Math. Program., Ser. A (2013) 137:409–452
DOI 10.1007/s10107-011-0497-4

FULL LENGTH PAPER

Improved lower bounds and exact algorithm for the capacitated arc routing problem

Enrico Bartolini · Jean-François Cordeau · Gilbert Laporte

Received: 31 March 2011 / Accepted: 15 October 2011 / Published online: 3 November 2011

Abstract In the capacitated arc routing problem (CARP), a subset of the edges of an undirected graph has to be serviced at least cost by a fleet of identical vehicles in such a way that the total demand of the edges serviced by each vehicle does not exceed its capacity. This paper describes a new lower bounding method for the CARP based on a set partitioning-like formulation of the problem with additional cuts. This method uses cut-and-column generation to solve different relaxations of the problem, and a new dynamic programming method for generating routes. An exact algorithm based on the new lower bounds was also implemented to assess their effectiveness. Computational results over a large set of classical benchmark instances show that the proposed method improves most of the best known lower bounds for the open instances, and can solve several of these for the first time.

Keywords Arc routing · Set partitioning · Valid inequalities · Column generation · Dynamic programming

Mathematics Subject Classification (2000) 90C10 · 90C39 · 90C57

E. Bartolini (✉) · J.-F. Cordeau · G. Laporte
HEC Montréal, and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, QC H3T 2A7, Canada
e-mail: Enrico.Bartolini@cirrelt.ca

J.-F. Cordeau
e-mail: Jean-Francois.Cordeau@cirrelt.ca

G. Laporte
e-mail: Gilbert.Laporte@cirrelt.ca

1 Introduction

In arc routing problems, the aim is to determine a least cost traversal of a subset of arcs or edges of a graph subject to some constraints. The Capacitated Arc Routing Problem (CARP), introduced by [26], is one of the best known arc routing problems. It consists in servicing a set of demands associated with *required* links using a fleet of capacitated vehicles based at a depot. When all links are required, the problem is known as the Capacitated Chinese Postman Problem [15]. Solving the CARP and computing a 1.5-approximation are both NP-hard [26]. In this paper, we restrict our attention to the undirected version of the CARP.

The CARP arises in a wide variety of contexts, including refuse collection, street sweeping, winter gritting, postal delivery, inspection of power lines, bridges and pipeline systems, meter reading, etc. The surveys of [2] and of [21,22], and the book of [19] describe several real-world applications and discuss different solution methodologies. More recent surveys are those of [42] and [17].

1.1 Literature review

The CARP has been much less studied than the Capacitated Vehicle Routing Problem (CVRP) which is its natural counterpart in a node setting. To date, most of the research on the CARP concerns heuristics and lower bounding procedures. The earliest lower bounds for the CARP are based on matching techniques and on dynamic programming [1,3,10,26,28,38,40,41].

To our knowledge, the first attempt to solve the CARP by means of polyhedral combinatorics and linear programming (LP) techniques is due to [8]. This paper presented a polyhedral study of the CARP, and introduced valid inequalities which were computationally tested within a cutting plane algorithm. Building upon this study, Belenguer and Benavent [9] have developed a branch-and-cut method based on a very sparse and compact formulation involving an exponential class of inequalities from [8], and have introduced a new class of valid inequalities. Their formulation does not always yield a feasible solution, but it can be used to provide a lower bound on the CARP optimum. Computational tests reported by Belenguer and Benavent [9] have shown that lower bounds computed by means of this formulation were significantly better than previous ones, and matched the best known upper bounds for 47 out of the 87 tested instances.

Column generation algorithms for the CARP were recently proposed by [27,31], and [36]. These methods are based on formulating the CARP using an exponential number of variables, each corresponding to a possible vehicle route. Gómez-Cabrero et al. used a set covering formulation of the CARP where columns correspond to non-elementary routes, i.e., a relaxation of routes where a link is allowed to be serviced more than once. The LP relaxation of this formulation is strengthened by valid inequalities for the CARP and solved by cut-and-column generation. Letchford and Oukil [31] also adopted a set covering formulation and proposed two column generation methods based on elementary and non-elementary routes, respectively, with the aim of exploiting the sparsity of the network in generating routes. Non-elementary routes were computed by dynamic programming, whereas elementary routes were computed by solving a mixed-integer problem. Martinelli et al. [36] applied a similar approach, but used a set partitioning formulation and dynamic programming to generate both elementary and non-elementary routes. A branch-and-price algorithm for the CARP was very recently proposed by [14]. This algorithm can be viewed as a two-phase method that combines the sparse formulation of Belenguer and Benavent, and a set partitioning formulation based on non-elementary routes. The idea is to first solve the sparse formulation using a cutting plane procedure, and then apply a branch-and-price algorithm based on a set partitioning formulation where the initial master is initialized with the cuts identified in the first phase.

An indirect approach for solving the CARP was suggested by [33] and [5]. These two papers independently proposed a method to transform a CARP instance with m required links into an equivalent CVRP with $2m+1$ nodes. In [33] the resulting CVRP is solved by branch-and-cut-and-price by adapting the algorithm of [23], whereas in [5] the resulting CVRP is modeled using a two-index formulation and solved by branch-and-cut.

These transformations appear at present to be the most promising methods for solving the CARP to optimality. They outperformed all other specific lower bounding procedures, and could solve to optimality several instances for the first time. However, we note that preliminary results reported by [14] show that their two-phase method is also competitive with the best previous methods, and is able to find new optima for some open benchmark instances.

1.2 Contributions of this paper

We propose a new lower bounding methodology for the CARP based on cut-and-column generation to solve a set partitioning (SP) formulation of the CARP where each column corresponds to a route. The LP relaxation of this formulation is strengthened using both valid inequalities which are specific to the CARP, and valid inequalities for the set partitioning polytope.

This method embeds four lower bounding procedures executed in sequence to progressively obtain better lower bounds. Two of these procedures are based on non-elementary CARP routes and two on elementary ones. CARP routes are generated by dynamic programming using a transformation of the original CARP instance into a Generalized Vehicle Routing Problem (GVRP) in which each required edge is represented by a cluster containing two nodes. The new dynamic programming algorithm uses bounding functions and different fathoming rules to reduce the size of the state space. To assess the effectiveness of the new lower bounds, we have implemented an exact algorithm that uses the final dual solution computed by the lower bounding method to generate a reduced integer problem which is guaranteed to contain an optimal solution, and is solved using an integer programming (IP) solver.

We present extensive computational results over five sets of benchmark CARP instances showing that the proposed lower bound is consistently better than previous ones. Our results also show that the new exact algorithm based on this lower bound is competitive with the best known ones, and can solve to optimality 27 previously unsolved instances with up to 159 required edges, a size that is likely to be intractable for methods based on a transformation of the problem into a CVRP.

1.3 Organization of this paper

The remainder of the paper is organized as follows. In Sect. 2 we formally describe the CARP and formulate it as a set partitioning-like integer model. In Sect. 3 we present the valid inequalities used to strengthen the SP formulation, and we show how some known valid inequalities for the CARP can be lifted when translated into this formulation. Section 4 provides an overview of our algorithm. It first computes a lower bound on the CARP optimum and then uses it to attempt to obtain an optimal solution. In Sect. 5, we describe a transformation of the CARP into an asymmetric GVRP and establish a one-to-one correspondence between optimal CARP routes and GVRP routes. In Sect. 6 we describe a dynamic programming algorithm for the generation of elementary CARP routes based on the results of Sect. 5. In Sect. 7, we detail four lower bounding procedures based on column generation which are used to compute the final lower bound. Computational results on the main test instances from the literature are reported in Sect. 8, followed by conclusions in Sect. 9.

2 Formal problem description and mathematical formulation

Let $G = (V, E)$ be a connected undirected graph where $V = \{0, \ldots, n\}$ is a set of $|V| = n + 1$ nodes, node 0 is a *depot*, and E is a set of edges with endpoints in V. With each edge $e \in E$ are associated a demand $q_e \geq 0$ and a travel cost $c_e \geq 0$. Edges requiring service are called *required edges*, and the subset of required edges is denoted by $E_R \subseteq E$. We denote by $m = |E_R|$ the number of required edges and following [8,9], we assume that required edges have a strictly positive demand. We denote by i_e and j_e the two endpoints of edge e, and by $A = \{(i_e, j_e), (j_e, i_e) : e \in E\}$ the set of arcs associated with E. Conversely, for any arc $a = (i_e, j_e) \in A$, the mapping $e(a)$ gives the corresponding edge e. We also denote by $A_R \subseteq A$ the subset of arcs associated with the required edge set E_R.

A walk $R = (v_0, e_0, v_1, e_1, \ldots, v_p)$ in G is a an alternating sequence of vertices $v_0, \ldots, v_p \in V$ and edges $e_k = \{v_k, v_{k+1}\} \in E$, $k = 0, \ldots, p - 1$. The set of edges traversed by a walk R is denoted by $E(R)$. A *CARP route* (or simply a route) is a walk R for which $v_0 = v_p = 0$, and that *services* a subset $S(R) \subseteq E(R) \cap E_R$ of required edges whose total demand does not exceed a *vehicle capacity* Q. Each route represents the walk of a vehicle based at the depot and servicing a non-empty subset of the required edges. An edge traversed by a route but not serviced by it, or an edge traversed more than once is said to be *deadheaded* by that route. Similarly, a path corresponding to a sequence of edges deadheaded by the same route is said to be deadheaded by that route.

The CARP is to find K routes in G such that each required edge is serviced by exactly one route, and the total cost of all edges traversed by the corresponding walks is minimized. In the CARP literature, K is either fixed, free, or bounded above. As in [33] and [9], we assume that K is a free decision variable.

Let $\mathscr{R}$ be the index set of all routes. For each route R_ℓ, $\ell \in \mathscr{R}$, let $a_{\ell e}$ be the number of times edge $e \in E_R$ is serviced by route R_ℓ, and let $b_{\ell e}$ be the number of times edge $e \in E$ is traversed by route R_ℓ. We denote by $S(R_\ell)$, or simply S_ℓ, the set of required edges serviced by route R_ℓ, and by $d_{\ell e} = b_{\ell e} - a_{\ell e}$ the number of times route R_ℓ deadheads edge $e \in E$. Let $c_\ell = \sum_{e \in E} b_{\ell e} c_e$ be the cost of route R_ℓ, $\ell \in \mathscr{R}$, and let x_ℓ be a binary variable equal to 1 if and only if route R_ℓ is in the solution. The CARP can be formulated as the following IP model.

$$z(SP) = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell \tag{1}$$

$$s.t. \sum_{\ell \in \mathscr{R}} a_{\ell e} x_\ell = 1, \ \forall e \in E_R, \tag{2}$$

$$\sum_{\ell \in \mathscr{R}} x_\ell \geq K^*, \tag{3}$$

$$x_\ell \in \{0, 1\}, \ \forall \ell \in \mathscr{R}. \tag{4}$$

Constraints (2) impose that each required edge is serviced by exactly one route, and (3) is a redundant constraint imposing a lower bound $K^* = \left\lceil \sum_{e \in E_R} q_e / Q \right\rceil$ on the number of routes needed to service all required edges.

Note that in any optimal SP solution, the sequence of edges on a route R_ℓ between any two required edges e_1 and e_2 serviced consecutively by R_ℓ corresponds to a shortest path between an endpoint of e_1 and an endpoint of e_2. Therefore, we assume that routes not satisfying this condition do not belong to $\mathscr{R}$. If there are several shortest paths of equal length between two nodes, we can a priori select one without excluding all optimal solutions. We henceforth write *the* shortest path. Let P_{ij} be the shortest path from node $i \in V$ to node $j \in V$. We denote by ζ_{ij}^ℓ the number of times route R_ℓ deadheads both paths P_{ij} and P_{ji} (i.e., the number of times it deadheads path P_{ij}, plus the number of times it deadheads path P_{ji}).

3 Valid inequalities for SP

The cost $z(LSP)$ of an optimal solution to the LP relaxation of SP (denoted by LSP) may yield a weak lower bound. We now describe three main classes of valid inequalities that we have incorporated into SP to yield a stronger formulation: odd edge cutset constraints, capacity constraints, and subset-row inequalities.

3.1 Odd edge cutset constraints

Odd edge cutset constraints were originally introduced by [8,9] and can be defined as follows. Given a feasible SP solution $\mathbf{x}$, let z_e be an integer aggregated variable representing the number of times edge e is deadheaded:

$$z_e = \sum_{\ell \in \mathscr{R}} d_{\ell e} x_\ell. \tag{5}$$

For any node set $S \subseteq V$, let $\delta(S) = \{e \in E : i_e \in S, j_e \in V\backslash S$ or $j_e \in S, i_e \in V\backslash S\}$, and let $\delta_R(S) = \delta(S) \cap E_R$. Define $\mathscr{S} = \{S \subseteq V\backslash\{0\} : |\delta_R(S)|$ is odd$\}$. Odd edge cutset inequalities are as follows:

$$\sum_{e\in\delta(S)} z_e \geq 1, \quad \forall S \in \mathscr{S}. \tag{6}$$

Because the solution graph induced by any feasible SP solution is Eulerian, inequalities (6) express the fact that if $|\delta_R(S)|$ is odd, then at least one edge incident to S must be deadheaded.

Building on the assumption that the shortest path between any two consecutive serviced edges is unique, inequalities (6) can be strengthened as follows. Given a feasible SP solution $\mathbf{x}$, define an aggregated variable y_{ij} representing the number of times both shortest paths P_{ij} and P_{ji} between nodes $i, j \in V$ are deadheaded, that is,

$$y_{ij} = \sum_{\ell\in\mathscr{R}} \zeta_{ij}^{\ell} x_\ell, \quad \forall i, j \in V. \tag{7}$$

Using variables y_{ij}, we obtain the following lifted odd edge cutset inequalities:

$$\sum_{\substack{i\in S\\ j\in V\backslash S}} y_{ij} \geq 1, \quad \forall S \in \mathscr{S}. \tag{8}$$

Constraints (8) are shown to be valid by extending the argument used to prove the validity of odd edge cutset inequalities. Because $|\delta_R(S)|$ is odd, each route crosses S an even number of times, and each required edge is serviced by exactly one route, then at least one path crossing node set S has to be deadheaded.

Let $E(P_{ij}) \subseteq E$ be the set of edges traversed by path P_{ij}. Note that because G is symmetric, we can assume $P_{ij} = P_{ji}$, and therefore variables y_{ij} and z_e can be linked by the following equations:

$$z_e = \sum_{\substack{i,j\in V,\ i<j,\\ e\in E(P_{ij})}} y_{ij}, \quad \forall e \in E. \tag{9}$$

It is then easy to see that an inequality (8) defined by set S corresponds to a lifting of the corresponding inequality (6) defined by the same node set. In fact, using equations (9), (6) can be rewritten as

$$\sum_{\substack{i,j\in V,\ i<j,\\ E(P_{ij})\cap\delta(S)\neq\emptyset}} y_{ij} \geq 1, \quad \forall S \in \mathscr{S}, \tag{10}$$

which is dominated by (8).

Note that inequalities (8) can be written in terms of the x_ℓ variables by using equations (7), and because their right-hand side is equal to one, their coefficients are positive

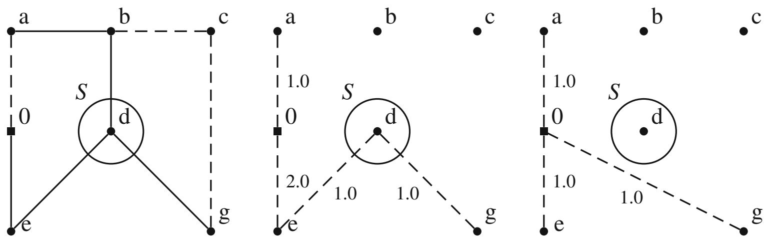


Fig. 1 Example of lifted inequality (8) defined by a node set S. From *left* to *right* input graph (*dashed edges* are non-required), support graph induced by variables z_e, and support graph induced by variables y_{ij}

integers, and x_ℓ variables are binary, they can further be lifted to obtain the following inequalities:

$$\sum_{\substack{i\in S\\ j\in V\setminus S}} \sum_{\substack{\ell\in\mathscr{R}\\ \zeta_{ij}^{\ell}>0}} x_\ell \geq 1, \quad \forall S \in \mathscr{S}. \tag{11}$$

However, we use inequalities (8) instead of (11) because it is straightforward to incorporate these inequalities exactly within the pricing problem.

The following example shows an inequality (6) defined by a set S dominated by the corresponding lifted inequality (8) defined by the same set. Consider the graph shown at the left of Fig. 1. In this graph, dashed lines represent non-required edges and a node set $S = \{d\}$ is defined. Suppose all edge costs correspond to Euclidean distances. A feasible LSP solution is given by the four routes $R_{\ell_1} = (0, a, b, d, g, d, e, 0)$ with $S_{\ell_1} = \{\{a, b\}, \{b, d\}, \{d, g\}\}$, $R_{\ell_2} = (0, e, d, g, d, e, 0)$ with $S_{\ell_2} = \{\{0, e\}, \{e, d\}, \{d, g\}\}$, $R_{\ell_3} = (0, a, b, d, e, 0)$ with $S_{\ell_3} = \{\{a, b\}, \{b, d\}, \{d, e\}\}$, and $R_{\ell_4} = (0, e, 0)$ with $S_{\ell_4} = \{\{0, e\}\}$, and setting $x_{\ell_1} = x_{\ell_2} = x_{\ell_3} = x_{\ell_4} = 0.5$. The center and right parts of Fig. 1 show the support graphs defined by vectors $\mathbf{z}$ and $\mathbf{y}$ associated with solution $\mathbf{x}$, respectively. It is easy to check that inequality (6) defined by node set S is satisfied by $\mathbf{z}$, whereas inequality (8) is violated by $\mathbf{y}$.

3.2 Capacity constraints

Let $\mathscr{F} = \{F \subseteq E_R : |F| \geq 2\}$ be the family of all subsets of required edges of cardinality at least 2, and let $\mathscr{R}(F) \subseteq \mathscr{R}$ be the index-subset of routes servicing at least one edge in the set $F \in \mathscr{F}$. Moreover, for any edge subset $F \in \mathscr{F}$, let $q(F) = \sum_{e\in F} q_e$ be the sum of the demands of the edges in F. Because at least $r(F) = \lceil q(F)/Q \rceil$ routes are required to service F, the following capacity constraints are valid for SP:

$$\sum_{\ell\in\mathscr{R}(F)} x_\ell \geq r(F), \quad \forall F \in \mathscr{F}. \tag{12}$$

Since a route is a closed walk starting and ending at the depot, inequalities (12) can be reformulated as follows. Let β_{ef}^{ℓ} be a binary coefficient equal to 1 if and only if route R_ℓ services in sequence required edges $e, f \in E_R$. Moreover, let β^ℓ_{0e} be equal to 1 if edge $e \in E_R$ is the first or last edge serviced by route R_ℓ ($\beta^\ell_{0e} = 2$ if route R_ℓ services only edge e). Given a feasible SP solution $\mathbf{x}$, define aggregated variables ξ_{ef} as follows:

$$\xi_{ef} = \sum_{\ell \in \mathscr{R}} \beta^\ell_{ef} x_\ell, \ \forall e, f \in E_R, \ \text{ and } \ \xi_{0e} = \sum_{\ell \in \mathscr{R}} \beta^\ell_{0e} x_\ell, \ \forall e \in E_R. \tag{13}$$

Inequalities (12) can then be written in the following weaker form:

$$\sum_{\substack{e \in F \\ f \in E_R \setminus F}} \xi_{ef} + \sum_{e \in F} \xi_{0e} \geq 2r(F), \quad \forall F \in \mathscr{F}. \tag{14}$$

The interest of inequalities (14) is that although they are weaker than (12) they are stronger than the CARP capacity constraints introduced in [8,9], and can also be incorporated exactly in the pricing problem.

For any node set $S \subseteq V \setminus \{0\}$, let $E_R(S) = \{e \in E_R : i_e, j_e \in S\}$. The CARP capacity constraints of Belenguer and Benavent are defined in terms of the aggregated variables z_e as follows:

$$\sum_{e \in \delta(S)} z_e \geq 2 \left\lceil \frac{q(E_R(S)) + q(\delta_R(S))}{Q} \right\rceil - |\delta_R(S)|, \quad \forall S \subseteq V \setminus \{0\}. \tag{15}$$

The following theorem shows that inequalities (14) (and therefore (12)) are stronger than the CARP capacity constraints (15).

Theorem 1 *Let $S \subseteq V \setminus \{0\}$, and define a corresponding edge set $F_S = \{E_R(S) \cup \delta_R(S)\}$. Let $\mathbf{x}$ be a feasible LSP solution, and let $\boldsymbol{\xi}$ and $\mathbf{z}$ be the corresponding vectors obtained through equations (13) and (5), respectively. An inequality (15) defined by node set S is dominated by inequality (14) defined by edge set F_S.*

Proof First note that if $\delta_R(S) = \emptyset$ any route $R_\ell, \ell \in \mathscr{R}$, servicing an edge $e \in F_S$ deadheads at least one edge in $\delta(S)$. Moreover, it deadheads at least two edges in $\delta(S)$ if it services a single edge of F_S. Therefore, because of the definition of variables ξ_{ef} and z_e the left hand side of inequality (15) defined by S is greater than or equal to that of inequality (14) defined by edge set F_S. Since in this case $r(F_S) = \lceil q(E_R(S))/Q \rceil$, inequality (14) dominates inequality (15). On the contrary, suppose that $\delta_R(S) \neq \emptyset$, and let us denote by $\ell_e \in \mathscr{R}$ the index of the route servicing only required edge $e, \forall e \in E_R$. Denote by $\mathscr{R}^1(\delta_R(S)) = \{\ell \in \mathscr{R} : \beta^\ell_{ef} = 1, \beta^\ell_{eg} = 1, e \in \delta_R(S), f, g \in E_R \setminus F_S\} \cup \{\ell \in \mathscr{R} : \beta^\ell_{0e} = 1, \beta^\ell_{ef} = 1, e \in \delta_R(S), f \in E_R \setminus F_S\} \cup \{\bigcup_{e \in \delta_R(S)} \ell_e\}$, and let $\mathscr{R}^2(\delta_R(S)) = \{\ell \in \mathscr{R} : \beta^\ell_{ef} = 1, \beta^\ell_{eg} = 1, e \in \delta_R(S), f \in E_R \setminus F_S, g \in E_R(S)\} \cup \{\ell \in \mathscr{R} : \beta^\ell_{0e} = 1, \beta^\ell_{ef} = 1, e \in \delta_R(S), f \in E_R(S)\} \cup \{\bigcup_{e \in \delta_R(S)} \ell_e\}$. Note that any route in the set $\mathscr{R}^1(\delta_R(S))$ deadheads at least one edge in $\delta(S)$. Moreover, each route $R_{\ell_e}, e \in E_R$, deadheads at least two edges in $\delta(S)$, and any route servicing in sequence two edges $e, f \in E_R$ such that $e \in E_R(S)$ and $f \in E_R \setminus F_S$ also has to deadhead at least one edge in $\delta(S)$. Therefore from the definition of variables z_e and ξ_{ef} we have

$$\sum_{e\in\delta(S)} z_e \geq \sum_{\substack{e\in E_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in E_R(S)} \xi_{0e} + \sum_{\ell\in\mathscr{R}^1(\delta_R(S))} x_\ell. \tag{16}$$

The definitions of sets $\mathscr{R}^1(\delta_R(S))$, $\mathscr{R}^2(\delta_R(S))$ and of variables ξ_{ef} imply

$$\sum_{\substack{e\in\delta_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in\delta_R(S)} \xi_{0e} = \sum_{\ell\in\mathscr{R}^1(\delta_R(S))} x_\ell + \sum_{\ell\in\mathscr{R}^2(\delta_R(S))} x_\ell, \tag{17}$$

and therefore, adding $\sum_{\ell\in\mathscr{R}^2(\delta_R(S))} x_\ell$ to both sides of (16) we obtain

$$\sum_{e\in\delta(S)} z_e + \sum_{\ell\in\mathscr{R}^2(\delta_R(S))} x_\ell \geq \sum_{\substack{e\in E_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in E_R(S)} \xi_{0e} + \sum_{\substack{e\in\delta_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in\delta_R(S)} \xi_{0e}. \tag{18}$$

Let $\mathscr{R}_e = \{\ell \in \mathscr{R} : a_{\ell e} = 1\}$. Because $\mathbf{x}$ satisfies constraints (2), and $\mathscr{R}^2(\delta_R(S)) \subseteq \bigcup_{e\in\delta_R(S)} \mathscr{R}_e$, we also have $\sum_{\ell\in\mathscr{R}^2(\delta_R(S))} x_\ell \leq |\delta_R(S)|$, and from (18) we obtain

$$\sum_{\substack{e\in E_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in E_R(S)} \xi_{0e} + \sum_{\substack{e\in\delta_R(S)\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in\delta_R(S)} \xi_{0e} \leq \sum_{e\in\delta(S)} z_e + |\delta_R(S)|, \tag{19}$$

and the inequality can be strict. Because $F_S = \{E_R(S) \cup \delta_R(S)\}$, the left-hand side of (19) can be rewritten as $\sum_{\substack{e\in F_S\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in F_S} \xi_{0e}$, and we have $2r(F_S) = 2\left\lceil \frac{q(E_R(S))+q(\delta_R(S))}{Q} \right\rceil$. Then, subtracting $2r(F_S)$ from both sides of (19) yields

$$\sum_{\substack{e\in F_S\\ f\in E_R\setminus F_S}} \xi_{ef} + \sum_{e\in F_S} \xi_{0e} - 2r(F_S) \leq \sum_{e\in\delta(S)} z_e + |\delta_R(S)| - 2r(F_S), \tag{20}$$

that is, the slack of inequality (14) defined by edge set F_S is at least as small as that of inequality (15) defined by set S. □

Note that inequalities (15) are formally the same as rounded capacity inequalities for the two-index formulation of the CVRP. Indeed, letting $E'_R = E_R \cup \{0\}$, it is easy to see that any vector $\boldsymbol{\xi}$ associated with a feasible SP solution $\mathbf{x}$ satisfies

$$\sum_{j\in E'_R} \xi_{ij} = 2, \qquad \forall i \in E_R, \tag{21}$$

$$\sum_{\substack{i\in F\\ j\in E'_R\setminus F}} \xi_{ij} \geq 2\left\lceil \frac{q(F)}{Q} \right\rceil, \forall F \subseteq E_R, |F| \geq 2 \tag{22}$$

$$\xi_{ij} \in \{0, 1\}, \qquad \forall i, j \in E_R, \tag{23}$$

$$\xi_{0i} \in \{0, 1, 2\}, \qquad \forall i \in E_R. \tag{24}$$

Thus, any valid inequality for the polytope associated with (22)–(24) is also valid for SP, and can be translated into the SP model using equations (13). In our preliminary experiments, we have considered the use of strengthened combs and framed capacity inequalities, presented in [35], using the CVRPSEP package of [34] to separate them. However, because the resulting improvement in the lower bounds was marginal, these inequalities were not used in the computational experiments reported in Sect. 8.

3.3 Subset-row inequalities

Let $\mathscr{C} = \{C \subseteq E_R : |C| = 3\}$ be the set of all required edge triplets, i.e., subsets of three required edges. For $C \in \mathscr{C}$, let $\mathscr{R}^2(C) \subseteq \mathscr{R}$ be the subset of routes servicing at least two edges in C, i.e., $\mathscr{R}^2(C) = \{\ell \in \mathscr{R} : |S_\ell \cap C| \geq 2\}$. Because each required edge has to be serviced by exactly one route, the following inequalities are valid for SP:

$$\sum_{\ell\in\mathscr{R}^2(C)} x_\ell \leq 1, \quad \forall C \in \mathscr{C}. \tag{25}$$

Inequalities (25) are a subset of the clique inequalities and a special case of the subset-row inequalities introduced by [29].

We denote by LRP the LP relaxation of the problem obtained by adding to SP all inequalities (8), (14), and (25), and by DRP its dual. Let $\pi_e, e \in E_R$, and π_0 be the dual variables associated with constraints (2) and (3), respectively, and let $\upsilon_F, F \in \mathscr{F}, w_S, S \in \mathscr{S}$, and $g_C, C \in \mathscr{C}$, be dual variables associated with constraints (14), (8) and (25), respectively. We denote a DRP solution by $(\boldsymbol{\pi}, \boldsymbol{\upsilon}, \mathbf{w}, \mathbf{g})$, where $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_m)$, $\boldsymbol{\upsilon} = (\upsilon_1, \ldots, \upsilon_{|\mathscr{F}|})$, $\mathbf{w} = (w_1, \ldots, w_{|\mathscr{S}|})$, and $\mathbf{g} = (g_1, \ldots, g_{|\mathscr{C}|})$.

4 Overview of the algorithm

This section provides an overview of our algorithm. This algorithm sequentially executes four lower bounding procedures, called BP1, BP2, BP3 and BP4, to solve increasingly tighter relaxations of LRP, and thus to compute a sequence of non-decreasing lower bounds. An exact algorithm then exploits the final lower bound to generate an optimal CARP solution. The four lower bounding procedures use column generation to dynamically generate the set of routes $\mathscr{R}$. They thus define an initial master problem in which the full route set $\mathscr{R}$ is substituted by a small subset $\overline{\mathscr{R}} \subseteq \mathscr{R}$. At each iteration they use the current master dual solution to solve the pricing problem which consists in finding the CARP route of minimum reduced cost with respect to the master dual solution. A detailed description of procedures BP1, BP2, BP3 and BP4 is provided in Sect. 7.

To compute the lower bounds, our algorithm first generates CARP routes by transforming the CARP into an equivalent asymmetric GVRP in which GVRP routes correspond to CARP routes. This transformation is described in Sect. 5, whereas Sect. 6 describes a dynamic programming algorithm, called GENROUTES, to actually generate the GVRP routes, and translate them into the corresponding CARP ones.

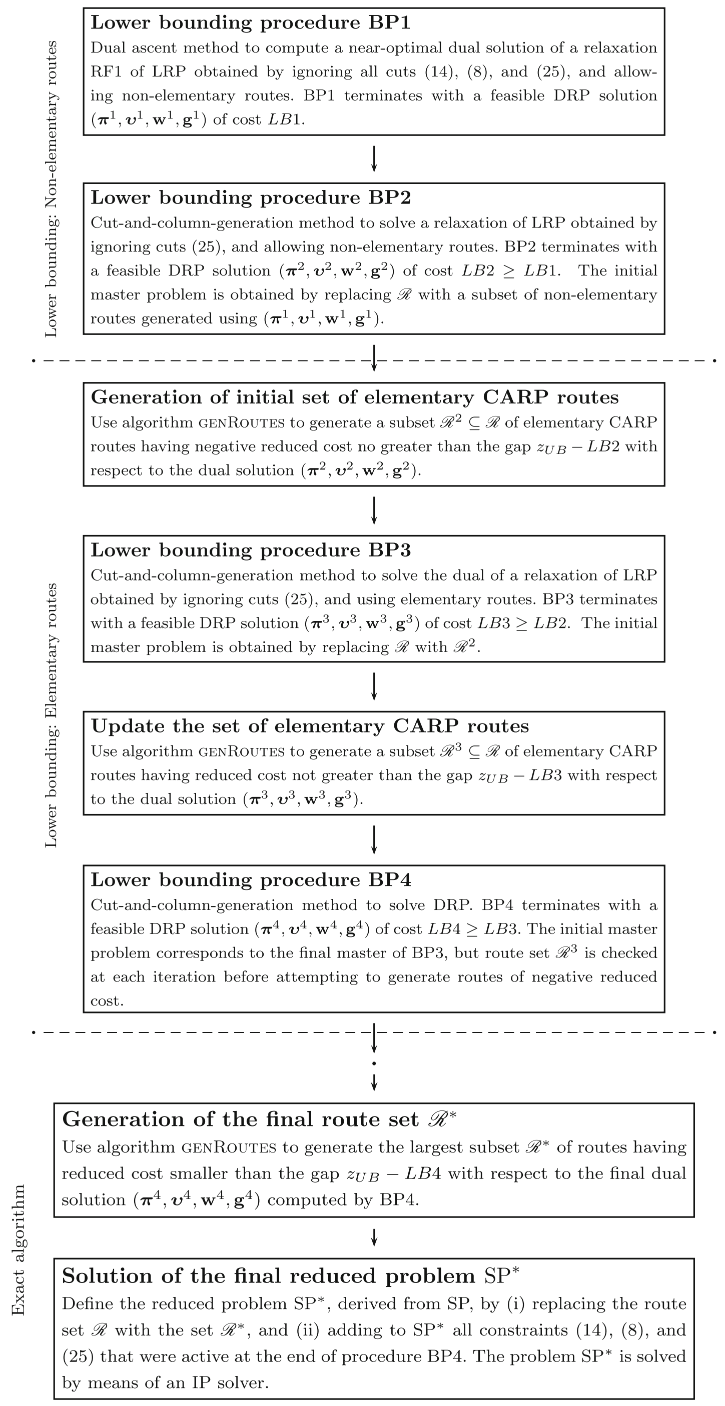
Lower bounding procedures BP1 and BP2 are based on non-elementary routes, whereas BP3 and BP4 use elementary routes. The algorithm described in this paper can thus be viewed as a three-step procedure (see Fig. 2) in which two lower bounding methods based on elementary and non-elementary CARP routes are first executed in sequence, and an exact algorithm then attempts to obtain a reduced integer problem from SP containing at least one optimal solution.

The algorithm requires the knowledge of a valid upper bound z_{UB} on the CARP optimum. It starts by executing in sequence procedures BP1, BP2, BP3, and BP4 to compute a final lower bound $LB4$ and a corresponding dual solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$. It then generates the largest subset $\mathscr{R}^* \subseteq \mathscr{R}$ of routes having a reduced cost smaller than the gap $z_{UB} - LB4$ with respect to the final dual solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$. An optimal CARP solution can then be obtained by solving a reduced problem derived from SP by replacing the entire route set $\mathscr{R}$ with $\mathscr{R}^*$. The full algorithm can be summarized as follows.

1. Execute in sequence the lower bounding procedures BP1, BP2, BP3, and BP4. If $LBx = z_{UB}, x \in \{1, 2, 3, 4\}$, then stop since z_{UB} is the cost of an optimal solution.
2. Set $\varrho = z_{UB} - LB4$ and $\Delta = \Delta^{MAX}$. Use algorithm GENROUTES to generate the largest subset $\mathscr{R}^*$ of routes having a reduced cost smaller than ϱ with respect to the final dual solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$ computed by BP4.
3. Let $\overline{\mathscr{S}}^4$, $\overline{\mathscr{F}}^4$, and $\overline{\mathscr{C}}^4$ be the subsets of inequalities (14), (8), and (25), respectively, active at the end of procedure BP4. Define the reduced problem SP* derived from SP by (i) replacing the route set $\mathscr{R}$ with the set $\mathscr{R}^*$, and (ii) adding to SP* all constraints $\overline{\mathscr{S}}^4$, $\overline{\mathscr{F}}^4$, and $\overline{\mathscr{C}}^4$. The problem SP* is solved by an IP solver.

In the following, we refer to Step 1 as a lower bounding method, and to Steps 2 and 3 as an exact algorithm.

Before solving the reduced problem SP* at Step 3, the algorithm replaces all inequalities (14) defined by edge sets in $\overline{\mathscr{F}}^4$ with the corresponding stronger inequalities (12), and solves the LP relaxation of SP*. The final lower bound $LB4$ and the corresponding dual solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$ are then updated, and all routes having a reduced cost greater than $z_{UB} - LB4$ with respect to $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$ are removed from $\mathscr{R}^*$.


Lower bounding: Non-elementary routes
Lower bounding: Elementary routes
Exact algorithm

Lower bounding procedure BP1
Dual ascent method to compute a near-optimal dual solution of a relaxation RF1 of LRP obtained by ignoring all cuts (14), (8), and (25), and allowing non-elementary routes. BP1 terminates with a feasible DRP solution $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$ of cost $LB1$.

Lower bounding procedure BP2
Cut-and-column-generation method to solve a relaxation of LRP obtained by ignoring cuts (25), and allowing non-elementary routes. BP2 terminates with a feasible DRP solution $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$ of cost $LB2 \geq LB1$. The initial master problem is obtained by replacing $\mathscr{R}$ with a subset of non-elementary routes generated using $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$.

Generation of initial set of elementary CARP routes
Use algorithm genRoutes to generate a subset $\mathscr{R}^2 \subseteq \mathscr{R}$ of elementary CARP routes having negative reduced cost no greater than the gap $z_{UB} - LB2$ with respect to the dual solution $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$.

Lower bounding procedure BP3
Cut-and-column-generation method to solve the dual of a relaxation of LRP obtained by ignoring cuts (25), and using elementary routes. BP3 terminates with a feasible DRP solution $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^3, \mathbf{g}^3)$ of cost $LB3 \geq LB2$. The initial master problem is obtained by replacing $\mathscr{R}$ with $\mathscr{R}^2$.

Update the set of elementary CARP routes
Use algorithm genRoutes to generate a subset $\mathscr{R}^3 \subseteq \mathscr{R}$ of elementary CARP routes having reduced cost not greater than the gap $z_{UB} - LB3$ with respect to the dual solution $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^3, \mathbf{g}^3)$.

Lower bounding procedure BP4
Cut-and-column-generation method to solve DRP. BP4 terminates with a feasible DRP solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$ of cost $LB4 \geq LB3$. The initial master problem corresponds to the final master of BP3, but route set $\mathscr{R}^3$ is checked at each iteration before attempting to generate routes of negative reduced cost.

Generation of the final route set $\mathscr{R}^*$
Use algorithm genRoutes to generate the largest subset $\mathscr{R}^*$ of routes having reduced cost smaller than the gap $z_{UB} - LB4$ with respect to the final dual solution $(\boldsymbol{\pi}^4, \boldsymbol{\upsilon}^4, \mathbf{w}^4, \mathbf{g}^4)$ computed by BP4.

Solution of the final reduced problem SP*
Define the reduced problem SP*, derived from SP, by (i) replacing the route set $\mathscr{R}$ with the set $\mathscr{R}^*$, and (ii) adding to SP* all constraints (14), (8), and (25) that were active at the end of procedure BP4. The problem SP* is solved by means of an IP solver.

Fig. 2 Overview of the algorithm

5 Transformation of the CARP into a GVRP

We first show how to transform the CARP into an equivalent asymmetric GVRP. It is indeed well known (see e.g., [13,24]) that a wide class of arc and node routing problems can be modeled as a GVRP. Here we describe this transformation in detail over an asymmetric graph $\widetilde{G}$, we establish a one-to-one correspondence between GVRP routes in $\widetilde{G}$ and CARP routes R_ℓ indexed in $\mathscr{R}$, and show that GVRP paths can be combined to yield CARP routes. As a result, we obtain a method for generating CARP routes as GVRP routes in $\widetilde{G}$. In this section, we also describe two relaxations of GVRP paths, namely the q-path relaxation and the ng-path relaxation (originally introduced for the CVRP), to be used by algorithm GENROUTES in Sect. 5.

5.1 Description of the transformation

Let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be a directed graph obtained from G as follows:

- $\widetilde{V}$ contains $2m + 1$ nodes partitioned into $m + 1$ clusters $\widetilde{V}_0, \dots, \widetilde{V}_m$, where $\widetilde{V}_0$ contains the depot. For each required edge $e \in E_R$, $\widetilde{V}_e$ contains two nodes associated with the two arcs $(i_e, j_e) \in A_R$ and $(j_e, i_e) \in A_R$, respectively. We denote by $\nu(a)$ the node $v \in \widetilde{V}$ associated with arc $a \in A_R$. For each node $v \in \widetilde{V}\backslash\{0\}$, we denote by $\alpha(v) \in A_R$ the arc associated with node v, and by $i(v)$ and $j(v)$ the initial and terminal endpoints of $\alpha(v)$, respectively. Moreover, we denote by $\varepsilon(v) = e(\alpha(v)) \in E_R$ the required edge corresponding to the arc $\alpha(v)$ associated with v. Note that $\widetilde{V}_{\varepsilon(v)}$ corresponds to the unique cluster containing v. For notational convenience, we also define $i(0) = j(0) = 0$, and $\varepsilon(0) = e_0$, where $e_0 = \{0, 0\}$ is a non-required dummy loop in G of cost $c_{e_0} = 0$.
- With each node $v \in \widetilde{V}$ is associated a demand $\tilde{q}_v = q_{\varepsilon(v)}$, ($\tilde{q}_0 = 0$).
- Let s_{ij} be the cost of the shortest path P_{ij} from node $i \in V$ to node $j \in V$. For each node pair $u, v \in \widetilde{V}\backslash\{0\}$ such that $\varepsilon(u) \neq \varepsilon(v)$, $\widetilde{E}$ contains two arcs (u, v) and (v, u) of cost $\tilde{c}_{uv} = s_{j(u)i(v)} + \frac{1}{2}c_{\varepsilon(u)} + \frac{1}{2}c_{\varepsilon(v)}$ and $\tilde{c}_{vu} = s_{j(v)i(u)} + \frac{1}{2}c_{\varepsilon(u)} + \frac{1}{2}c_{\varepsilon(v)}$, respectively. Moreover, for each $v \in \widetilde{V}\backslash\{0\}$, $\widetilde{E}$ contains two arcs $(0, v)$ and $(v, 0)$ of cost $\tilde{c}_{0v} = s_{0i(v)} + \frac{1}{2}c_{\varepsilon(v)}$ and $\tilde{c}_{v0} = s_{j(v)0} + \frac{1}{2}c_{\varepsilon(v)}$, respectively.

A *GVRP path* $P = (v_1, \dots, v_p)$ is a simple path in $\widetilde{G}$ starting from node $v_1 = b(P)$, visiting the nodes of $V(P) = \{v_1, \dots, v_p\}$, ending at node $v_p = e(P)$, and such that (i) its demand $q(P) = \sum_{k=1}^{p} \tilde{q}_{v_k}$ does not exceed Q, and (ii) $|V(P) \cap \widetilde{V}_e| \leq 1, \forall e \in E_R$. We denote by $A(P)$ and by $\mathscr{V}(P)$ the set of arcs and the index set of clusters traversed by path P, respectively. The cost of path P is defined as $\tilde{c}(P) = \sum_{(u,v)\in A(P)} \tilde{c}_{uv}$. A GVRP path P is called *forward* if $b(P) = 0$, and it is called *backward* if $e(P) = 0$. A GVRP path $\widetilde{R}$ such that $b(\widetilde{R}) = e(\widetilde{R}) = 0$ is called a *GVRP route*. In the following, we denote by $\widetilde{\mathscr{R}}$ the index-set of all GVRP routes in $\widetilde{G}$.

It is easy to see that any GVRP route $\widetilde{R}_\ell$, $\ell \in \widetilde{\mathscr{R}}$, can be obtained by combining a forward GVRP path P and a backward GVRP path $\overline{P}$ such that $e(P) = b(\overline{P}) = v$, for some $v \in \widetilde{R}_\ell$, and satisfying (A) $q(P) + q(\overline{P}) \leq Q + \tilde{q}_v$ and (B) $\mathscr{V}(P) \cap \mathscr{V}(\overline{P}) = \{0, \varepsilon(v)\}$. Note that although $\widetilde{G}$ is asymmetric, because of the definition of costs $\{\tilde{c}_{uv}\}$, any GVRP backward path $\overline{P} = (v_p, \dots, v_1, 0)$ in $\widetilde{G}$ corresponds

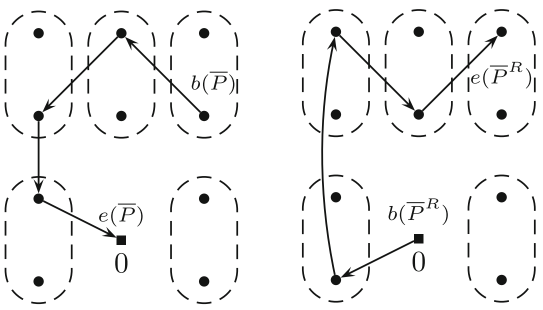


Fig. 3 Example of reverse path in $\widetilde{G}$: a backward path $\overline{P}$ on the *left*, and its reverse $\overline{P}^R$ on the *right*

to a forward path $\overline{P}^R$ of the same cost $\tilde{c}(\overline{P}^R) = \tilde{c}(\overline{P})$ obtained by setting $\overline{P}^R = (0, \widetilde{V}_{\varepsilon(v_1)} \backslash \{v_1\}, \ldots, \widetilde{V}_{\varepsilon(v_p)} \backslash \{v_p\})$. The forward path $\overline{P}^R$ is called the *reverse path* of the backward path $\overline{P}$. Figure 3 depicts a backward path $\overline{P}$ and its reverse path $\overline{P}^R$. It follows that any GVRP route $\widetilde{R}_\ell$, $\ell \in \widetilde{\mathscr{R}}$, can be obtained by combining two forward paths P and $\overline{P}^R$ such that $\overline{P}^R$ is the reverse of a backward path $\overline{P}$, and the pair $\{P, \overline{P}\}$ satisfies conditions (A) and (B), plus the following condition: (C) $e(\overline{P}^R) = \widetilde{V}_{\varepsilon(v)} \backslash e(P)$. In the following we refer to conditions (A), (B), and (C) as *route feasibility conditions*.

5.2 GVRP paths and CARP routes

The definition of a CARP route given in Sect. 2 gives rise to a number of redundant routes associated with those walks that traverse a serviced edge more than once, and this redundancy is not removed by the assumption that the path traversed between two edges serviced in sequence is the shortest one.

Consider a CARP route R_ℓ, and let $E(R_\ell) \subseteq E$ be the set of edges traversed by this route. We call a *line* of route R_ℓ a subset of edges $L \subseteq E(R_\ell)$ such that: (i) each edge $e \in L$ is traversed exactly twice by R_ℓ in opposite directions, (ii) edges in L form a simple open chain in G, and (iii) L is maximal with respect to inclusion. Each line L of a route R_ℓ such that $|S_\ell \cap L| = r$ gives rise to 2^r routes equivalent to R_ℓ (i.e., having the same cost and servicing the same edges) obtained by switching the service of each edge e in $S_\ell \cap L$ from the first to the second time it is visited by R_ℓ, or vice versa. The number of these equivalent routes can be huge, especially for routes containing more than one line.

A simple method to avoid this drawback is to impose an arbitrary ordering of service for all edges on a line. We say that a CARP route R_ℓ is a *forward-service route* if all edges in S_ℓ are serviced the first time they are traversed. Conversely, R_ℓ is called a *backward-service route* if all required edges in S_ℓ are serviced by R_ℓ the last time they are traversed. Similarly, we define a forward-service and backward-service CARP open route. Clearly, it is possible to assume that in any CARP solution all CARP routes are forward-service (equivalently, backward-service) routes.

It is easy to see that any GVRP path $P = (0, v_1, \ldots, v_p)$ corresponds to a *CARP open route* (a CARP route not ending at the depot) that services in sequence edges $\varepsilon(v_1), \ldots, \varepsilon(v_p)$ through arcs $\alpha(v_1), \ldots, \alpha(v_p)$, and deadheads paths

$P_{0i(v_1)}, \ldots, P_{j(v_{p-1})i(v_p)}$. Obviously, the same correspondence holds between any GVRP route $\widetilde{R}_\ell, \ell \in \widetilde{\mathscr{R}}$, and a CARP route $R_\ell, \ell \in \mathscr{R}$. Moreover, assuming that the path deadheaded by any CARP route $R_\ell, \ell \in \mathscr{R}$, between two edges serviced in sequence is the shortest one, and that R_ℓ is a CARP forward-service route, the converse is also true. We can therefore establish a one-to-one correspondence between CARP routes in G indexed by $\mathscr{R}$ and GVRP routes in $\widetilde{G}$ indexed by $\widetilde{\mathscr{R}}$, and between their costs. In the following, we denote by $\widetilde{R}_\ell, \ell \in \widetilde{\mathscr{R}}$, the GVRP route corresponding to CARP route $R_\ell, \ell \in \mathscr{R}$.

Note that any GVRP route $\widetilde{R}_\ell, \ell \in \mathscr{R}$, corresponding to a forward-service CARP route R_ℓ can be obtained by combining a forward GVRP path P and a reverse GVRP path $\overline{P}^R$ satisfying route feasibility conditions, and such that P corresponds to a forward-service CARP open route and $\overline{P}^R$ corresponds to a backward-service CARP open route. In the next section, we describe a dynamic programming algorithm that exploits this correspondence to generate CARP routes corresponding to GVRP routes in $\widetilde{G}$. This algorithm uses two relaxations of GVRP paths, namely q-paths and ng-paths, to compute lower bounds on the cost of the least cost GVRP route that can be obtained by combining a GVRP path P with any reverse path. These relaxations are now briefly described.

5.3 q-path relaxation of GVRP paths

The q-path relaxation was introduced by [16] for the CVRP. In the context of the CARP, a similar relaxation was proposed by [10], and also adopted to relax CARP routes by [27,31], and [14].

A *GVRP* (q, v)*-path* in $\widetilde{G}$ is a not necessarily simple path starting from node 0, ending at node $v \in \widetilde{V}$, and whose total demand q does not exceed Q. A GVRP $(q, 0)$-path is called a *GVRP* q*-route*. Let $f(q, v)$ be the cost of the least cost GVRP (q, v)-path in $\widetilde{G}$ ending in $v \in \widetilde{V}$, and let $\pi(q, v) \in E_R$ be the cluster visited just prior to v in this path. Let $g(q, v)$ be the cost of the least cost GVRP (q, v)-path ending at v with the constraint that the cluster $\chi(q, v)$ visited just before v in this second path is not equal to $\pi(q, v)$. We say that a GVRP (q, v)-path contains a loop of k consecutive clusters (*k-cluster-loop*) if it visits in sequence k nodes $v_1, \ldots, v_k$ such that $\varepsilon(v_1) = \varepsilon(v_k)$. The functions $f(q, v)$, and $g(q, v)$ can be computed in pseudo-polynomial time by an extension of the dynamic programming recursion described by [16], imposing the restriction that the corresponding (q, v)-paths do not contain 2-cluster-loops.

5.4 ng-path relaxation of GVRP paths

The ng-path relaxation was introduced by [6] for the CVRP and the vehicle routing problem with time windows. This relaxation can be extended to the GVRP as follows. For each cluster $\widetilde{V}_e, e \in E_R$, let $\widetilde{N}_e$ be a subset of clusters (selected according to some criterion) such that $V_e \in \widetilde{N}_e$ and $|\widetilde{N}_e| \leq \Delta(\widetilde{N}_e)$, where $\Delta(\widetilde{N}_e)$ is an a priori defined parameter. Clusters in $\widetilde{N}_e$ are called *neighbor clusters* of $\widetilde{V}_e$. Let Π be the family of all cluster subsets. Using sets N_e, associate with each GVRP path

$P = (0, v_1, \ldots, v_p)$ a cluster subset $\Pi(P) \in \Pi$ containing cluster $\widetilde{V}_{\varepsilon(v_p)}$ plus every other cluster $\widetilde{V}_{\varepsilon(v_k)}, k = 1, \ldots, p$, that belongs to all sets $\widetilde{N}_{\varepsilon(v_{k+1})}, \ldots, \widetilde{N}_{\varepsilon(v_p)}$. Formally, we have $\Pi(P) = \{\widetilde{V}_{\varepsilon(v_k)} \in \mathscr{V}(P) : \widetilde{V}_{\varepsilon(v_k)} \in \bigcap_{s=k+1,\ldots,p} \widetilde{N}_{\varepsilon(v_s)}, \quad k = 1, \ldots, p-1\} \cup \{\widetilde{V}_{\varepsilon(v_p)}\}$. Note that $\Pi(P)$ is a subset of the clusters visited by P that depends on the order in which these clusters are visited by P. For any path P in $\widetilde{G}$, let P' be the subpath of P obtained by removing the last node from P. A *GVRP ng-path* (NG, q, v) is a not necessarily simple path P in $\widetilde{G}$ of demand q, starting from node 0, ending at node $v \in \widetilde{V}$, and such that $\Pi(P) = NG$, and $\varepsilon(v) \notin \Pi(P')$. Let $f(NG, q, v)$ be the cost of the least cost ng-path (NG, q, v). Because of the definition of an ng-path (NG, q, v), a lower bound on the cost $\tilde{c}(P)$ of any forward GVRP path P can be obtained using functions $f(NG, q, v)$ as follows:

$$\tilde{c}(P) \geq \min_{NG \subseteq \mathscr{V}(P) \cap N_{\varepsilon(e(P))}} \{f(NG, q(P), e(P))\}. \tag{26}$$

The functions $f(NG, q, v), \forall NG \in \Pi, \forall v \in \tilde{V}, 0 \leq q \leq Q$, can be computed by means of a dynamic programming recursion similar to that described in [6]. The computational complexity of this recursion, unlike that used for q-path functions $f(q, v)$ and $g(q, v)$, is not pseudo-polynomial because of the exponential size of Π. However, in practice it is possible to considerably reduce the time spent for computing functions $f(NG, q, v)$ by limiting the size of sets $\widetilde{N}_e, \forall e \in E_R$, and using dominance rules as described by [7].

Note that, depending on the definition of sets $\widetilde{N}_e$, ng-paths are allowed to contain two-cluster loops. These loops can be eliminated using the method described by [16] for computing q-paths without 2-vertex loops. However, we do not implement this method, as in practice it is often sufficient to set $\Delta(\widetilde{N}_e) \geq 10$ to avoid such loops.

6 Dynamic programming algorithm genRoutes

We now describe a dynamic programming algorithm, called genRoutes, used by our lower bounding method to generate CARP routes.

For each route $R_\ell, \ell \in \mathscr{R}$, define the set $\mathscr{C}(R_\ell) = \{C \in \mathscr{C} : |C \cap S_\ell| \geq 2\}$, and let $\boldsymbol{\beta}_\ell(F) = \sum_{\substack{e \in F \\ f \in E_R \setminus F}} \beta^\ell_{ef} + \sum_{e \in F} \beta^\ell_{0e}, \forall F \in \mathscr{F}$, and $\zeta_\ell(S) = \sum_{\substack{i,j \in S \\ i<j}} \zeta^\ell_{ij}, \forall S \in \mathscr{S}$. Given a DRP solution $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$, the reduced cost $\bar{c}^r(R_\ell)$, or simply $\bar{c}^r_\ell$, of CARP route $R_\ell, \ell \in \mathscr{R}$, with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ is

$$\bar{c}^r_\ell = c_\ell - \sum_{e \in S_\ell} a_{e\ell}\pi_e - \sum_{F \in \mathscr{F}} \boldsymbol{\beta}_\ell(F)\upsilon_F - \sum_{S \in \mathscr{S}} \zeta_\ell(S) w_S - \sum_{C \in \mathscr{C}(R_\ell)} g_C - \pi_0. \tag{27}$$

genRoutes generates GVRP routes in $\widetilde{G}$ corresponding to forward-service CARP routes using bounding functions based on the ng-path and q-path relaxations. Given dual values $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$, and two user-defined parameters Δ and ϱ, genRoutes outputs a subset $\mathscr{D} \subseteq \mathscr{R}$ containing at most Δ forward-service CARP routes R_ℓ having reduced cost $\bar{c}^r_\ell$ not exceeding ϱ, and such that $\min_{\ell \in \mathscr{R} \setminus \mathscr{D}}\{\bar{c}^r_\ell\} \geq \min_{\ell \in \mathscr{D}}\{\bar{c}^r_\ell\}$.

6.1 Description of GENROUTES

Let $\mathscr{P}$ be the set of all GVRP forward paths in $\widetilde{G}$, and let $\mathscr{P}_v \subseteq \mathscr{P}$ be the subset of paths ending at vertex $v \in \widetilde{V}$. GENROUTES is a two-phase procedure that first generates path sets $\mathscr{P}_v$, $v \in \widetilde{V}$, and then combines these paths to extract the route set $\mathscr{D}$.

GENROUTES associates with each GVRP path $P \in \mathscr{P}$ a *modified path cost* $\bar{d}(P)$ such that, for each pair of paths $P, \overline{P} \in \mathscr{P}$ that can be combined to obtain a GVRP route $\widetilde{R}_\ell$ corresponding to R_ℓ, the modified costs satisfy $\bar{d}(P) + \bar{d}(\overline{P}) = \bar{d}(\widetilde{R}_\ell) \leq \bar{c}^r_\ell$. Therefore, the set $\mathscr{D}$ of CARP routes can be obtained by generating all GVRP routes having a modified cost not exceeding ϱ.

To compute the modified path costs, we associate a modified arc cost $\bar{d}_{uv}$ to each arc $(u, v) \in \widetilde{A}$, computed as

$$\bar{d}_{uv} = \tilde{c}_{uv} - \frac{1}{2}\bar{\pi}_{\varepsilon(u)} - \frac{1}{2}\bar{\pi}_{\varepsilon(v)} - \sum_{\substack{F \in \mathscr{F} s.t. \\ \varepsilon(u) \in F, \varepsilon(v) \notin F}} \bar{v}_F - \sum_{\substack{S \in \mathscr{S} s.t. \\ j(u) \in S, i(v) \notin S}} \bar{w}_S, \quad \forall u, v \in \widetilde{V} \tag{28}$$

For any edge triplet $C = \{e_1, e_2, e_3\} \in \mathscr{C}$, let $\widetilde{C} = \{\widetilde{V}_{e_1}, \widetilde{V}_{e_2}, \widetilde{V}_{e_3}\}$ be the corresponding cluster triplet in $\widetilde{G}$, and define $\mathscr{C}^k(P) = \{C \in \mathscr{C} : |\widetilde{C} \cap \mathscr{V}(P)| = k\}$ for each GVRP path P. Using modified arc costs $\bar{d}_{uv}$, the modified path cost $\bar{d}(P)$ associated with path $P \in \mathscr{P}$ is computed as

$$\bar{d}(P) = \sum_{(u,v) \in A(P)} \bar{d}_{uv} - \sum_{C \in \mathscr{C}^3(P)} \bar{g}_C - \sum_{\substack{C \in \mathscr{C}^2(P) \\ \varepsilon(e(P)) \notin C}} \bar{g}_C. \tag{29}$$

The following lemma shows that the modified costs of any two GVRP paths that can be combined to obtain a GVRP route $\widetilde{R}_\ell$ corresponding to R_ℓ provide a lower bound on $\bar{c}^r_\ell$.

Lemma 1 *Let P and $\overline{P}$ be two GVRP forward paths satisfying route feasibility conditions $(A) - (C)$, and let R_ℓ, $\ell \in \widetilde{\mathscr{R}}$, be the GVRP route obtained by combining P and $\overline{P}$. The reduced cost $\bar{c}^r_\ell$ of CARP route R_ℓ, $\ell \in \mathscr{R}$, corresponding to $\widetilde{R}_\ell$, satisfies $\bar{c}^r_\ell \geq \bar{d}(P) + \bar{d}(\overline{P})$.*

Proof See Appendix A.

Let $P \in \mathscr{P}$ be any GVRP path, and let $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{v}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ be any DRP solution. A *completion bound* on P with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{v}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ is defined as a lower bound on the reduced cost with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{v}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ of any CARP route corresponding to a GVRP route containing P. When generating the path set $\mathscr{P}$, GENROUTES uses ng-path and q-path bounding functions to associate with each GVRP path $P \in \mathscr{P}$ a completion bound $\overline{LB}^{ng}(P)$ on P with respect to the input DRP solution $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$. The lower bound $LB^{ng}(P)$ is used to fathom all paths P generated such that $LB^{ng}(P) > \varrho$, and is computed according to the following.

Lemma 2 *A valid lower bound $\overline{LB}^{ng}(P)$ on the reduced cost $\bar{c}^r_\ell$ with respect to a DRP solution $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ of any CARP route R_ℓ, $\ell \in \mathscr{R}$, corresponding to a GVRP route $\widetilde{R}_\ell$ containing path P can be computed as*

$$LB^{ng}(P) = \bar{d}(P) + \min_{\substack{NG \subseteq \Pi \ s.t. \ NG \cap \mathscr{V}(P) = \{V_{\varepsilon(e(P))}\} \\ q \le Q - q(P) + \tilde{q}_{e(P)}}} \{ f(NG, q, e(P)) \}, \tag{30}$$

where functions $f(NG, q, v)$ are computed using modified arc costs $\bar{d}_{uv}$.

Proof See Appendix A.

Although ng-path functions usually provide a better bound than q-path functions, because the computation of $\overline{LB}^{ng}(P)$ can be time-consuming due to the minimization over Π, we first compute a weaker completion bound $\overline{LB}^{q}(P)$ using functions $f(q, v)$ and $g(q, v)$ as follows:

$$\overline{LB}^{q}(P) = \bar{d}(P) + \min_{\tilde{q}_{e(P)} \le q \le Q - q(P) + \tilde{q}_{e(P)}} \begin{cases} f(q, e(P)), & \text{if } \pi(q, e(P)) \notin \mathscr{V}(P) \\ g(q, e(P)), & \text{otherwise,} \end{cases} \tag{31}$$

where $f(q, v)$ and $g(q, v)$ are computed using modified costs $\bar{d}_{uv}$. All paths P such that $\overline{LB}^{q}(P) > \varrho$ are fathomed.

The two phases of GENROUTES are called GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$, respectively. GEN$\mathscr{P}$ corresponds to a Dijkstra-like algorithm that builds path sets $\mathscr{P}_v$, $v \in \widetilde{V}$, by generating a sequence of GVRP forward paths having non decreasing completion bound with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$. COMBINE$\mathscr{P}$ is an iterative procedure that is executed after GEN$\mathscr{P}$, and generates a sequence of path pairs $(P, \overline{P})$ from the sets $\mathscr{P}_v$, $v \in \widetilde{V}$, that satisfy route feasibility conditions, and that have non-decreasing modified cost $\bar{d}(P) + \bar{d}(\overline{P}) \le \varrho$. These pairs are combined to obtain the route set $\mathscr{D}$. GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$ use four fathoming rules, fathoming 1 to fathoming 4, in order to reduce the number of generated paths and pairs. A detailed description of procedures GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$ and of the fathoming rules is provided in Appendix B.

7 Lower bounding procedures

We have developed four lower bounding procedures BP1, BP2, BP3, and BP4 to compute the final lower bound $LB4$ on the CARP optimum. Procedure BP1 is an extension of a procedure originally proposed by [4] for the CVRP that uses a dual ascent heuristic instead of the simplex algorithm to solve the master problem. Procedures BP2, BP3, and BP4 use the same simplex-based cut-and-column generation method, called LPCG, to solve different relaxations of LRP. A detailed description of LPCG is provided in Appendix C.

7.1 Lower bounding procedure BP1

Procedure BP1 solves a relaxation of problem LRP, called RF1, where constraints (14), (8), and (25) are ignored, and the route set $\mathscr{R}$ is enlarged to contain all CARP q-routes in G.

Here, we briefly define CARP q-routes, and relate them with corresponding GVRP q-routes in $\widetilde{G}$ (see Sect. 5.3). A CARP q-route is defined as a CARP route where the constraint that a required edge is serviced at most once is relaxed. A CARP q-route *ends at arc* a if $e(a) \in E_R$ is the last edge serviced, through arc a, before returning to the depot for the last time. Each CARP q-route ending at arc a corresponds to a GVRP q-route in $\widetilde{G}$ of the same cost *ending at node* $v = \nu(a)$, i.e., visiting node $v = \nu(a) \in \widetilde{V}$ just before returning to node 0. Conversely, given any GVRP q-route $\widetilde{R} = (0, v_1, \dots, v, 0)$ ending at node v, a corresponding CARP q-route ending at arc $\alpha(v)$ is obtained by servicing in sequence edges $\varepsilon(v_1), \dots, \varepsilon(v)$ through arcs $\alpha(v_1), \dots, \alpha(v)$, and deadheading the shortest paths $P_{0i(v_1)}, \dots, P_{j(v)0}$. It is obvious how to define a CARP (q, a)-path in G, $e(a) \in E_R$, using the same correspondence with a GVRP $(q, \nu(a))$-path in $\widetilde{G}$. We say that a CARP q-route contains a loop of k consecutive edges (k-loop) if it services in sequence k required edges $e_1, \dots, e_k = e_1$ (note that a 1-loop corresponds to servicing twice the same edge consecutively).

It is worth noting that the q-path relaxation adopted in this paper is slightly different from that commonly adopted in the existing CARP literature. Indeed, since we assume that a CARP route always deadheads shortest paths, a CARP (q, a)-path as defined in this paper always ends at a serviced edge $e(a)$, whereas the usual q-path relaxation for the CARP is obtained by defining, for each vertex $i \in V$, a (q, i)-path of demand q ending at vertex i. This last definition has the advantage of exploiting the sparsity of the graph, yielding a pricing algorithm for CARP q-routes which can be executed in $O(Q|E| + |V| \log |V|)$ time (see [31]) rather than in $O(Q|E_R|^2)$ time as in this paper. However, our definition enables the generation of CARP q-routes as corresponding GVRP q-routes in $\widetilde{G}$, and because there are no arcs between nodes of a same cluster in $\widetilde{G}$, imposing that GVRP q-routes cannot contain loops of two consecutive clusters implies that the corresponding CARP q-routes do not contain 3-loops. That is, pricing GVRP q-routes with 2-cycle elimination yields CARP q-routes without 3-loops. Therefore, the q-path relaxation adopted in this paper, which is quite fast to solve in practice, can yield better lower bounds than the classical one even if 2-cycle elimination is used.

Procedure BP1 is a dual ascent method that uses column generation to compute a near optimal solution $\boldsymbol{\pi}^1$ of cost $LB1$ to the dual of RF1, called DRF1. The final DRP solution of cost $LB1$ corresponding to $\boldsymbol{\pi}^1$ is denoted by $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$, where $\boldsymbol{\upsilon}^1 = 0$, $\mathbf{w}^1 = 0$, $\mathbf{g}^1 = 0$. Procedure BP1 is based on the following theorem.

Theorem 2 *Let* $\lambda_e \in \mathbb{R}, \forall e \in E_R$, *and* $\lambda_0 \in \mathbb{R}^+$ *be penalties associated with constraints* (2) *and* (3)*, respectively. A feasible DRF1 solution* $\boldsymbol{\pi}$ *of cost* $z(DRF1(\lambda))$ *can be obtained by means of the following equations:*

$$\pi_e = q_e \min_{\substack{\ell \in \mathscr{R} \\ a_{e\ell} > 0}} \left\{ \frac{(c_\ell - \lambda(S_\ell) - \lambda_0)}{q(S_\ell)} \right\} + \lambda_e, \quad \forall e \in E_R, \tag{32}$$

$$\pi_0 = \lambda_0, \tag{33}$$

where $q(S_\ell) = \sum_{e \in S_\ell} a_{e\ell} q_e$ *and* $\lambda(S_\ell) = \sum_{e \in S_\ell} a_{e\ell} \lambda_e$.

Proof Consider any route $R_{\ell'}$, $\ell' \in \mathscr{R}$. Because $a_{e\ell'} > 0, \forall e \in S_{\ell'}$, we have

$$q_e \min_{\substack{\ell \in \mathscr{R} \\ a_{e\ell} > 0}} \left\{ \frac{(c_\ell - \lambda(S_\ell) - \lambda_0)}{q(S_\ell)} \right\} \leq q_e \frac{(c_{\ell'} - \lambda(S_{\ell'}) - \lambda_0)}{q(S_{\ell'})}, \quad \forall e \in S_{\ell'}. \tag{34}$$

Using expressions (32) and (33), from (34) we obtain

$$\sum_{e \in E_R} a_{e\ell'} \pi_e + \pi_0 \leq \sum_{e \in E_R} a_{e\ell'} q_e \frac{(c_{\ell'} - \lambda(S_{\ell'}) - \lambda_0)}{q(S_{\ell'})} + \lambda(S_{\ell'}) + \lambda_0 = c_{\ell'}. \tag{35}$$

□

Procedure BP1 uses subgradient optimization and column generation to heuristically solve $\max_{\boldsymbol{\lambda}}\{z(DRF1(\boldsymbol{\lambda}))\}$ as follows. Let $\overline{\text{RF1}}$ be a restricted problem obtained from RF1 by substituting $\mathscr{R}$ with a subset $\overline{\mathscr{R}} \subseteq \mathscr{R}$, and let $\bar{\boldsymbol{\pi}}$ be a solution to the dual $\overline{\text{DRF1}}$ of problem $\overline{\text{RF1}}$. We denote by $\bar{\phi}(a)$ the cost of the least cost GVRP q-route ending at $\nu(a)$ with respect to the modified arc costs $\bar{d}_{uv}$. The value $\bar{\phi}(a)$ corresponds to the reduced cost of the least reduced cost CARP q-route with respect to $\bar{\boldsymbol{\pi}}$. Cost $\bar{\phi}(a)$ is computed as $\bar{\phi}(a) = \min_{\bar{q}=q_{\nu(a)},\ldots,Q}\{f(\bar{q}, \nu(a)) + \bar{d}_{\nu(a)0}\}$, where functions $f(q, v)$ are calculated using the modified arc costs $\bar{d}_{uv}$ obtained according to Eq. (28) by setting $\bar{\boldsymbol{\upsilon}} = 0$ and $\bar{\mathbf{w}} = 0$

BP1 initializes $LB1 = 0, \boldsymbol{\lambda} = 0, \overline{\mathscr{R}} = \emptyset$, and executes a fixed number $maxit_1$ of macro-iterations which perform the following steps:

1. execute a fixed number $maxit_2$ of subgradient iterations using Theorem 2 (where $\mathscr{R}$ is replaced by $\overline{\mathscr{R}}$) to compute a dual solution $\bar{\boldsymbol{\pi}}$ of cost $\bar{z}$ to $\overline{\text{RF1}}$, and modifying the penalties $\boldsymbol{\lambda}$ using the subgradient method;
2. generate a subset $\mathscr{N}$ containing the CARP q-route of minimum reduced cost $\bar{\phi}_a$ ending at arc $a, \forall a \in A_R$. If $\mathscr{N} = \emptyset$ and $\bar{z}$ is greater than $LB1$, then update $LB1 = \bar{z}$, and $\boldsymbol{\pi}^1 = \bar{\boldsymbol{\pi}}$. Otherwise, update $\mathscr{R} = \mathscr{R} \cup \mathscr{N}$.

At Step 1, a subgradient $\bar{\boldsymbol{\psi}}$ on $z(\overline{DRF1}(\boldsymbol{\lambda}))$ can be computed as follows. Let $\ell_e \in \overline{\mathscr{R}}$, $e \in E_R$, be the index of a route R_{ℓ_e} yielding the minimum in (32), and define a not necessarily feasible LRF1 solution $\bar{\mathbf{x}}$ setting $\bar{x}_\ell = \sum_{e \in E_R} a_{e\ell}(q_e/q(S_{\ell_e})), \forall \ell \in \mathscr{R}$. A valid subgradient on $z(\overline{DRF1}(\boldsymbol{\lambda}))$ is given by setting $\bar{\psi}_e = \sum_{\ell \in \overline{\mathscr{R}}} a_{e\ell} \bar{x}_\ell - 1$, and $\bar{\psi}_0 = \sum_{\ell \in \overline{\mathscr{R}}} \bar{x}_\ell - K^*$.

7.2 Lower bounding procedure BP2

Procedure BP2 solves a relaxation of LRP, denoted by RF2, not containing constraints (25) and in which the set $\mathscr{R}$ of CARP routes is enlarged to contain all CARP q-routes.

RF2 is solved by a lower bounding procedure called BP2 which is based on LPCG (see Appendix C). The initial master problem $\overline{\text{RF2}}$ is obtained from RF2 by substituting $\mathscr{R}$ with a subset $\overline{\mathscr{R}} \subseteq \mathscr{R}$ containing a CARP q-route ending in arc a, for each arc $a \in A_R$, having minimum reduced cost with respect to the dual solution $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$ computed by BP1. The subsets $\overline{\mathscr{S}}$, $\overline{\mathscr{F}}$, and $\overline{\mathscr{C}}$ of the cuts in the initial master problem are initialized as $\overline{\mathscr{S}} = \{\{i\} : i \in V, |\delta_R(i)| \text{ is odd}\}$, and $\overline{\mathscr{F}} = \emptyset, \overline{\mathscr{C}} = \emptyset$. Moreover, the pools $\mathfrak{R}$ and $\mathfrak{B}$ of inactive routes and cuts used by LPCG are initialized as $\mathfrak{R} = \emptyset$, and $\mathfrak{B} = \emptyset$. At each iteration of LPCG, the route subset $\mathscr{N}$ is obtained by computing a CARP q-route of minimum reduced cost $\bar{\phi}_a$ ending at arc $a, \forall a \in A_R$, with respect to the current master dual solution. Values $\bar{\phi}_a, \forall a \in A_R$, are computed as $\bar{\phi}(a) = \min_{\bar{q}=q_{\nu(a)},\ldots,Q}\{f(\bar{q}, \nu(a)) + \bar{d}_{\nu(a)0}\}$, where functions $f(q, v)$ are calculated using the modified arc costs $\bar{d}_{uv}$ computed according to (28).

In the following, we denote by $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$, $\mathbf{g}^2 = 0$, the final DRP solution of cost $LB2$ computed by procedure BP2.

7.3 Lower bounding procedure BP3

Procedure BP3 solves a relaxation RF3 of problem LRP obtained by ignoring constraints (25). RF3 is solved by a lower bounding procedure called BP3 which is based on LPCG, and applies algorithm GENROUTES to generate CARP routes.

Let $M(\mathscr{R}^2)$ and $M(\mathscr{P}_v^2)$ be two parameters defined a priori. Before starting LPCG, BP3 applies GENROUTES to generate a subset $\mathscr{R}^2 \subseteq \mathscr{R}$ containing at most $M(\mathscr{R}^2)$ CARP routes having reduced cost with respect to $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$ less than or equal to the gap $z_{UB} - LB2$. Let $\mathscr{P}^2$ be the set of all paths generated by GENROUTES to compute $\mathscr{R}^2$. When generating $\mathscr{R}^2$, we impose that $|\mathscr{P}_v^2| \leq M(\mathscr{P}_v^2), \forall v \in \widetilde{V}$, and both fathoming 3 and fathoming 4 (see Appendix B) are applied within GENROUTES using DRP solutions $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$ and $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$. The path set $\mathscr{P}^2$ is called *optimal* if $|\mathscr{P}_v^2| < M(\mathscr{P}_v^2), \forall v \in \widetilde{V}$. Note that if $\mathscr{P}^2$ is optimal, it is not required to use GENROUTES within BP3, as all routes that can be part of an optimal solution can be generated by combining paths in the set $\mathscr{P}^2$.

The initial master problem of LPCG is obtained setting $\overline{\mathscr{R}} = \mathscr{R}^2$, and $\overline{\mathscr{S}} = \emptyset$, $\overline{\mathscr{F}} = \emptyset$, and $\overline{\mathscr{C}} = \emptyset$. The route pool $\mathfrak{R}$ is initialized as $\mathfrak{R} = \emptyset$, and the cut pool $\mathfrak{B}$ initially contains all inequalities found by procedure BP2. Let $M(\mathscr{N})$ be a parameter of LPCG defined a priori. At each iteration of LPCG, a subset $\mathscr{N} \subseteq \mathscr{R}$ containing at most $M(\mathscr{N})$ routes of negative reduced cost is generated according to the following two cases:

1. $\mathscr{P}^2 \neq \emptyset$:
   - (i) Extract from $\mathscr{P}^2$ a subset $\overline{\mathscr{P}}_v \subseteq \mathscr{P}_v^2$ containing the $M(\mathscr{P}_v)$ ($M(\mathscr{P}_v)$ defined a priori) paths of minimum modified cost $\bar{d}(P)$ ending at node $v, \forall v \in \widetilde{V}$, and such that $\overline{LB}^{ng}(P) \leq 0, \forall P \in \mathscr{P}_v, v \in \widetilde{V}$.
   - (ii) Use procedure COMBINE$\mathscr{P}$ to combine paths in $\overline{\mathscr{P}}_v, \forall v \in \widetilde{V}$, setting $\Delta = M(\mathscr{N})$ and $\varrho = 0$.
   - (iii) If $|\mathscr{N}| = 0$ we have to consider two cases:
     (a) $\mathscr{P}^2$ is not optimal: set $\mathscr{P}^2 = \emptyset$, and go to case 2.

(b) $\mathscr{P}^2$ is optimal and $|\overline{\mathscr{P}}_v| < |\mathscr{P}_v^2|$ for some $v \in \widetilde{V}$: set $M(\mathscr{P}_v) = \infty$ and return to (i).

2. $\mathscr{P}^2 = \emptyset$:
   - (i) Use GENROUTES to compute route subset $\mathscr{N}$ by setting $\Delta = M(\mathscr{N})$ and $\varrho = 0$, and imposing $|\mathscr{P}_v| \leq M(\mathscr{P}_v), \forall v \in \widetilde{V}$, within GEN$\mathscr{P}$.
   - (ii) If $|\mathscr{N}| = 0$ and $|\mathscr{P}_v| = M(\mathscr{P}_v)$ for some $v \in \widetilde{V}$, then set $M(\mathscr{P}_v) = \infty$ and repeat case 2.

In both cases 1 and 2, whenever $|\mathscr{N}| = \emptyset$ or GENROUTES runs out of memory procedure BP3 terminates prematurely.

When generating $\mathscr{N}$, fathoming 3 and fathoming 4 are applied within GENROUTES using both DRP solutions $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$ and $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$. Moreover, any route R such that $c(R) \geq c(R_\ell)$, for some $\ell \in \overline{\mathscr{R}} \cup \mathfrak{R}$, and such that $S(R) = S(R_\ell)$ is removed from $\mathscr{N}$. In the following, we denote by $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^3, \mathbf{g}^3)$, $\mathbf{g}^3 = 0$, the final DRP solution of cost $LB3$ computed by procedure BP3, and by $\overline{\mathscr{R}}^3$, $\overline{\mathscr{S}}^3$ and $\overline{\mathscr{F}}^3$ the subsets of routes and inequalities (8) and (14), respectively, of the final master $\overline{\text{RF3}}$ at the end of BP3.

7.4 Lower bounding procedure BP4

Procedure BP4 is similar to BP3 except that it incorporates constraints (25). It is executed after BP3 and starts by generating a subset $\mathscr{R}^3 \subseteq \mathscr{R}$ containing at most $M(\mathscr{R}^3)$ CARP routes having reduced cost with respect to $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^3, \mathbf{g}^3)$ less than or equal to the gap $z_{UB} - LB3$, where $M(\mathscr{R}^3)$ is defined a priori. Fathoming 3 and fathoming 4 are applied within GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$ using DRP solutions $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$, $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$, and $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^3, \mathbf{g}^3)$. When generating $\mathscr{R}^3$, we impose that $|\mathscr{P}^3| \leq M(\mathscr{P}^3)$, where $\mathscr{P}^3$ is the set of all paths that are generated by GENROUTES to compute route set $\mathscr{R}^3$, and $M(\mathscr{P}^3)$ is a parameter defined a priori. Path set $\mathscr{P}^3$ is called *optimal* if $|\mathscr{P}^3| < M(\mathscr{P}^3)$.

BP4 uses the route set $\mathscr{R}^3$ to update the pool $\mathfrak{R}$ as $\mathfrak{R} = \mathfrak{R} \cup \mathscr{R}^3$, whereas the master problem is initialized by setting $\overline{\mathscr{R}} = \overline{\mathscr{R}}^3$, and $\overline{\mathscr{S}} = \overline{\mathscr{S}}^3$, $\overline{\mathscr{F}} = \overline{\mathscr{F}}^3$. At each iteration of LPCG, the subset $\mathscr{N} \subseteq \mathscr{R}$ is generated according to the same strategy as in procedure BP3 where $\mathscr{P}^2$ is replaced with $\mathscr{P}^3$, and fathoming rules 3 and 4 are applied within GENROUTES using DRP solutions $(\boldsymbol{\pi}^1, \boldsymbol{\upsilon}^1, \mathbf{w}^1, \mathbf{g}^1)$, $(\boldsymbol{\pi}^2, \boldsymbol{\upsilon}^2, \mathbf{w}^2, \mathbf{g}^2)$, and $(\boldsymbol{\pi}^3, \boldsymbol{\upsilon}^3, \mathbf{w}^4, \mathbf{g}^3)$.

8 Computational experiments

All algorithms were coded in C and compiled under Visual Studio 2008. CPLEX 12.1 [18] was used as the LP solver in procedures BP2, BP3, BP4, and as the IP solver in the exact algorithm. All computational experiments were run on an Intel Xeon E5310 Workstation clocked at 1.6 GHz with 8 Gb RAM running Windows Server 2003 x64 Edition.

8.1 Test instances

We have used five sets of CARP benchmark instances called *egl* [9], *val* [10], *gdb* [25], *kshs* [30], and *bmcv* [12]. The sets *val*, *gdb*, and *kshs* correspond to Capacitated Chinese Postman Problems in that all edges of the corresponding graphs are required. Data set *egl* is based on data from a winter gritting application in Lancashire, England [32,20] and contains 24 instances based on two networks created by changing the set of required edges and the capacities of the vehicles. These instances involve up to 140 nodes and 190 edges, and the demand quantities for the required edges are proportional to their costs. Data set *val* contains 34 instances with up to 50 vertices and 97 edges obtained from 10 randomly generated graphs by changing the capacity of the vehicles. Data sets *gdb* and *kshs* contain 23 randomly generated instances with up to 27 nodes and 55 edges, and six instances with up to 10 nodes and 15 edges, respectively. Finally, the data set *bmcv* is made up of 100 instances with up to 97 vertices and 142 edges obtained by partitioning the inter-city road network of Flanders, Belgium, into districts in the context of winter gritting. The latter instances are further partitioned into four classes, called C, D, E and F. Instances D and F are defined over the same graphs as instances C and E but double the vehicle capacity. Data sets *egl*, *val*, *gdb*, and *kshs* are publicly available at http://www.uv.es/~belengue/carp.html, whereas instances *bmcv* were kindly provided to us by [11].

8.2 Computational results

In Tables 1, 2, 3, 4, 5 we compare the results obtained by the four lower bounding procedures BP1, BP2, BP3, and BP4, and by our exact algorithm (columns headed "Our algorithm") with those obtained by the methods of [5] (columns "BM"), [33] (columns "LPU"), [9] (columns "BB"), and [12] (columns "BMCV").

The algorithm of Baldacci and Maniezzo works by transforming the CARP into an equivalent CVRP which is solved by branch-and-cut using a two-index formulation strengthened by the generation of several classes of valid inequalities for the CVRP. The algorithm of Longo et al. is also based on a transformation of the CARP into a CVRP. However, these authors model this CVRP as a set covering-like problem which is solved by branch-and-cut-and-price. This formulation is strengthened using rounded capacity cuts, framed capacity, and strengthened comb cuts. The algorithm of Belenguer and Benavent is a cutting plane method based on a sparse formulation described by integer variables representing the number of times an edge is deadheaded. This formulation contains capacity constraints, odd edge cutset constraints, and disjoint path inequalities. Finally, the lower bounds computed by Beullens et al. are obtained by first applying the cutting plane algorithm of Belenguer and Benavent, but ignoring disjoint path inequalities, and then using the CPLEX IP solver to obtain an integer solution.

In all instances, edge costs are represented as integers, and therefore the final lower bounds obtained by procedures BP1, BP2, BP3, and BP4 are rounded up to the nearest integer. Tables 1, 2, 3, 4, 5 display the following data for each instance. Column "$Ins.$" reports the instance name whereas column "z_{UB}" reports the best known upper bound

Table 1 Lower bounds on *gdb* and *kshs* instances

Ins.	z_{UB}	Our algorithm										LPU
		$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3$	t_3	$\%LB4$	t_4	$\lvert\mathscr{R}^*\rvert$	t^*	t^{*1}
gdb1	316	91.14	0.7	100.00	0.9	100.00	0.9	100.00	0.9	–	0.9	3.1
gdb2	339	93.81	0.1	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	1.4
gdb3	275	92.36	0.1	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	0.8
gdb4	287	93.73	0.1	100.00	0.1	100.00	0.1	100.00	0.1	–	0.1	0.8
gdb5	377	96.55	0.2	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	1.0
gdb6	298	95.97	0.1	100.00	0.1	100.00	0.1	100.00	0.1	–	0.1	0.3
gdb7	325	90.15	0.1	100.00	0.3	100.00	0.3	100.00	0.3	–	0.3	1.8
gdb8	348	94.83	0.3	100.00	1.2	100.00	1.2	100.00	1.2	–	1.2	32.5
gdb9	303	97.03	0.5	100.00	1.2	100.00	1.2	100.00	1.2	–	1.2	31.0
gdb10	275	93.09	0.1	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	6.3
gdb11	395	93.16	0.6	100.00	2.6	100.00	2.6	100.00	2.6	–	2.6	1364.6
gdb12	458	96.51	0.1	99.13	0.1	99.13	0.2	99.78	0.2	65	0.2	5.2
gdb13	536	98.13	0.2	99.63	0.3	99.63	0.6	99.63	0.7	3642	2.6	93.0
gdb14	100	100.00	0.1	100.00	0.1	100.00	0.1	100.00	0.1	–	0.1	0.2
gdb15	58	100.00	0.2	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	3.8
gdb16	127	96.06	0.3	100.00	0.4	100.00	0.4	100.00	0.4	–	0.4	26.4
gdb17	91	95.60	0.2	100.00	0.3	100.00	0.3	100.00	0.3	–	0.3	34.8
gdb18	164	100.00	0.5	100.00	0.5	100.00	0.5	100.00	0.5	–	0.5	56.4
gdb19	55	100.00	0.0	100.00	0.0	100.00	0.0	100.00	0.0	–	0.0	0.1
gdb20	121	94.21	0.2	100.00	0.3	100.00	0.3	100.00	0.3	–	0.3	4.5
gdb21	156	97.44	0.3	100.00	0.4	100.00	0.4	100.00	0.4	–	0.4	20.9
gdb22	200	98.50	0.4	100.00	0.6	100.00	0.6	100.00	0.6	–	0.6	112.1
gdb23	233	100.00	0.7	100.00	0.7	100.00	0.7	100.00	0.7	–	0.7	145.5
Average		96.01	0.3	99.95	0.5	99.95	0.5	99.97	0.5		0.6	84.6
kshs1	14661	94.64	0.6	100.00	0.6	100.00	0.6	100.00	0.6	–	0.6	0.8
kshs2	9863	90.53	0.3	100.00	0.4	100.00	0.4	100.00	0.4	–	0.4	0.5
kshs3	9320	91.18	0.1	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	1.0
kshs4	11498	97.04	0.0	99.89	0.1	100.00	0.1	100.00	0.1	–	0.1	1.0
kshs5	10957	94.53	0.2	100.00	0.3	100.00	0.3	100.00	0.3	–	0.3	0.7
kshs6	10197	91.64	0.2	100.00	0.3	100.00	0.3	100.00	0.3	–	0.3	1.5
Average		93.26	0.2	99.98	0.3	100.00	0.3	100.00	0.3		0.3	0.9

[1] Total computing time spent by the branch-and-cut-and-price of Longo et al. for solving the problem to optimality by transforming it into a CVRP

(either taken from [39], or computed by our exact algorithm; bold values indicate an improvement over the best known upper bound). Column "z_{LB}" reports the best lower bound found by our algorithm. Columns "$\%LBx$", $x = 1, \dots, 4$, report the percentage ratio of lower bounds LBx computed by procedures BPx (i.e., $\%LBx = LBx/z_{UB} \cdot 100$); columns "$t_x$", report the total (cumulative) time in seconds spent

Table 2 Lower bounds on *bmcv* instances, classes C and E

$Ins.$	z_{UB}	z_{LB}	Our algorithm												BMCV $\%LB^2$
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$		t_3	$\%LB4^1$		t_4	$\|\mathscr{R}^*\|$	t^*	
C01	4150	4105	93.52	3.4	98.92	42.6	98.92	M	2571.3	98.92		2571.3	–	–	98.31
C02	3135	3135	95.18	1.0	100.00	5.0	100.00		5.0	100.00		5.0	–	5.0	100.00
C03	2575	2567	94.37	2.0	99.57	16.3	99.57		219.9	99.69		509.0	55344	547.7	98.06
C04	3510	3478	93.73	2.7	99.09	30.8	99.09	M	1999.1	99.09		1999.1	–	–	98.43
C05	5365	5365	95.56	2.0	99.48	20.2	99.48		1834.1	100.00		2461.7	–	2461.7	98.88
C06	2535	2532	93.45	1.0	99.57	15.2	99.57		248.6	99.88		652.8	211326	1277.8	98.42
C07	4075	4063	93.33	0.9	98.67	14.4	98.80		372.8	99.71		814.9	9365	827.1	98.53
C08	4090	4083	94.28	1.6	99.44	21.5	99.44		59.0	99.83		90.6	5415	94.6	97.80
C09	5260	5233	95.25	4.0	99.49	73.9	99.49	M	3270.9	99.49		3270.9	–	–	99.14
C10	4700	4660	93.91	0.8	98.53	5.4	98.66		47.0	99.15		103.8	33557	127.8	98.30
C11	4635	4583	95.19	5.1	98.79	56.0	98.88		1472.8	98.88	M	9795.7	–	–	98.17
C12	4240	4209	94.29	1.8	98.70	42.2	98.73		1410.4	99.27		6386.2	M	–	97.64
C13	2955	2940	94.65	1.1	98.71	9.6	98.82		304.1	99.49		2301.2	372209	4200.2	97.97
C14	4030	4030	96.15	1.2	99.55	16.2	99.73		214.6	100.00		380.9	–	380.9	98.51
C15	4940	4912	95.91	8.6	99.43	101.3	99.43	M	5829.1	99.43		5829.1	–	–	98.08
C16	1475	1475	93.29	0.8	99.93	6.4	99.93		20.5	100.00		24.2	–	24.2	99.66
C17	3555	3555	93.50	0.7	100.00	3.4	100.00		3.4	100.00		3.4	–	3.4	99.44
C18	5620	5577	95.46	9.5	99.23	133.6	99.23	M	6637.5	99.23		6637.5	–	–	98.75
C19	3115	3096	93.90	2.7	99.39	23.0	99.39	M	1341.4	99.39		1341.4	–	–	98.39
C20	2120	2120	93.07	1.4	100.00	12.1	100.00		12.1	100.00		12.1	–	12.1	100.00
C21	3970	3960	95.29	3.3	99.75	59.1	99.75	M	3084.1	99.75		3084.1	–	–	99.50
C22	2245	2245	95.59	1.2	100.00	5.0	100.00		5.0	100.00		5.0	–	5.0	100.00

Table 2 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm											BMCV $\%LB^2$
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$		t_3	$\%LB4^1$	t_4	$\|\mathscr{R}^*\|$	t^*	
C23	4085	4032	91.55	5.7	98.70	59.3	98.70	M	6057.7	98.70	6057.7	–	–	98.29
C24	3400	3384	96.32	5.6	99.53	46.4	99.53	M	2971.3	99.53	2971.3	–	–	99.12
C25	2310	2310	95.02	0.7	100.00	2.1	100.00		2.1	100.00	2.1	–	2.1	100.00
Average			94.47	2.7	99.38	32.8	99.40		1599.7	99.58	2292.4			98.78
Solved													14	4
E01	4910	4885	95.97	2.2	99.12	30.3	99.14		2467.8	99.49	13009.0	M	–	98.37
E02	3990	3978	96.67	1.4	99.27	9.3	99.57		74.3	99.70	185.1	53650	251.6	99.25
E03	2015	2015	92.31	1.1	100.00	8.5	100.00		8.5	100.00	8.5	–	8.5	100.00
E04	4155	4154	95.96	3.7	99.78	40.9	99.81		1010.8	99.98	2046.4	14445	2062.4	99.28
E05	4585	4585	96.10	1.6	99.76	15.6	99.93		49.4	100.00	61.2	–	61.2	99.35
E06	2055	2055	96.69	0.8	100.00	4.3	100.00		4.3	100.00	4.3	–	4.3	100.00
E07	4155	4133	95.84	0.8	98.92	6.4	99.09		94.2	99.47	291.1	26687	319.0	97.11
E08	4710	4702	94.82	1.3	99.55	18.3	99.66		104.8	99.83	165.6	40983	200.2	98.51
E09	5820	5780	95.57	5.7	99.31	53.2	99.31	M	2428.5	99.31	2428.5	–	–	98.71
E10	3605	3605	97.53	0.8	100.00	3.7	100.00		3.7	100.00	3.7	–	3.7	100.00
E11	4655	4637	94.44	4.6	99.61	49.4	99.61	M	8881.2	99.61	8881.2	–	–	99.46
E12	**4180**	4161	96.34	1.5	98.64	20.3	99.02		1005.1	99.55	4761.4	611460	6585.4	97.25
E13	3345	3337	95.87	1.0	99.07	9.7	99.16		312.5	99.76	1217.6	76847	1362.1	99.25
E14	4115	4115	94.63	1.4	99.95	10.5	100.00		67.1	100.00	67.1	–	67.1	99.27
E15	4205	4189	92.70	8.9	99.62	134.3	99.62	M	6743.7	99.62	6743.7	–	–	99.17
E16	3775	3755	96.58	2.4	99.47	20.3	99.47	M	1787.4	99.47	1787.4	–	–	98.94
E17	2740	2740	92.01	0.7	100.00	2.7	100.00		2.7	100.00	2.7	–	2.7	100.00

Table 2 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm												BMCV
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$		t_3	$\%LB4^1$		t_4	$\|\mathscr{R}^*\|$	t^*	$\%LB^2$
E18	3835	3825	94.00	5.8	99.74	104.8	99.74	M	5999.8	99.74		5999.8	–	–	99.74
E19	3235	3222	94.19	2.8	99.60	37.6	99.60	M	2270.5	99.60		2270.5	–	–	98.92
E20	2825	2802	93.81	1.8	99.01	27.0	99.19		2616.3	99.19	M	10029.8	–	–	98.58
E21	3730	3728	94.66	3.6	99.95	26.3	99.95	M	2348.3	99.95		2348.3	–	–	99.87
E22	2470	2470	94.45	2.2	99.84	10.2	100.00		149.4	100.00		149.4	–	149.4	98.79
E23	3710	3686	91.29	4.5	99.35	92.4	99.35	M	6350.0	99.35		6350.0	–	–	99.06
E24	4020	4001	93.63	5.9	99.53	48.0	99.53	M	2682.9	99.53		2682.9	–	–	97.76
E25	1615	1615	95.73	0.4	100.00	1.2	100.00		1.2	100.00		1.2	–	1.2	100.00
Average			94.87	2.7	99.56	31.4	99.63		1898.6	99.73		2859.9			99.07
Solved														14	5

[1] M: Algorithm GENROUTES runs out of memory
[2] Lower bound obtained by [12] by optimally solving a relaxation of the problem

Table 3 Lower bounds on *bmcv* instances, classes D and F

$Ins.$	z_{UB}	z_{LB}	Our algorithm											BMCV $\%LB^2$
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$		t_3	$\%LB4^1$	t_4	$\|\mathscr{R}^*\|$	t^*	
D01	3215	3215	92.07	6.4	100.00	83.9	100.00		83.9	100.00	83.9	–	83.9	100.00
D02	2520	2520	93.57	2.4	100.00	18.8	100.00		18.8	100.00	18.8	–	18.8	100.00
D03	2065	2065	92.74	2.0	100.00	49.4	100.00		49.4	100.00	49.4	–	49.4	100.00
D04	2785	2785	89.48	5.8	100.00	70.0	100.00		70.0	100.00	70.0	–	70.0	100.00
D05	3935	3935	91.69	3.8	100.00	21.9	100.00		21.9	100.00	21.9	–	21.9	100.00
D06	2125	2125	89.18	2.8	100.00	15.8	100.00		15.8	100.00	15.8	–	15.8	100.00
D07	3115	3078	90.69	1.6	97.66	26.3	98.07		911.9	98.81	4478.4	418361	10446.1	96.79
D08	3045	2995	91.07	1.7	98.36	67.2	98.36	M	6714.5	98.36	6714.5	–	–	97.70
D09	4120	4120	93.11	12.4	100.00	102.8	100.00		102.8	100.00	102.8	–	102.8	100.00
D10	3340	3335	96.14	1.9	99.79	17.7	99.79		76.6	99.85	104.6	5789	107.2	99.70
D11	3745	3745	90.33	10.9	100.00	122.5	100.00		122.5	100.00	122.5	–	122.5	100.00
D12	3310	3310	89.52	4.1	100.00	77.7	100.00		77.7	100.00	77.7	–	77.7	100.00
D13	2535	2535	88.36	2.1	100.00	18.0	100.00		18.0	100.00	18.0	–	18.0	100.00
D14	3280	3272	90.00	2.8	99.76	49.0	99.76	M	5261.1	99.76	5261.1	–	–	99.70
D15	3990	3990	90.60	17.2	100.00	336.3	100.00		336.3	100.00	336.3	–	336.3	100.00
D16	1060	1060	98.11	0.8	100.00	8.6	100.00		8.6	100.00	8.6	–	8.6	100.00
D17	2620	2620	90.46	0.7	100.00	8.6	100.00		8.6	100.00	8.6	–	8.6	100.00
D18	4165	4165	92.70	26.2	100.00	454.9	100.00		454.9	100.00	454.9	–	454.9	100.00
D19	2400	2393	93.00	3.8	99.71	115.6	99.71	M	3413.1	99.71	3413.1	–	–	98.75
D20	1870	1870	91.18	3.3	100.00	27.4	100.00		27.4	100.00	27.4	–	27.4	100.00
D21	3050	2985	91.61	8.5	97.87	156.2	97.87	M	5292.3	97.87	5292.3	–	–	96.39
D22	1865	1865	92.12	3.0	100.00	19.7	100.00		19.7	100.00	19.7	–	19.7	100.00

Table 3 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm											BMCV $\%LB^2$
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$		t_3	$\%LB4^1$	t_4	$\|\mathscr{R}^*\|$	t^*	
D23	3130	3114	91.57	12.5	99.49	976.4	99.49	M	6897.1	99.49	6897.1	–	–	99.36
D24	2710	2676	91.77	14.2	98.75	304.5	98.75	M	7105.6	98.75	7105.6	–	–	98.15
D25	1815	1815	91.96	0.9	100.00	11.1	100.00		11.1	100.00	11.1	–	11.1	100.00
Average			91.72	6.1	99.65	126.4	99.67		1484.8	99.70	1628.6			99.46
Solved													19	17
F01	4040	4040	87.97	8.5	100.00	88.1	100.00		88.1	100.00	88.1	–	88.1	100.00
F02	3300	3300	94.00	2.9	100.00	18.9	100.00		18.9	100.00	18.9	–	18.9	100.00
F03	1665	1665	92.31	2.6	100.00	23.4	100.00		23.4	100.00	23.4	–	23.4	100.00
F04	3485	3476	89.07	7.4	99.74	126.3	99.74	M	4109.4	99.74	4109.4	–	–	99.71
F05	3605	3605	90.85	4.1	100.00	27.5	100.00		27.5	100.00	27.5	–	27.5	100.00
F06	1875	1875	90.72	2.1	100.00	12.1	100.00		12.1	100.00	12.1	–	12.1	100.00
F07	3335	3335	92.08	1.8	100.00	11.2	100.00		11.2	100.00	11.2	–	11.2	100.00
F08	3705	3690	89.26	2.3	99.60	50.3	99.60	M	3385.6	99.60	3385.6	–	–	99.73
F09	4730	4730	90.97	15.8	100.00	136.6	100.00		136.6	100.00	136.6	–	136.6	100.00
F10	2925	2925	92.99	2.0	100.00	8.4	100.00		8.4	100.00	8.4	–	8.4	100.00
F11	3835	3835	88.89	13.7	100.00	133.5	100.00		133.5	100.00	133.5	–	133.6	100.00
F12	3395	3390	91.22	4.3	99.85	77.1	99.85	M	3125.7	99.85	3125.7	–	–	99.71
F13	2855	2855	91.24	2.0	100.00	14.3	100.00		14.3	100.00	14.3	–	14.3	100.00
F14	3330	3330	90.15	3.0	100.00	26.3	100.00		26.3	100.00	26.3	–	26.3	100.00
F15	3560	3560	87.05	25.1	100.00	279.3	100.00		279.3	100.00	279.3	–	279.3	100.00
F16	2725	2725	96.70	4.8	100.00	27.2	100.00		27.2	100.00	27.2	–	27.2	100.00
F17	2055	2055	94.31	0.7	100.00	6.0	100.00		6.0	100.00	6.0	–	6.0	100.00

Table 3 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm											BMCV
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3$[1]		t_3	$\%LB4$[1]	t_4	$\|\mathscr{R}^*\|$	t^*	$\%LB$[2]
F18	3075	3063	91.71	14.0	99.61	318.5	99.61	M	9900.6	99.61	9900.6	–	–	99.51
F19	2525	2500	91.56	6.6	99.01	134.9	99.01	M	5814.4	99.01	5814.4	–	–	98.42
F20	2445	2445	90.47	5.1	100.00	42.1	100.00		42.1	100.00	42.1	–	42.1	100.00
F21	2930	2930	92.35	9.2	100.00	70.4	100.00		70.4	100.00	70.4	–	70.4	100.00
F22	2075	2075	88.63	2.8	100.00	18.1	100.00		18.1	100.00	18.1	–	18.1	100.00
F23	3005	2994	90.48	9.5	99.63	340.4	99.63	M	9260.5	99.63	9260.5	–	–	99.33
F24	3210	3210	90.69	15.2	100.00	146.2	100.00		146.2	100.00	146.2	–	146.2	100.00
F25	1390	1390	92.01	0.8	100.00	2.1	100.00		2.1	100.00	2.1	–	2.1	100.00
Average			91.11	6.6	99.90	85.6	99.90		1467.5	99.90	1467.5			99.86
Solved													19	19

[1] M: Algorithm GENROUTES runs out of memory
[2] Lower bound obtained by Beullens et al. by optimally solving a relaxation of the problem

Table 4 Lower bounds on *egl* instances

$Ins.$	z_{UB}	z_{LB}	Our algorithm										LPU			BM			BB	
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$	t_3	$\%LB4^1$	t_4	$\|\mathscr{R}^*\|^1$	t^*	$\%LB^2$	t	t^*	$\%LB^3$	t	t^*	$\%LB^4$	t
e1-A	3548	3548	95.69	3.9	100.00	15.9	100.00	15.9	100.00	15.9	–	15.9	100.00	144.2	144.2	100.00	12.7	12.7	99.07	32.0
e1-B	4498	4487	94.40	3.5	99.33	16.1	99.49	736.4	99.76	2527.8	43699	2619.5	99.33	52.6	–	99.24	23.7	450.3	98.62	35.8
e1-C	5595	5580	96.94	1.6	99.11	9.6	99.34	332.6	99.73	714.7	10465	732.7	99.05	46.1	–	98.89	38.5	–	97.46	22.4
e2-A	5018	5012	93.80	10.9	99.88	104.6	99.88	M 1485.1	99.88	1485.1	–	–	99.86	521.2	–	99.86	159.1	182.0	99.52	50.5
e2-B	6317	6284	94.63	5.4	99.48	36.9	99.48	M 2368.4	99.48	2368.4	–	–	99.41	198.5	–	99.27	121.0	–	98.92	19.1
e2-C	8335	8319	97.37	3.5	99.08	14.1	99.47	564.9	99.81	3934.9	93018	4204.1	98.79	66.9	–	97.96	153.2	–	97.35	18.8
e3-A	5898	5898	95.07	15.9	100.00	72.6	100.00	72.6	100.00	72.6	–	72.6	100.00	924.9	924.9	100.00	143.5	143.5	99.51	86.8
e3-B	7775	7711	96.15	9.5	99.01	59.8	99.18	1078.7	99.18	M 8894.6	–	–	99.00	375.6	–	98.88	350.2	–	98.34	58.3
e3-C	10292	10244	96.75	5.3	98.98	22.7	99.13	658.0	99.53	2345.4	M	–	98.75	142.1	–	98.45	594.4	–	97.35	19.9
e4-A	6444	6395	94.94	19.2	99.24	184.3	99.24	M 3856.8	99.24	3856.8	–	–	99.24	1171.6	–	99.18	391.2	–	98.88	43.1
e4-B	**8961**	8935	95.50	9.4	99.26	46.8	99.43	830.1	99.71	8455.4	M	–	99.14	419.0	–	99.04	698.2	–	98.30	15.9
e4-C	**11562**	11493	97.31	7.4	99.05	28.1	99.21	644.1	99.40	2060.6	M	–	98.83	202.7	–	98.37	682.3	–	97.53	16.5
s1-A	5018	5018	95.72	7.2	99.92	84.7	99.96	794.6	100.00	1232.3	–	1232.3	99.92	750.4	–	99.95	89.7	91.8	99.48	1085.5
s1-B	6388	6388	95.13	4.4	99.87	64.3	99.94	315.4	100.00	344.1	–	344.1	99.86	204.5	–	99.85	148.2	–	97.07	216.1
s1-C	8518	8517	96.82	3.5	99.66	26.5	99.94	155.8	99.99	173.9	21158	195.8	99.55	67.0	–	99.46	457.1	–	97.56	102.7
s2-A	9884	9825	96.32	36.2	99.40	369.5	99.40	tl	99.40	tl	–	–	99.39	3260.3	–	–	–	–	98.95	521.8
s2-B	13100	13017	97.05	21.8	99.03	134.5	99.24	830.4	99.37	3777.8	M	–	98.99	896.7	–	–	–	–	98.37	170.2
s2-C	16425	16407	98.57	14.7	99.63	59.3	99.75	867.9	99.89	2468.6	602630	15082.9	99.56	408.9	–	–	–	–	98.76	274.2
s3-A	10220	10145	95.67	40.1	99.27	435.9	99.27	M 5066.1	99.27	5066.1	–	–	99.25	1680.4	–	–	–	–	98.09	206.3
s3-B	13682	13648	97.28	26.1	99.54	155.5	99.64	1536.0	99.75	10721.5	M	–	99.52	1639.5	–	–	–	–	99.06	504.3
s3-C	**17188**	17163	98.62	17.6	99.57	71.9	99.70	1261.9	99.85	4091.0	438007	13201.8	99.49	635.3	–	–	–	–	98.73	182.7

Table 4 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm										LPU			BM			BB	
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3$[1]	t_3	$\%LB4$[1]	t_4	$\lvert\mathscr{R}^*\rvert$[1]	t^*	$\%LB$[2]	t	t^*	$\%LB$[3]	t	t^*	$\%LB$[4]	t
s4-A	12268	12141	96.27	57.6	98.96	459.9	98.96	M 7506.0	98.96	7506.0	–	–	98.98	14318.1	–	–	–	–	98.04	324.7
s4-B	16321	16098	96.12	37.2	98.54	219.3	98.63	2590.8	98.63	tl	–	–	98.60	2761.1	–	–	–	–	97.62	1173.0
s4-C	**20481**	20430	98.00	24.8	99.52	125.2	99.63	655.9	99.75	3355.9	M	–	99.48	1120.0	–	–	–	–	98.53	211.0
Average[5]			96.25	16.1	99.39	117.4	99.50	2026.0	99.61	4344.5			99.33	1333.6		–	–		98.38	224.6
Average[6]			95.75	7.4	99.46	52.5	99.58	927.3	99.71	2565.5			99.38	352.5		99.23	270.9		98.33	121.6
Solved												10			2			5		0

[1] M: algorithm GENROUTES runs out of memory
[2] Lower bound obtained by Longo et al. without branching by transforming the problem into a CVRP
[3] Lower bound obtained by Baldacci and Maniezzo without branching by transforming the problem into a CVRP
[4] Lower bound obtained by Belenguer and Benavent without branching by solving a relaxation of the problem
[5] Average over all instances considered by Longo et al.
[6] Average over all instances considered by Baldacci and Maniezzo

Table 5 Lower bounds on *val* instances

$Ins.$	z_{UB}	z_{LB}	Our algorithm										LPU			BM			BB	
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3^1$	t_3	$\%LB4^1$	t_4	$\|\mathscr{R}^*\|^1$	t^*	$\%LB^2$	t	t^*	$\%LB^3$	t	t^*	$\%LB^4$	t
val1A	247	247	89.07	1.4	100.00	5.9	100.00	5.9	100.00	5.9	–	5.9	100.00	98.3	98.3	100.00	2.0	2.0	100.00	0.5
val1B	247	247	90.28	0.8	100.00	4.8	100.00	4.8	100.00	4.8	–	4.8	100.00	54.6	54.6	100.00	2.1	2.1	100.00	0.6
val1C	319	314	92.79	0.1	98.12	1.7	98.12	15.4	98.43	39.7	54992	133.8	97.81	771.9	8916.8	97.35	45.8	890.4	96.87	0.7
val2A	298	298	91.61	1.0	100.00	5.5	100.00	5.5	100.00	5.5	–	5.5	100.00	79.4	5.5	100.00	0.5	0.5	100.00	0.1
val2B	330	330	92.12	0.6	99.70	8.1	100.00	290.3	100.00	290.3	–	290.3	99.70	169.0	671.3	99.61	9.9	11.1	100.00	0.2
val2C	528	528	98.11	0.1	100.00	0.2	100.00	0.2	100.00	0.2	–	0.2	100.00	1.0	1.0	99.67	4.8	7.8	99.62	0.4
val3A	105	105	88.57	0.8	100.00	5.5	100.00	5.5	100.00	5.5	–	5.5	100.00	127.6	127.6	100.00	1.5	1.5	100.00	0.2
val3B	111	111	90.99	0.3	100.00	3.6	100.00	3.6	100.00	3.6	–	3.6	100.00	134.3	134.3	100.00	1.3	1.3	100.00	0.2
val3C	162	162	95.68	0.2	99.38	1.3	100.00	3.0	100.00	3.0	–	3.0	99.38	3.2	328.9	100.00	5.3	5.3	99.38	0.9
val4A	522	522	91.76	12.5	100.00	80.0	100.00	80.0	100.00	80.0	–	80.0	100.00	2475.3	2475.3	100.00	60.8	60.8	100.00	0.5
val4B	534	534	92.13	6.0	100.00	49.5	100.00	49.5	100.00	49.5	–	49.5	100.00	1178.4	1178.4	100.00	54.8	54.8	100.00	0.3
val4C	550	550	93.64	5.3	100.00	28.2	100.00	28.2	100.00	28.2	–	28.2	100.00	824.6	824.6	100.00	86.7	86.7	100.00	1.0
val4D	652	649	95.09	1.8	99.39	20.2	99.39	1386.6	99.54	7926.6	M	–	99.39	76.6	–	98.40	186.1	–	98.77	7.2
val5A	566	566	92.93	7.9	100.00	34.2	100.00	34.2	100.00	34.2	–	34.2	100.00	629.4	629.4	100.00	53.0	53.0	100.00	1.4
val5B	589	588	93.04	5.8	99.83	48.8	99.83	M 2737.4	99.83	2737.4	–	–	99.83	388.1	–	99.44	99.8	4202.3	100.00	2.1
val5C	617	613	93.68	4.9	99.35	37.8	99.35	912.0	99.35	1293.4	M	–	99.35	274.8	–	98.63	114.2	–	99.19	1.7
val5D	718	717	96.10	1.6	99.58	9.3	99.72	108.6	99.86	280.6	295009	702.0	99.72	62.8	–	99.10	155.0	–	99.44	1.0
val6A	330	330	91.52	2.3	100.00	11.3	100.00	11.3	100.00	11.3	–	11.3	100.00	158.7	158.7	100.00	9.8	9.8	100.00	0.6
val6B	340	337	91.47	0.9	99.12	10.8	99.12	M 1647.3	99.12	1647.3	–	–	99.12	169.3	–	98.50	24.3	201.6	99.41	2.3
val6C	424	421	95.75	0.5	98.82	2.2	98.82	17.3	99.29	26.7	29798	55.7	99.06	119.4	–	98.58	14.0	312.0	98.58	1.8
val7A	382	382	92.15	3.1	100.00	24.6	100.00	24.6	100.00	24.6	–	24.6	100.00	319.4	319.4	100.00	9.2	9.2	100.00	0.3
val7B	386	386	91.97	2.6	100.00	13.3	100.00	13.3	100.00	13.3	–	13.3	100.00	163.8	163.8	100.00	8.3	8.3	100.00	0.2
val7C	437	437	92.45	0.9	99.08	23.8	99.08	319.7	100.00	1760.0	–	1760.0	99.77	607.6	132.0	99.70	15.5	92.6	99.77	3.2

Table 5 continued

$Ins.$	z_{UB}	z_{LB}	Our algorithm										LPU			BM			BB	
			$\%LB1$	t_1	$\%LB2$	t_2	$\%LB3$[1]	t_3	$\%LB4$[1]	t_4	$\|\mathscr{R}^*\|$[1]	t^*	$\%LB$[2]	t	t^*	$\%LB$[3]	t	t^*	$\%LB$[4]	t
val8A	522	522	93.87	6.0	100.00	20.9	100.00	20.9	100.00	20.9	–	20.9	100.00	359.4	359.4	100.00	32.7	32.7	100.00	0.3
val8B	531	531	94.73	4.0	100.00	16.4	100.00	16.4	100.00	16.4	–	16.4	100.00	168.9	168.9	100.00	34.7	34.7	100.00	0.1
val8C	657	655	97.11	1.2	99.54	6.6	99.54	52.7	99.70	226.7	607971	2496.3	99.54	376.9	–	98.52	98.1	–	99.39	3.7
val9A	450	450	90.67	30.2	100.00	212.8	100.00	212.8	100.00	212.8	–	212.8	100.00	17722.6	17722.6	100.00	406.5	406.5	100.00	4.32
val9B	453	453	91.39	14.4	100.00	114.7	100.00	114.7	100.00	114.7	–	114.7	100.00	4520.03	4520.03	100.00	430.6	430.6	100.00	1.97
val9C	459	459	91.72	9.7	100.00	63.2	100.00	63.2	100.00	63.2	–	63.2	100.00	1460.89	1460.89	100.00	517.8	517.8	100.00	2.07
val9D	516	512	94.38	2.7	99.22	31.1	99.22	579.8	99.22	8237.1	M	–	99.22	305.16	–	97.21	785.9	–	98.64	21.84
val10A	637	637	92.78	31.1	100.00	214.7	100.00	214.7	100.00	214.7	–	214.7	100.00	13336.6	13336.6	99.88	394.1	879.3	100.00	1.67
val10B	645	645	92.87	17.9	100.00	156.5	100.00	156.5	100.00	156.5	–	156.5	100.00	13719.4	13719.4	99.43	327.3	4250.4	100.00	1.44
val10C	655	655	93.13	10.9	100.00	96.8	100.00	96.8	100.00	96.8	–	96.8	100.00	5078.57	5078.57	99.44	468.3	5560.1	100.00	0.83
val10D	734	734	94.82	4.0	100.00	60.8	100.00	60.8	100.00	60.8	–	60.8	100.00	473.63	473.63	99.16	684.7	–	99.73	8.38
Average			92.95	5.7	99.74	41.9	99.77	273.3	99.83	755.7			99.76	1953.2		99.49	151.3		99.67	2.2
Solved												29			26			28		22

[1] M: algorithm GENROUTES runs out of memory
[2] Lower bound obtained by Longo et al. without branching by transforming the problem into a CVRP
[3] Lower bound obtained by Baldacci and Maniezzo without branching by transforming the problem into a CVRP
[4] Lower bound obtained by Belenguer and Benavent without branching by solving a relaxation of the problem

by procedures “BPx” (e.g., t_4 is the total time spent for computing $LB1$, $LB2$, $LB3$, and $LB4$). Columns “$\%LB$” under headings LPU, BM, BB, and BMCV report the percentage ratio of the best lower bound obtained by the respective authors, and columns “t”, give the corresponding computing times in seconds (computing times for BMCV are not available). Lower bound ratios reported for Longo et al., Baldacci and Maniezzo, and Belenguer and Benavent are computed with respect to the root lower bounds, whereas those of Beullens et al. are relative to the best lower bound at termination. Columns “$|\mathscr{R}^*|$” give the cardinality of the final route set $\mathscr{R}^*$ computed by the exact method described in Section 4. Finally, column “t^*” provides the total computing time for the instances solved to optimality.

A time limit of 4 h was imposed to all lower bounding procedures described in this paper. An entry tl in column “t_x” indicates that the time limit was reached. An additional time limit of 6 h was then allowed to the execution of the exact algorithm described in Sect. 4. An entry $_M$ in columns “LBx” and in column “$|\mathscr{R}^*|$” indicates that GENROUTES ran out of memory in generating the route set $\mathscr{N}$ within procedure BPx and route set $\mathscr{R}^*$ at Step 2 of the exact algorithm, respectively. In these cases, the algorithm terminated prematurely.

All computing times reported for Longo et al., and Baldacci and Maniezzo are relative to an Intel Pentium IV clocked at 2.8 GHz, and to an Intel Pentium IV clocked at 2.4 GHz, respectively. In order to allow a fair comparison between the running times of these algorithms and ours, we have used the CPU2000 benchmarks, reported by the Standard Performance Evaluation Corporation (SPEC 2005), which are publicly available at http://www.spec.org/cpu/results/. According to these benchmarks the SPECint and SPECfp scores for the Intel Xeon E5310 used in this paper are 1,680 and 1,619, whereas the scores for the Pentium IV at 2.8 GHz of Longo et al. are 976 and 915, and those of the Pentium IV at 2.4 GHz of Baldacci and Maniezzo are 852 and 840. Therefore, we estimate that our computer is approximately 1.8 times faster than that of Longo et al., and twice faster than that of Baldacci and Maniezzo. Belenguer and Benavent used a SUN Sparc 20. We could not find benchmarks for this computer, but our machine is clearly much faster, probably by at least one order of magnitude.

The following parameter settings were used by our algorithm to obtain all results reported in Tables 1, 2, 3, 4, 5: $maxit1 = 100$, and $maxit2 = 50$ in procedure BP1; $\eta = 10$ and $\gamma = 50$ in algorithm LPCG (see Appendix C); $M(\mathscr{R}^2) = 100000$, $M(\mathscr{P}_v^2) = \lceil \frac{10000000}{|E_R|} \rceil$, in procedure BP3; $M(\mathscr{R}^3) = 50000$ and $M(\mathscr{P}^3) = 50000000$, in procedure BP4; $M(\mathscr{N}) = 500$ and $M(\mathscr{P}_v) = 50000$, in both procedures BP3 and BP4. Finally, we set $\Delta^{MAX} = 1000000$ in the exact method.

8.3 Analysis of the computational results

The results reported in Tables 1, 2, 3, 4, 5 show that the new lower bounds are tighter than the previous best ones. Lower bound $LB2$ is on average better and significantly faster than all previous best ones. An exception is data-set val where $LB2$ is on average slightly worse than the lower bound of LPU but runs approximately 20 times faster. The final lower bound $LB4$ is more time consuming, but is on average significantly tighter than previous ones on all data-sets tested. Indeed, $LB4$ is never worse than previous lower bounds, except for two instances (F8 from data-set *bmcv*, and s4-A from data-set *egl*). On average, it is always within less than 0.5% from the best known upper bounds.

Concerning the exact method, our results confirm the usefulness of the new lower bounds. A total of 27 instances could be solved for the first time by the exact algorithm: six from data-set *egl*, and 21 from data-set *bmcv*. Five new upper bounds were also found, two of which are optimal. The two *val* instances 5D and 8C are solved for the first time by our exact algorithm and by the algorithm of [14]. The latter algorithm also optimally solves instances 4D and 5C for the first time. Note that because in the instances from data-set *bmcv* edge costs are multiples of 5, the lower bound found by our algorithm for instance E21 can be raised to 3730, thus proving the optimality of the best known upper bound. At present, 13 instances from *egl*, and 33 from *bmcv* remain open. Tables 1, 4, and 5 show that our exact algorithm outperforms the branch-and-cut-and-price of Longo et al., solving 11 more instances from data-sets *egl* and *val*, and being on average faster. It is also competitive with the branch-and-cut of Baldacci and Maniezzo, solving six more instances from data-sets *egl* and *val*, even though it is on average slower on data-set *egl*. Note that the model used by Baldacci and Maniezzo assumes that the number of vehicles K is fixed.

It is interesting to note that on data-set *val* the improvement of lower bound $LB3$ over $LB2$ is rather marginal. This seems to suggest that for these instances, pricing elementary routes instead of non-elementary ones may not be worth the extra computational effort. Indeed, BP2 alone yields in this case very strong lower bounds within a rather limited computing time, suggesting that a branch-and-cut-and-price based on non-elementary routes may prove to be an effective solution method. The preliminary results reported by Bode and Irnich seem to confirm this hypothesis, although the incorporation of BP2 as a lower bounding method could provide some advantages because of the stronger cuts and q-route relaxation.

9 Conclusions

We have developed a new lower bounding methodology and an exact algorithm based on a set partitioning-like formulation of the CARP strengthened by additional valid inequalities. We have reported results of an extensive computational study over a set of CARP benchmark instances showing that the new bounding scheme yields very tight lower bounds, and significantly improves most of the best known lower bounds for the open benchmark instances. To the best of our knowledge, this study represents the first implementation and computational evaluation of a direct solution method for the CARP based on column generation, and combining elementary CARP routes and cutting planes. The effectiveness of our proposed lower bounding algorithm was assessed by embedding it into an exact algorithm which proved to be competitive with the best known exact algorithms. It was capable of solving 27 open instances for the first time.

The methodology described in this paper can be extended in different ways. As is usually the case for cut-and-column-generation methods, major improvements can come from a more efficient column generation or stronger cutting planes. The most recent studies on the classical CVRP and the VRP with time windows [4,6,23,29] have reported major improvements in the effectiveness of state-of-the-art solution methods stemming from new developments in column generation techniques, and from a better integration between pricing and cutting. However, in the case of the CARP the situation is less clear. Polyhedral and mathematical programming approaches to the CARP can still be considered in their infancy, and pure branch-and-cut approaches suffer from the lack of a practical formulation containing a polynomial number of variables. This paper demonstrates that a direct cut-and-column-generation approach to the CARP using elementary CARP routes and cutting planes derived from different relaxations constitutes a promising solution approach. However, pricing elementary CARP routes seems to be particularly challenging and, unlike what happens for the CVRP, a set partitioning-like formulation for the CARP using elementary routes, but without incorporating additional cuts, yields a rather weak lower bound (in our preliminary experiments, the average lower bound ratio was ∼93.5% for data-set *val*, and ∼97.5% for data-set *egl*). We therefore believe that improvements to the effectiveness of cut-and-column-generation methods for the CARP can come both from the development of new pricing techniques, and from the integration of new cutting planes.

Acknowledgments This work was partly funded by the Canadian Natural Sciences and Engineering Research Council under grants 227837-09 and 39682-10. This support is gratefully acknowledged. Thanks are due to the Associate Editor and the referees for their valuable comments.

Appendix A: Proofs of Lemmas 1 and 2

Lemma 1 *Let P and $\overline{P}$ be two GVRP forward paths satisfying route feasibility conditions $(A)-(C)$, and let R_ℓ, $\ell \in \widetilde{\mathscr{R}}$, be the GVRP route obtained by combining P and $\overline{P}$. The reduced cost $\bar{c}^r_\ell$ of CARP route R_ℓ, $\ell \in \mathscr{R}$, corresponding to $\widetilde{R}_\ell$, satisfies $\bar{c}^r_\ell \geq \bar{d}(P) + \bar{d}(\overline{P})$.*

Proof Because $\widetilde{R}_\ell$ corresponds to R_ℓ, we have $\mathscr{V}(P) \cup \mathscr{V}(\overline{P}) = \{\widetilde{V}_e \subset \widetilde{V} : e \in S_\ell\}$, and $\mathscr{C}(R_\ell) = \mathscr{C}^3(P) \cup \mathscr{C}^3(\overline{P}) \cup \mathscr{C}^2(P) \cup \mathscr{C}^2(\overline{P})$. From the definition of costs $\{\bar{d}_{uv}\}$, we have

$$\bar{c}^r_\ell = \sum_{(u,v)\in P} \bar{d}_{uv} - \sum_{C \in \mathscr{C}(R_\ell)} \bar{g}_C. \tag{36}$$

Consider any edge triplet $C \in \mathscr{C}(R_\ell)$, and let $\widetilde{C}$ be the corresponding cluster triplet in $\widetilde{V}$, we have the following two cases:

(i) $|\widetilde{C} \cap \mathscr{V}(P)| = 3$: because $\mathscr{V}(P) \cap \mathscr{V}(\overline{P}) = \{\widetilde{V}_0, \widetilde{V}_{\varepsilon(e(P))}\}$, we have $|\widetilde{C} \cap \mathscr{V}(\overline{P})| \leq 1$. The same holds for the symmetric case when $|\widetilde{C} \cap \mathscr{V}(\overline{P})| = 3$.

(ii) $|\widetilde{C} \cap \mathscr{V}(P)| = 2$: because $\mathscr{V}(P) \cap \mathscr{V}(\overline{P}) = \{\widetilde{V}_0, \widetilde{V}_{\varepsilon(e(P))}\}$, we have $|\widetilde{C} \cap \mathscr{V}(\overline{P})| \leq 1 + |\widetilde{C} \cap \widetilde{V}_{\varepsilon(e(P))}|$.

From (i) it follows that $\mathscr{C}^3(P) \cap \mathscr{C}^3(\overline{P}) = \emptyset$, and from (ii) we have $\mathscr{C}^2(P) \cap \mathscr{C}^2(\overline{P}) \subseteq \{C \in \mathscr{C} : \widetilde{V}_{\varepsilon(e(P))} \in \widetilde{C}\}$. Therefore, $\sum_{C\in\mathscr{C}(R_\ell)} \bar{g}_C \leq \sum_{C\in\mathscr{C}^3(P)} \bar{g}_C +$

$\sum_{\substack{C\in\mathscr{C}^2(P)\\ \varepsilon(e(P))\notin C}} \bar{g}_C + \sum_{C\in\mathscr{C}^3(\overline{P})} \bar{g}_C + \sum_{\substack{C\in\mathscr{C}^2(\overline{P})\\ \varepsilon(e(P))\notin C}} \bar{g}_C$, and because $\mathbf{g} \le 0$ from (36) we have $\bar{c}^r_\ell \ge \bar{d}(P) + \bar{d}(\overline{P})$. □

Lemma 2 *A valid lower bound $\overline{LB}^{ng}(P)$ on the reduced cost $\bar{c}^r_\ell$ with respect to a DRP solution $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ of any CARP route R_ℓ, $\ell \in \mathscr{R}$, corresponding to a GVRP route $\widetilde{R}_\ell$ containing path P can be computed as*

$$LB^{ng}(P) = \bar{d}(P) + \min_{\substack{NG\subseteq\Pi \ s.t.\ NG\cap\mathscr{V}(P)=\{V_{\varepsilon(e(P))}\}\\ q\le Q-q(P)+\tilde{q}_{e(P)}}} \{f(NG, q, e(P))\}, \tag{37}$$

where functions $f(NG, q, v)$ are computed using modified arc costs $\bar{d}_{uv}$.

Proof Let R be the CARP route having smallest reduced cost $\bar{c}^r(R)$ with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ and corresponding to a GVRP route $\widetilde{R}$ containing P. Moreover, let $\overline{P}$ be the GVRP path that gives $\widetilde{R}$ when combined with P. Denote by $\bar{d}\,'(\overline{P})$ the cost of path $\overline{P}$ using modified arc costs $\bar{d}_{uv}$ (i.e., $\bar{d}\,'(\overline{P}) = \sum_{(uv)\in A(\overline{P})} \bar{d}_{uv}$), from expression (26) we have

$$\bar{d}\,'(\overline{P}) \ge \min_{NG\subseteq\mathscr{V}(\overline{P})\cap\widetilde{N}_{\varepsilon(e(\overline{P}))}} \left\{f(NG, q(\overline{P}), e(\overline{P}))\right\}. \tag{38}$$

Because P and $\overline{P}$ satisfy route feasibility conditions $(A) - (C)$ we have that $\varepsilon(e(P)) = \varepsilon(e(\overline{P}))$, say $\varepsilon(e(P)) = e$, and $\mathscr{V}(\overline{P}) \cap \mathscr{V}(P) = \{\widetilde{V}_0, \widetilde{V}_e\}$, and therefore

$$\{NG \subseteq \Pi : NG \subseteq \mathscr{V}(\overline{P}) \cap \widetilde{N}_e\} \subseteq \{NG \subseteq \Pi : NG \cap \mathscr{V}(P) = \{\widetilde{V}_e\}\}. \tag{39}$$

Moreover, from condition (A),

$$q(\overline{P}) \le Q - q(P) + q_e. \tag{40}$$

Therefore, from (38) using (39), (40) we obtain

$$\bar{d}\,'(\overline{P}) \ge \min_{\substack{NG\cap\mathscr{V}(P)=\{\widetilde{V}_e\}\\ q\le Q-q(\overline{P})+q_e}} \left\{f(NG, q, e(\overline{P}))\right\}. \tag{41}$$

Finally, because $\mathbf{g} \le 0$ and from Lemma 1 we obtain

$$\bar{c}^r(R) \ge \bar{d}(P) + \bar{d}(\overline{P}) \ge \bar{d}(P) + \bar{d}\,'(\overline{P}). \tag{42}$$

From (41) and (42) we obtain $LB^{ng}(P) \le \bar{c}^r(R)$. □

Appendix B: Detailed description of procedures GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$

We now describe in detail the two procedures GEN$\mathscr{P}$ and COMBINE$\mathscr{P}$ that correspond to the two phases of algorithm GENROUTES.

B.1 Procedure GEN$\mathscr{P}$

Procedure GEN$\mathscr{P}$ corresponds to a Dijkstra-like algorithm that generates a sequence of GVRP forward paths $(P^1, \ldots, P^h)$ satisfying the following conditions:

$(C1)$ $\overline{LB}^{ng}(P^1) \leq \cdots \leq \overline{LB}^{ng}(P^h) \leq \varrho$;
$(C2)$ $\tilde{c}(P^k) \leq \tilde{c}(P^s)$, for each path P^s such that $V(P^k) = V(P^s)$, $1 \leq k, s \leq h$;
$(C3)$ $\tilde{q}(P^k) \leq \lceil Q/2 \rceil + \tilde{q}_{e(P^k)}$, $1 \leq k \leq h$;
$(C4)$ P^k, $1 \leq k \leq h$, corresponds to either a forward-service CARP open route, or a backward-service CARP open route.

GEN$\mathscr{P}$ uses a set $\mathscr{T}$ of temporary paths which is initialized as $\mathscr{T} = \{P^0\}$, where $P^0 = (0)$ is an empty path visiting only node 0. At each iteration of GEN$\mathscr{P}$, a path $P \in \mathscr{T}$ having smallest value $\overline{LB}^{ng}(P)$ is extracted from $\mathscr{T}$ and inserted in $\mathscr{P}_{e(P)}$. A new path P_v is then created for each $v \in \widetilde{V}$ by inserting v at the end of P. Path P_v is rejected if it does not satisfy $(C1)$–$(C4)$, otherwise it is inserted in $\mathscr{T}$. GEN$\mathscr{P}$ terminates when either $\mathscr{T} = \emptyset$, or $\overline{LB}^{ng}(P) > \varrho$.

In order to reduce the number of generated paths, GEN$\mathscr{P}$ uses three fathoming rules: fathoming 1, fathoming 2, and fathoming 3. Fathoming 1 and 2 are dominance rules used to eliminate GVRP paths that are not optimal with respect to the arc costs $\tilde{c}_{uv}$: fathoming 1 performs a two-opt check of the cluster sequence visited by a path, whereas fathoming 2 keeps the cluster sequence fixed and checks whether the path is the shortest one visiting this sequence. Fathoming 3 requires the knowledge of a valid upper bound z_{UB} on $z(SP)$, and a feasible DRP solution $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$ of cost $\hat{z}$. It is based on the observation that the reduced cost $\hat{c}^r_\ell$ with respect to $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$ of any route R_ℓ being part of a feasible solution of cost z_{UB} satisfies $c^r_\ell \leq z_{UB} - \hat{z}$. Fathoming 3 was introduced by [6].

Fathoming 1. Let $P \in \mathscr{P}$ be a forward GVRP path such that $|P| \geq 4$ and let $e^3(P), e^2(P), e^1(P)$, be the three nodes preceding $e(P)$ in this path (i.e., $P = (0, v_1, \ldots, e^3(P), e^2(P), e^1(P), e(P))$). Path P can be fathomed if the subpath $(e^3(P), e^2(P), e^1(P), e(P))$ has a cost greater than the subpath $(e^3(P), e^1(P), e^2(P), e(P))$ with respect to arc costs $\tilde{c}_{uv}$.

Fathoming 2. Let $P = (0, v_1, \ldots, v_p) \in \mathscr{P}$ be a forward GVRP path, and let P^* be the shortest GVRP path in $\widetilde{G}$ using arc costs $\tilde{c}_{uv}$, visiting the cluster sequence $(\widetilde{V}_0, \widetilde{V}_{\varepsilon(v_1)}, \ldots, \widetilde{V}_{\varepsilon(v_{p-1})})$, and ending in v_p. Path P can be fathomed if $\tilde{c}(P) > \tilde{c}(P^*)$.

Fathoming 3. Let $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$ be a feasible DRP solution of cost $\hat{z}$, and let $\widehat{LB}^{ng}(P)$, $P \in \mathscr{P}$, be a completion bound on P with respect to $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$. Any path $P \in \mathscr{P}$ such that $\widehat{LB}^{ng}(P) > z_{UB} - \hat{z}$ can be fathomed as it cannot generate any GVRP route corresponding to a CARP route of any SP solution of cost less than or equal to z_{UB}.

The completion bound $\widehat{LB}^{ng}(P)$ required by fathoming 3 is computed as described in Sect. 6 using (37) and replacing $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$ with $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$. The shortest path P^* and its cost $\tilde{c}(P^*)$ required by fathoming 2 can be computed in $O(|P|)$ time as follows. Let $\tilde{s}(P, v_k)$ be the cost of the shortest path in $\widetilde{G}$ starting at 0, visiting cluster sequence $(\widetilde{V}_0, \widetilde{V}_{\varepsilon(v_1)}, \ldots, \widetilde{V}_{\varepsilon(v_k)})$, $k \leq p-1$, and ending at vertex v_p. Moreover, define $\bar{v} = V_{\varepsilon(v)} \backslash \{v\}, \forall v \in \widetilde{V}$. We have, $\tilde{s}(P, v_1) = \tilde{c}_{0v_1}, \tilde{s}(P, \bar{v}_1) = \tilde{c}_{0\bar{v}_1}$, and $\tilde{s}(P, v_k) = \min\{\tilde{s}(P, v_{k-1}) + \tilde{c}_{v_{k-1}v_k}, \tilde{s}(P, \bar{v}_{k-1}) + \tilde{c}_{\bar{v}_{k-1}v_k}\}, \tilde{s}(P, \bar{v}_k) =$

$\min\{\tilde{s}(P, v_{k-1}) + \tilde{c}_{v_{k-1}\bar{v}_k}, \tilde{s}(P, \bar{v}_{k-1}) + \tilde{c}_{\bar{v}_{k-1}\bar{v}_k}\}$. The cost $\tilde{c}(P^*)$ is then given by setting $\tilde{c}(P^*) = \min\{\tilde{s}(P, v_{p-1}) + \tilde{c}_{v_{p-1}v_p}, \tilde{s}(P, \bar{v}_{p-1}) + \tilde{c}_{\bar{v}_{p-1}v_p}$.

Note that both fathoming 1 and 2 can be viewed as special cases of condition $(C2)$. However, because of duals $\boldsymbol{v}$ and $\mathbf{w}$, given any two paths $P^k, P^s, 1 \le k, s, \le h$, it is often the case that $\tilde{c}(P^k) \le \tilde{c}(P^s)$, but $\bar{d}(P^k) \ge \bar{d}(P^s)$, and the path sequence $(P^1, \dots, P^h)$ generated by GEN$\mathscr{P}$ does not in general satisfy $\tilde{c}(P^1) \le \cdots \le \tilde{c}(P^h)$. Therefore, using fathoming 1 and 2 can result in practice in a significant reduction in the computing time and memory required by procedure GEN$\mathscr{P}$.

It is clear that by removing condition $(C3)$ procedure GEN$\mathscr{P}$ can be used directly to generate GVRP routes. In our computational experiments, we have observed a clear trade-off between the speedup obtained by imposing condition $(C3)$ when generating path set $\mathscr{P}$ and the time spent combining these paths to obtain routes. In particular, whenever $|\mathscr{P}|$ becomes large (say $|\mathscr{P}| > 10^6$) we found it convenient to turn GENROUTES into a single phase procedure that directly generates GVRP routes using GEN$\mathscr{P}$. In this case, it is possible to modify condition $(C4)$ to also fathom all GVRP paths corresponding to backward-service CARP open routes. Directly generating routes using GEN$\mathscr{P}$ also avoids the drawback of combining path pairs giving rise to routes that would be rejected by fathoming 1 or 2.

B.2 Procedure COMBINE$\mathscr{P}$

Procedure COMBINE$\mathscr{P}$ is an iterative procedure which is executed after GEN$\mathscr{P}$ to combine paths $\mathscr{P}_v, v \in \widetilde{V}$, and is designed to avoid symmetries when combining path pairs to extract route set $\mathscr{D}$. It is based on the observation that any forward-service CARP route having reduced cost less than or equal to ϱ corresponds to a GVRP route $\widetilde{R}_\ell, \ell \in \widetilde{\mathscr{R}}$, that can be decomposed into two forward GVRP paths P and $\overline{P}$ such that:

$(C5)$ $\bar{d}(P) + \bar{d}(\overline{P}) \le \varrho$;

$(C6)$ P and $\overline{P}$ satisfy route feasibility conditions $(A) - (C)$ (see Sect. 5.1);

$(C7)$ $q(\overline{P}) - \tilde{q}_{e(P)} \le q(P) \le q(\overline{P}) + \tilde{q}_{e(P)}$.

COMBINE$\mathscr{P}$ initializes $\mathscr{D} = \emptyset$ and generates a sequence of GVRP forward path pairs $\mathscr{L} = ((P^{r_1}, \overline{P}^{s_1}), (P^{r_2}, \overline{P}^{s_2}), \dots, (P^{r_p}, \overline{P}^{s_p}))$ such that each pair $(P^{r_k}, \overline{P}^{s_k}) \in \mathscr{L}$ satisfies $(C5)$–$(C7)$, and $\bar{d}(P^{r_1}) + \bar{d}(\overline{P}^{s_1}) \le \bar{d}(P^{r_2}) + \bar{d}(\overline{P}^{s_2}) \le \cdots \le \bar{d}(P^{r_p}) + \bar{d}(\overline{P}^{s_p})$. For each pair $h, 1 \le h \le p$, a corresponding CARP route R^h is obtained, and if $\bar{c}^r(R^h) \le \varrho$, then R^h is added to $\mathscr{D}$. Procedure COMBINE$\mathscr{P}$ terminates whenever either $|\mathscr{D}| \ge \Delta$, or no more pairs $(P, \overline{P})$ satisfying $(C5)$–$(C7)$ can be extracted from path sets $\mathscr{P}_v$.

Step-by-step description of COMBINE$\mathscr{P}$. Let $\mathscr{P}_v = (P_v^1, \dots, P_v^{h_v}), v \in \widetilde{V}$, be the set of all GVRP paths ending at node v computed by procedure GEN$\mathscr{P}$. In the following, we assume that each $\mathscr{P}_v$ is ordered by non-decreasing values of modified path costs, i.e., $\bar{d}(P_v^1) \le \cdots \le \bar{d}(P_v^{h_v})$. For the sake of simplicity, we denote by P_e^k the path in position k of ordered set $\mathscr{P}_v$ such that $\varepsilon(v) = e$ and $i(v) < j(v)$, and by $\overline{P}_e^k$ the path in position k of ordered set $\mathscr{P}_v$ such that $\varepsilon(v) = e$ and $i(v) > j(v)$, $\forall e \in E_R$. Thus, P_e^k and $\overline{P}_e^k$ correspond to CARP open routes ending at edge e and servicing it through arc (i_e, j_e) and (j_e, i_e), respectively. Moreover, we define $\bar{v} = V_{\varepsilon(v)} \setminus \{v\}, \forall v \in \widetilde{V}$. A step-by-step description of procedure COMBINE$\mathscr{P}$ is as follows.

1. Initialize $\mathscr{L} = ((P_e^1, \overline{P}_e^1) : P_e^1, \overline{P}_e^1$ satisfy (C5) − (C7)), $\forall e \in E_R$.
2. If $\mathscr{L} = \emptyset$, then STOP. Otherwise, extract from $\mathscr{L}$ a pair $(P_e^r, \overline{P}_e^s)$ having smallest value $\bar{d}(P_e^r) + \bar{d}(\overline{P}_e^s)$, and execute the following steps.
3. Let R_e^{rs} be the CARP route obtained by combining paths $P_e^r, \overline{P}_e^s$. If $\bar{c}^r(R_e^{rs}) \le \varrho$ and $c(R_e^{rs}) < c(R), \forall R \in \mathscr{D}$ such that $S(R) = S(R_e^{rs})$, then, add R to $\mathscr{D}$. If $|\mathscr{D}| \ge \Delta$, STOP. Otherwise, execute the following steps.
4. Add the following pairs to $\mathscr{L}$:
   (a) $(P_e^{r'}, \overline{P}_e^s)$, where $r' = \min\{\bar{r} = r+1, \ldots, |\mathscr{P}_v|$ s.t. $\bar{d}(P_e^{\bar{r}}) + \bar{d}(\overline{P}_e^s) \le \varrho$, and $P_e^{\bar{r}}, \overline{P}_e^s$ satisfy (C5) − (C7), if $r - s \ge 0$.
   (b) $(P_e^r, \overline{P}_e^{s'})$, where $s' = \min\{\bar{s} = s+1, \ldots, |\mathscr{P}_{\bar{v}}|$ s.t. $\bar{d}(P_e^r) + \bar{d}(\overline{P}_e^{\bar{s}}) \le \varrho$, and $P_e^r, \overline{P}_e^{\bar{s}}$ satisfy (C5) − (C7), if $r - s \le 0$.
5. If $r - s = 1$, or $r = s$ and $r - s' < -1$; then set $r = r + 1$, $s = s + 1$, add pair $(P_e^r, \overline{P}_e^s)$ to $\mathscr{L}$, and go to Step 4. Otherwise, go to Step 2.

A dominance rule similar to fathoming 3 described for GEN$\mathscr{P}$ is also applied at Step 3 of COMBINE$\mathscr{P}$. Specifically, the following fathoming procedure is applied for each route R_e^{rs} generated at Step 3.

Fathoming 4. Let $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$ be a feasible DRP solution of cost $\hat{z}$, and let $\hat{c}_\ell^r$ be the reduced cost with respect to $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\upsilon}}, \hat{\mathbf{w}}, \hat{\mathbf{g}})$ of any CARP route $R_\ell, \ell \in \mathscr{R}$. Any route R_ℓ such that $\hat{c}_\ell^r > z_{UB} - \hat{z}$ can be fathomed as it cannot be part of any SP solution of cost less than or equal to z_{UB}.

Appendix C: Cut-and-column generation algorithm LPCG

LPCG is a simplex-based cut-and-column generation method for solving LRP which constitutes the skeleton of procedures BP2, BP3, and BP4. It can be briefly described as follows.

Let η, γ, and $M(\mathscr{N})$ be a priori defined parameters. LPCG starts by defining an initial master problem $\overline{LRP}$ obtained from LRP by substituting the route set $\mathscr{R}$ with a subset $\overline{\mathscr{R}} \subseteq \mathscr{R}$, and the sets $\mathscr{S}$, $\mathscr{F}$, and $\mathscr{C}$ of inequalities (8), (14), and (25) with subsets $\overline{\mathscr{S}} \subseteq \mathscr{S}$, $\overline{\mathscr{F}} \subseteq \mathscr{F}$, and $\overline{\mathscr{C}} \subseteq \mathscr{C}$. At each iteration, it solves $\overline{LRP}$ using the simplex algorithm to obtain an optimal $\overline{LRP}$ solution $\bar{\mathbf{x}}$, and a corresponding $\overline{\text{DRP}}$ solution $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$. It then generates at most η constraints (14), (8), (25) violated by $\bar{\mathbf{x}}$, adds them to $\overline{\mathscr{S}}$, $\overline{\mathscr{F}}$, and $\overline{\mathscr{C}}$, respectively, and executes a new iteration. Whenever no violated inequality is found, or after at least γ inequalities have been added to $\overline{LRP}$ in subsequent iterations, LPCG solves the pricing problem to generate a subset $\mathscr{N} \subset \mathscr{R}$ containing at most $M(\mathscr{N})$ CARP routes having negative reduced cost with respect to $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\upsilon}}, \bar{\mathbf{w}}, \bar{\mathbf{g}})$. The new routes are then added to $\overline{\mathscr{R}}$. LPCG terminates whenever $\mathscr{N} = \emptyset$, and no violated inequalities (14), (8), or (25) were identified in the last iteration. In order to keep the LP compact, at each iteration, LPCG removes from $\overline{LRP}$ all variables and cuts that were inactive in the last θ iterations (say, $\theta = 20$), and moves them into a route pool $\mathfrak{R}$ and and a cut pool $\mathfrak{B}$, respectively. These pools are checked at each iteration before attempting to solve the pricing problem and before generating violated inequalities.

Violated inequalities at each iteration of LPCG are detected in the order (14), (8), and (25), and the corresponding separation problems are solved as follows.

Capacity constraints (14). Inequalities (14) are separated as CVRP rounded capacity inequalities over a weighted undirected graph $\hat{G}(\xi) = (\hat{V}(\xi), \hat{E}(\xi))$ defined by the aggregated variables ξ_{ef} obtained through equations (13). The graph $\hat{G}(\xi)$ is defined as follows. The node set $\hat{V}(\xi)$ contains a node 0 and a node u^e of demand q_e for each required edge $e \in E_R$. The edge set $\hat{E}(\xi)$ contains an edge $\{u^e, u^f\}$ of weight ξ_{ef} for each pair e, f such that $\xi_{ef} > 0$, and an edge $\{0, e\}$ of weight ξ_{0e} for each edge $e \in E_R$ such that $\xi_{0e} > 0$. Let $q(\hat{S})$ be the total demand of nodes in $\hat{S}$, for any $\hat{S} \subseteq \hat{V}(\xi)$, and let $\xi(\delta(\hat{S}))$ be the total weight of edges in $\hat{E}(\xi)$ crossing $\hat{S}$. Any set $\hat{S} \subseteq \hat{V}(\xi)\backslash\{0\}$ such that $\xi(\delta(\hat{S})) < \lceil q(\hat{S})/Q \rceil$ corresponds to a violated inequality (14) defined by an edge set $F = \{e \in E_R : u^e \in \hat{S}\}$. We use the package CVRPSEP [34] to identify sets in $\hat{G}(\xi)$ corresponding to violated inequalities (14).

Odd edge cutset constraints (8). Inequalities (8) can be separated exactly similarly to CARP odd edge cutset constraints (6) over a weighted undirected graph $G(\bar{y}) = (V(\bar{y}), E(\bar{y}))$ defined by the aggregated variables y_{ij} obtained through equations (7). Node set $V(\bar{y})$ of $G(\bar{y})$ corresponds to V, and edge set $E(\bar{y})$ contains an edge $\{i, j\}$ of weight $\bar{y}_{ij}$ for each node pair $i, j, i < j$, such that $\bar{y}_{ij} > 0$. For each node in $i \in V(\bar{y})$, node i is labeled "odd" if $|\delta_R(i)|$ is odd, and "even" otherwise. Let $\bar{y}(\delta(S))$ be the total weight of edges in $E(\bar{y})$ crossing S. Any set $S \subseteq V(\bar{y})\backslash\{0\}$ containing an odd number of odd nodes and such that $\bar{y}(\delta(S)) < 1$ corresponds to a violated inequality (8) defined by a node set S. To determine set S, we use the algorithm of [37] to compute the minimum weight odd cut $(S : V(\bar{y})\backslash S)$ in $G(\bar{y})$. Note that the size of graph $G(\bar{y})$ can be reduced by iteratively shrinking edges having weight greater than or equal to one, and if $G(\bar{y})$ is not connected the search can be restricted by considering each connected component of $G(\bar{y})$ separately.

Subset row inequalities (25). Let $\overline{\mathscr{R}}^{\varepsilon} = \{\ell \in \overline{\mathscr{R}} : \bar{x}_\ell > \varepsilon\}$, where ε is an a priori defined parameter. Inequalities (25) are separated by enumerating all triplets $\mathscr{C}$ and checking the degree of violation of the corresponding inequality with respect to the routes in $\overline{\mathscr{R}}^{\varepsilon}$. When detecting violated inequalities (25), an attempt is made to avoid generating violated inequalities that will likely be redundant in subsequent iterations. To this end, let $\mathscr{C}'$ be the set of triplets corresponding to all violated inequalities identified in the current iteration, and let $\omega(C)$ be the degree of violation of inequality (25) defined by $C, \forall C \in \mathscr{C}'$. All triplets C such that, for some $C' \in \mathscr{C}'$, $\omega(C) < \omega(C')$, and $|C' \cap C| = 2$ are then removed from $\mathscr{C}'$.

References

1. Amberg, A., Voß, S.: A hierarchical relaxations lower bound for the capacitated arc routing problem. In: Sprague H.R. (ed.) Proceedings of the 35th Hawaii International Conference on System Sciences, vol. 3, pp. 1–10. IEEE Computer Society, Washington, DC (2002)
2. Assad, A.A., Golden, B.L.: Arc routing methods and applications. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Network Routing, North-Holland, Amsterdam (1995)
3. Assad, A.A., Pearn, W.L., Golden, B.L.: The capacitated postman problem: lower bounds and solvable cases. Am. J. Math. Manage. Sci. **7**, 63–88 (1987)

4. Baldacci, R., Christofides, N., Mingozzi, A.: An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Math. Program. A **115**, 351–385 (2008)
5. Baldacci, R., Maniezzo, V.: Exact methods based on node-routing formulations for undirected arc-routing problems. Networks **47**, 52–60 (2006)
6. Baldacci, R., Mingozzi, A., Roberti, R.: New route relaxation and pricing strategies for the vehicle routing problem. Oper. Res. (2011, to appear)
7. Baldacci, R., Mingozzi, A., Roberti, R.: New state space relaxations for solving the traveling salesman problem with time windows. INFORMS J. Comput. doi:10.1287/ijoc.1110.0456 (2011)
8. Belenguer, J.M., Benavent, E.: The capacitated arc routing problem: valid inequalities and facets. Comput. Optim. Appl. **10**, 165–187 (1998)
9. Belenguer, J.M., Benavent, E.: A cutting plane algorithm for the capacitated arc routing problem. Comput. Oper. Res. **30**, 705–728 (2003)
10. Benavent, E., Campos, V., Corberán, A., Mota, E.: The capacitated arc routing problem: lower bounds. Networks **22**, 669–690 (1992)
11. Beullens, P., Muyldermans, L.: Personal communication (2010)
12. Beullens, P., Muyldermans, L., Cattrysse, D., Van Oudheusden, D.: A guided local search heuristic for the capacitated arc routing problem. Eur. J. Oper. Res. **147**, 629–643 (2003)
13. Blais, M., Laporte, G.: Exact solution of the generalized routing problem through graph transformations. J. Oper. Res. Soc. **54**, 906–910 (2003)
14. Bode, C., Irnich, S.: Cut-first branch-and-price-second for the capacitated arc-routing problem. Technical Report LM-2011-01 (preliminary version), Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Mainz, Germany (2011)
15. Christofides, N.: The optimum traversal of a graph. Omega **1**, 719–732 (1973)
16. Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxation. Math. Program. **20**, 255–282 (1981)
17. Corberán, A., Prins, C.: Recent results on arc routing problems: an annotated bibliography. Networks **56**, 50–69 (2010)
18. CPLEX: *ILOG CPLEX 12.1 Callable Library*. (ILOG, Nevada 2008)
19. Dror, M. (ed.): Arc Routing: Theory, Solutions and Applications. Kluwer, Boston (2000)
20. Eglese, R.W.: Routeing winter gritting vehicles. Discr. Appl. Math. **48**, 231–244 (1994)
21. Eiselt, H.A., Gendreau, M., Laporte, G.: Arc routing problems, part I: The Chinese postman problem. Oper. Res. **43**, 231–242 (1995)
22. Eiselt, H.A., Gendreau, M., Laporte, G.: Arc routing problems, part II: The rural postman problem. Oper. Res. **43**, 399–414 (1995)
23. Fukasawa, R., Longo, H., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E., Werneck, R.F.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Math. Program. A **106**, 491–511 (2006)
24. Ghiani, G., Improta, G.: An efficient transformation of the generalized vehicle routing problem. Eur. J. Oper. Res. **122**, 11–17 (2000)
25. Golden, B.L., DeArmon, J.S., Baker, E.K.: Computational experiments with algorithms for a class of routing problems. Comput. Oper. Res. **10**, 47–59 (1983)
26. Golden, B.L., Wong, R.T.: Capacitated arc routing problems. Networks **11**, 305–315 (1981)
27. Gómez-Cabrero, D., Belenguer, J.M., Benavent, E.: Cutting plane and column generation for the capacitated arc routing problem. Presented at ORP3 MEETING, Valencia, Spain (2005)
28. Hirabayashi, R., Nishida, N., Saruwatari, Y.: Node duplication lower bounds for the capacitated arc routing problems. J. Oper. Res. Soc. Jpn **35**, 119 (1992)
29. Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D.: Subset-row inequalities applied to the vehicle routing problem with time windows. Oper. Res. **56**, 497–511 (2008)
30. Kiuchi, M., Shinano, Y., Hirabayashi, R., Saruwatari, Y.: An exact algorithm for the capacitated arc routing problem using parallel branch and bound method. In: Abstracts of the 1995 Spring National Conference of the Operations Research Society of Japan, pp. 28–29 (1995)
31. Letchford, A.N., Oukil, A.: Exploiting sparsity in pricing routines for the capacitated arc routing problem. Comput. Oper. Res. **36**, 2320–2327 (2009)
32. Li, L.Y.O., Eglese, R.W.: An interactive algorithm for vehicle routing for winter-gritting. J. Oper. Res. Soc. **47**, 217–228 (1966)
33. Longo, H., Poggi de Aragão, M., Uchoa, E.: Solving capacitated arc routing problems using a transformation to the CVRP. Comput. Oper. Res. **33**, 1823–1837 (2006)

34. Lysgaard, J.: CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Technical Report Working paper 03-04, Department of Management Science and Logistics, Aarhus School of Business, Denmark (2003)
35. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math. Program. A **100**, 423–445 (2004)
36. Martinelli, R., Pecin, D., Poggi de Aragão, M., Longo, H.: Column generation bounds for the capacitated arc routing problem. Presented at XLII SBPO, Bento Gonçalves, Brazil (2010)
37. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and *b*-matchings. Math. Oper. Res. **1**, 67–80 (1982)
38. Pearn, W.L.: New lower bounds for the capacitated arc routing problem. Networks **18**, 181–191 (1988)
39. Santos, L., Coutinho-Rodrigues, J., Current, J.R.: An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transport. Res. B **44**, 246–266 (2010)
40. Win, Z.: Contributions to Routing Problems. PhD thesis, University of Augsburg, Germany (1987)
41. Wøhlk, S.: New lower bound for the capacitated arc routing problem. Comput. Oper. Res. **33**, 3458–3472 (2006)
42. Wøhlk, S.: A decade of capacitated arc routing. In: Golden, B.L., Raghavan, S., Wasil, A. (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, New York (2008)