarchical branching scheme is developed that is able to finally guarantee integer CARP solutions while being compatible with the pricing algorithm.

For a formal definition of the CARP, we assume an undirected graph \( G = (V, E) \) with node set \( V \) and edge set \( E \). Non-negative integer demands \( q_e \geq 0 \) are located on edges \( e \in E \). Those edges with positive demand form the subset \( E_R \subseteq E \) of required edges that have to be serviced exactly once. A fleet \( K \) of \( |K| \) homogeneous vehicles stationed at depot \( d \in V \) with capacity \( Q \) is given. The problem is to find minimum cost vehicle tours which start and end at the depot \( d \), service all required edges exactly once, and respect the vehicle capacity \( Q \). The tour costs consist of service costs \( c^{serv}_e \) for required edges \( e \) that are serviced and deadheading cost \( c_d \) whenever an edge \( e \) is traversed without servicing.

Throughout the paper we use the following standard notation. For any subset \( S \subseteq V \) we denote by \( \delta(S) \) the set of edges with exactly one endpoint in \( S \) and by \( \delta_R(S) = \delta(S) \cap E_R \). For the sake of brevity, we write \( \delta(i) \) instead of \( \delta(\{i\}) \). \( E(S) \) is the set of edges with both endpoints in \( S \) and \( E_R(S) = E(S) \cap E_R \). For any subset \( F \subseteq E \) and any parameter or variable \( y \), let be \( y(F) = \sum_{e \in F} y_e \).

The remainder of this paper is structured as follows. Section 2 reviews existing exact approaches for the CARP. Section 3 describes the Dantzig-Wolfe decomposition. The key components (cutting, pricing, branching) of the bap algorithm and their implementation are described in Section 4. Section 5 analyzes the interplay between cycle elimination and branching. Computational results in Section 6 show the capability of the new approach. Final conclusions are drawn in Section 7.

2. Review of Models and Methods

In this section, we outline successful MIP-based exact algorithms for the CARP that have been presented in the literature. Two types of approaches can be distinguished: Full exact methods determine an optimal integer solution and show optimality by proving that its cost is a lower bound. Methods that use compact MIP models with aggregated variables (see below) also provide a lower bound, but are not able to determine a solution to the CARP. Instead, optimality is proved with the help of a heuristic whenever the heuristic solution matches the lower bound.

Some authors (e.g. Belenguer and Benavent, 1998; Longo et al., 2006) assume that \( |K| \) is just a lower bound on the fleet size, others (e.g. Belenguer and Benavent, 1998) fix the number \( |K| \) of vehicles. We also assume a fixed fleet size with \( |K| \) vehicles, but point out that this assumption can affect the strength of the lower bounds and the computing times.

2.1. Node-Routing Transformation

Several researchers developed and applied transformations of arc-routing problems into node-routing problems (Pearn et al., 1987; Longo et al., 2006; Baldacci and Maniezzo, 2006). These approaches transform each required edge of the CARP into two or three associated nodes so that the number of nodes in the capacitated vehicle-routing problem (CVRP) is \( 2|E_R| + 1 \) or \( 3|E_R| + 1 \), respectively. The resulting CVRP instance is then solved with any CVRP algorithm.

Exact algorithms for the CVRP have been intensively studied in the past. The currently most successful algorithms are based on branch-and-cut (Lysgaard et al., 2004), branch-and-price-and-cut (Fukasawa et al., 2006), and a set-partitioning and cut-generation based approach that finally applies a standard IP solver (Baldacci et al., 2010).

A very successful variation of the solution approach of Baldacci et al. (2010), tailored to the CARP, is the recent work of Bartolini et al. (2011). As a prerequisite, an upper bound \( ub \) is required. First, the
CARP is transformed into a generalized VRP (GVRP). Here each required edge \( e \in E_R \) is represented by two nodes \( i_e, j_e \) (one for each direction of service) so that any GVRP solution must cover exactly one node of the cluster \( \{i_e, j_e\} \). Second, an extended set covering formulation with lifted odd cuts, rounded capacity cuts (Belenguer and Benavent, 2003), and subset-row inequalities (Jepsen et al., 2008) is solved by a sequence of lower bounding procedures (producing non-decreasing lower bounds \( lb_1, lb_2, lb_3, \) and \( lb_4 \)). Finally, as in the approach of Baldacci et al. (2010), all feasible CARP routes with reduced cost not exceeding the integrality gap \( ub - lb_4 \) are enumerated and a corresponding set-partitioning problem is solved using a MIP solver. In essence, the method is a VRP method as pricing and enumeration are performed on a transformed network that is dense and requires the covering of nodes.

Although these approaches are rather successful, Letchford and Oukil (2009) mentioned the following drawbacks: Even for relatively small CARP instances the resulting VRP is defined over a larger number of nodes and edges. In particular, the increase in the number of edges can be quadratic as the VRP graph is complete. As real-world CARP instances are based on street networks with typically very sparse underlying graphs, this effect is significant. Furthermore, specific graph structures allowing tailored CARP algorithms get lost by the transformation and the resulting VRP instance might feature further symmetries.

### 2.2. Two-Index Formulation

Belenguer and Benavent (1998) were the first to develop and analyze the following IP formulation for the CARP. This formulation is also referred to as sparse or two-index formulation. For every pair of an edge \( e \) and a vehicle \( k \) there are service and deadheading variables: \( x_{e}^{k} \) is equal to 1 if vehicle \( k \) services edge \( e \in E_R \) and 0 otherwise. The variable \( y_{e}^{k} \) counts the number of times vehicle \( k \) traverses edge \( e \) without servicing. Auxiliary variables \( p_{i}^{k} \) for each node \( i \) and vehicle \( k \) are needed to ensure even node degrees. The two-index formulation is:

\[
\min \sum_{k \in K} c^{serv, \top} x_{e}^{k} + \sum_{k \in K} c^{\top} y_{e}^{k} \tag{1}
\]

\[
\text{s.t.} \quad \sum_{k \in K} x_{e}^{k} = 1 \quad \text{for all } e \in E_R \tag{2}
\]

\[
x_{e}^{k}(\delta_{R}(S)) + y_{e}^{k}(\delta(S)) \geq 2x_{f}^{k} \quad \text{for all } S \subseteq V \setminus \{d\}, f \in E_R(S), k \in K \tag{3}
\]

\[
x_{e}^{k}(\delta_{R}(i)) + y_{e}^{k}(\delta(i)) = 2p_{i}^{k} \quad \text{for all } i \in V, k \in K \tag{4}
\]

\[
y_{e}^{k} \leq Q \quad \text{for all } k \in K \tag{5}
\]

\[
p_{i}^{k} \in \mathbb{Z}_+^{\lvert V \rvert}, x_{e}^{k} \in \{0,1\}^{\lvert E_R \rvert}, y_{e}^{k} \in \mathbb{Z}_+^{\lvert E \rvert} \quad \text{for all } k \in K \tag{6}
\]

The objective (1) is the minimization of all service and deadheading costs. Since each required edge is serviced exactly once, as stated by (2), service costs \( c^{serv} \) have no impact on optimal decisions. The formulation in (Belenguer and Benavent, 1998) therefore omits service costs. Constraints (3) are the subtour-elimination constraints (SEC). As discussed in (Belenguer and Benavent, 1998, p. 169), the given SEC still allow disconnected components being deadheaded. A corresponding infeasible integer solution, denoted as extended \( k \)-route, is not optimal and can be excluded by adding constraints of the form (3) with \( 2x_{f}^{k} \) replaced by \( 2y_{f}^{k} \). The parity constraints (4) ensure that each vehicle can leave a node \( i \) after entering. The capacity constraints are given by (5) and integrality constraints by (6).

The two-index formulation has two major drawbacks: First, the number of variables increases with the fleet size \( \lvert K \rvert \). Second, the inherent symmetry with respect to the numbering of vehicles lets branch-and-bound based algorithms perform poorly. The computational results in (Belenguer and Benavent, 2003; Ahr, 2004) show that the two-index formulation can work well for small \( \lvert K \rvert \leq 5 \), but is not suited to solve CARP instances with a larger fleet.
2.3. One-Index Formulation

The one-index formulation, first considered independently by Letchford (1997) and Belenguer and Benavent (1998), solely uses aggregated deadheading variables

\[ y_e = \sum_{k \in K} y_k^e \in \mathbb{Z}_+, \]

one for each edge \( e \in E \).

A formulation with deadheading variables alone seems appealing due to the small number of variables and, even more important, due to the eliminated symmetry regarding the numbering of the vehicles \( k \in K \). Notably, no IP formulation with aggregated deadheading variables \( y_e \in \mathbb{Z}_+ \) alone is known for the CARP. In fact, the integer polyhedron of the following CARP model is a relaxation of the CARP and can therefore contain infeasible integer solutions (see Belenguer and Benavent, 2003, p. 709). Even worse, a feasible integer solution to the CARP that is represented by the deadheading variables \( y_e \) of the one-index formulation is not helpful. The reconstruction of tours from deadheading variables is \( \mathcal{NP} \)-hard (and typically also a hard problem in practice). The bap phase of our solution approach can be interpreted as such a flow decomposition algorithm.

The usefulness of the one-index formulation is, however, that its LP-relaxation often produces a very tight lower bound:

\[
\begin{align*}
\min & \quad c^\top y \\
\text{s.t.} & \quad y(\delta(S)) \geq 2K(S) - |\delta_R(S)| & \text{for all } \emptyset \neq S \subseteq V \setminus \{d\} \\
& \quad y(\delta(S)) \geq 1 & \text{for all } \emptyset \neq S \subseteq V, |\delta_R(S)| \text{ odd} \\
& \quad y \in \mathbb{Z}^{|E|}_+ 
\end{align*}
\]

The objective (7) just takes the cost for deadheadings into account as service costs are constant. The capacity inequalities (8) require that there are at least \( 2K(S) \) traversals (services and deadheadings) over the cutset \( \delta(S) \). Thus, \( K(S) \) is the minimum number of vehicles necessary to serve \( E_R(S) \cup \delta_R(S) \) which can be approximated by \( \lceil q(E_R(S) \cup \delta_R(S))/Q \rceil \) and computed exactly by solving a bin-packing problem. The odd-cut inequalities (9) require at least one deadheading if there is an odd number of required edges in the cut \( \delta(S) \).

In combination with a powerful CARP heuristic, the one-index formulation provides a possible exact algorithm: Only if the heuristic comes across an optimal solution, the lower bound provided by (7)–(10) might prove optimality (as benchmark problems typically have integral costs, a gap less than one suffices).

Disjoint-path inequalities are another class of valid inequalities first considered by Belenguer and Benavent (2003). For the development of our model, it suffices to know that the general form of all valid inequalities of the one-index formulation is

\[
\sum_{e \in E} d_{es} y_e \geq r_s, \quad s \in \mathcal{S},
\]

where \( s \) is the index referring to a particular inequality, \( d_{es} \) is the coefficient of edge \( e \) in the inequality, and \( \mathcal{S} \) the set of all valid inequalities. Some details and references on separation procedures are provided in Section 4.1 and in the appendix.

2.4. Extended Set-Covering Approach

Gómez-Cabrero et al. (2005) use a set-covering approach in which the standard set covering constraints are supplemented with the capacity inequalities (8) and odd-cut inequalities (9). As for the one-index formulation, the focus is on generating an excellent lower bound by solving the LP-relaxation of the model. As the number of tours and cuts grows exponentially with the size of the instance, both row and column generation is required. Opposed to our approach, cutting planes are added after a solution to the LP-relaxation of an initial set-covering model has been computed. Details on pricing out new routes and
separating violated cuts will be discussed in comparison with the proposed cut-first bap-second approach in Section 4.2.

Let $c_r$ indicate the cost of a route $r \in \Omega$ and let $\bar{x}_{er} \in \{0, 1\}$ and $\bar{y}_{er} \in \mathbb{Z}_+$ be the number of times that route $r$ services and deadheads through edge $e$, respectively. The extended set-covering model for the CARP has binary decision variables $\lambda_r$ for each route $r \in \Omega$ and is defined as follows:

$$\min \sum_{r \in \Omega} c_r \lambda_r$$

subject to

$$\sum_{r \in \Omega} \bar{x}_{er} \lambda_r \geq 1 \quad \text{for all } e \in E_R$$

$$\sum_{r \in \Omega} d_{sr} \lambda_r \geq r_s \quad \text{for all } s \in S$$

$$\lambda_r \in \{0, 1\} \quad \text{for all } r \in \Omega$$

The extension to the standard set-covering model (12), (13) and (15) is the addition of transformed cuts (14) derived from (11), i.e., the constraints of the one-index formulation. Herein, $d_{sr}$ is the coefficient of the transformed cut $s \in S$ for route $r$, which is $d_{sr} = \sum_{e \in E} d_{es} \bar{y}_{er}$.

Gómez-Cabrero et al. (2005) allow non-elementary tours in the sense that a tour may service a required edge more than once. This relaxation makes pricing out new tours a relatively easy problem (pseudo polynomial). Using 2-loop elimination techniques, first proposed for the CARP in (Benavent et al., 1992), massive cycling on service edges can be prevented. The consequence of allowing non-elementary tours is however that coefficients $\bar{x}_{er}$ of tours are not necessarily binary, but can be non-negative integers. With these additional tour variables in the set-covering formulation, the lower bound of the LP-relaxation is weakened. However, the bounds obtained with the LP-relaxation of (12)–(15) sometimes outperform those of the one-index formulation. The approach is therefore attractive but incomplete as Gómez-Cabrero et al. (2005) do not present a branching scheme which is in general needed to determine an optimal integer solution. The devising of such a branching scheme is one of the major contributions of this paper.

Another set covering-based solution approach was presented, discussed, and empirically analyzed by Letchford and Oukil (2009). The main focus of this paper is on the impact that elementary pricing has on lower bounds. The authors neither extend their set-covering formulation with valid cuts, nor devise a branching scheme to produce integer CARP solutions. The final conclusion that can be drawn from the paper is that elementary routes improve the lower bounds at the cost of a dramatic increase in computation times. Comparing the results of (Letchford and Oukil, 2009) and (Gómez-Cabrero et al., 2005), the contribution of elementary pricing to the quality of the lower bounds is on average smaller than the impact of the cuts.

3. Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition is one of the most successful techniques when it comes to solving vehicle and crew routing and scheduling problems (Desaulniers et al., 2005; Lübbecke and Desrosiers, 2005). The advantages of solving the resulting integer master program follow from (i) a typically stronger lower bound, (ii) the elimination of symmetry in vehicles or crew members, and (iii) the possibility to handle non-linear cost structures for routes and schedules. In the CARP case the first two aspects apply.

In the following, we propose the decomposition of the two-index formulation (1)–(6) together with the valid cuts (11). As in other routing problems, we assume that the covering/partitioning constraints (2) are the coupling constraints. Additionally, the valid cuts (11) (or any subset of active cuts) are coupling constraints.

3.1. Column-Generation Formulation

First we analyze the domain $D = \{(x, y, p) : \text{fulfilling (3)–(6)}\}$ in order to describe the general structure of the pricing problem as well as the extreme points of $D$ that correspond to the variables of the column-generation formulation a.k.a. extensive formulation (see Lübbecke and Desrosiers, 2005). The domain $D$ is
separable by vehicle (index $k$) and can therefore be described as the cartesian product of domains $D^k = \{(x^k, y^k, p^k) : \text{that fulfill (3)--(6)}\}$ for $k \in K$. As vehicles are assumed identical in the CARP, all domains $D^k$ are identical.

Let $X = \text{conv}(D^k)$ be the polyhedron given by the convex hull of integral points in $D^k$. $X$ consists of all convex combinations of its extreme points plus all non-negative linear combinations of its extreme rays (Schrijver, 2003). For the sake of simplicity, we argue with the well-studied directed case (Ahuja et al., 1993) to describe what extreme points and rays of $X$ are in the CARP case.

Modeling constrained directed shortest paths can be done on directed networks, where the nodes represent states (combinations of resource consumptions and nodes), and arcs connect those states resulting from feasible movements (see Irnich and Desaulniers, 2005). When modeling constrained shortest paths, the depot is typically split into a source and a sink node, one unit of flow is sent from source to sink, and flow conservation must hold in all nodes. Here, the set of extreme points consists of all efficient feasible routes. The attribute efficient means that the route does not deadhead along a cycle in $G$. Such a cycle corresponds to a ray of $X$. Extreme rays are the simple deadheading cycles in $G$. Simple means that all nodes of the cycle have degree 2. Thus, any route is composed of an efficient route plus a non-negative combination (possibly null) of simple cycles.

It is clear that optimal solutions to the CARP do not include routes $r$ with deadheading cycles. Hence, only efficient routes are needed for the formulation of the master program. However, we will see later on that the inclusion of simple deadheading cycles of the form $C_e = (e, e)$ for edges $e \in E$ is helpful (see Section 3.4).

We use the identical notation for routes $r \in \Omega$, coefficients of route costs $c_r$, service $\bar{x}_{er}$, deadheading $\bar{y}_{er}$, and cuts $d_{sr}$ as in Section 2.4. The integer master program (IMP) of the CARP reads as follows:

$$\text{min } \sum_{k \in K} c^T \bar{x}_{er} \lambda^k \quad \text{subject to} \quad \sum_{k \in K} \sum_{r \in \Omega} \bar{x}_{er} \lambda^k = 1 \quad \text{for all } e \in E_R$$

$$\sum_{k \in K} \sum_{r \in \Omega} d_{sr} \lambda^k \geq r_s \quad \text{for all } s \in S$$

$$\sum_{r \in \Omega} 1^T \lambda^k = 1 \quad \text{for all } k \in K$$

$$\lambda^k \geq 0 \quad (\in \mathbb{R}^{\mid \Omega \mid}) \quad \text{for all } k \in K$$

$$\bar{x}_{er} = \sum_{r \in \Omega} \bar{x}_{er} \lambda^k, \quad y^k = \sum_{r \in \Omega} \bar{y}_{er} \lambda^k \quad \text{for all } e \in E_R / e \in E, k \in K$$

$$x^k \in \{0, 1\}^{\mid E_R \mid}, y^k \in \mathbb{Z}^{\mid E \mid}_+ \quad \text{for all } k \in K$$

The objective (16) minimizes over the costs of all tours. Equalities (17) ensure that every required edge is covered exactly once. The reformulated cuts are given by (18). Equalities (19) are convexity constraints and require each vehicle to perform a CARP tour. Constraints (21) couple the variables for service and deadheading with the tour variables and constraints (22) ensure the integrality of the solution.

The LP-relaxation of (16)--(22) is the master program (MP). As there is no need to keep the coupling constraints (21) when integrality is relaxed, MP reduces to (16)--(20). Column generation solves MP by iteratively reoptimizing a restricted master program (RMP) over a proper subset of variables (columns) and generating missing variables with negative reduced costs.

### 3.2. Pricing Problem and Relaxations

The task of the pricing problem is exactly the generation of one or several variables with negative reduced cost or proving that no such variable exists. Let dual prices $\pi = (\pi_e)_{e \in E_R}$ to the partitioning constraints (17), $\beta = (\beta_k)_{k \in K}$ to the cuts (18), and $\mu = (\mu^k)_{k \in K}$ to the convexity constraints (19) be given. Omitting the index $k$ of the vehicle, the pricing problem is

$$z_{PP} = \min \bar{c}^\pi x + \bar{c}^π y - \mu \quad \text{s.t.} \quad (3)--(6),$$
where reduced costs for service and deadheading can be associated to the edges:

\[ \tilde{c}_e^{serv} = c_e^{serv} - \pi_e \text{ for all } e \in E_R \quad \text{and} \quad \tilde{c}_e = c_e - \sum_{s \in S} d_{es} \beta_s \text{ for all } e \in E. \] (23)

It is known that the determination of a minimum reduced cost route \( r \in \Omega \) for the CARP is an \( \mathcal{NP} \)-hard problem. Even if practically tractable for small and mid-sized instances, computation times can become long (Letchford and Oukil, 2009). In order to keep the computational effort small, several researchers proposed the use of non-elementary CARP routes. We follow this idea in our cut-first bap-second approach and briefly discuss the impact of a relaxed pricing problem.

In contrast to elementary routes, a non-elementary route \( r \) services at least one of the required edges \( e \in E_R \) more than once, i.e., has coefficient \( x_{er} \geq 2 \). The effect for the MP is the enlargement of the set \( \Omega \) which typically degrades the quality of the lower bound. The advantage is, however, that pricing becomes an easy problem. The currently most efficient non-elementary pricing algorithm was recently presented by Letchford and Oukil (2009) and has worst-case complexity \( \mathcal{O}(Q(|E| + |V| \log |V|)) \), while elementary pricing is \( \mathcal{NP} \)-hard in the strong sense. In fact, pricing elementary or non-elementary routes plays with the tradeoff between the hardness of pricing and the quality of the lower bound resulting in branch-and-bound trees that can significantly differ in size.

Each route \( r \), either elementary or non-elementary, is given as a point \( (\bar{x}_{er}, \bar{y}_{er}, \bar{p}_{er}) \) satisfying the constraints (3)–(6) except for the binary requirements for \( x_{er} \). The point represents a solution in which an edge \( e \) is traversed \( \bar{x}_{er} + \bar{y}_{er} \) times. The corresponding multiple copies of these edges form a Eulerian graph. Since a Eulerian tour is generally not unique, the tour does not automatically imply a particular sequence in which edges are serviced. In the following, we allow a point being represented by one or several corresponding service sequences. Such a sequence arises naturally when routes are determined as shortest paths in pricing (see Section 4.2), which is the only efficient method currently available. More precisely, we write \( s_r = (e_1^r, e_2^r, \ldots, e_{p_r}^r) \) for the sequence in which the required edges \( e_1^r, e_2^r, \ldots, e_{p_r}^r \in E_R \) are serviced by route \( r \).

Whenever consecutively serviced edges are identical, i.e., \( e_i^r = e_{i+1}^r \) for an index \( i \), the route contains a so-called 2-loop. The simplest form of a 2-loop is the subroute \( (i, j, i) \) on a required edge \( e = \{i, j\} \). Opposed to node routing, where the only 2-loops are subpaths \( (i, j, i) \) by any deadheading path between these endpoints. Thus, using the concept of tasks (on edges), in more detail described in (Irnich and Desaulniers, 2005), 2-loop free CARP routes are routes without task 1-cycles. Pricing 2-loop free routes can be done as efficiently as non-elementary pricing. We outline this approach in Section 4.2.

### 3.3. Aggregation

In order to eliminate the symmetry from (16)–(22) with respect to the vehicles, aggregation over \( k \in K \) identical subproblems can be applied. The aggregated service, deadheading, and route variables are

\[ x_e = \sum_{k \in K} x_{ek}^k \text{ for } e \in E_R, \quad y_e = \sum_{k \in K} y_{ek}^k \text{ for } e \in E, \quad \lambda_r = \sum_{k \in K} \lambda_r^k \text{ for } r \in \Omega. \]

This leads to the following aggregated integer master program (agg-IMP):

\[ \begin{align*}
\min & \quad \sum_{r \in \Omega} c_r \lambda_r \\
\text{s.t.} & \quad \sum_{r \in \Omega} \bar{x}_{er} \lambda_r = 1 \quad \text{for all } e \in E_R, \\
& \quad \sum_{r \in \Omega} d_{sr} \lambda_r \geq r_s \quad \text{for all } s \in S \\
& \quad 1^T \lambda = |K|, \\
& \quad \lambda \geq 0, \lambda \in \mathbb{Z}^{[|K|]}.
\end{align*} \] (24–26)

A crucial point in our approach is that aggregation complicates the determination of feasible integer CARP solutions. We assume that a solution \( \bar{\lambda} \) of the LP-relaxation of (24)–(26) is given. Clearly, if all variables \( \lambda \)
are binary, an integer solution of the CARP is found. However, the development of a branching scheme to force a solution to become integral is intricate for the CARP (see also Section 4.3). First, it is not possible to uniquely deduce the disaggregated values of $x^k_e$ and $y^k_e$ from $\lambda$. Moreover, the values of the aggregated service variables are useless as $x^\lambda_e = 1$ holds. Finally, the aggregated deadheading variables $\bar{y}_e$ do not allow the reconstruction of tours, as already discussed for the one-index formulation in Section 2.3.

A second important aspect is that integrality of all variables $y_e$ does not automatically force the tour variables $\lambda$ to be binary. This implies that branching on the aggregated original variables is generally not sufficient to guarantee integrality. It is widely known that branching on the route variables $\lambda_e$ has the disadvantages that it destroys the structure of the pricing problem and branches tend to be highly unbalanced (Villeneuve and Desaulniers, 2005). The effect is a typically untractable pricing problem together with a huge branch-and-bound tree. Instead integrality of agg-IMP must be controlled by additional constraints that are not included in the given formulation (24)–(26).

A first alternative is related to the branching rule that Ryan and Foster (1981) suggested for the LP-relaxation of a set-partitioning problem: in any fractional solution there exist two rows, say $e$ and $e'$, such that the fractional solution does not uniquely determine whether $e$ and $e'$ are covered by the same or by different columns. For the formulation agg-MP, it suffices to have

$$h_{ee'} = \sum_{r \in \Omega} (\bar{x}_{er} \bar{x}_{er}) \lambda_r \in \{0, 1\} \quad \text{for all } e, e' \in E_R.$$  

The variable $h_{ee'}$ indicates whether the two required edges $e, e' \in E_R$ are served separately or by the same tour. These additional binary constraints ensure binary route variables $\lambda$. The condition $h_{ee'} = 0$ is equivalent to $x^k_e + x^k_{e'} \leq 1$ for all $k \in K$ in the two-index formulation, while $h_{ee'} = 1$ is equivalent to $x^k_e = x^{K}_e$ for all $k \in K$. Even if these conditions can be expressed in the original variables, they destroy the structure of the pricing problem.

In our approach we use a second alternative to ensure integrality based on undirected follower information. Compared to Ryan and Foster’s rule it has the advantage that the structure of the pricing problem can be preserved. The follower conditions are given by

$$f_{ee'} = \sum_{r \in \Omega} f_{ee'r} \lambda_r \in \{0, 1\} \quad \text{for all } e, e' \in E_R \tag{27}$$

where $f_{ee'} = |\{1 \leq q < p' : (e, e') = \{e_q, e'_{q+1}\}\}$ counts how often the two edges $e$ and $e'$ are served in succession by route $r \in \Omega$. Note that $f$ is symmetric, i.e., $f_{ee'} = f_{ee'}$ holds.

The follower information suffices to ensure integrality of the route variables $\lambda$ as can be seen as follows: Let $H \subseteq E_R \times E_R$ be the binary relation representing the values $h_{ee'}$, i.e., $(e, e') \in G$ if and only if $h_{ee'} = 1$. Similarly, let $F$ be the follower relation with $(e, e') \in F$ if and only if $f_{ee'} = 1$. Then $H$ is the reflexive and transitive closure of $F$. Hence, binary values of $f_{ee'}$ imply binary values of $h_{ee'}$, and therewith binary route variables $\lambda_r$. The algorithmic procedures that impose follower ($f_{ee'} = 1$) and non-follower ($f_{ee'} = 0$) conditions to the master and pricing problems are explained in Section 4.3.

3.4. Dual-Optimal Inequalities

The equation (23) for the reduced costs of deadheading along the edge $e \in E$ shows that for large values of dual prices $\beta$, the reduced cost $\bar{c}_e$ might become negative. Thus, an extreme ray corresponding to the cycle $C_e = (e, e)$ must be priced out.

Conversely, if an additional variable which represents that cycle $C_e$ is already present in agg-MP, its reduced cost must be non-negative for any RMP solution. This implies that also the reduced costs $\bar{c}_e$ are non-negative because $2\bar{c}_e$ is the reduced cost of the cycle $C_e$.

The goal of this subsection is to briefly discuss the theoretical implications of adding variables for cycles $C_e$. Let $z_e \geq 0$ be the variable that represents the cycle $C_e = (e, e)$ corresponding to an extreme
ray of the pricing polyhedron $X$. The addition of $z_e$ to the LP-relaxation of agg-MP leads to the following extended aggregated master program (eMP):

$$\begin{align*}
\min & \quad \sum_{r \in \Omega} c_r \lambda_r + \sum_{e \in E} (2c_e)z_e \\
\text{s.t.} & \quad \sum_{r \in \Omega} \bar{x}_{er} \lambda_r = 1 \quad \text{for all } e \in E_R \\
& \quad \sum_{r \in \Omega} d_{sr} \lambda_r + \sum_{e \in E} (2d_{es})z_e \geq r_s \quad \text{for all } s \in S \\
& \quad 1^T \lambda = |K| \\
& \quad \lambda \geq 0, z \geq 0
\end{align*}$$

(28) (29) (30) (31) (32)

Obviously, deadheading twice through $e$ produces a cost of $2c_e$ in (28), it has no impact on partitioning (29), but can have non-zero coefficients $2d_{es}$ in the cuts (30). As $C_e$ is an extreme ray, it has coefficient zero in the generalized convexity constraint (31).

The additional variables $z_e$ allow extended $k$-routes (Belenguer and Benavent, 1998) in eMP, the primal problem, and correspond to inequalities in the dual problem. In fact, the presence of $z_e$ in eMP gives a dual inequality of the form

$$\sum_{s \in S} d_{es} \beta_s \leq c_e \quad \text{for all } e \in E.$$ 

The concept of dual-optimal inequalities was first presented by Ben Amor et al. (2006). In our case, the positive impact of the dual cuts is twofold:

1. The reduced costs of deadheading edges are non-negative which provides algorithmic advantages in every pricing iteration as will be shown in Section 4.2.
2. The dual variables $\beta_s$ are stabilized because their values are restricted by the dual-optimal inequalities.

The result is a typically faster convergence of the column generation process (du Merle et al., 1999; Ben Amor et al., 2006).

4. Cut-First Branch-and-Price-Second Approach

An outline of the overall cut-first bap-second approach is shown in Algorithm 1. Its three key components are the cut generation procedure, the pricer, and the branching scheme. These components will be explained in the following.

**Algorithm 1: Cut-First Branch-and-Price-Second Algorithm**

1. Solve LP-relaxation of one-index formulation (7)–(10) with a cutting-plane algorithm
2. Identify binding cuts (odd cuts, capacity cuts, disjoint-path inequalities DP1, DP2, DP3) and odd/capacity cuts with rhs>0 for singleton sets $S = \{i\}$ with $i \in V \setminus \{d\}$; the index set of these cuts is $S$
3. Solve extended aggregated integer master program eMP (28)–(32) with branch-and-price; use set $S$ from step 2 for valid inequalities (30)

4.1. Cutting

Before starting the bap phase, the one-index formulation (7)–(10) is solved with a cutting-plane algorithm in order to identify the cuts that are finally binding. Details about the particular separation procedures, the order in which they are invoked, the number of separated and finally binding cuts, and computation times are given in the appendix.

Here, we briefly sketch the separation routines. To find violated odd cuts (9) the efficient separation algorithm of Letchford et al. (2008) is applied. Capacity inequalities (8) are separated using the heuristic algorithm of Belenguer and Benavent (2003) and an exact MIP-based algorithm of Ahr (2004). In both
cases, the minimum number of vehicles $K(S)$ is approximated by $\lceil q(E_R(S) \cup \delta_R(S))/Q \rceil$. Disjoint-path inequalities (of type dp1, dp2 and dp3) were presented by Belenguer and Benavent (2003) together with a rather complex heuristic scheme to separate violated inequalities. We implemented a similar heuristic trying to be as close as possible to the original algorithm.

Only those cuts that are finally binding in the one-index formulation (plus odd cuts for singleton sets) are active in the eMP. No additional cuts are added later in the column-generation procedure even if non-binding or other cuts might become violated in the branch-and-bound tree. Conversely, non-binding cuts are not eliminated from eMP.

4.2. Pricing

For pricing out non-elementary routes, Letchford and Oukil (2009) proposed labeling algorithms that work on the original CARP graph $G$ and so exploit the sparsity of the network. They introduce both an exact and a heuristic pricer that consider labels with identical capacity consumption $q$ in the sequence $q = 0, 1, 2, \ldots, Q$. Both algorithms have two types of path-extension steps (see Irnich and Desaulniers, 2005, for details):

1. An extension with service on a required edge $e \in E_R$ always creates new labels where the consumed capacity increases by $q_e > 0$. The reduced cost $\tilde{c}_e^{extv}$ is added to the cost component.
2. An extension along a deadheading edge $e \in E$ does not alter the capacity consumed and adds the value $\tilde{c}_e$ to the cost of the partial path. The dual inequalities of the eMP guarantee that $\tilde{c}_e \geq 0$ holds. All extensions over deadheading edges (for a given capacity consumption $q$) can be performed together using the Dijkstra algorithm.

As a result, the overall time complexity of the exact pricing routine is $O(Q(|E| + |V| \log |V|))$.

We adapted the presented pricing routines in order to eliminate 2-loops. This technique is straightforward following the ideas presented in (Houck et al., 1980; Benavent et al., 1992). The resulting worst-case time complexity is still $O(Q(|E| + |V| \log |V|))$.

4.3. Branching

Let $\lambda$ be a fractional solution to eMP at a branch-and-bound node with associated values $\tilde{x}$ and $\tilde{y}$ (eqs. (21)). To obtain an integer solution a branching scheme has to be devised. Our hierarchical branching scheme consists of three levels of decisions:

1. branching on node degrees
2. branching on edge flows
3. branching on followers and non-followers

The idea behind this scheme is that decisions from the first two levels are more global decisions that typically have a stronger impact on the lower bounds. The decisions of the third level are more local, but they alone guarantee integrality (see Section 3.3).

First, if there exists a node $i \in V$ with node degree $d_i = \tilde{x}(\delta(i)) + \tilde{y}(\delta(i))$ not even (either fractional or odd), the two branches $x(\delta(i)) + y(\delta(i)) \leq 2p$ and $x(\delta(i)) + y(\delta(i)) > 2p + 2$ are created with $p \in \mathbb{Z}_+$ defined by $2p < d_i < 2p + 2$. Second, if for an edge $e \in E$ the edge flow $\tilde{\phi}_e = \tilde{x}_e + \tilde{y}_e$ is fractional, the two branches $x_e + y_e \leq \lceil \tilde{\phi}_e \rceil$ and $x_e + y_e > \lceil \tilde{\phi}_e \rceil + 1$ are generated. Both types of branching decisions only have an impact on the master program, where a linear constraint must be added. This constraint has the same form as the cuts (30). Consequently, the equations (23) can still be used to compute the reduced cost of service and deadheading edges. Third, if for any two required edges $e$ and $e'$ the follower information $f_{ee'}$ (see eq. (27)) is fractional, two branches with constraints $f_{ee'} = 0$ and $f_{ee'} = 1$ are induced. This means that all routes variables $\lambda$ not respecting the constraint are removed from eMP. Moreover, no routes violating these decisions must be priced out. We guarantee compatible routes by modifying the underlying graph on which pricing is carried out. The network modifications that we describe in Section 4.3.2 do not destroy the structure of the pricing problem.
The specific variable to branch on is determined as follows. For branching on node degrees, we first compute for each node \( i \in V \) the distance of \( \bar{d}_i \) to the next even integer, i.e., \( \min\{2p + 2 - \bar{d}_i, \bar{d}_i - 2p\} \) for an integer \( p \) with \( 2p \leq \bar{d}_i < 2p + 2 \). We select the node \( i^* \) for which

\[
\frac{\min\{2p + 2 - \bar{d}_i, \bar{d}_i - 2p\}}{\alpha + \beta 2p}
\]

is maximal. We experimented with different values for \( \alpha \) and \( \beta \). For example, \( \alpha = 1 \) and \( \beta = 0 \) chooses the largest absolute distance of the node degree to the next even integer. In the final implementation we have chosen \( \alpha = 6 \) and \( \beta = 1 \) and ties are broken arbitrarily. The idea of this rule is that the distance of \( \bar{d}_i \) to the next even integer is biased towards selecting nodes with a smaller node degree. For branching on edge flows, an edge \( e^* \) with fractional flow \( \bar{y}_{e^*} \) closest to 0.5 is chosen.

In order to describe the rule for the third level, we partition the set \( E_R \) of required edges. The active branch-and-bound constraints induce the follower relation \( F \) and the non-follower relation \( N \), i.e., \( F = \{(e, e') : f_{ee'} \text{ is fixed to } 1\} \) and \( N = \{(e, e') : f_{ee'} \text{ is fixed to } 0\} \). The reflexive and transitive closure of \( F \cup N \) defines the partitioning \( E_R = E_R^1 \cup E_R^2 \cup \ldots \cup E_R^m \). The subsets \( E_R^k \) characterize which required edges are directly or indirectly connected by active follower and non-follower constraints. If there exists no follower or non-follower relation for an edge \( e \in E_R \), the subset consists of that edge alone.

For the selection of a variable \( f_{e, e^*} \), we search for the pair \((e^*, e'^*)\) with fractional value \( f_{e, e^*} \) closest to 0.5 inducing a subset of size less than or equal to 5. If \( e^* \) and \( e'^* \) are already in the same subset, the addition of an associated constraint will not alter the given partitioning. Otherwise, the two subsets of \( e^* \) and \( e'^* \), say \( E_R^k \) and \( E_R^m \) are merged resulting in a single subset of size \( |E_R^k| + |E_R^m| \). If no induced subset is of size less than or equal to 5, we only consider pairs with smallest induced subset and proceed as described before. The idea behind this rule is that small subsets are algorithmically attractive because we have to enumerate permutations of their edges to map branching decisions to the pricing network (see Section 4.3.2).

4.3.1. Integer Solutions from Follower Information

The analysis undertaken in Section 3.3 has shown that branching on followers alone guarantees binary master program variables \( \lambda \). A crucial point in the proof is that routes were assumed being elementary. In fact, for relaxed pricing with 2-loop free routes the variables \( \lambda \) can still be fractional. A detailed example is provided in the appendix.

With a relaxed pricing, it can happen that all follower variables \( f_{e, e^*} \) are binary but route variables \( \bar{\lambda} \) are still fractional. In this case, nevertheless, an integer solution can implicitly be obtained from the fractional master program eMP. In fact, the binary follower information implies a unique partitioning and sequences of required edges that can be utilized to construct a solution.

We assume that the relation \( F \) is given by those pairs of required edges that are followers in the eMP, i.e., fulfill \( f_{e, e'^*} = 1 \). The reflexive and transitive closure \( \bar{F} \) of the follower relation \( F \) defines exactly \( |K| \) subsets of the required edges, i.e., \( E_R = \bigcup_{k \in K} E_R^k \). A required edge \( e \in E_R \) can be in follower relation to either no other edge, to a single edge, or to exactly two edges. In other words, the graph \((E_R, \bar{F})\) has exactly \( |K| \) components consisting of paths (possibly with length 0). Hence, \( F \) implies for each subset \( E_R^k \) a sequence \( s_k = (e_1^k, e_2^k, \ldots, e_\ell^k) \) of required edges. This sequence is unique except for reversal.

The final step is the determination of cost-minimal routes \( r_k \) that service the required edges exactly in the sequence \( s_k \). This task consists of two types of decisions, the identification of deadheading paths connecting subsequent required edges and the fixing of directions in which required edges are serviced. A cost-minimal route can be computed as a shortest path in the auxiliary network depicted in Figure 1. We assume that for each pair of nodes a shortest deadheading path between these nodes is pre-computed such that \( c_{ij} \) is the shortest deadheading distance between the nodes \( i \) and \( j \). Recall that \( c_{e, e'^*} \) is the cost for servicing a required edge \( e \). The auxiliary network consists of two copies for each required edge \( e_1, \ldots, e_t \) modeling the two possible directions when servicing. Starting and ending at nodes representing the depot \( d \), dashed arcs model the deadheadings between the required edges. Thus, the length of a path in the auxiliary network is exactly the cost of the shortest route that services \( e_1, \ldots, e_t \) in the given sequence \( s_k \). The appendix provides a detailed example.
4.3.2. Network Modifications

This section explains how follower and non-follower constraints can be handled in the pricing problem. The key property of our approach is that the active constraints can solely be implemented by network modifications. More precisely, only deletions and additions of required edges are performed on the original pricing network \( G = (V, E) \). The basic structure of the pricing problem remains unchanged. We assume that active branch-and-bound constraints are given by the follower relation \( F = \{(e, e') : f_{ee'} \text{ is fixed to 1}\} \) and the non-follower relation \( N = \{(e, e') : f_{ee'} \text{ is fixed to 0}\} \).

For the sake of clarity, we first consider a single follower or non-follower constraint \((F \text{ or } N = \{(e, e')\})\). Afterwards the general case with multiple active constraints will be described.

On the non-follower branch \( f_{ee'} = 0 \), servicing edge \( e \) immediately followed by edge \( e' \) is forbidden. Note that in the undirected network \( G = (V, E) \) all constraints are symmetric so that \( e \) and \( e' \) can be interchanged and that the two services do not need to be directly connected but can be connected with any deadheading path. This constraint can be implemented using the concept of task-2-loop free paths as presented in (Irnich and Desaulniers, 2005; Irnich and Villeneuve, 2006). All required edges represent different tasks except for \( e \) and \( e' \) which represent the same task. Any task-2-loop free path ensures that the non-follower constraint \( f_{ee'} = 0 \) is respected.

On the follower branch \( f_{ee'} = 1 \), the service of edge \( e \) must immediately follow the service of edge \( e' \) (or vice versa). First, the two original required edges are deleted from the network (deadheading along \( e \) and \( e' \) remains possible). Second, four new edges are added to the network. They model the consecutive service to \( e = \{i, j\} \) and \( e' = \{k, l\} \). Since the direction of service is unknown for both edges, there are four possible paths:

- path \( p_1 = (i, j) + \text{deadheading from } j \text{ to } k + (k, l) \)
- path \( p_2 = (i, j) + \text{deadheading from } j \text{ to } l + (l, k) \)
- path \( p_3 = (j, i) + \text{deadheading from } i \text{ to } k + (k, l) \)
- path \( p_4 = (j, i) + \text{deadheading from } i \text{ to } l + (l, k) \)

The four additional edges \( \{i, l\}, \{i, k\}, \{j, l\} \) and \( \{j, k\} \) represent these paths. Consequently, they all have the same resulting demand \( q_e + q_{e'} \) and the same associated task. The latter implies that no two of these four edges can be served consecutively. The costs of the new edges is calculated by summing up serviced costs \( \tilde{c}_{ee'} + \tilde{c}_{ee'} \) plus the costs for deadheading along the respective paths.

The modifications become even more intricate if several follower and non-follower conditions are active. In general the proceeding is outlined in Algorithm 2:

The computation of a minimum-cost path \( p \) is similar to the shortest-path computation in the auxiliary network described in 4.3.1. We just elaborate the differences: First, costs \( c_e \) and \( c_{ee'} \) have to be replaced by reduced costs \( \tilde{c}_e \) and \( \til\til{c}_{ee'} \) defined by (23). Note that dual-optimal inequalities (see Section 3.4) guarantee that all reduced deadheading costs are non-negative. Therefore, the distances \( \til\til{c}_{ij} \) of the shortest deadheading paths can be computed with the Dijkstra algorithm (between every pair of nodes \( i \) and \( j \)). Second, the path \( p \) starts in node \( i \) and ends in node \( j \). Hence, the deadheading part from the depot \( d \) to the first required edge \( e_1 \) and from the last required edge \( e_t \) to the depot \( d \) is omitted in the auxiliary network. Moreover,
Algorithm 2: Network modification for follower and non-follower constraints

**input**: A network with costs \( \tilde{c}_{e}^{\text{serv}} \) and \( \tilde{c}_{e} \), active follower and non-follower relations \( F \) and \( N \)

**output**: A modified network

Compute the partitioning \( E_{R}^{1} \cup E_{R}^{2} \cup \ldots \cup E_{R}^{q} \) of \( E_{R} \) induced by \( F \) and \( N \)

for \( \ell = 1, \ldots, q \) do

if \( |E_{R}^{\ell}| > 1 \) then

Delete all required edges \( e \in E_{R}^{\ell} \) from the network

Compute all feasible sequences \( s = (e_{1}, e_{2}, \ldots, e_{t}) \) on all subsets of edges in \( E_{R}^{\ell} \)

for all sequences \( s = (e_{1}, e_{2}, \ldots, e_{t}) \) do

for the four pairs \( (i, j) \) of endpoints of \( e_{1} = \{i_{1}, i_{2}\} \) and \( e_{t} = \{j_{1}, j_{2}\} \) do

Compute the minimum-cost path \( p \) servicing sequence \( s \)

\( p \) starts at node \( i \) and ends in node \( j \)

Add a required edge \( \{i, j\} \) to the network

with demand \( \sum_{m=1}^{t} q_{em} \), associated task is \( \ell \)

The directions of \( e_{1} \) and \( e_{t} \) are fixed in each iteration (third for-loop). A detailed example of the network modification is included in the appendix.

The enumeration of all possible sequences \( s \) may produce a network with an exponential number of edges. By selecting edges in the follower branching rule carefully, we try to keep the resulting subsets small. Therewith, we postpone the exponential growing of the network to some deeper branches of the tree (that were not reached in our experiments).

A final remark is that parallel edges might result from feasible sequences \( s \) that have identical first and last edges. In this case, for identical demand \( \sum q_{ep} \), a single cost-minimal edge can be chosen while the other edges can be discarded.

5. **Cycle Elimination**

It is known for a long time that for routing problems cycle-elimination techniques can improve lower bounds of master programs (Houck et al., 1980; Feillet et al., 2004; Irnich and Villeneuve, 2006; Desaulniers et al., 2008). For the CARP, Letchford and Oukil (2009) analyzed the impact of elementary CARP routes. Although their study shows that elementary routes can improve lower bounds, a comparison with the cutting-plane approaches of Benavent et al. (1992), Ahr (2004), and Gómez-Cabrero et al. (2005) did not reveal the full potential of cycle elimination. The reason is that no odd-cut inequalities (9), capacity inequalities (8), and disjoint-path inequalities were present in the column-generation formulation. While Section 6.1 will quantify lower bound improvements due to cycle elimination, this section focuses on the interplay between cycle elimination and branching from a conceptual point of view.

In Section 3.2, task-2-loop elimination was introduced to improve the master program lower bound. Task-2-loop (i.e., task-1-cycle) elimination excludes the occurrence of two consecutive required edges labeled with the same task. Thus, by giving edges the same task the non-follower branching rule was implemented (see Section 4.3.2). The crucial point is here that branching requires task-\( k \)-loop exactly for \( k = 2 \). For \( k > 2 \), the non-follower branching rule would be incorrect as also not direct following occurrences of the two tasks would be excluded.

In conclusion, branching only works with task-2-loop elimination while using task-\( k \)-loop for \( k > 2 \) helps to improve master program lower bounds. At first glance, both requirements (\( k = 2 \) and \( k \) as large as possible) seem incompatible in the presented bap approach. However, we can resolve this incompatibility by differentiating between the two corresponding classes of tasks:

- tasks \( T_{F}^{E} \) for modeling the elementary routes
• tasks \( T^B \) for branching

In essence, a shortest-path problem where paths are elementary w.r.t. \( T^E \) and task-2-loop free w.r.t. \( T^B \) must be solved. Relaxations to the elementarity w.r.t. \( T^E \) can again play with the tradeoff between hardness of the problem and strength of the relaxation. We outline the meaning and handling of the two task sets in the following.

For tasks \( T^E \), each required edge \( e \in E_R \) forms an individual task so that the sets can be considered identical (\( T^E = E_R \)). Branching does not modify this definition. A branch with a follower constraint, however, triggers a network modification: if a sequence of required edges is modeled by one or four new edges (see Section 4.3.2) such edges get the associated task sequence. The concept of task sequences associated to arcs (and nodes) has already been sketched in (Irnich and Desaulniers, 2005, p. 40).

For tasks \( T^B \), only active non-follower constraints have to be considered and cause the insertion of tasks into the set \( T^B \). More precisely, only those tasks \( \ell \) that are assigned to the new edges modeling a sequence through the same component \( E^B_\ell \) (see last step of Algorithm 2) have to be included in \( T^B \). Consequently, the task set \( T^B \) is empty at the root node of the bap tree. The \( k \)-cycle elimination method, as presented in (Irnich and Villeneuve, 2006), are therefore applicable when the root in bap is solved.

Our ongoing research is on combining \( T^B \)-2-loop and \( T^E \)-\( k \)-cycle elimination (and is going to be presented in a separate paper). It is obvious that not only \( k \)-cycle elimination but also alternative relaxations for \( T^E \)-elementarity are promising, e.g., partial elementarity or NG-route relaxations (Desaulniers et al., 2008; Baldacci et al., 2009).

6. Computational Results

We tested our cut-first bap-second approach on the four standard benchmark sets for the CARP. The same benchmark sets (or a subset of instances) was also used to analyze the methods presented in the review (Section 2). The complete data with additional information can be downloaded from http://www.uv.es/~belengue/carp/. The two benchmark sets kshs and gdb contain 6 and 23 artificially generated instances. While the underlying graphs are sparse for the gdb set, some kshs instances are defined over a complete graph. Two other benchmark sets, bccm and eg1, have 34 and 24 instances, respectively. The latter set is based on parts of the road network of Cumbria.

The following results were performed on a standard PC with an Intel(R) Core(TM) i7-2600 at 3.4 GHz with 16 GB of main memory. The algorithm was coded in C++ and the callable library of CPLEX 12.2 was used to iteratively solve LPs and MIPs (reoptimization and separation) in the cutting phase and to reoptimize the RMP.

For the computational analysis we use the following notation:

- \( lb_* \): lower bound obtained by algorithm of *;
- \( BB \): cutting-plane algorithm of Belenguer and Benavent (2003); A= Ahr (2004); L= linear relaxation (root) by Longo et al. (2006); LO= linear relaxation (root, elementary routes) by Letchford and Oukil (2009); MPS= dual ascent and cutting-plane algorithm of Martinelli et al. (2011)
- \( lb_{\text{root}} \): lower bound obtained in our linear relaxation (root) or at termination of phase II (tree)
- \( lb_{\text{tree}} \): best known upper bound
- \( ub_{\text{best}} \): first paper where the currently best upper bound was computed; H= Hertz et al. (2000); La= Lacomme et al. (2001); BE= Brandão and Eglese (2008); PD= Polacek et al. (2008); SCC= Santos et al. (2010); BLC= Baldacci et al. (2010); Be= Beullens et al. (2003)
- \( opt \): value of optimal solution
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bound
- \( UB \): upper bounds
For the sake of brevity, all computational results regarding phase I (cutting-plane algorithm) are presented in the appendix.

6.1. Linear Relaxation Results

For phase II, the following two column-generation acceleration techniques were implemented: Before calling the exact pricer, we apply the heuristic of Letchford and Ouikil (2009) (consider only tours with consecutive services) and a pricing heuristic where weakly dominant labels are ignored (see Irnich and Desaulniers, 2005, p. 58). Moreover, the partitioning constraints (29) in eMP are replaced by covering constraints (≥ 1) together with the constraint $\sum_{r \in \Omega} \sum_{e \in E} x_{er} \lambda_r \leq |E_R|$ in order to stabilize the column generation process (du Merle et al., 1999).

We begin with the presentation of the results on the linear relaxation of the master program eMP (28)-(32). These results are shown in the first three sections of the Tables 1-2 (results for the small-sized benchmark sets kshs and gdb are presented in the appendix).
Table 1: Results for the **bccm** instances at the end of phase II.
| instance | | | \( |D_{B}| \) | \( |E| \) | \( |K| \) | \( D_{B,B} \) | \( D_{A,A} \) | \( D_{B,O} \) | \( D_{B,O}^{\text{opt}} \) | \( D_{B,O}^{\text{phase I}} \) | \( D_{B,O}^{\text{phase II}} \) | \( D_{B,O}^{\text{comp. by comp.}} \) | \( |E|_{\text{comp. by comp.}} \) | \( \text{time phase I} \) | \( \text{time phase II} \) | \( |D_{B}| \) | \( |E| \) | \( |K| \) | \( |D_{B}| \) | \( |E| \) | \( |K| \) |
| e1-a | 77 | 98 | 5 | 3515 | 3516 | 3548 | 3425 | 3527 | 3548 | 3545 | - | 3545 | La | Lo | 49.3 | 1572 | 135 | 35/0/79 |
| e1-b | 77 | 98 | 7 | 4436 | 4436 | 4468 | 4291 | 4463 | 4498 | 4464 | - | 4498 | La | BM | 102.5 | 1733 | 617 | 588/12/16 |
| e1-c | 77 | 98 | 10 | 5453 | 5481 | 5542 | 5472 | 5513 | 5595 | 5523 | 5545 | - | 5525 | La | BLC | 145.5 | 4248 | 3102/291/852 |
| e2-a | 77 | 98 | 7 | 4994 | 4963 | 5011 | 4832 | 4995 | 5012 | 4995 | - | 5018 | La | BM | 25.3 | 1966 | 312 | 268/4/38 |
| e2-b | 77 | 98 | 10 | 6249 | 6271 | 6280 | 6105 | 6273 | 6284 | 6272 | 6301 | - | 6317 | BE | - | 56.9 | 3822 | 1996/641/1184 |
| e2-c | 77 | 98 | 14 | 8114 | 8155 | 8234 | 8187 | 8165 | 8335 | 8202 | 8244 | - | 8325 | BE | BLC | 118.9 | 5921 | 5915/0/0 |
| e3-a | 77 | 98 | 7 | 4994 | 4963 | 5011 | 4832 | 4995 | 5012 | 4995 | - | 5018 | La | BM | 38.3 | 2041 | 3042/294/852 |
| e3-b | 77 | 98 | 10 | 6249 | 6271 | 6280 | 6105 | 6273 | 6284 | 6272 | 6301 | - | 6317 | BE | BLC | 118.9 | 5921 | 5915/0/0 |
| e3-c | 77 | 98 | 17 | 10019 | 10119 | 10163 | 10036 | 10138 | 10244 | 10144 | 10191 | - | 10929 | PD | - | 51.2 | 3822 | 1996/641/1184 |
| e4-a | 77 | 98 | 9 | 6372 | 6378 | 6395 | 6233 | 6373 | 6388 | 6388 | 6388 | - | 6388 | La | Lo | 38.3 | 2041 | 3042/294/852 |
| e4-b | 77 | 98 | 14 | 8809 | 8838 | 8884 | 8787 | 8838 | 8935 | 8852 | 8892 | - | 8961 | BE | BLC | 19.4 | 3822 | 1996/641/1184 |
| e4-c | 77 | 98 | 19 | 10025 | 10091 | 10143 | 10036 | 10138 | 10244 | 10144 | 10191 | - | 10929 | PD | - | 51.2 | 3822 | 1996/641/1184 |
| s1-a | 140 | 190 | 7 | 4992 | 4975 | 5014 | 4985 | 5010 | 5018 | 5011 | - | 5018 | La | BM | 208.3 | 8462 | 100 | 73/3/23 |
| s1-b | 140 | 190 | 10 | 6201 | 6180 | 6379 | 6284 | 6368 | 6368 | 6388 | 6388 | - | 6388 | BE | BLC | 454.5 | 364 | 360/3/1 |
| s1-c | 140 | 190 | 14 | 8310 | 8280 | 8400 | 8342 | 8404 | 8518 | 8417 | 8441 | - | 8518 | La | BLC | 605.3 | 436 | 436/0/0 |
| s2-a | 140 | 190 | 14 | 9780 | 9718 | 9824 | 9667 | 9737 | 9825 | 9790 | 9803 | - | 9884 | SCC | - | 283.1 | 434 | 434/0/0 |
| s2-b | 140 | 190 | 20 | 12286 | 12835 | 12968 | 12801 | 12968 | 12968 | 12970 | 13100 | - | 13100 | BE | - | 1136.3 | 2344 | 2344/0/0 |
| s2-c | 140 | 190 | 27 | 16221 | 16216 | 16353 | 16262 | 16274 | 16425 | 16343 | 16352 | 16425 | BE | BLC | 2154.6 | 2640 | 2615/0/0 |
| s3-a | 140 | 190 | 15 | 10025 | 9991 | 10143 | 9925 | 10083 | 10145 | 10143 | 10160 | - | 10220 | SCC | - | 927.8 | 408 | 408/0/0 |
| s3-b | 140 | 190 | 22 | 13554 | 13520 | 13616 | 13508 | 13598 | 13631 | 13628 | 13682 | - | 13682 | PD | - | 1291.5 | 1519 | 1519/0/0 |
| s3-c | 140 | 190 | 29 | 16969 | 16958 | 17100 | 17014 | 17019 | 17188 | 17058 | 17097 | 17188 | BLC | BLC | 2750.9 | 4019 | 3997/0/0 |

Table 2: Results for the egl instances at the end of phase II.
The focus of the following analysis is on lower bounds. The lower bounds $lb_{\text{root}}$ obtained with our column-generation algorithm in the root node are always at least as good as the bounds $lb_A$ obtained by Ahr (2004). Compared to the lower bounds $lb_{BB}$ of Belenguer and Benavent (2003), the values $lb_{\text{root}}$ are sometimes worse (for the bccm instances 2b and 6b as well as eg1 instance s2-a). We suspect that this is due to the fact that the separation of disjoint-path inequalities in phase I is performed with a complex heuristic that certainly differs at some points from the one used to compute $lb_{BB}$ (see appendix for the discussion of the phase I results). Conversely, for several instances better lower bounds were obtained by column generation, i.e., $lb_{\text{root}} > \max\{lb_{BB}, lb_A\}$. This is the case for one kshs instance (kshs4), two gdb instances (gdb8 and gdb12), nine bccm instances, and all eg1 instances except for s2-a.

Comparing the pure set-partition formulation (without additional cuts) of Letchford and Oukil (2009) with our approach, $lb_{LO}$ never exceeds $lb_{\text{root}}$ except for the two eg1 instances e4-c and s1-c. It seems that the addition of cuts to eMP is most of the time more important than using elementary routes alone.

Longo et al. (2006) presented detailed lower bound results only for bccm and eg1 instances. Their lower bounds obtained in the root-node of the CVRP branch-and-price-and-cut algorithm are generally better than our bounds (7 times for bccm and always for eg1 except for s3-a with an identical bound). For the bccm instances 1c, 3c, and 5b, however, $lb_{\text{root}}$ exceeds $lb_{L}$.

Some concluding remarks on the comparison with (Longo et al., 2006) can be given: Our computation times for root nodes are significantly smaller than those for the CVRP, sometimes by more than factor 100 (for example bccm instance 1c). Since the branch-and-price-and-cut algorithm of Longo et al. (2006) is based on the algorithm by Fukasawa et al. (2006), CVRP pricing problems are solved with (node-)k-cycle elimination for $k \in \{2, 3, 4\}$. This partially explains their better lower bounds at the cost of a more complex and time-consuming pricing.

Finally, the strongest relaxation bound $lb_{BCL}$ obtained by Bartolini et al. (2011) almost always exceeds $lb_{\text{root}}$ as their relaxation is stronger. Elementary routes and different types of cuts (including the non-robust subset-row inequalities on the master program) are combined in the approach. The only cuts not used in their implementation are disjoint-path inequalities. This fact can explain why for the two bccm instances 1c and 5b $lb_{BCL} < lb_{\text{root}}$ holds.

Cycle Elimination Result

Recall from Section 5 that we can provide results for pricing with k-loop elimination only for the linear relaxation eMP due to the incompatibility of the branching scheme with k-loop elimination for $k \geq 3$. Table 3 summarizes the impact of cycle elimination on the root node lower bound and the root node computation time. We present results only for the eg1 instances because these are the only ones where cycle elimination had a significant impact. We suspect that the presence of non-required edges conveys the appearance of cycles.

As could be expected, the results exactly reflect the trade-off between lower bound improvement and hardness of solving the respective linear relaxation. For $k = 5$-loop elimination, solving eMP becomes time consuming (in one case more than 10 hours). On average, the increase of the lower bound is 11.0 going from $k = 2$ to $k = 3$, 5.4 from $k = 3$ to $k = 4$, and 2.8 from $k = 4$ to $k = 5$. While for some instances the increase is marginal, it can become substantial (e.g. for e2-c an overall increase of $25 + 36 + 6 = 67$). On the downside, average computation times increase also by factor 2.3 from $k = 2$ to $k = 3$, 3.0 from $k = 3$ to $k = 4$, and 12.9 from $k = 4$ to $k = 5$.

For the instance e1-a, the 5-loop elimination entirely closes the integrality gap (remaining gap = 0). In seven other cases, 4-loop or 5-loop elimination gives a stronger root node relaxation than the CVRP root node relaxation computed in (Longo et al., 2006) (e1-b, e2-b, e2-c, e3-b, e3-c, e4-c, s1-a, s3-a, and s4-c; see also Table 2). Interestingly, these are often instances with relatively small computation times.

Concluding, the computational results indicate that the use of cycle-free routes is one of the key devices to improve lower bounds. In addition to the study of Letchford and Oukil (2009) it has become clear now that loop-elimination is still beneficial when cutting planes are already added to the eMP.
Table 3: Cycle Elimination for the egl instances.

<table>
<thead>
<tr>
<th>name</th>
<th>ub</th>
<th>opt</th>
<th>$\Delta_{best}$</th>
<th>$\Delta_{opt}$</th>
<th>time</th>
<th>phase II</th>
<th>$\Delta_{best}$</th>
<th>$\Delta_{opt}$</th>
<th>time</th>
<th>phase II</th>
<th>$\Delta_{best}$</th>
<th>$\Delta_{opt}$</th>
<th>time</th>
<th>phase II</th>
<th>remain.</th>
<th>gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1-a</td>
<td>3548</td>
<td>3548</td>
<td>3546 (+1)</td>
<td>3546 (+0)</td>
<td>368</td>
<td>3548 (+2)</td>
<td>2575</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e1-b</td>
<td>4498</td>
<td>4464</td>
<td>4465 (+1)</td>
<td>4467 (+2)</td>
<td>196</td>
<td>4470 (+3)</td>
<td>2474</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e1-c</td>
<td>5595</td>
<td>5523</td>
<td>5528 (+5)</td>
<td>5532 (+4)</td>
<td>56</td>
<td>5535 (+3)</td>
<td>1127</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2-a</td>
<td>5018</td>
<td>4996</td>
<td>4996 (+0)</td>
<td>4999 (+3)</td>
<td>1247</td>
<td>4999 (+0)</td>
<td>22301</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2-b</td>
<td>6317</td>
<td>6273</td>
<td>6280 (+7)</td>
<td>6283 (+3)</td>
<td>272</td>
<td>6283 (+0)</td>
<td>3611</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e3-a</td>
<td>8335</td>
<td>8202</td>
<td>8227 (+25)</td>
<td>8263 (+36)</td>
<td>55</td>
<td>8269 (+6)</td>
<td>1083</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e3-b</td>
<td>5898</td>
<td>5894</td>
<td>5895 (+1)</td>
<td>5895 (+0)</td>
<td>991</td>
<td>5895 (+0)</td>
<td>25083</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e3-c</td>
<td>7775</td>
<td>7684</td>
<td>7699 (+15)</td>
<td>7704 (+5)</td>
<td>190</td>
<td>7709 (+5)</td>
<td>4246</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4-a</td>
<td>10292</td>
<td>10145</td>
<td>10176 (+31)</td>
<td>10182 (+6)</td>
<td>65</td>
<td>10183 (+1)</td>
<td>1169</td>
<td>109</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4-b</td>
<td>8961</td>
<td>8852</td>
<td>8862 (+10)</td>
<td>8865 (+3)</td>
<td>244</td>
<td>8870 (+5)</td>
<td>2195</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4-c</td>
<td>11562</td>
<td>11411</td>
<td>11438 (+27)</td>
<td>11463 (+25)</td>
<td>159</td>
<td>11465 (+2)</td>
<td>1774</td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1-a</td>
<td>5018</td>
<td>5011</td>
<td>5012 (+1)</td>
<td>5012 (+1)</td>
<td>687</td>
<td>5013 (+1)</td>
<td>2517</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1-b</td>
<td>6388</td>
<td>6370</td>
<td>6373 (+3)</td>
<td>6376 (+3)</td>
<td>2073</td>
<td>6376 (+0)</td>
<td>8635</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1-c</td>
<td>8518</td>
<td>8418</td>
<td>8457 (+39)</td>
<td>8468 (+11)</td>
<td>157</td>
<td>8475 (+7)</td>
<td>755</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2-a</td>
<td>9884</td>
<td>9791</td>
<td>9795 (+4)</td>
<td>9795 (+0)</td>
<td>1977</td>
<td>9798 (+3)</td>
<td>26124</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2-b</td>
<td>13100</td>
<td>12949</td>
<td>12955 (+6)</td>
<td>12960 (+5)</td>
<td>848</td>
<td>12963 (+3)</td>
<td>5976</td>
<td>137</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s2-c</td>
<td>16425</td>
<td>16314</td>
<td>16332 (+18)</td>
<td>16338 (+6)</td>
<td>357</td>
<td>16340 (+2)</td>
<td>3612</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3-a</td>
<td>10220</td>
<td>10143</td>
<td>10145 (+2)</td>
<td>10145 (+0)</td>
<td>2718</td>
<td>10147 (+2)</td>
<td>4194</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3-b</td>
<td>13682</td>
<td>13598</td>
<td>13604 (+6)</td>
<td>13605 (+1)</td>
<td>1024</td>
<td>13606 (+1)</td>
<td>12671</td>
<td>76</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s3-c</td>
<td>17188</td>
<td>17058</td>
<td>17089 (+31)</td>
<td>17090 (+1)</td>
<td>416</td>
<td>17095 (+5)</td>
<td>3271</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4-a</td>
<td>12268</td>
<td>12126</td>
<td>12129 (+3)</td>
<td>12129 (+0)</td>
<td>1248</td>
<td>12132 (+3)</td>
<td>32172</td>
<td>139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4-b</td>
<td>16321</td>
<td>16066</td>
<td>16071 (+5)</td>
<td>16073 (+2)</td>
<td>688</td>
<td>16077 (+4)</td>
<td>15905</td>
<td>248</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s4-c</td>
<td>20481</td>
<td>20340</td>
<td>20362 (+22)</td>
<td>20375 (+13)</td>
<td>352</td>
<td>20383 (+8)</td>
<td>3932</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.2. Branch-and-Price and Integer Solution Results

The final branch-and-bound component that has to be specified is the node selection rule. In order to increase the overall lower bound as fast as possible, nodes are processed according to the best-first search strategy. In case of a tie, the last node added is selected first. We found that this strategy is fundamental for finding integer solutions early because there often exist large subtrees having nodes with identical lower bounds (some kind of plateaus). In these cases, the rule implies that the subtree is processed in a depth-first manner. Another crucial point for the effectiveness of the branching scheme is that the follower son node is always selected before its non-follower brother node (two unprocessed son nodes have identical lower bounds (see appendix for khs and gdb instances). All khs and gdb instances were solved to proved optimality often in less than one second. We did not provide any upper bounds to help bap to terminate early. Instead, an optimal CARP solution was computed for these instances and the lower bound had to be raised to close the integrality gap (gap < 1).

For the bccm benchmark set, results are shown in Table 1. The information about optimal solutions differs from that given in (Letchford and Oukil, 2009) because results from a working paper by Ghiani et al. (2007) are unreliable. This working paper has been withdrawn (Laporte, 2011).

For all bccm instances we can either show that the upper bound is at the same time a lower bound or find an integer solution and prove its optimality. For the instance 9d, we have found an optimal solution with value 515 during experiments with other branching schemes. Hence, for 7a, 7c, and 9d the final setup was only able to prove optimality due to the tree lower bound. For five instances (4d, 5c, 5d, 8c, and 9d), optimality was proven for the first time. This means the solution of all bccm benchmark problems that were open at the time of writing. Overall, we are able to determine optimal integer solutions in 32 out of 35 cases. In contrast, the node-routing approaches of Longo et al. (2006), Baldacci and Maniezzo (2006), and Bartolini et al. (2011) determine and prove optimal solutions in five, six, and 29 cases, respectively.

Results for the eg1 test set are presented in Table 2. With the given setup, five instances (e1-a, e1-b, e2-a, e3-a, and s1-a) are solved to proved optimality in 4 hours. These instances were already solved either by Longo et al. (2006) or Baldacci and Maniezzo (2006). For s1-b we are able to close the gap, but we were unable to determine an optimal solution. Thus, we prove optimality of s1-b by lower bound using the upper bound presented in (Brandio and Eglese, 2008).

(Bartolini et al., 2011) are able to solve ten of the 24 eg1 instances (however, e2-a remains unsolved in their approach). The better lower bounds \( b_L \) in the final bounding procedure allow them to close the gap for five additional instance e1-c, e2-c, s1-c, s2-c, and s3-\( c \).

The lower bound \( b_{\text{lower}}^\text{tree} \) resulting from the partial solution of the branch-and-bound tree exceeds the best known lower bounds \( b_L \) presented by Longo et al. (2006) for 14 instances. The bound \( b_{\text{lower}}^\text{tree} \) also exceeds the lower bound presented by Bartolini et al. (2011) in six cases (e2-b, e3-b, e4-a, s3-a, s4-a, and s4-b). The appendix contains a table that lists the best known lower and upper bounds with references.

7. Conclusions

In this paper we proposed a cut-first branch-and-price-second algorithm for the CARP. The strength of the new column-generation formulation results from strong lower bounds, symmetry elimination, efficient pricing, and an effective branching scheme. Strong lower bounds are obtained by the combination of cuts from the one-index formulation and the Dantzig-Wolfe reformulation inducing a set-partitioning type master program. This aggregated master program avoids vehicle-specific variables and therewith eliminates symmetry. Even so, the reconstruction of individual vehicle routes is possible. The generation of new routes can be done efficiently because all pricing computations can be performed on the original sparse underlying street network. A second fundamental finding is that negative reduced costs on deadheading edges can be prevented by adding dual-optimal inequalities to the extensive formulation. Non-negative reduced costs are algorithmically advantageous as parts of the pricing can now be carried out using Dijkstra’s algorithm without the need to prevent cycling. Finally, the new branching scheme is the first one that ensures integral
CARP solutions, while the structure of the pricing problem remains unchanged. Contrary to the node-routing case, the integrality of the aggregated variables of a original (i.e. compact) formulation generally do not imply integer master program variables. The key device to finally obtain integer routes is branching on followers of required edges, where in one branch two required edges have to be serviced consecutively, and in the other branch subsequent service is forbidden. While branching on follower constraints is common in node routing, the novelty of our approach is the handling of follower constraints referring to edges that might not be directly connected.

Computational experiments show that the proposed cut-first branch-and-price-second algorithm gives considerable results for all four benchmark sets. Several earlier exact approaches proved optimality of known heuristic solutions by matching lower and upper bounds, but were not able to deliver optimal CARP routes. Our branching scheme however enables us to compute feasible integer solutions and optimal ones in many cases. As a result, all open benchmark instances of Belenguer and Benavent (1998) are solved now. For the (Eglese and Li, 1992) benchmark set, optimality of one more instance was shown (independently the results of Bartolini et al. (2011)) and some lower bounds were improved.

We see the following possible avenues for future research: A deeper analysis of the polyhedral structure might lead to new strong valid inequalities and might help to further strengthen the initial lower bound. All cuts might also be separated in the branch-and-bound tree and not only once at the beginning. Another topic, as already suggested by Letchford and Oukil (2009), are approaches that replace the weaker 2-loop free pricing with stronger relaxations. Since columns in the master program often contain 3-loops and longer loops, $k$-loop elimination for $k \geq 3$ would probably improve the lower bounds. We pointed out that this is a non-trivial task due to the incompatibility between the suggested branching rule on (non-)followers and $k$-loop elimination for $k \geq 3$. Various other relaxations of the elementary pricing problem that eliminate these loops should be analyzed (Irnich and Villeneuve, 2006; Desaulniers et al., 2008; Baldacci et al., 2009) for which we expect that pricing times remain acceptable as pricing on the original sparse graph is possible.

References


Appendix

A. Integer Solutions from Fractional Solutions

Example: Consider a CARP instance with nodes \( \{1, \ldots, 11\} \) and 19 edges that are all required. The depot is located in node 1. The edge demands are all equal to 1 and the vehicle capacity is \( Q = 5 \). Four vehicles are needed to service these edge demands. In Figure 2, a solution of eMP in one of the branch-and-bound nodes is shown. This eMP solution consists of five (fractional) routes that are 2-loop free. Below, the corresponding node sequences ("\( = \)" indicates a service and "\( \sim \)" a deadheading) and service sequences \( s_r \) are shown. Additionally, the value of the corresponding route variable \( \lambda \) is given.

- Route 1 is \((1 = 7 = 8 = 11 = 5 = 6 = 1)\) servicing \( s_1 = \{(1,7),\{7,8\},\{8,11\},\{11,5\},\{5,6\}\}\) \(\lambda_1 = 1\)
- Route 2 is \((1 = 6 = 7 = 8 = 10 = 9 = 2 = 1)\) servicing \( s_2 = \{(1,6),\{6,7\},\{8,10\},\{10,9\}\}\) \(\lambda_2 = 1\)
- Route 3 is \((1 = 2 = 9 = 11 = 10 = 1)\) servicing \( s_3 = \{(1,2),\{2,9\},\{9,11\},\{11,10\},\{10,1\}\}\) \(\lambda_3 = 1\)
- Route 4 is \((1 = 4 = 3 = 5 = 3 = 4 = 1)\) servicing \( s_4 = \{(1,4),\{4,3\},\{3,5\},\{3,4\},\{4,1\}\}\) \(\lambda_4 = 0.5\)
- Route 5 is \((1 = 4 = 2 = 3 = 5 = 3 = 2 = 4 = 1)\) servicing \( s_5 = \{(4,2),\{2,3\},\{3,5\},\{3,2\},\{2,4\}\}\) \(\lambda_5 = 0.5\)

The eMP uses routes 1, 2, and 3 one time each, but the last two routes 4 and 5 are used only 0.5 times. The only edge serviced by two routes is \(\{3,5\}\) (by route 4 and 5). Moreover, the edges \(\{1,4\},\{2,3\},\{2,4\}\) and \(\{3,4\}\) are serviced twice within the same route, either by route 4 or 5, i.e., \(f(1,4),(3,4) = f(3,4),(3,5) = f(2,4),(2,3) = f(2,3),(3,5) = 2\) (see equation (27)).

Note that this convex combination of routes produces integer edge flows on all edges (see Figure 2). The implied follower information is

\[ f(1,4),(3,4) = f(3,4),(3,5) = f(3,5),(2,3) = f(2,3),(2,4) = 1. \]

Example: In the previously given example, the reflexive and transitive closure \( \bar{F} \) defined by the five routes separates the set of required edges into four subsets \( E_{R}^{1} \cup E_{R}^{2} \cup E_{R}^{3} \cup E_{R}^{4} \). The follower relation defined by routes 4 and 5 results in one subset \( E_{R}^{5} = \{(1,4),\{2,3\},\{2,4\},\{3,4\},\{3,5\}\}\). This implies four sequences of required edges:

\( s_1 = \{(1,7),\{7,8\},\{8,11\},\{11,5\},\{5,6\}\}\)
\( s_2 = \{(1,6),\{6,7\},\{8,10\},\{10,9\}\}\)
\( s_3 = \{(2,9),\{9,11\},\{11,10\},\{10,1\}\}\)
\( s_4 = \{(1,4),\{4,3\},\{3,5\},\{3,2\},\{2,4\}\}\)

The costs of the corresponding routes equal the costs of the fractional solution in the previous example.

Figure 2: Fractional Solution
B. Network Modifications

Example: We consider a graph $G$ with eight nodes \{a, ..., g\} and twelve edges (see Figure 3). We assume the following active (non)-follower decisions given by $F = \{(e_1, e_2), (e_3, e_4)\}$ and $N = \{(e_1, e_3)\}$ with $e_1 = \{a, b\}$, $e_2 = \{c, d\}$, $e_3 = \{e, f\}$ and $e_4 = \{g, h\}$. The edges $e_1$, $e_2$, $e_3$ and $e_4$ induce the subset $\{e_1, e_2, e_3, e_4\}$ of $E_R$. The set of all possible subsequences $s$ is

$$\{(e_1, e_2), (e_3, e_4), (e_1, e_2, e_3, e_4), (e_1, e_4, e_3), (e_2, e_1, e_4, e_3)\}$$

(For example, the sequences $(e_2)$ and $(e_2, e_3, e_4)$ are infeasible as they violate the follower condition $f_{e_1, e_2} = 1$.) For each possible subsequence and for each pair of start and end nodes, a shortest-path problem in the auxiliary network has to be solved. These twenty (five sequences and four pairs) new edges are added to the network depicted in Figure 4.
C. Computational Results of Phase I

Recall from Section 4 that in phase I the LP-relaxation of the one-index formulation (7)–(10) is solved. The results obtained at the end of this phase are presented in the Tables 4–7. We use the following notation:

- $lb_*$: lower bound obtained by algorithm of *
- $lb_A$: cutting-plane algorithm of Ahr (2004)
- $lb_L$: linear relaxation (root) by Longo et al. (2006)
- $lb_{LO}$: linear relaxation (root with elementary routes) by Letchford and Oukil (2009)
- $lb_{MPS}$: dual ascent and cutting-plane method of Martinelli et al. (2011)
- $lb_{own}$: lower bound obtained in phase I

<table>
<thead>
<tr>
<th>init cuts</th>
<th>number of initial cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>number of odd cuts and capacity cuts separated on components</td>
</tr>
<tr>
<td>odd cuts</td>
<td>number of separated odd cuts/number of binding odd cuts</td>
</tr>
<tr>
<td>cap cuts</td>
<td>number of separated/binding capacity cuts</td>
</tr>
<tr>
<td>dp1</td>
<td>number of separated/binding dp1 cuts</td>
</tr>
<tr>
<td>dp2</td>
<td>number of separated/binding dp2 cuts</td>
</tr>
<tr>
<td>comp. time</td>
<td>in time seconds of phase I</td>
</tr>
</tbody>
</table>

The meaning of all entries in the tables and the way we produced the results is explained in the following:

We initialize the one-index formulation with all odd cut constraints $y(S) \geq 1$ for singleton sets $S = \{v\}$ where $v \in V$ is an odd node. Additionally, odd cut and capacity cut constraints $y(S) \geq k(S) - |\delta_R(S)|$ are added to the model with the initialization procedure described in (Belenguer and Benavent, 2003). The number of all these inequalities is shown in column “init cuts”.

We start every iteration of the separation procedure with a fast separation heuristic that considers the components of the support graph $(V, E_R \cup \{e \in E : y_e > 0\})$ in order to generate candidate sets $S$. Associated violated odd cuts or capacity cuts are added first. Their number is shown in column “component”.

If no cut has been found so far, the efficient separation routine of Letchford et al. (2008) is used to separate odd cuts exactly. Moreover, capacity cuts are then separated in a heuristic way with the procedure of Belenguer and Benavent (2003) by solving a maximum-flow problem both on an auxiliary graph and perturbed variants of that auxiliary graph (the edge demand $q_e$ is perturbed). This procedure is guaranteed to find all violated fractional capacity cuts. Those sets $S$ that do not violate the fractional cut $y(S) \geq q(E_R(S) \cup \delta_R(S))/Q - \delta_R(S)$ are checked to violate $y(S) \geq k(S) - |\delta_R(S)|$.

If no cut is found, the algorithm of Ahr (2004) is then used to exactly separate capacity cuts.

The separation of disjoint-path inequalities is the most time consuming component. After analyzing the computation times, we decided to start with the separation of disjoint-path inequalities of type dp2. We proceed with the separation procedures for disjoint-path inequalities of type dp1 followed by dp3.

In all separation procedures, we use a threshold of $\epsilon = 0.01$ for minimum violation, i.e., a cut is only violated if its rhs exceeds the lhs by at least $\epsilon$. The only exception from this rule is the MIP-based separation algorithm for capacity cuts of Ahr (2004), where the threshold is set to $\epsilon = 0.1$.

Using this cascade of separation procedures, we never found violated cuts of type dp3 (last procedure in the cascade). Therefore, the Tables 4–7 do not report results on disjoint-path inequalities of type dp3. However, when we change the order so that dp3 cuts were considered before dp1 and dp2, we find at least some violated inequalities of type dp3.

The number of separated cuts and the number of cuts used to initialize the eMP (phase II) are shown in the columns “odd cuts”, “capacity cuts”, “dp1”, and “dp2”, respectively. The last column presents the computation time of phase I.

Next, we compare our results with the results from the literature. For the ksha and gdb instances, we obtained at least the lower bounds $lb_A$ and $lb_{BB}$ as presented in (Ahr, 2004; Belenguer and Benavent, 2003).

In the case of gdb8, we are able to increase the lower bound (this is possible as Belenguer and Benavent (2003) do not use an exact separation procedure for capacity cuts and Ahr (2004) did not separate disjoint-path inequalities). For gdb8 and gdb12, one binding dp1 inequality was separated (but not of type dp2).
The computation times for both sets of instances are all small (sometimes too small to be recorded properly; note that all times are rounded up with precision 0.1).

For the \texttt{bccm} instances, the lower bounds $lb_{\text{own}}$ is always at least as strong as $lb_A$. This is consistent as we use the same MIP approach as Ahr (2004) to exactly separate capacity cuts. Compared to (Belenguer and Benavent, 2003), in three cases ($2b$, $6b$ and $9d$), the lower bounds $lb_{BB}$ are two units better than our bound $lb_{\text{own}}$. We suspect that this is due to the fact that the separation of disjoint-path inequalities is performed with our version of the heuristic that certainly differs from the one used to compute $lb_{BB}$. For $10d$, we are able to increase the lower bound compared to $lb_{BB}$ by one unit.

The lower bounds $lb_L$ of Longo et al. (2006) obtained in the root node of their CVRP branch-and-price-and-cut algorithm are at least as good as our lower bounds obtained at the end of the cutting plane algorithm (11 cases better). The only exception is the instance $5b$ were our bound is slightly better.

For the \texttt{egl} benchmark set, our lower bounds $lb_{\text{own}}$ are at least as strong as the both lower bounds $lb_A$ and $lb_{BB}$ (except for $s2-a$). As for the \texttt{bccm} instances, the reported lower bound $lb_L$ are always better than our lower bounds. The recent paper by Martinelli et al. (2011) reports computational results only for the \texttt{egl} instances. There, the lower bounds at the end of the cutting-plane algorithms are sometimes worse (9 cases) and sometimes better (9 cases) than our lower bounds. This might be due to the fact that they do not separate disjoint-path inequalities, but might use smaller thresholds $\varepsilon > 0$ for violation.

Concerning the computation time presented in Tables 6 and 7, our implementation could further be accelerated. The work of Martinelli et al. (2011) shows that a warm-start of the cutting-plane algorithm can further reduce computation times. They use a dual-ascent heuristic for the warm-start. However, even without such a warm-start the computation times for phase I are typically much smaller than those of phase II.
| name  | $|V|$ | $|E|$ | $|K|$ | $b_{DP}$ | $b_{A}$ | $b_{NUM}$ | init cuts | odd cuts | cap cuts | dp1 | dp2 | time |
|-------|-----|-----|-----|-------|-------|-------|--------|--------|--------|-----|-----|-----|
| gdb1  | 12  | 22  | 5   | 316   | 316   | 316   | 7      | 8      | 0/11  | 1/2 | 0/0 | 0/0 | 0.2 |
| gdb2  | 12  | 26  | 6   | 339   | 339   | 339   | 6      | 4      | 0/7   | 0/1 | 0/0 | 0/0 | 0.3 |
| gdb3  | 12  | 22  | 5   | 275   | 275   | 275   | 8      | 2      | 0/8   | 0/1 | 0/0 | 0/0 | 0.2 |
| gdb4  | 11  | 19  | 4   | 287   | 287   | 287   | 9      | 0      | 0/8   | 1/2 | 0/0 | 0/0 | 0.2 |
| gdb5  | 13  | 26  | 6   | 377   | 377   | 377   | 7      | 7      | 0/11  | 2/3 | 0/0 | 0/0 | 0.2 |
| gdb6  | 12  | 22  | 5   | 298   | 298   | 298   | 6      | 2      | 0/6   | 1/2 | 0/0 | 0/0 | 0.3 |
| gdb7  | 12  | 22  | 5   | 325   | 325   | 325   | 7      | 10     | 0/15  | 1/2 | 0/0 | 0/0 | 0.1 |
| gdb8  | 27  | 46  | 10  | 344   | 344   | 346   | 19     | 17     | 1/21  | 25/14| 1/1 | 0/0 | 1.8 |
| gdb9  | 27  | 51  | 10  | 303   | 303   | 303   | 17     | 15     | 0/21  | 14/10| 0/0 | 0/0 | 0.8 |
| gdb10 | 12  | 25  | 4   | 275   | 275   | 275   | 8      | 3      | 0/10  | 0/0 | 0/0 | 0/0 | 0.4 |
| gdb11 | 22  | 45  | 5   | 395   | 395   | 395   | 18     | 5      | 0/20  | 0/1 | 0/0 | 0/0 | 0.4 |
| gdb12 | 13  | 23  | 7   | 450   | 450   | 450   | 8      | 3      | 0/5   | 2/3 | 1/1 | 0/0 | 0.4 |
| gdb13 | 10  | 28  | 6   | 536   | 536   | 536   | 6      | 2      | 0/6   | 0/1 | 0/0 | 0/0 | 0.4 |
| gdb14 | 7   | 21  | 5   | 100   | 100   | 100   | 1      | 0      | 0/0   | 0/1 | 0/0 | 0/0 | 0.1 |
| gdb15 | 7   | 21  | 4   | 58    | 58    | 58    | 1      | 0      | 0/0   | 0/1 | 0/0 | 0/0 | 0.1 |
| gdb16 | 8   | 28  | 5   | 127   | 127   | 127   | 9      | 0      | 0/7   | 0/1 | 0/0 | 0/0 | 0.4 |
| gdb17 | 8   | 28  | 5   | 91    | 91    | 91    | 9      | 0      | 0/8   | 0/0 | 0/0 | 0/0 | 0.5 |
| gdb18 | 9   | 36  | 5   | 164   | 164   | 164   | 1      | 0      | 0/0   | 0/1 | 0/0 | 0/0 | 0.5 |
| gdb19 | 8   | 11  | 3   | 55    | 55    | 55    | 5      | 1      | 0/5   | 0/1 | 0/0 | 0/0 | 0.1 |
| gdb20 | 11  | 22  | 4   | 121   | 121   | 121   | 5      | 3      | 0/6   | 0/1 | 0/0 | 0/0 | 0.2 |
| gdb21 | 11  | 33  | 6   | 156   | 156   | 156   | 5      | 1      | 0/5   | 0/1 | 0/0 | 0/0 | 0.5 |
| gdb22 | 11  | 44  | 8   | 200   | 200   | 200   | 5      | 1      | 0/5   | 0/1 | 0/0 | 0/0 | 1.0 |
| gdb23 | 11  | 55  | 10  | 233   | 233   | 233   | 1      | 0      | 0/0   | 1/2 | 0/0 | 0/0 | 1.6 |

Table 5: Results for the gdb instances at the end of phase I.
| name | $|V|$ | $|E|$ | $|K|$ | $b_{BB}$ | $b_{LA}$ | $b_{LB}$ | $b_{own}$ | init cuts | component | odd cuts | cap cuts | dp1 | dp2 | time |
|------|-----|-----|-----|------|------|------|------|-------|---------|---------|---------|------|-----|------|
| 1a   | 24  | 39  | 2   | 247  | 247  | 247  | 247  | 16    | 16      | 0/31    | 0/0     | 0/0 | 0/0 | 0.1 |
| 1b   | 24  | 39  | 3   | 247  | 247  | 247  | 247  | 16    | 16      | 0/31    | 0/0     | 0/0 | 0/0 | 0.2 |
| 1c   | 24  | 39  | 8   | 309  | 309  | 312  | 309  | 16    | 15      | 0/23    | 14/13   | 0/0 | 0/0 | 0.5 |
| 2a   | 24  | 34  | 2   | 298  | 298  | 298  | 298  | 17    | 4       | 0/19    | 2/3     | 0/0 | 0/0 | 0.1 |
| 2b   | 24  | 34  | 3   | 330  | 328  | 329  | 328  | 17    | 6       | 0/15    | 3/6     | 0/0 | 0/0 | 0.1 |
| 2c   | 24  | 34  | 8   | 526  | 526  | 528  | 526  | 17    | 11      | 0/9     | 13/14   | 0/0 | 0/0 | 0.4 |
| 3a   | 24  | 35  | 2   | 105  | 105  | 105  | 105  | 16    | 9       | 0/21    | 1/2     | 0/0 | 0/0 | 0.1 |
| 3b   | 24  | 35  | 3   | 111  | 111  | 111  | 111  | 16    | 10      | 0/19    | 2/5     | 0/0 | 0/0 | 0.3 |
| 3c   | 24  | 35  | 7   | 161  | 159  | 161  | 161  | 17    | 16      | 0/23    | 1/31    | 0/0 | 3/3 | 0.7 |
| 4a   | 41  | 69  | 3   | 522  | 522  | 522  | 522  | 24    | 11      | 0/29    | 3/4     | 0/0 | 0/0 | 0.3 |
| 4b   | 41  | 69  | 4   | 534  | 534  | 534  | 534  | 24    | 11      | 0/28    | 3/4     | 0/0 | 0/0 | 0.5 |
| 4c   | 41  | 69  | 5   | 550  | 550  | 550  | 550  | 24    | 12      | 0/28    | 6/7     | 0/0 | 0/0 | 0.5 |
| 4d   | 41  | 69  | 9   | 644  | 642  | 648  | 644  | 24    | 19      | 6/30    | 57/25   | 3/2 | 1/1 | 8.0 |
| 5a   | 34  | 65  | 3   | 566  | 566  | 566  | 566  | 19    | 26      | 0/43    | 1/2     | 0/0 | 0/0 | 0.3 |
| 5b   | 34  | 65  | 4   | 589  | 586  | 588  | 589  | 20    | 33      | 0/47    | 3/4     | 0/0 | 1/1 | 0.5 |
| 5c   | 34  | 65  | 5   | 612  | 610  | 613  | 612  | 20    | 33      | 0/46    | 6/6     | 0/0 | 1/1 | 0.8 |
| 5d   | 34  | 65  | 9   | 714  | 714  | 716  | 714  | 21    | 38      | 1/37    | 11/11   | 0/0 | 0/0 | 0.9 |
| 6a   | 31  | 50  | 3   | 330  | 330  | 330  | 330  | 19    | 27      | 0/41    | 0/1     | 0/0 | 0/0 | 0.1 |
| 6b   | 31  | 50  | 4   | 338  | 336  | 337  | 336  | 19    | 27      | 0/38    | 1/2     | 0/0 | 0/0 | 0.3 |
| 6c   | 31  | 50  | 10  | 418  | 414  | 420  | 418  | 19    | 26      | 1/22    | 38/20   | 1/0 | 1/1 | 2.0 |
| 7a   | 40  | 66  | 3   | 382  | 382  | 382  | 382  | 21    | 7       | 0/26    | 0/0     | 0/0 | 0/0 | 0.1 |
| 7b   | 40  | 66  | 4   | 386  | 386  | 386  | 386  | 21    | 7       | 0/26    | 1/1     | 0/0 | 0/0 | 0.3 |
| 7c   | 40  | 66  | 9   | 436  | 436  | 436  | 436  | 22    | 6       | 0/21    | 27/21   | 2/2 | 1/0 | 3.1 |
| 8a   | 30  | 63  | 3   | 522  | 522  | 522  | 522  | 18    | 6       | 0/22    | 0/1     | 0/0 | 0/0 | 0.1 |
| 8b   | 30  | 63  | 4   | 531  | 531  | 531  | 531  | 18    | 6       | 0/21    | 2/3     | 0/0 | 0/0 | 0.4 |
| 8c   | 30  | 63  | 9   | 653  | 654  | 654  | 653  | 20    | 13      | 0/17    | 20/15   | 2/2 | 6/6 | 2.1 |
| 9a   | 50  | 92  | 3   | 450  | 450  | 450  | 450  | 32    | 34      | 0/64    | 0/0     | 0/0 | 0/0 | 0.4 |
| 9b   | 50  | 92  | 4   | 453  | 453  | 453  | 453  | 32    | 21      | 0/50    | 0/1     | 0/0 | 0/0 | 0.3 |
| 9c   | 50  | 92  | 5   | 459  | 459  | 459  | 459  | 32    | 21      | 0/49    | 0/1     | 1/1 | 0/0 | 1.1 |
| 9d   | 50  | 92  | 10  | 509  | 505  | 512  | 507  | 32    | 26      | 0/50    | 15/14   | 1/1 | 1/1 | 3.0 |
| 10a  | 50  | 97  | 3   | 637  | 637  | 637  | 637  | 32    | 18      | 0/48    | 0/1     | 0/0 | 0/0 | 0.2 |
| 10b  | 50  | 97  | 4   | 645  | 645  | 645  | 645  | 32    | 41      | 15/83   | 1/2     | 0/0 | 0/0 | 1.1 |
| 10c  | 50  | 97  | 5   | 655  | 655  | 655  | 655  | 32    | 51      | 0/77    | 3/4     | 0/0 | 0/0 | 0.8 |
| 10d  | 50  | 97  | 10  | 732  | 731  | 734  | 733  | 33    | 35      | 38/64   | 38/15   | 4/2 | 6/4 | 104.0 |

Table 6: Results for the bccm instances at the end of phase I.
| e1-a | e1-b | e1-c | e2-a | e2-b | e2-c | e3-a | e3-b | e3-c | e4-a | e4-b | e4-c | s1-a | s1-b | s1-c | s2-a | s2-b | s2-c | s3-a | s3-b | s3-c | s4-a | s4-b | s4-c |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   | 77   |
| 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 |
| 3515 | 3516 | 3517 | 3527 | 3538 | 3545 | 3546 | 3547 | 3548 | 3549 | 3550 | 3551 | 3552 | 3553 | 3554 | 3555 | 3556 | 3557 | 3558 | 3559 | 3560 | 3561 | 3562 |
| 41   | 65   | 133/146| 317/152| 7/5 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 | 0/0 |
| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146| 133/146|
| 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   | 49   |
| 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  | 146  |

Table 7: Results for the egal instances at the end of phase I.
This section presents the results on the linear relaxation of the master program eMP (phase II) for the small-sized benchmark sets *kshs* and *gdb*. We use the same notation as in the journal article:

| lower bound obtained by algorithm of *; BB= cutting-plane algorithm of Belenguer and Benavent (2003); A= Ahr (2004); L= linear relaxation (root) by Longo et al. (2006); LO= linear relaxation (root, elementary routes) by Letchford and Oukil (2009); MPS= dual ascent and cutting-plane algorithm of Martinelli et al. (2011) |
| lower bound obtained in our linear relaxation (root) or at termination of phase II (tree) comp. by first paper where the currently best upper bound was computed; H= Hertz et al. (2000); La= Lacomme et al. (2001); BE= Brandão and Eglese (2008); PD= Polacek et al. (2008); SCC= Santos et al. (2010); BLC= Baldacci et al. (2010); Be= Beullens et al. (2003) |
| value of optimal solution proved by first paper where optimality is proved due to |\( |ub - lb| < 1 \) | proved by sol first paper where optimality is proved by computing an integer solution; B= Benavent et al. (1992); BB= Belenguer and Benavent (2003); Lo= Longo et al. (2006); BM= Baldacci and Maniezzo (2006); LO= Letchford and Oukil (2009); own= own solution |
| computation in time seconds for phase I or phase II B&B nodes number of solved nodes in our branch-and-bound tree branching D/E/F number of different branching decisions: D= node degree; E= edge flow; F= follower |

<table>
<thead>
<tr>
<th>name</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>kshs1</td>
<td>8</td>
<td>15</td>
<td>4</td>
<td>14661</td>
<td>14661</td>
<td>14.661</td>
<td>14661</td>
<td>BB</td>
<td>Lo</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>kshs2</td>
<td>10</td>
<td>15</td>
<td>4</td>
<td>9863</td>
<td>9863</td>
<td>9.863</td>
<td>9863</td>
<td>BB</td>
<td>Lo</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>kshs3</td>
<td>6</td>
<td>15</td>
<td>4</td>
<td>9320</td>
<td>9320</td>
<td>9.320</td>
<td>9320</td>
<td>BB</td>
<td>Lo</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>kshs4</td>
<td>8</td>
<td>15</td>
<td>4</td>
<td>11098</td>
<td>11098</td>
<td>11.498</td>
<td>11498</td>
<td>Lo</td>
<td>Lo</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>kshs5</td>
<td>8</td>
<td>15</td>
<td>3</td>
<td>10957</td>
<td>10957</td>
<td>10.957</td>
<td>10957</td>
<td>BB</td>
<td>Lo</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>kshs6</td>
<td>9</td>
<td>15</td>
<td>3</td>
<td>10197</td>
<td>10197</td>
<td>10.197</td>
<td>10197</td>
<td>BB</td>
<td>Lo</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 8: Results for the *kshs* instances at the end of phase II.
| name  | $|V|$ | $|E|$ | $|K|$ | $\theta_B$ | $\theta_A$ | $\theta_{opt}$ | opt. | proved by lb | proved by sol | time phase I | time phase II | $BB$ & $Lo$ | nodes | branching | $D$, $E$, $F$ |
|-------|-----|-----|-----|--------|--------|--------|------|-------------|--------------|-------------|-------------|------------|---------|----------|-------------|
| gdb1  | 12  | 22  | 5   | 316    | 316    | 316    | 316  | BB          | Lo           | 0.2         | 0.1         | 9           | 0/0     | 8        |              |
| gdb2  | 12  | 26  | 6   | 339    | 339    | 339    | 339  | BB          | Lo           | 0.3         | 0.1         | 6           | 0/0     | 5        |              |
| gdb3  | 12  | 22  | 5   | 275    | 275    | 275    | 275  | BB          | Lo           | 0.2         | 0.1         | 7           | 0/0     | 6        |              |
| gdb4  | 11  | 19  | 4   | 287    | 287    | 287    | 287  | BB          | Lo           | 0.2         | 0.1         | 3           | 0/0     | 2        |              |
| gdb5  | 13  | 26  | 6   | 377    | 377    | 377    | 377  | BB          | Lo           | 0.2         | 0.1         | 7           | 0/0     | 6        |              |
| gdb6  | 12  | 22  | 5   | 298    | 298    | 298    | 298  | BB          | Lo           | 0.3         | 0.1         | 5           | 0/0     | 4        |              |
| gdb7  | 12  | 22  | 5   | 325    | 325    | 325    | 325  | BB          | Lo           | 0.1         | 0.1         | 9           | 0/0     | 8        |              |
| gdb8  | 27  | 46  | 10  | 344    | 344    | 347    | 348  | Lo          | Lo           | 1.8         | 0.8         | 12          | 5/0     | 6/0      |              |
| gdb9  | 27  | 51  | 10  | 303    | 303    | 303    | 303  | BB          | Lo           | 0.8         | 1.3         | 11          | 6/0     | 4/0      |              |
| gdb10 | 12  | 25  | 4   | 275    | 275    | 275    | 275  | BB          | Lo           | 0.4         | 0.1         | 8           | 0/0     | 7        |              |
| gdb11 | 22  | 45  | 5   | 395    | 395    | 395    | 395  | BB          | Lo           | 0.4         | 1.0         | 21          | 0/0     | 20/0     |              |
| gdb12 | 13  | 23  | 7   | 450    | 450    | 451    | 456  | Lo          | Lo           | 0.4         | 0.2         | 15          | 12/0    | 2/0      |              |
| gdb13 | 10  | 28  | 6   | 536    | 536    | 536    | 536  | BB          | Lo           | 0.4         | 0.2         | 4           | 0/0     | 3/0      |              |
| gdb14 | 7   | 21  | 5   | 100    | 100    | 100    | 100  | BB          | Lo           | 0.1         | 0.1         | 3           | 0/0     | 2/0      |              |
| gdb15 | 7   | 21  | 4   | 58     | 58     | 58     | 58   | BB          | Lo           | 0.1         | 0.1         | 6           | 1/0     | 4/0      |              |
| gdb16 | 8   | 28  | 5   | 127    | 127    | 127    | 127  | BB          | Lo           | 0.4         | 0.2         | 14          | 2/0     | 11/0     |              |
| gdb17 | 8   | 28  | 5   | 91     | 91     | 91     | 91   | BB          | Lo           | 0.5         | 0.2         | 20          | 1/0     | 18/0     |              |
| gdb18 | 9   | 36  | 5   | 164    | 164    | 164    | 164  | BB          | Lo           | 0.5         | 0.2         | 17          | 0/0     | 16/0     |              |
| gdb19 | 8   | 11  | 3   | 55     | 55     | 55     | 55   | BB          | Lo           | 0.1         | 0.1         | 1           | 0/0     | 0/0      |              |
| gdb20 | 11  | 22  | 4   | 121    | 121    | 121    | 121  | BB          | Lo           | 0.2         | 0.2         | 6           | 0/0     | 5        |              |
| gdb21 | 11  | 33  | 6   | 156    | 156    | 156    | 156  | BB          | Lo           | 0.5         | 0.2         | 12          | 2/0     | 9        |              |
| gdb22 | 11  | 44  | 8   | 200    | 200    | 200    | 200  | BB          | Lo           | 1.0         | 0.4         | 22          | 3/0     | 18/0     |              |
| gdb23 | 11  | 55  | 10  | 233    | 233    | 233    | 233  | BB          | Lo           | 1.6         | 0.6         | 23          | 3/0     | 19/0     |              |

Table 9: Results for the $gdb$ instances at the end of phase II.
E. Best Known Lower and Upper Bounds for the $egl$ Instances

The following table summarizes the best known lower and upper bounds for the $egl$ instances and list the corresponding references:

<table>
<thead>
<tr>
<th>name</th>
<th>$lb_{best}$</th>
<th>comp. by</th>
<th>$ub_{best}$</th>
<th>comp. by</th>
<th>opt</th>
<th>proved by</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1-a</td>
<td>-</td>
<td>-</td>
<td>3548</td>
<td>Lacomme et al. (2001)</td>
<td>3548</td>
<td>Longo et al. (2006)</td>
</tr>
<tr>
<td>e1-c</td>
<td>-</td>
<td>-</td>
<td>5595</td>
<td>Lacomme et al. (2001)</td>
<td>5595</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>e2-a</td>
<td>-</td>
<td>-</td>
<td>5018</td>
<td>Lacomme et al. (2001)</td>
<td>5018</td>
<td>Baldacci and Maniezzo (2006)</td>
</tr>
<tr>
<td>e2-b</td>
<td>6301</td>
<td>own</td>
<td>6317</td>
<td>Brandão and Eglese (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>e2-c</td>
<td>-</td>
<td>-</td>
<td>8335</td>
<td>Brandão and Eglese (2008)</td>
<td>8335</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>e3-a</td>
<td>-</td>
<td>-</td>
<td>5898</td>
<td>Lacomme et al. (2001)</td>
<td>5898</td>
<td>Longo et al. (2006)</td>
</tr>
<tr>
<td>e3-b</td>
<td>7728</td>
<td>own</td>
<td>7775</td>
<td>Polacek et al. (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>e3-c</td>
<td>10244</td>
<td>Bartolini et al. (2011)</td>
<td>10292</td>
<td>Polacek et al. (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>e4-a</td>
<td>6408</td>
<td>own</td>
<td>6444</td>
<td>Santos et al. (2010)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>e4-b</td>
<td>8935</td>
<td>Bartolini et al. (2011)</td>
<td>8961</td>
<td>Bartolini et al. (2011)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>e4-c</td>
<td>11493</td>
<td>Bartolini et al. (2011)</td>
<td>11562</td>
<td>Bartolini et al. (2011)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s1-a</td>
<td>-</td>
<td>-</td>
<td>5018</td>
<td>Lacomme et al. (2001)</td>
<td>5018</td>
<td>Baldacci and Maniezzo (2006)</td>
</tr>
<tr>
<td>s1-b</td>
<td>-</td>
<td>-</td>
<td>6388</td>
<td>Brandão and Eglese (2008)</td>
<td>6388</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>s1-c</td>
<td>-</td>
<td>-</td>
<td>8518</td>
<td>Lacomme et al. (2001)</td>
<td>8518</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>s2-a</td>
<td>9825</td>
<td>Bartolini et al. (2011)</td>
<td>9884</td>
<td>Santos et al. (2010)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s2-b</td>
<td>13017</td>
<td>Bartolini et al. (2011)</td>
<td>13100</td>
<td>Brandão and Eglese (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s2-c</td>
<td>-</td>
<td>-</td>
<td>16425</td>
<td>Brandão and Eglese (2008)</td>
<td>16425</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>s3-a</td>
<td>10160</td>
<td>own</td>
<td>10220</td>
<td>Santos et al. (2010)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s3-b</td>
<td>13648</td>
<td>Bartolini et al. (2011)</td>
<td>13682</td>
<td>Polacek et al. (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s3-c</td>
<td>-</td>
<td>-</td>
<td>17188</td>
<td>Bartolini et al. (2011)</td>
<td>17188</td>
<td>Bartolini et al. (2011)</td>
</tr>
<tr>
<td>s4-a</td>
<td>12149</td>
<td>own</td>
<td>12268</td>
<td>Santos et al. (2010)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s4-b</td>
<td>16105</td>
<td>own</td>
<td>16321</td>
<td>Polacek et al. (2008)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>s4-c</td>
<td>20430</td>
<td>Bartolini et al. (2011)</td>
<td>20481</td>
<td>Bartolini et al. (2011)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 10: Bounds for the $egl$ instances.
F. Optimal solutions for the bccm instances:

Optimal solutions for the bccm instances that have not been presented before are the following:

4d  \(z = 650\)

veh 1  1→2→3→4→5→6→12→11→17→16→15→10→9→3→2→1
veh 2  1→2→3→9→10→4→5→11→16→15→14→13→7→8→9→3→2→1
veh 3  1→2→3→9→14→24→25→31→35→36→32→26→19→16→15→10→9→3→2→1
veh 4  1→7→13→23→24→30→29→23→14→9→3→2→1
veh 5  1→2→3→9→10→11→17→18→22→28→27→21→22→21→20→19→16→15→10→9→3→2→1
veh 6  1→2→3→9→10→15→25→27→32→31→30→29→23→14→9→3→2→1
veh 7  1→2→3→9→10→15→25→31→35→34→38→39→36→37→33→27→20→17→11→10→9→3→2→1
veh 8  1→2→3→9→10→15→16→19→20→27→33→37→41→40→36→40→39→35→34→29→23→14→9→3→2→1
veh 9  1→2→8→9→3→2→1

5c  \(z = 617\)

veh 1  1→2→3→9→15→21→22→23→28→27→31→30→29→18→12→6→1
veh 2  1→6→18→19→25→29→30→25→26→20→14→8→14→13→2→1
veh 3  1→7→13→19→20→21→27→32→31→32→33→34→28→22→15→14→13→12→6→1
veh 4  1→2→3→10→3→4→10→16→17→11→5→4→11→10→9→8→2→1
veh 5  1→6→7→13→14→15→16→23→24→11→17→24→34→33→28→22→21→20→19→12→6→1

5d  \(z = 718\)

veh 1  1→2→13→19→12→18→6→1
veh 2  1→6→7→1
veh 3  1→2→3→4→11→17→11→5→4→10→9→3→2→1
veh 4  1→2→8→9→15→14→8→14→13→12→6→1
veh 5  1→2→3→10→11→24→17→16→15→14→13→7→1
veh 6  1→2→3→10→16→23→24→34→28→23→22→21→20→14→13→12→6→1
veh 7  1→6→12→13→14→15→21→27→32→33→34→28→22→15→14→13→12→6→1
veh 8  1→6→18→19→20→26→31→30→29→18→6→1
veh 9  1→6→12→19→25→26→27→28→33→32→31→30→25→29→18→6→1

8c  \(z = 657\)

veh 1  1→2→3→10→9→8→1
veh 2  1→5→8→7→6→4→5→1
veh 3  1→9→10→19→18→19→15→8→1
veh 4  1→2→9→10→15→14→13→17→14→8→1
veh 5  1→5→4→7→12→16→21→24→28→29→22→25→30→23→18→15→9→2→1
veh 6  1→5→4→6→11→20→27→28→21→29→17→14→8→1
veh 7  1→8→13→12→11→16→17→18→15→9→2→1
veh 8  1→2→9→15→18→23→26→30→29→30→26→18→15→9→2→1
veh 9  1→8→13→16→20→24→27→20→21→22→18→15→9→2→1

9d  \(z = 515\)

veh 1  1→5→2→3→6→7→12→11→10→1
veh 2  1→9→14→15→16→1
veh 3  1→15→24→30→38→36→35→34→33→25→20→21→13→9→1
veh 4  1→5→2→3→4→7→12→19→32→40→32→31→17→16→1
veh 5  1→5→6→11→18→19→18→17→30→29→28→22→14→9→1
veh 6  1→5→10→17→24→25→22→13→8→9→1
veh 7  1→9→14→23→28→36→44→45→49→48→45→46→38→39→31→17→16→1
veh 8  1→15→23→29→37→38→39→40→47→39→30→29→24→15→1
veh 9  1→9→14→22→21→26→25→33→41→42→43→44→35→27→28→23→15→1
veh 10  1→15→23→28→27→26→34→42→43→48→49→50→46→50→45→44→37→29→24→15→1

Note: "=" indicates a service and "−" a deadheading
References


