European Journal of Operational Research 300 (2022) 837–851

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The min-max close-enough arc routing problem

Nicola Bianchessi[a], Ángel Corberán[b], Isaac Plana[c,*], Miguel Reula[d], José M. Sanchis[e]

[a] Department of Computer Science, Università degli Studi di Milano, Italy
[b] Departament d'Estadística i Investigació Operativa, Universitat de València, Spain
[c] Departament de Matemàtiques per a l'Economia i l'Empresa, Universitat de València, Spain
[d] Departament d'Estadística i Investigació Operativa, Universitat de València, Spain
[e] Departamento de Matemática Aplicada, Universidad Politécnica de Valencia, Spain

A R T I C L E I N F O

Article history:
Received 10 May 2021
Accepted 22 October 2021
Available online 29 October 2021

Keywords:
Routing
Close-enough arc routing problem
Branch-and-price
Branch-and-cut
Min-max

A B S T R A C T

Here we introduce the Min-Max Close-Enough Arc Routing Problem, where a fleet of vehicles must serve a set of customers while trying to balance the length of the routes. The vehicles do not need to visit the customers, since they can serve them from a distance by traversing arcs that are "close enough" to the customers. We present two formulations of the problem and propose a branch-and-cut and a branch-and-price algorithm based on the respective formulations. A heuristic algorithm used to provide good upper bounds to the exact procedures is also presented. Extensive computational experiments to compare the performance of the algorithms are carried out.

1. Introduction

In this article we introduce an arc routing problem where a fleet of homogeneous vehicles has to serve a set of customers in such a way that the lengths of the routes are balanced. Each customer is associated with a subset of "close-enough" arcs and by traversing any of these arcs the vehicle serves the customer. The problem, called the Min-Max Close Enough Arc Routing Problem (MM-CEARP), consists of finding a set of vehicle routes, all of them starting and ending at the depot, jointly servicing all the customers, and such that the length of the longest route is minimized.

The MM-CEARP is NP-hard as it generalizes the Close Enough Arc Routing Problem (CEARP), the single-vehicle version of the problem in which the total distance traveled is minimized. The CEARP was first introduced as the Generalized Directed Rural Postman Problem (GDRPP) by Drexl (2007), who proved it to be NP-hard by showing that any Directed Rural Postman Problem (DRPP) instance can be solved by transforming it into a GDRPP one that has one customer containing one single arc for each required arc of the DRPP instance. The CEARP was also studied by Shuttleworth, Golden, Smith, & Wasil (2008), Drexl (2014), Hà, Bostel, Langevin, & Rousseau (2012, 2014), Ávila, Corberán, Plana, & Sanchis (2016), and Cerrone, Cerulli, Golden, & Pentangelo (2017). The exact algorithms proposed in the last three references are the most successful methods for the CEARP. They are capable of solving to optimality instances with up to 500 vertices, 1500 arcs, and between 500 and 15,000 customers. A stochastic version of the CEARP was studied by Renaud, Absi, & Feillet (2017), while Aráoz, Fernández, & Franquesa (2017) considered the special case in which the customers are associated with clusters of edges that define pairwise-disjoint connected subgraphs. A real-life application of the CEARP arises in automated meter reading of water or gas consumption. Instead of visiting customers one by one, the vehicles only need to traverse a street that is close enough to the meters in order to receive the consumption data via radio frequency identification (RFID). This application was first described in Gulczynski, Heath, & Price (2006), although in the context of node routing problems. The papers by Shuttleworth et al. (2008) and Hà et al. (2012, 2014) were the first that studied the automated meter reading problem in the context of a street network. The paper by Eglese, Golden, & Wasil (2014) is an interesting summary of the models and solution methods proposed since the late 1970s in meter reading. Another application of the CEARP can be found in inventory management in large companies. In Duric, Jovanovic, & Sibalija (2018) a system that allows aerial drones to read RFID tags from tens of meters away and identify the location of the tags with a small average error is described. Therefore, to carry out the inventory, the drone does not need to traverse all the aisles of the warehouse to collect data. Drones with RFID receivers or integrated cameras are identified by Aráoz et al. (2017) as the most suitable

* Corresponding author.
E-mail address: isaac.plana@uv.es (I. Plana).

https://doi.org/10.1016/j.ejor.2021.10.047
0377-2217/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

devices to perform network maintenance and surveillance tasks. The drones do not need to fly over the nodes or lines to be monitored, but only to approach the target at a certain distance. The authors noted that only a subset of the edges of a network should be traversed in network maintenance quality control. They also argued that CEARP is the most appropriate problem for modeling location arc routing problems in which facilities need to be located in some given areas and connected between them via a route.

More recently, the CEARP with several vehicles has been subject of investigation in Ávila, Corberán, Plana, & Sanchis (2017) and Corberán, Plana, Reula, & Sanchis (2019, 2021a). These articles deal with the Distance-Constrained Close Enough Arc Routing Problem (DC-CEARP). This problem consists of finding a set of routes of total minimum cost, leaving from and entering at the depot and servicing all the customers, while ensuring that the length of each route does not exceeds a given maximum distance. In Ávila et al. (2017), the authors present and compare four formulations for the problem and four branch-and-cut algorithms based on them. The best method is able to optimally solve instances with up to five vehicles, 196 vertices, 450 arcs, and 150 customers. In Corberán, Plana, Reula, & Sanchis (2021a), a new formulation that combines the best features of the previous ones is presented and its associated polyhedron is studied. Based on that study, an exact algorithm improving the existing ones is also proposed. In Corberán et al. (2019) a matheuristic providing good feasible solutions for the DC-CEARP is described.

In the context of the meter-reading application, most real instances are so large that a vehicle is not capable of servicing all the customers within the working time period. Then, several routes for a vehicle or for a fleet of vehicles need to be designed and their working times balanced. For example, Shuttleworth et al. (2008) report the solution of a real instance with 150,000 customers and 16,500 street segments that were partitioned into 18 routes. Min-max objectives are quite common in routing problems because they lead to more realistic models, since minimizing the length of the longest route tends to balance the length or cost of the planned routes. Moreover, if the travel times are proportional to the travel distance, this objective tries to minimize the time at which the last customer is served. As Ahr noted in Ahr (2004), “this kind of objective is preferable when the aim is to serve each customer as early as possible”. The min-max objective for several arc routing problems was first proposed in Frederickson, Hecht, & Kim (1978). These authors introduced the Min-Max K-Chinese Postman Problem (Min-Max K-CPP) and proved that it is NP-hard and proposed a (2-1/K)-approximation algorithm. More recently Ahr & Reinelt (2002) presented several lower bounds and heuristics for this problem and a Tabu Search procedure that produces very good solutions (Ahr & Reinelt, 2006). In Ahr (2004) some more results on the Min-Max K-CPP, including an exact solution method based on a branch-and-cut approach, are presented. Furthermore, Applegate, Cook, Dash, & Rohe (2002) considered a min-max problem in a newspaper delivery context. The chapter by Benavent, Corberán, Plana, & Sanchis (2014) summarizes the results obtained for some important min-max arc routing problems.

The main contribution of the paper at hand is to introduce in the literature the MM-CEARP, focusing on its modeling and its exact solution. More precisely, we propose two different models for the problem: an arc-based formulation making use of arc and servicing variables, and a route-based set covering formulation. Then, on the basis of the proposed models, we present a branch-and-cut (BC) algorithm as well as a branch-and-price (BP) algorithm. As for the BP algorithm, an additional contribution comes from the definition of the first-level rule used in the branching scheme. In the route-based formulation, the sets of feasible routes associated with the vehicles are identical. The proposed branching scheme allows to recover integer solutions at the expenses of a diversification of the sets of feasible routes. Nevertheless, the first-level rule does not introduce symmetries in the solution space, does not alter the structure of the pricing problems, and, finally, allows the pricing problems to continue sharing the same feasible region. In turn, as long as only this rule is applied, this allows to design the sequential solution of the pricing problems (at each the column generation iteration) to potentially avoid solving some of them. Furthermore, the first-level rule consists of an application of the Ryan and Foster's branching rule (see Ryan & Foster, 1981), which is itself something not typical when (i) columns of the master program refer to elements of distinct sets and/or (ii) the BP algorithm is addressing a routing problem. In particular, as for (ii), we have been able to efficiently handle the implications arising from the application of such a kind of rule thanks to the BC algorithm used to solve the pricing problems to optimality. Again something not typical for BP algorithms addressing routing problems, where the leading technique used to solve the pricing problems consists of dynamic programming algorithms.

The rest of the paper is organized as follows. In Section 2, we formally define the MM-CEARP and present for the problem an arc-based and a route-based formulation. Solution algorithms to address the problem are then presented. In Section 3 we present a BC algorithm addressing the arc-based formulation, whereas in Section 4 we describe a BP algorithm based on the set covering formulation. A heuristic used to compute solutions with which initializing the exact algorithms is described in Section 5. To compare the exact algorithms, extensive computational experiments on benchmark instances are reported in Section 6. Conclusions are drawn in Section 7. In order to ease the reading, a list of the main sets, parameters, and variables used along this work is reported in Table 7 at the end of the paper.

2. Problem definition and formulation

Let $G=(V,A)$ be a strongly connected directed graph with set of vertices V, where vertex 1 denotes the depot, and set of arcs A, and let $d_{ij}\geq 0$ be the an integer value representing the length/distance associated with the traversal of arc $(i,j)\in A$. There is a fleet of K identical vehicles based at the depot and a set of L customers. Each customer $c\in\{1,\dots,L\}$ has associated a set of arcs $H_c\subseteq A$ from which it can be served. We consider that a customer c is served if there is a vehicle k that traverses at least one arc in H_c. Note that the subsets H_c do not need to be disjoint nor induce connected subgraphs. The Min-Max Close-Enough Arc Routing Problem consists of finding a set of K routes, starting and ending at the depot, servicing all the customers and minimizing the length of the largest route.

In what follows, $\mathbb{K}=\{1,\dots,K\}$ will represent the set of vehicles and $\mathbb{H}=\{1,\dots,L\}$ the set of customers. Given sets $S,S_1,S_2\subset V$, we define $(S_1,S_2)=\{(i,j)\in A: i\in S_1\ j\in S_2\}$, $\delta^+(S)=(S,V\setminus S)$, $\delta^-(S)=(V\setminus S,S)$, $\delta(S)=\delta^+(S)\cup\delta^-(S)$, and $A(S)=\{(i,j)\in A: i,j\in S\}$. Finally, given a vector x indexed on the arcs, and given a set F of arcs, $x(F)=\sum_{(i,j)\in F}x_{ij}$.

2.1. Arc-based formulation

In this section we present an ILP formulation for the MM-CEARP, very similar to one of the four proposed by Ávila et al. (2017) for the DC-CEARP, which uses an artificial variable w to model the minimization of the maximum length route and the following two sets of variables:

x_{ij}^k = number of times that the vehicle k traverses arc $(i,j)\in A$,

$$z_c^k=\begin{cases}1, & \text{if customer } c \text{ is served by vehicle } k\\ 0, & \text{otherwise.}\end{cases}$$

Table 1
Characteristics of the MM-CEARP instances.

	# Inst	Max K	$\lvert V\rvert$	$\lvert A\rvert$		$\lvert A_R\rvert$		$\lvert A_{NR}\rvert$		$\lvert\mathbb{H}\rvert$	
				Min	Max	Min	Max	Min	Max	Min	Max
Albaida	81	8	116	259	305	124	172	109	162	18	33
Madrigueras	105	11	196	453	544	224	305	197	281	22	47
Random50	24	5	50	296	300	105	292	7	193	10	100
Random75	48	8	75	448	450	143	438	10	305	15	150

The first MM-CEARP formulation is:

$$\min \quad w \tag{1a}$$

$$s.t. \quad \sum_{k\in\mathbb{K}} z_c^k = 1 \qquad \forall c \in \mathbb{H} \tag{1b}$$

$$\sum_{(i,j)\in A} d_{ij} x_{ij}^k \le w \qquad \forall k \in \mathbb{K} \tag{1c}$$

$$x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall\, i \in V,\ \forall k \in \mathbb{K} \tag{1d}$$

$$\sum_{(i,j)\in H_c} x_{ij}^k \ge z_c^k \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{1e}$$

$$x^k(\delta^+(S)) \ge z_c^k - x^k(H_c \cap A(V\setminus S)) \qquad \forall S \subset V\setminus\{1\},\ \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{1f}$$

$$x_{ij}^k \ge 0 \ \text{ and integer} \qquad \forall (i,j) \in A,\ \forall k \in \mathbb{K} \tag{1g}$$

$$z_c^k \in \{0,1\} \qquad \forall c \in \mathbb{H},\ \forall k \in \mathbb{K} \tag{1h}$$

Eq. (1b) forces the service of all customers while inequalities (1c) imply that the length of any route is less than or equal to w, and, together to the objective function, that the length of the longest route is minimized. Constraints (1d) are the well known symmetry equations for each vertex in V. Inequalities (1e) ensure that if a vehicle serves a customer c, at least one arc in H_c must be traversed. The connectivity of each route is guaranteed by inequalities (1f). If vehicle k does not serve customer c, $z_c^k = 0$ and the inequality is trivially satisfied. Otherwise, if vehicle k serves customer c by traversing an arc in $H_c \cap A(V\setminus S)$, then it does not need to traverse the cut-set $\delta(S)$ and the inequality is also trivially satisfied. Only when vehicle k serves customer c by traversing an arc not in $H_c \cap A(V\setminus S)$ (hence, traversing an arc in $\delta(S)$ or in $A(S)$), the vehicle has to traverse $\delta(S)$ and, therefore, the inequality is satisfied. Note that there is an exponential number of such inequalities. Finally, (1g) and (1h) define the domain of the variables

In what follows we present the parity inequalities proposed in Ávila et al. (2017) and Corberán et al. (2021a) for the DC-CEARP. They are also valid for our problem and will be used to strengthen the linear relaxation of the above formulation.

Parity inequalities

Parity inequalities are implied by the fact that any cutset has to be traversed by each vehicle an even, or zero, number of times. Note that symmetry Eq. (1d) guarantee that every node has even degree in the graph induced by any integer solution $x \in \mathbb{Z}^{|A|}$. However, if x is fractional, this is not necessarily true and, therefore, parity inequalities can help to cut this kind of "solutions".

Let $S \subseteq V\setminus\{1\}$ and $F^{\mathbb{H}} = \{c_1, c_2, \dots, c_q\}$, where $q \ge 3$ and odd, satisfying

- $H_{c_i} \cap H_{c_j} \cap \delta(S) = \emptyset$ and
- $H_{c_i} \cap \delta(S) \ne \emptyset \quad \forall i \in \{1,\dots,q\}$

The following inequality is called *disaggregate z-parity inequality*, because is associated with a single vehicle k, and is valid for the MM-CEARP

$$x^k(\delta(S)) \ge \sum_{i=1}^{q} \Big(2z_{c_i}^k - 1 - 2x^k(H_{c_i}\setminus\delta(S))\Big) + 1. \tag{2a}$$

Basically, the inequality establish that if vehicle k serves customer c_i and does not traverse any edge in $H_{c_i}\setminus\delta(S)$, then k serves c_i by traversing at least an arc in $H_{c_i}\cap\delta(S)$. If this is true for the q customers in $F^{\mathbb{H}}$, vehicle k has to traverse $\delta(S)$ at least q times, and, since q is odd, k has to traverse the cutset at least one more time.

Parity inequalities can be generalized to any subset of vehicles as follows. Given a subset of vehicles $\Omega = \{k_1,\dots,k_p\}$, the associated Ω-*aggregate z-parity inequality* is

$$\sum_{k\in\Omega} x^k(\delta(S)) \ge \sum_{i=1}^{q} \Big(\sum_{k\in\Omega} 2z_{c_i}^k - 1 - 2\sum_{k\in\Omega} x^k(H_{c_i}\setminus\delta(S))\Big) + 1. \tag{2b}$$

If $\Omega = \mathbb{K}$, we have the *aggregate parity inequality*

$$\sum_{k\in\mathbb{K}} x^k(\delta(S)) \ge \sum_{i=1}^{q} \Big(1 - 2\sum_{k\in\mathbb{K}} x^k(H_{c_i}\setminus\delta(S))\Big) + 1. \tag{2c}$$

2.2. Route-based formulation

As illustrated in Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance (1998), most routing problems can be formulated in a natural way as set partitioning problems where the columns (of the coefficient matrix) correspond to feasible routes for the vehicles and each row (of the coefficient matrix) corresponds to the requirement that a customer must be served *exactly* once. Alternatively, the problem can be formulated as a set covering problem in which it is required that each customer is served *at least* once. Note that, if a subcolumn of a feasible column defines another feasible column with lower cost, an optimal solution to the set covering problem will define an optimal set partitioning solution and, hence, it is possible to work with any of the two formulations. However the set covering formulation has the following advantages:

- its linear programming relaxation is numerically more stable and thus easier to solve, and
- it is trivial to construct a feasible integer solution from a solution to the linear programming relaxation.

According to these insights, we modeled the MM-CEARP by means of a route-based set covering formulation that leads the bases for the BP algorithm discussed in Section 4.

Let R^k be the set of feasible routes for vehicle $k \in \mathbb{K}$. Feasibility takes into account constraints (1c)–(1h) for each vehicle $k \in \mathbb{K}$. For each $r \in R^k$, let d^{kr} be the length of the route. Moreover, for each customer $c \in \mathbb{H}$ and each $r \in R^k$, let s_c^{kr} be a binary parameter equal to 1 if the route r serves customer c and 0 otherwise. Then, let's consider a set of variables associated with the use of the routes:

$$\lambda^{kr} = \begin{cases} 1, & \text{if the route } r \in R^k \text{ is assigned to the vehicle } k \in \mathbb{K}, \\ 0, & \text{otherwise,} \end{cases}$$

and another set of variables modeling the length of the route assigned to each vehicle:

$w^k =$ length of route assigned to vehicle $k \in \mathbb{K}$.

Using this notation, the MM-CEARP can be formulated as follows:

$$\min \quad w^1 \tag{3a}$$

$$s.t. \sum_{k\in\mathbb{K}} \sum_{r\in R^k} s_c^{kr}\lambda^{kr} \geq 1 \qquad \forall c \in \mathbb{H} \tag{3b}$$

$$\sum_{r\in R^k} \lambda^{kr} = 1 \qquad \forall k \in \mathbb{K} \tag{3c}$$

$$\sum_{r\in R^k} d^{kr}\lambda^{kr} - w^k \leq 0 \qquad \forall k \in \mathbb{K} \tag{3d}$$

$$w^k - w^{k+1} \geq 0 \qquad \forall k = 1, \dots, K-1 \tag{3e}$$

$$\lambda^{kr} \in \{0,1\} \qquad \forall k \in \mathbb{K}, \ \forall r \in R^k \tag{3f}$$

The objective function (3a) minimizes the length of the longest route. This is ensured by constraints (3d) together with (3e). Actually, constraints (3d) define the lengths of the routes assigned to the vehicles. Then, constraints (3e) impose the lengths of the routes associated with vehicles from 1 to K to be sorted in non-increasing order. The mandatory service of the customers is established in inequalities (3b). The convexity constraints (3c) imply that a single route $r \in R^k$ is assigned to each vehicle $k \in \mathbb{K}$. Finally, constraints (3f) define the domain for the λ^{kr} variables. Constraints $w^k \geq 0$ are implied by inequalities (3d).

Note that sets R^k, $k \in \mathbb{K}$, are all identical. Nevertheless, we decided to index them (by vehicle index) to have a notation allowing us to better explain the BP algorithm (see Section 4). In particular, the reason for using such a notation will be clarified in Section 4.2.

3. Branch-and-cut algorithm

In this section, we describe the branch-and-cut algorithm for solving the MM-CEARP, which relies on the arc-based formulation presented in Section 2.1 and the use of mixed-integer programming (MIP) solver.

3.1. Separation algorithms

Here we describe the separation algorithms that have been used to identify inequalities that are violated by the current LP solution at any iteration of the cutting-plane phase of the branch-and-cut algorithm, which includes separation methods for identifying violated connectivity (1f) and aggregated parity inequalities (2c).

Connectivity inequalities

To identify violated connectivity inequalities (1f) we have used a heuristic procedure proposed in Ávila et al. (2017) for the DC-CEARP. Given a solution (x^{k*}, z^{k*}) of the linear relaxation corresponding to a vehicle k, we first build the graph induced by the arcs $a \in A$ such that $x_a^{k*} \geq \varepsilon$, where ε is a given parameter. If the support graph is not weakly connected, let $C_1, \dots, C_q$ be its weakly connected components. For each C_i, let S be its associated set of vertices. We look for the customer $c \in \mathbb{H}$ such that $z_c^{k*} - x^{k*}(H_c \cap A(V \setminus S)$ is maximized. If $x^{k*}(\delta^+(S)) < z_c^{k*} - x^{k*}(H_c \cap A(V \setminus S)$ the corresponding connectivity constraint (1f) is violated. This procedure has a computational complexity $O(K|A||\mathbb{H}|)$.

A second heuristic (described in Corberán et al. (2021a)) based on the Gomory–Hu algorithm and working in $O(|V|^3|A|)$ time is also applied.

Parity inequalities

To separate parity inequalities (2c) we have implemented the following heuristic algorithm with complexity $O(|A|^2)$. Note that these inequalities can be written as

$$\sum_{k\in\mathbb{K}} x^k(\delta(S)) \geq \sum_{k\in\mathbb{K}} \sum_{i=1}^{q} \Big(z_{c_i}^k - 2x^k\big(H_{c_i} \setminus \delta(S)\big)\Big) + 1.$$

If (x^{k*}, z^{k*}) are the values of the variables associated with vehicle k in the solution of the linear relaxation, let $(\bar{x}^*, \bar{z}^*)$ be the aggregated solution, that is, $\bar{x}_a^* = \sum_{k\in\mathbb{K}} x_a^{k*}$ and $\bar{z}_c^* = \sum_{k\in\mathbb{K}} z_c^{k*} = 1$. First, we create the graph induced by the arcs $a \in A_R = H_1 \cup \ldots \cup H_L$ with $\bar{x}_a^* \geq 1 + \varepsilon$ and by the arcs $a \notin A_R$ with $\bar{x}_a^* > \varepsilon$, where ε is a given parameter. Let $C_1, \dots, C_k$ be the weakly connected components of this graph. Then, given a connected component C_i and its associated set of vertices S, we compute $\bar{x}^*(\delta(S) \cap A_R)$ and check if this value is close to an odd number, that is, $2n + 0.75 \leq \bar{x}^*(\delta(S) \cap A_R) \leq 2n + 1.25$. If so, the heuristic tries to select $q = 2n+1$ customers among those having arcs in the cutset in order to form set $F^{\mathbb{H}}$ as described in Section 2.1. To do so, we iteratively add customers to $F^{\mathbb{H}}$ in decreasing order of the $\bar{z}_c^* - 2\bar{x}^*(H_c \setminus \delta(S))$ values, such that the sets $H_c \cap \delta(S)$ are disjoint with those associated with the previously selected customers, until we reach the desired number q of customers. If there are not enough customers that can be selected, we choose another component. Otherwise, we check if the inequality (2c) is violated.

3.2. Initial relaxation and cutting-plane algorithm

The initial LP relaxation contains all the inequalities in the formulation except for the connectivity inequalities, which are exponential in number. At each cutting plane iteration, the separation algorithms are applied in the following order:

1. Connectivity inequalities separation algorithm based on connected components with $\epsilon = 0, 0.25, 0.5, 0.75$.
2. Connectivity inequalities separation algorithm based on Gomory–Hu.
3. Only at the root node, parity inequalities separation algorithm with $\epsilon = 0, 0.25, 0.5, 0.75$.

This cutting-plane algorithm is applied at each node of the tree until no new violated inequalities are found. When this happens, we branch using the strong branching strategy provided by the MIP solver. This strategy branches on variables and allows to assign different priorities to them. Variables with higher priority are the first ones used for branching. We have assigned a higher priority to the z_c^k variables.

4. Branch-and-price algorithm

When a set covering problem is addressed by means of a BP algorithm, its formulation, in our case formulation (3), is usually referred as master program (MP). In the BP algorithm, at each node of the branch-and-bound tree, the linear relaxation of the MP (LMP), eventually augmented by branching constraints, is solved iteratively by means of column generation. The starting point is to define the LMP over a subset $\tilde{R} \subseteq \bigcup_{k\in\mathbb{K}} R^k$ of the feasible routes for the vehicles. This restricted version of LMP is usually called reduced linear master program (RLMP). At each iteration, column generation alternates between the optimization of the RLMP and the solution of pricing problems (PPs). The former allows to retrieve optimal dual variable values with respect to set $\tilde{R}$. The latter, on the bases of the dual variable values, generates negative reduced cost route variables λ^{kr} to be included in the RLMP, if any. When no negative reduced cost variable is found, the optimal solution of the RLMP is also the optimal solution of the LMP Desaulniers, Desrosiers, & Solomon (2005). Branching is finally required to ensure the integrality of the solution.

4.1. Column generation

Let us consider the linear relaxation of (3) at the root node of the branch-and-bound tree. The dual variables associated with the constraints (3b), (3c), (3d), and (3e) are respectively:

- $\mu_c \in \mathbb{R}^+$, for each customer $c \in \mathbb{H}$,
- $\theta_k \in \mathbb{R}$, for each vehicle $k \in \mathbb{K}$,
- $\rho_k \in \mathbb{R}^-$, for each vehicle $k \in \mathbb{K}$,
- $\sigma_k \in \mathbb{R}^+$, for each $k = 1, \dots, K-1$.

Using these dual variables in their respective domain, we are able to express the formulation of the dual of LMP as follows:

$$\max \sum_{c\in\mathbb{H}} 1\cdot\mu_c + \sum_{k\in\mathbb{K}} 1\cdot\theta_k + \sum_{k\in\mathbb{K}} 0\cdot\rho_k + \sum_{k=1}^{K-1} 0\cdot\sigma_k \tag{4a}$$

$$s.t. \quad \sum_{c\in\mathbb{H}} s_c^{kr}\mu_c + \theta_k + d^{kr}\rho_k \le 0 \qquad \forall k\in\mathbb{K},\ \forall r\in R^k \tag{4b}$$

$$-\rho_1 + \sigma_1 \le 1 \tag{4c}$$

$$-\rho_k + \sigma_k - \sigma_{k-1} \le 0 \qquad \forall k = 2,\dots,K-2 \tag{4d}$$

$$-\rho_K - \sigma_{K-1} \le 0 \tag{4e}$$

where there is a constraint (4b) for each variable λ^{kr} of the primal formulation, and constraints (4c)–(4e) are related with each w^k variable, $k \in \mathbb{K}$.

Thus, based on the dual formulation (4), we can see that there is one distinct PP for each vehicle $k \in \mathbb{K}$. In particular, given the duals $(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho}, \boldsymbol{\sigma})$, the PP for vehicle $k \in \mathbb{K}$ consists of finding a minimum reduced cost route to be assigned to the vehicle, where the reduced cost $\bar{c}^{kr}(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho})$ of route $r \in R^k$ to be assigned to the vehicle is defined as:

$$\bar{c}^{kr}(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho}) = -\sum_{c\in\mathbb{H}} s_c^{kr}\mu_c - \theta_k - d^{kr}\rho_k \tag{5}$$

A solution (a route) corresponds to a negative reduced cost λ^{kr} variable if its value (reduced cost) is less than 0.

4.1.1. Pricing problem modeling

In order to define the pricing problem, we consider the same variables (with the same meaning) as those used in formulation (1). The PP associated with vehicle $k \in \mathbb{K}$ can then be formulated as follows:

$$\min \quad -\sum_{c\in\mathbb{H}} \mu_c z_c^k - \sum_{(i,j)\in A} d_{ij} x_{ij}^k \rho_k \tag{6a}$$

$$s.t. \quad x^k(\delta^+(i)) = x^k(\delta^-(i)) \qquad \forall\, i \in V \tag{6b}$$

$$x^k(\delta^+(S)) \ge z_c^k - x^k(H_c \cap A(V\setminus S)) \qquad \forall S \subset V\setminus\{1\},\ \forall c \in \mathbb{H} \tag{6c}$$

$$\sum_{(i,j)\in H_c} x_{ij}^k \ge z_c^k \qquad \forall c \in \mathbb{H} \tag{6d}$$

$$x_{ij}^k \ge 0 \ \text{ and integer} \qquad \forall (i,j) \in A \tag{6e}$$

$$z_c^k \in \{0,1\} \qquad c \in \mathbb{H} \tag{6f}$$

where $\mu_c \ge 0$, $\theta_k \in \mathbb{R}$, $\rho_k \le 0$ and, hence, $-d_{ij}\rho_k \ge 0$ for each $(i,j) \in A$. The objective function (6a) aims at minimizing the reduced cost of the route. Constraints (6b) are the symmetry equations for each vertex, while constraints (6c) are used to ensure the connectivity of the optimal solution. Consistency between the x_{ij}^k and z_c^k variables is imposed through constraints (6d).

An optimal solution to (6) corresponds to a negative reduced cost variable if its value is less than θ_k.

Moreover, when an upper bound W is available for w^1, R^k can be restricted to include feasible routes such that $d^{kr} \le W-1$ (d_{ij} is an integer value for each $(i,j) \in A$), and we can include in formulation (6) the following constraint:

$$\sum_{(i,j)\in A} d_{ij} x_{ij}^k \le W - 1 \tag{6g}$$

Note that the valid inequalities (6b)–(6f), which defines the feasible region of the PP associated with vehicle $k \in \mathbb{K}$, are the same as the inequalities (1d)–(1h) appearing in formulation (1) for each vehicle $k \in \mathbb{K}$. Thus, the disaggregate z-parity inequalities (2a) are also valid for model (6).

4.1.2. A branch-and-cut algorithm for the pricing problem

In Bianchessi, Corberán, Plana, Reula, & Sanchis (2021), the authors introduce the *Profitable Close Enough Arc Routing Problem* (PCEARP). Let $G = (V, A)$ be a directed and strongly connected graph with a cost $c_{ij} \ge 0$ associated with each arc $(i,j) \in A$ and a distinguished vertex 1 as the depot. Let $\mathbb{H}$ be the set of customers, each of them has an associated set of arcs $H_c \subseteq A$ in such a way a customer c is served when at least one of the arcs in H_c is traversed. Associated with each customer c there is a profit $p_c \ge 0$ that is collected (only once) if the customer is served. The PCEARP consists of finding a tour starting and ending at the depot and maximizing the difference between the total profit collected and the cost of the route. Therefore, for each vehicle $k \in \mathbb{K}$, the pricing problem can be seen as a PCEARP with the additional constraint (6g). In fact, it is possible to rewrite the objective function as a maximization problem with

- $p_c = \mu_c \ge 0$,
- $c_{ij} = -d_{ij}\rho_k \ge 0$.

Finally, it is worth observing that all the PPs share the same feasible region at the root node of the branch-and-bound tree. However, as will be explained in Section 4.2, branching rules may differentiate the pricing problem feasible regions in the subtree arising from their application.

We solve the pricing problem by using a branch-and-cut algorithm similar to the one described in Bianchessi et al. (2021) for solving the PCEARP.

When solving the pricing problem, it may be advantageous to save as many routes (columns) as we can find. Therefore, every time that the branch-and-cut algorithm finds an integer solution with negative reduced cost, we store it in order to add it to the restricted master problem. Furthermore, for each stored route, we study if it traverses any arc $a \in H_c$ associated with a customer c having $\mu_c = 0$. If this happens, we mark this customer as served by the route.

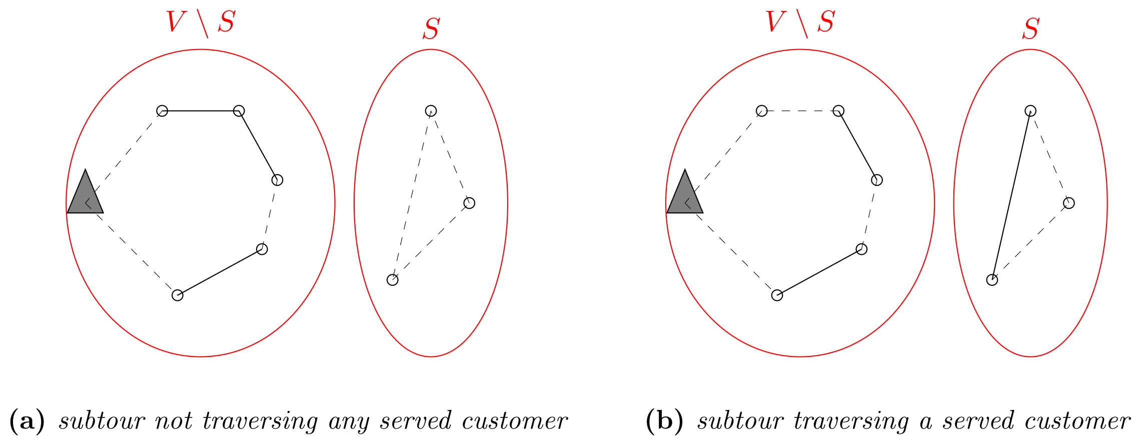

$V \setminus S$
S
(a) subtour not traversing any served customer
(b) subtour traversing a served customer

Fig. 1. Solutions of the PP with subtours that satisfy connectivity inequalities (6c).

Initial relaxation The initial relaxation considered in order to apply the BC includes constraints (6b)–(6g) (minus the integrality conditions). In particular, let S_c be the set of vertices incident with the arcs in H_c, $c \in \mathbb{H}$. The initial relaxation includes only connectivity constraints (6c) associated with sets S_c such that $1 \notin S_c$.

Moreover, in order to obtain routes useful for the LMP, inequalities (6h) and (6i) are also included in the initial relaxation. Inequality (6h) forces the vehicle to leave the depot, while inequality (6i) ensures that at least one customer will be served by the vehicle.

$$x^k(\delta^+(1)) \geq 1 \tag{6h}$$

$$\sum_{c=1,\ldots,L} z_c^k \geq 1. \tag{6i}$$

Separation algorithms In the branch-and-cut algorithm for the pricing problem we separate connectivity (6c) and parity (2a) inequalities. The separation algorithm used for parity inequalities is similar as the one described in Section 3.1 but without aggregating the solution.

For the connectivity inequalities we apply the first separation heuristic described in Section 3.1 with a modification that will be described in what follows.

Note that constraints (6c) do not guarantee that all solutions of the formulation will be connected, since there are still two situations in which disconnected subtours may appear. The first one is if $\rho_k = 0$. In this case, a solution can contain cycles with arcs that do not belong to any served customer (see Fig. 1a, where the triangle represents the depot and the solid lines represent arcs of a served customer), but these cycles can be removed without affecting the reduced cost of the solution. The other situation may occur when there is a cycle disconnected from the depot, but for any customer with $z_c^k = 1$ there is at least one arc in H_c traversed and connected to the depot (see Fig. 1b). But this solution will not be optimal, since this cycle can be removed from the solution while still servicing the same customers, thus decreasing the reduced cost of the route.

However, as will be explained in Section 4.2, the branching rules of the branch-and-price procedure may introduce some lower bound on the use of some arcs, that is, on some x_{ij}^k variables. If there is a disconnected subtour containing one of these arcs with a lower bound greater than 0, it is not possible to remove this cycle from the solution. For this reason, for each connected component C_i, if it does not contain the depot, we check if there is an arc incident with vertices in C_i having a lower bound greater than 0. If such an arc is found, the inequality $x^k(\delta(C_i)) \geq 1$ is a valid inequality that is violated by this solution.

Cutting-plane algorithm The cutting-plane algorithm applies the following separation algorithms in the order in which are listed:

1. Heuristic separation algorithm for connectivity inequalities with $\varepsilon = 0,\ 0.25,\ 0.5,\ 0.75$.
2. Heuristic separation algorithm for parity inequalities with $\varepsilon = 0,\ 0.25,\ 0.5$ (only at the root node).

The cutting-plane algorithm is applied at each node of the tree until no new violated inequalities are found. Again, we branch using the MIP-solver implementation of the strong branching strategy by giving higher priority to the z_c^k variables.

Primal heuristic To obtain a higher number of columns and good lower bounds that can help reducing the size of the branch-and-cut search tree, we have implemented a heuristic algorithm, whose pseudocode is shown in Algorithm 1 , using the fractional solutions

Algorithm 1: Primal heuristic.

```
   Input: $G$, $\mathbb{H}$, $(x^*, z^*)$
   Output: A feasible route $R$
1  $\mathcal{A} \leftarrow \emptyset$ ;
2  $\mathcal{H} \leftarrow \mathbb{H}$;
3  $A^1 \leftarrow \{a \in A_R :\ x_a^* \geq 0.9\}$;
4  $A^2 \leftarrow \{a \in A_R :\ 0.7 \leq x_a^* < 0.9\}$;
5  $A^3 \leftarrow \{a \in A_R :\ 0.5 \leq x_a^* < 0.7\}$;
6  for  i=1 to 3 do
7      $p(a) \leftarrow \sum_{c \in \mathcal{H}:\ a \in H_c} \mu_c \quad \forall a \in A^i$;
8      while $A^i \neq \emptyset$ AND $\max\{p(a) :\ a \in A^i\} > 0$ do
9          $\bar{a} \leftarrow argmax_{a \in A^i}\{p(a)\}$;
10         $\mathcal{A} \leftarrow \mathcal{A} \cup \{\bar{a}\}$;
11         $A^i \leftarrow A^i \setminus \{\bar{a}\}$;
12         Remove from $\mathcal{H}$ all the customers served by arc $\bar{a}$;
13         Recalculate $p(a)\ \forall a \in A^i$;
14 Apply an insertion heuristic to construct a route $R$ with the
   arcs in $\mathcal{A}$;
15 if the value of $R$ is better than the current lower bound of
   the branch-and-cut and $R$ satisfies (6g) then
16     Stop;
17 else
18     Solve the DGRP on $G$ with required arcs $\mathcal{A}$, the depot as
       required vertex, and costs $d_{ij}\rho_k$;
19     if the solution of DGRP satisfies (6g) then
20         $R \leftarrow$ Optimal solution of the DGRP;
```

of the LPs at the nodes of the tree.

Let (x^*, z^*) be a fractional solution. The subset of arcs $A_R = H_1 \cup \ldots \cup H_L$ is split into four different subsets according to their

x^* value. The first subset includes arcs a with $x^*_a \geq 0.9$. The second, those with $x^*_a \in [0.7, 0.9)$, the third the arcs with $x^*_a \in [0.5, 0.7)$, and the last subset includes the arcs $a \in A_R$ with $x^*_a < 0.5$, which will not be considered in the procedure.

We start by building a solution by iteratively selecting arcs from the first subset. For each arc a of this subset, we calculate the profit obtained from traversing it, given by the sum of the profit μ_c of the customers c not yet served such that $a \in H_c$. Then, the arc with the maximum profit is added to $\mathcal{A}$ and removed from its corresponding subset. Moreover, the customers served by traversing this arc are labeled as served. This procedure is repeated until the first subset is empty or there are no new customers that can be served traversing the remaining arcs. Then, we repeat the procedure with the second subset and, if necessary, with the third one.

Once the set $\mathcal{A}$ has been obtained, a route traversing this subset of arcs is built. The route is initialized by randomly selecting an arc in $\mathcal{A}$. Then, all the remaining arcs are allocated using a deterministic completion procedure. For each unassigned arc $a \in \mathcal{A}$, we compute the cost of inserting the arc in the route in the best possible position and add the one with the minimum insertion cost. We proceed until all the arcs in $\mathcal{A}$ are allocated. Once the route is complete, we check which customers are served. If the resulting solution improves the current lower bound, we stop. Otherwise, we solve a Directed General Routing Problem (DGRP) in which all the arcs in $\mathcal{A}$ are marked as “required” and the depot is a “required vertex”, using the exact procedure described in Ávila, Corberán, Plana, & Sanchis (2015). The DGRP consists of finding a minimum cost route that traverses all the required arcs and visits all the required nodes at least once. As before, we study the customers served by the obtained route and check if it improves the current lower bound.

This algorithm is executed at every 100 iterations of the cutting-plane procedure at the root node. Once the root node has been studied, it is executed once every 20 nodes up to node number 200, once every 50 nodes between nodes 201 and 501, and once every 200 nodes beyond that number.

4.1.3. Solution of the PPs

Let $\mathcal{K} = \{v_1, \dots, v_K\}$ be the set of vehicles sorted in non-ascending order with respect to their corresponding ρ_{v_k} values. At each column generation iteration, the PPs are considered sequentially starting from the problem associated with vehicle v_1. Let $\mathcal{R}^{v_k}$ be the set of routes found by solving the PP for vehicle $v_k \in \mathcal{K}$. For each route $r \in \mathcal{R}^{v_k}$, we check if it corresponds to a negative reduced cost $\lambda^{v_k r}$ variable (column), meaning that we check if its cost is less than θ_{v_k}. Additionally, we check if the route r corresponds to a negative reduced cost column for any of the other vehicles. In this way, as long as all the PPs share the same feasible region, we avoid to solve subsequent PPs corresponding to vehicles v_t, $t > k$, such that $|\rho_{v_k} - \rho_{v_t}| < \epsilon$, with $\epsilon \to 0^+$. This does not hold anymore once a branching rule which diversifies the pricing problem feasible regions is applied (see Section 4.2). When this happens, in each node of the subtree arising from the application of such a branching rule, all the PPs have to be eventually solved at each column generation iteration. The sequential solution of the PPs terminates as soon as negative reduced cost columns are found after solving one of them, or when all the PPs have been solved and no negative reduced cost column has been generated.

4.1.4. Heuristic column generation

In many iterations of the column generation algorithm (especially in the initial ones), there are many routes with negative reduced cost that can be efficiently found by means of an heuristic algorithm. Hence, at each column generation iteration, the solution mechanism outlined in Section 4.1.3 is first applied considering a Greedy Randomized Adaptive Search Procedure (GRASP) as solver for the PPs. In case no negative reduced cost column is generated by means of the heuristic, the mechanism is reapplied by solving the PPs to optimality through the BC algorithm described in Section 4.1.2. It is worth noting that in the construction and local search phases, we do not keep only the route with the lowest reduced cost, but also all those routes generated by the algorithm that have a negative reduced cost, which will be stored in set $\mathcal{R}_h$.

The heuristic, the pseudocode of which can be seen in Algorithm 2 , is initialized by computing an initial value $\nu(a)$ for

Algorithm 2: Heuristic column generation.

```
   Input: $t_l$, $m\_iter\_LS$, $\alpha$, $\eta$
   Output: $\mathcal{R}_h$
1  $iter \leftarrow 0$;
2  $\mathcal{R}_h \leftarrow \emptyset$;
3  $i \leftarrow 1$;
4  Generate set $\mathcal{A}_{ini}$;
5  while $t_l$ is not reached AND $i \leq |\mathcal{A}_{ini}|$ do
6      $a \leftarrow \mathcal{A}_{ini}(i)$;
7      $(\mathcal{R}_c, R_{iter}) \leftarrow$ Construction $(a, \alpha)$;
8      $\mathcal{R}_h \leftarrow \mathcal{R}_h \cup \mathcal{R}_c$;
9      $\mathcal{R}_{ls} \leftarrow$ IteratedLocalSearch $(R_{iter}, \eta, m\_iter\_LS)$;
10     $\mathcal{R}_h \leftarrow \mathcal{R}_h \cup \mathcal{R}_{ls}$;
11     $i \leftarrow i + 1$;
```

each arc $a \in A_R$ as the cost of going from the depot to the arc and back minus the profit of serving all customers traversed in this trip. Out of all these arcs, we build a subset $\mathcal{A}_{ini}$ with the $\min\{50, \frac{|A_R|}{2}\}$ arcs of minimum value $\nu(a)$, and we sort them randomly. At each iteration of the GRASP, a route is initialized by selecting an arc in $\mathcal{A}_{ini}$ and then, completed and improved, respectively, by using the *Construction* and *IteratedLocalSearch* subroutines described in the following.

Construction Initial solutions are built using an adaptation of the path-scanning procedure, which selects the subset of customers associated with the smallest $\nu(a)$. For each customer not yet assigned, the corresponding profit ψ_c is computed as the difference in the reduced cost of the route if c is served through the route. A restricted candidate list (RCL) is built including the customers candidates associated with the smallest profits and taking into account a threshold parameter α. Given $\psi_{\min}$ and $\psi_{\max}$, a customer is included in the RCL if

$$\psi_c \geq \alpha(\psi_{\max} - \psi_{\min}) + \psi_{\min}.$$

As usual, parameter α controls the greediness of the selection ($\alpha = 0$ pure greedy; $\alpha = 1$ pure random). Looking for a tiny randomization component within a greedy procedure, our algorithm has been implemented with a fixed $\alpha = 0.1$. *Construction* runs until it is verified that, with the remaining unassigned customers, it is not possible to achieve a route with negative reduced cost. Note also that all the negative reduced cost routes generated during the construction phase are kept as solutions of the current PP.

IteratedLocalSearch With the best solution built in the constructive phase (route R_{iter}), a local search procedure tries to improve it by exploring neighbor solutions. It consists of a Destroy and Repair method based on the one described in Corberán et al. (2019). Here, in the destruction phase of the algorithm, η ($\eta \in \{1, 3\}$) arcs are removed from the route but always keeping at least one arc in it. Then, in the repair phase, a best improvement strategy was adopted, according to which the customer associated with the minimum profit μ_c is inserted in the route. The reconstruction phase uses the same stopping criteria as *Construction*. Let ϕ_a be the set of arcs in A_R in the current route (R_{iter}), the *IteratedLocalSearch* procedure stops when $m\_iter\_LS = \min\{\frac{L}{2}, \frac{|\phi_a|}{2}\}$ iterations without improving the solution are performed.

GRASP is repeated until a maximum computing time (t_l = 2 seconds) is exceeded or a route for each arc in $\mathcal{A}_{ini}$ has been built.

4.1.5. Restricted master heuristic

In order to speed up the BP algorithm, and also improve the convergence speed of the column generation, we implemented a *restricted master heuristic* as defined in Joncour, Michel, Sadykov, Sverdlov, & Vanderbeck (2010). The basic idea behind restricted master heuristics is to solve, by means of a general mixed integer linear programming (MILP) solver, the MP (in our case model (3)) defined over a subset of the available columns. When the heuristic succeeds in solving the MP defined over a given set of columns, a feasible solution to the problem becomes available.

We run the restricted master heuristic every Δ column generation iterations and whenever an optimal solution for the RLMP has been computed.

The heuristic is immediately terminated when it is triggered and the current (optimal) solution is not feasible, i.e., the current base includes an active dummy column (see Section 4.1.6). Otherwise, let $\bar{R}$ be the set of the routes associated with the $\bar{\lambda}^{kr}$ variables that are active w.r.t. the current (optimal) feasible solution to the RLMP. The routes in $\bar{R}$ are used to initialize an integer program similar to (3) to be solved by means of a general MILP solver. In the new integer program, $\boldsymbol{\lambda}$ variables are no more indexed by vehicle. The program considers binary variables λ^r assuming value 1 if route r is selected to be assigned to one of the vehicles. According to the new definition of the $\boldsymbol{\lambda}$ variables, coefficients s_c^r and d^r play respectively the role of coefficients s_c^{kr} and d^{kr} in (3). The program reads as follows:

$$\bar{W} = \min \quad w \tag{7a}$$

$$s.t. \quad \sum_{r\in\bar{R}} s_c^r \lambda^r \geq 1 \qquad \forall c \in \mathbb{H} \tag{7b}$$

$$\sum_{r\in\bar{R}} \lambda^r \leq K \tag{7c}$$

$$d^r\lambda^r - w \leq 0 \qquad \forall r \in \bar{R} \tag{7d}$$

$$\lambda^r \in \{0, 1\} \qquad \forall r \in \bar{R} \tag{7e}$$

$$w \geq 0 \tag{7f}$$

The new $\boldsymbol{\lambda}$ variables allow to aggregate constraints (3c) in (3) and formulate them as (7c), to disregard constraints (3e), and, in general, to avoid symmetries in the solution space. This comes at the expense of an increase in the number of constraints (7d) required to define the maximum length.

Whenever an optimal solution to (7) is found, a new upper bound $\bar{W}$ (for w^1) becomes available. Hence, column generation is restarted to solve the RLMP by considering only vehicle routes with length smaller than, or equal to, $\bar{W} - 1$.

4.1.6. Overall algorithm overview

The RLMP is initialized by means of set of columns $\hat{\mathcal{C}} \cup \mathcal{C}$. Set $\hat{\mathcal{C}}$ includes a high cost dummy column for each customer $c \in \mathbb{H}$, by means of which constraints (3b) are satisfied. In particular, the column for a given customer $c \in \mathbb{H}$ has a coefficient 1 on the row corresponding to the constraint associated with the customer, whereas all the other coefficients of the column are 0. Similarly, a high cost dummy column is further included in $\hat{\mathcal{C}}$ for each vehicle $k \in \mathbb{K}$ to satisfy constraints (3c). All dummy columns have null length. At the root node of the branch-and-bound tree, $\mathcal{C}$ is empty. In any other node of the tree, $\mathcal{C}$ includes the columns that were in the optimal basis of the RLMP at the father node, and that correspond to routes that are feasible with respect to the active branching constraints and the current value of W.

Then, iteratively, the RLMP is solved to optimality or proved to be infeasible.

First, at each iteration, the RLMP is solved and the dual variable values retrieved. If no dummy column appears in the current basis, all columns in $\hat{\mathcal{C}}$ are extracted from (the constraint matrix of) the RLMP.

Let $\mathcal{PC}$ be the set of columns generated so far during the execution of the whole BP algorithm (the so called pool of columns). Columns in $\mathcal{PC}$ correspond to routes that are feasible with respect to the current value of W. Before solving the PPs, negative reduced cost columns are searched for among those included in $\mathcal{PC}$. $\mathcal{PC}$ is scanned sequentially and at most K negative reduced cost columns are selected to be inserted into the RLMP. A column in $\mathcal{PC}$ is eligible for selection if (i) the corresponding route is feasible with respect to the active branching constraints, (ii) none of the previous selected columns is associated with the same PP associated with it, and (iii) the corresponding route services at least one customer not serviced by any of the routes corresponding to previous selected columns. Let $\hat{\mathcal{PC}}$ be the subset of columns selected and extracted from $\mathcal{PC}$. If $|\hat{\mathcal{PC}}| > 0$, columns in $\hat{\mathcal{PC}}$ are inserted into the RLMP and a new iteration is started, otherwise the PPs are solved in order to eventually find new negative reduced cost columns.

An attempt is made to solve the PPs, as described in Section 4.1.3, by means of the MS-GRASP heuristic (Section 4.1.4). In case no negative reduced cost column is generated, the solution mechanism described in Section 4.1.3 is reapplied by solving the PPs to optimality through the BC algorithm (Section 4.1.2). If, again, no negative reduced cost column is found, it means that either the optimal solution of the current RLMP is also optimal for the LMP, or the LMP is infeasible (some dummy columns are in the basis corresponding to the dual variable values for the current iteration). In both cases the column generation algorithm terminates. Otherwise, the negative reduced cost columns generated are inserted into the RLMP and a new iteration is started. When the column generation algorithm ends, all the non-dummy columns defining RLMP are (re)inserted in $\mathcal{PC}$.

Finally, as mentioned in Section 4.1.5, the column generation algorithm takes advantage of a restricted master heuristic. This helps in speeding up its convergence as well as the convergence of the whole BP algorithm. Actually, the cardinality of the set $\bigcup_{k\in\mathbb{K}} R^k$ depends on the value of W. The lesser the value of W, the smaller the cardinality of the set, and this eventually allows to solve faster the PPs with both the MS-GRASP heuristic and the BC algorithm. In turn, the smaller the cardinality of the set $\bigcup_{k\in\mathbb{K}} R^k$, the greater the dual bound generated by solving to optimality the RLMP, and, in general, tighter dual bounds associated with the nodes of the tree imply a faster convergence of the BP algorithm. The restricted master heuristic is run every Δ column generation iterations, before solving the RLMP and retrieving the dual variable values, and whenever an optimal solution for the RLMP has been computed. Each time a new improving feasible solution is found, a new upper bound $\bar{W}$ (for w^1) becomes available. The value of W is updated accordingly. All the columns in the RLMP corresponding to routes that do not satisfy constraint (6g) with the updated value of W are removed from its constraint matrix. Similarly, columns in

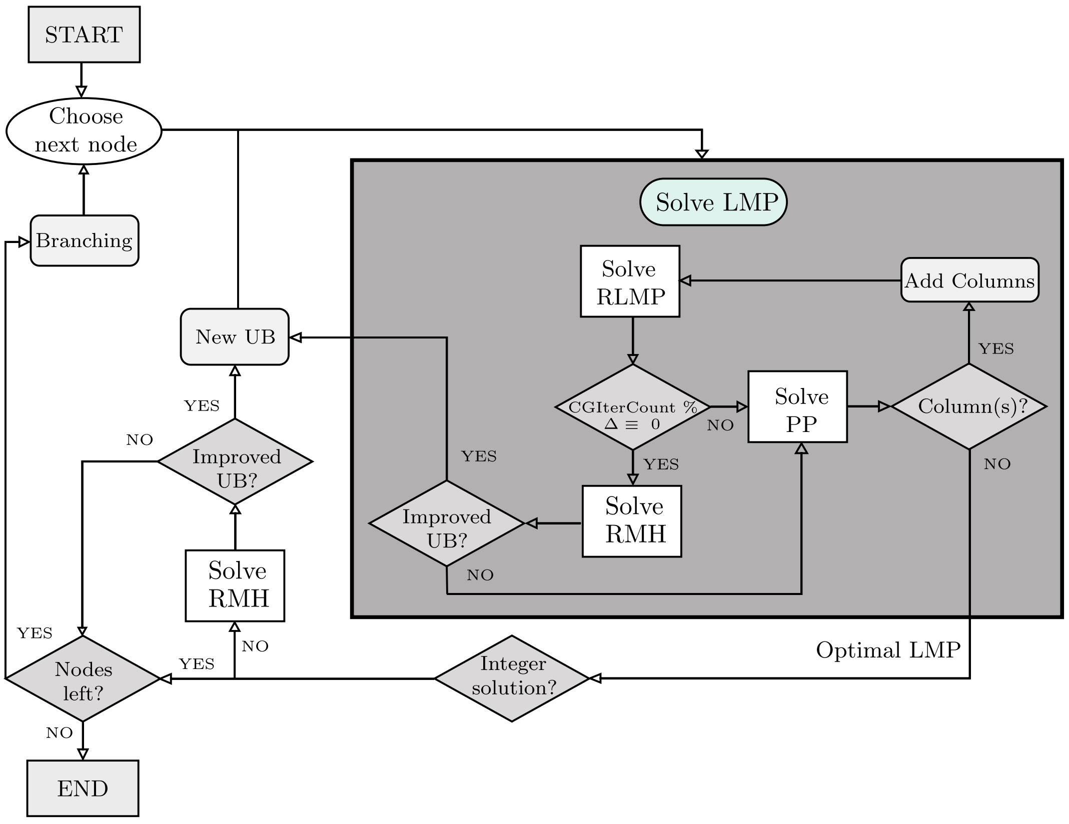


Fig. 2. Flow chart of the branch-and-price algorithm.

$\mathcal{PC}$ corresponding to routes which are no more feasible with respect to the updated value of W are deleted from the set. Columns in $\hat{\mathcal{C}}$ are reinserted into the RLMP. The solution process of the RLMP is then restarted. In particular, when the improving feasible solution is found after having computed the optimal solution for the current RLMP, the whole column generation algorithm is restarted.

Fig. 2 depicts the flow chart of the branch-and-price algorithm.

4.2. Branching rules

Let $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ be the optimal solution of the current RMP. When $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ is fractional, we apply a two-level hierarchical branching scheme. Branching rules are presented in the following in order of priority.

First, we consider an application of the Ryan and Foster's branching rule (see Ryan & Foster, 1981). For each pair of customers c' and c'', we define $\alpha_{c'c''} = \sum_{k\in\mathbb{K}}\sum_{r\in R^k} s^{kr}_{c'} s^{kr}_{c''} \lambda^{kr}$ as the sum of the λ^{kr} variable values associated with routes that serve both customers c' and c''. We select the fractional value $\alpha^*_{c'c''}$ closest to 0.5 such that $0 < \alpha^*_{c'c''} < 1$. On one branch, we set $\alpha^*_{c'c''} = 0$, meaning that the customers c' and c'' must be served in different routes (and hence by different vehicles). Whereas, on the other branch, we set $\alpha^*_{c'c''} = 1$, meaning that the two customers have to be served in the same route by the same vehicle. When $\alpha^*_{c'c''}$ is set to 0, the constraint $z^k_{c'} + z^k_{c''} \le 1$ is inserted in the formulation (6) associated with each vehicle $k \in \mathbb{K}$. For the case $\alpha^*_{c'c''} = 1$, the formulation (6) associated with each vehicle $k \in \mathbb{K}$ is modified by inserting constraint $z^k_{c'} - z^k_{c''} = 0$. This rule does not introduce symmetries in the solution space, does not alter the structure of the PPs, and, finally, allows the PPs to continue sharing the same feasible region. Thus, as long as only this rule is applied, at each column generation iteration it is possible to design the sequential solution of the PPs to potentially avoid solving some of them (see Section 4.1.3).

When the solution is fractional and no pair of customers c' and c'' exists such that $0 < \alpha^*_{c'c''} < 1$, we branch on the fractional use of an arc by vehicle $k \in \mathbb{K}$. For each $r \in R^k$, let b^{kr}_{ij} be an integer parameter equal to the number of times the vehicle k traverses arc (i, j) while traveling along route r. We consider values $\beta^k_{ij} = \sum_{r\in R^k} b^{kr}_{ij}\lambda^{kr}$ and select β^{k*}_{ij} such that $\beta^{k*}_{ij} - \lfloor\beta^{k*}_{ij}\rfloor$ is the closest to 0.5. On one branch, the formulation (6) associated with vehicle k is modified by considering an upper bound $\lfloor\beta^{k*}_{ij}\rfloor$ on the use of arc (i, j) and constraint $\sum_{r\in R^k} b^{kr}_{ij}\lambda^{kr} \le \lfloor\beta^{k*}_{ij}\rfloor$ is inserted in the LMP. Then, on the other branch, a lower bound $\lfloor\beta^{k*}_{ij}\rfloor + 1$ on the use of arc (i, j) is considered in the formulation (6) for vehicle k, and the additional constraint $\sum_{r\in R^k} b^{kr}_{ij}\lambda^{kr} \ge \lfloor\beta^{k*}_{ij}\rfloor + 1$ is inserted in the LMP. The new constraints inserted in the LMPs of the two branches give rise to additional dual variables, $\gamma^{UB^k}_{ij} \le 0$ and $\gamma^{LB^k}_{ij} \ge 0$, respectively, that have to be considered in the definition of the reduced cost of the routes in R^k. Let A^{UB^k} (A^{LB^k}) be the subset of arcs for which dual variables $\gamma^{UB^k}_{ij}$ ($\gamma^{LB^k}_{ij}$) exist. The objective function (6a) of the PP associated with vehicle k becomes:

$$\min\ -\sum_{c\in\mathbb{H}}\mu_c z^k_c - \sum_{(i,j)\in A} d_{ij}x^k_{ij}\rho_k - \theta_k - \sum_{(i,j)\in A^{UB^k}} \gamma^{UB^k}_{ij} x^k_{ij} - \sum_{(i,j)\in A^{LB^k}} \gamma^{LB^k}_{ij} x^k_{ij}$$

This branching rule is sufficient to guarantee the integrality of the solutions. In fact, integer flows on arcs for each vehicle guarantee that the $\boldsymbol{\lambda}$ variables are integer (see Barnhart et al., 1998). However, the application of this type of rule differentiates the pricing problem feasible regions. Thus, in each node of the subtree arising from their applications, all the PPs have to be eventually solved at each column generation iteration (see Section 4.1.3). This is also

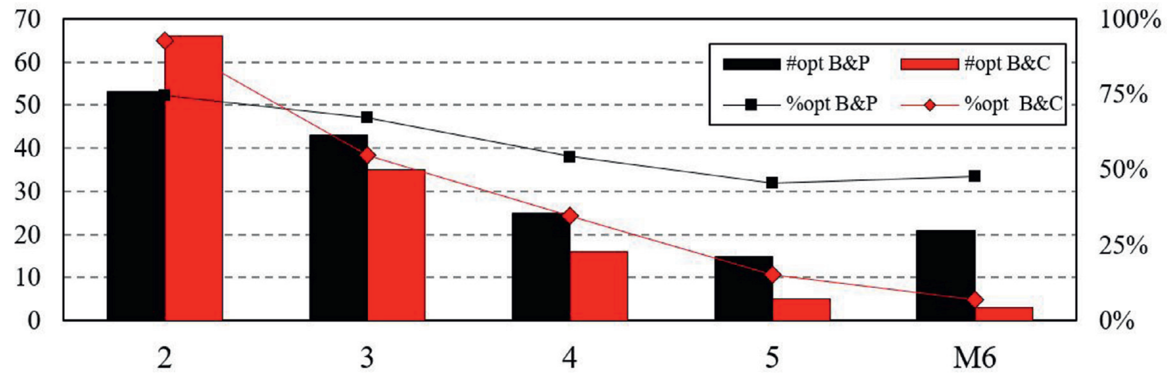


Fig. 3. Instances per vehicle solved optimally.

the reason why we decided from the beginning to index sets R^k by vehicle index (see Section 2.2).

The search tree is explored according to a best-first search strategy.

It is worth mentioning that, even if the optimal solution $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ is fractional, it may happen that at most K distinct routes are selected and fractionally assigned to the different vehicles. Since every λ^{kr} variable in the current RMP represents a route $r \in R^k$ which has to be feasible w.r.t. the current upper bound W for w^1 (i.e., $d^{kr} < W$), the optimal solution $(\overline{\boldsymbol{\lambda}}, \overline{\mathbf{w}})$ can be converted into an integer feasible solution to (3) whose value improves the current upper bound W. Whenever this happens, we convert the optimal fractional solution into the corresponding integer solution and update W accordingly.

5. Primal bound heuristic

In the branch-and-cut and branch-and-price algorithms described in the previous sections, we use the upper bounds provided by the following heuristic, which is based on the multi-start iterated local search matheuristic for the DC-CEARP described in Corberán et al. (2019). The pseudocode of this heuristic can be found in Algorithm 3 .

Algorithm 3: Primal bound heuristic.

```
Input: $G$, $\mathbb{H}$, $i_{\max}$, $t_l$
Output: $S_{best}$
1  $i \leftarrow 0$;
2  $S_{best} \leftarrow \emptyset$;
3  while $t_l$ is not reached AND $i \le i_{\max}$ do
4      for each initialization strategy do
5          $S \leftarrow \emptyset$;
6          Add the first customer to $S$ according to the initialization strategy;
7          for each unassigned customer $c \in \mathbb{H}$ do
8              Let $k_0 \in \mathbb{K}$ be the longest route of $S$;
9              Calculate the cost of inserting $c$ in all the possible routes in $\mathbb{K} \setminus \{k_0\}$ and insert $c$ in the cheapest position;
10         Apply ILS using 2-Exchange and Destroy-Repair to $S$;
11         if $S$ is better than $S_{best}$ then
12             $S_{best} \leftarrow S$;
13             $i \leftarrow 0$;
14     $i \leftarrow i + 1$;
```

In Corberán et al. (2019), three different criteria for initializing the routes are proposed: *random initialization*, *random selection among the best applicants*, and *weighted selection among the best applicants*. The *random initialization* criterion chooses the first customer of each route completely at random, *weighted selection among the best applicants* chooses the first customer randomly among a set of customers closest to the depot, and *weighted selection among the best applicants* assigns weights to the customers according to their distance to the depot and chooses the initial customer for each route with probability proportional to these weights. We generate one solution with each different initialization criterion.

Since the goal in the MM-CEARP is to minimize the length of the largest route, and there is no maximum length for the routes, we complete the routes by following the *parallel completion* strategy described in Corberán et al. (2019) with some modifications. Let k_0 be the longest route among those partially constructed. For all the arcs serving customers that have not been assigned yet, we calculate their insertion cost in all the possible positions of all the routes except k_0, and insert the arc according to the cheapest insertion. The customers served by this arc are marked as served. These steps are repeated until all the customers have been assigned to a route.

The constructed solutions are used as initial solutions of an Iterated Local Search (ILS) heuristic. The local search operators used are the exchange of pairs of arcs belonging to two different routes the routes (*2-Exchange*), and the reconstruction of partial solutions, *Destroy-Repair*. Both are well-known perturbation operators, and their implementation details can be found in Corberán et al. (2019).

New solutions are iteratively generated and improved using the above operators until a time limit t_l is reached or a certain number of iterations $i_{\max}$ are performed without improving the best solution. After some preliminary tests, we decided to set the maximum number of iterations ($i_{\max} = 10$) and the time limit ($t_l = 100$ seconds) in order to balance the amount of time and the quality of the solution.

6. Computational experiments

In this section we test the performance of the two exact algorithms proposed in this work. We have conducted all the computational experiments on a desktop PC with an Intel(R) Core(TM) i7 clocked at 3.4 GHz CPU, with 32 GBytes of RAM and running Windows 10 Enterprise 64 bits. The branch-and-cut algorithms described in Sections 3 and 4.1.2 have been implemented in C++ using IBM ILOG CPLEX 12.10 with Concert Technology, and compiled in release mode with MS Visual Studio Community 2015. Also the BP algorithm have been implemented in C++ and compiled in release mode with MS Visual Studio Community 2015. In particular, the callable library of CPLEX 12.10 was used for (re-)optimizing the

Table 2
Number of instances grouped by dataset and number of vehicles.

# Veh	Albaida	Madrigueras	Random50	Random75	TOTAL
2	24	24	11	12	**71**
3	21	22	9	12	**64**
4	17	17	3	9	**46**
5	9	16	1	7	**33**
6	6	12	–	4	**22**
7	3	7	–	3	**13**
8	1	4	–	1	**6**
9	–	1	–	–	**1**
10	–	1	–	–	**1**
11	–	1	–	–	**1**
	81	**105**	**24**	**48**	**258**

RLMPs. Finally, all the algorithms have been compiled by allowing a single thread of execution.

The experiments were carried out with a CPU time limit of two hours. For the BC algorithm described in Section 3, we turned off CPLEX heuristic algorithms and activated CPLEX own cuts (including zero-half cuts) in automatic mode. We fixed to zero the tolerance of the optimality gap and selected the best bound branching strategy.

The instances used and the computational results obtained are described in what follows.

6.1. Instances

In Ávila et al. (2017), four different data sets for the Distance-Constrained CEARP (DC-CEARP) were proposed. Two of them were based on the street networks of two Spanish towns, *Albaida* and *Madrigueras*, and the other were based random graphs with 50 and 75 vertices, *Random50* and *Random75* respectively. In total, 72 instances were defined. Moreover, by considering the number of vehicles allowed to serve the customers, ranging between 2 and 5, the total number of instances addressed was 251.

Contrary to what happens in the DC-CEARP, where there is a maximum distance for each route that determines the minimum number of vehicles needed, in the Min-Max CEARP there is no such limitation. Thus, the 72 DC-CEARP instances can be solved for any number of vehicles. However, depending on the characteristics of the instance, it may not make sense to use a very high number of vehicles. It may happen that a customer (or a set of customers) is very far from the depot, so the length of the longest route can be determined by the trip to serve this customer and in this case the optimal objective value will not decrease if we increase the number of vehicles. To address this issue, given an instance, we compute for each customer the length of the shortest route traveling from the depot to an arc of the customer, serving it and going back to the depot. Then, the longest of these routes provides a trivial lower bound for the MM-CEARP associated with the instance. Each instance is solved iteratively with $k = 2, 3, \dots$ vehicles. If for a given value of k, the optimal solution cost is equal to the trivial lower bound, we set $k-1$ as the maximum number of vehicles for this instance. A total of 258 instances have been defined with a minimum of 2 vehicles and a maximum of 11.

The characteristics of these instances, grouped by sets, are shown in Table 1. The number of instances per set is given in column *# Inst*, and the maximum number of vehicles for which the instances in this set have been solved in column *Max K*. The remaining columns report the minimum and maximum number of arcs, arcs in $A_R = H_1 \cup \dots \cup H_L$, arcs in $A_{NR} = A \setminus A_R$, and customers, respectively, for the instances in each set.

Table 2 summarizes the distribution of the instances according to the number of vehicles. Since the number of instances with more than 5 vehicles is very limited, in the analysis of the computational results outlined in the next section we grouped the 44 instances with 6 or more vehicles and denoted the group as M6.

All these instances, as well as their best known solutions, can be downloaded from Corberán, Plana, & Sanchis (2021b).

6.2. Computational results

This section presents the results of our computational experiments to compare the performance of the BC and BP algorithms.

Table 3 provides the results of each exact algorithm grouped by the number of vehicles of the instance. Note that the BP results are given only for the 230 out of the 258 instances where this method was able to solve the linear master problem at the root node and thus provide a lower bound. In this table, the columns “Gap 0” and “Time 0” show the average percentage gap at the end of the root node and its average computing time in seconds, while “Gap” and “Time” provide the same information regarding the final lower bound and the total time. All gaps are calculated with respect to the upper bounds computed by each algorithm. If *LB* and *UB* denote the lower and upper bounds found by a given algorithm in an instance, the gap is computed as $\frac{UB-LB}{UB} \times 100$. Column “Nodes” shows the average number of nodes of the enumeration tree.

The BC algorithm shows a fairly good performance in the instances with up to 4 vehicles, where it solves 117 out of the 181 instances to optimality and produces feasible solutions that are, in the worst case, 6.2% far from optimal on average in less than 5000 seconds. In particular, it achieves the best results for the instances with 2 vehicles, being able to solve 66 out of 71 instances in very short computing times. However, the performance of this algorithm degrades when the number of vehicles increases. This does not happen for the BP algorithm, which shows a more robust behavior and outperforms the BC algorithm for the instances with 3 or more vehicles. This can be clearly seen in the Fig. 3, which shows for the BC and BP algorithms the variation of the number and percentage of optima as a function of the number of vehicles.

In order to study the behavior of the algorithms w.r.t. computing time, we use the performance profiles described by Dolan & Moré (2002). Let $\mathcal{S}$ be the set of algorithms and $\mathcal{P}$ the set of instances. Let $t_{p,s}$ be the computing time required by algorithm $s \in \mathcal{S}$ to solve instance $p \in \mathcal{P}$. The *performance ratio* is then defined as $r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$. If algorithm s is not able to solve the instance p within the time limit, we set $r_{p,s} = \infty$. Then, the *performance profile* of each algorithm is defined as

$$\rho_s(\tau) = \frac{|\{p \in \mathcal{P} : r_{p,s} \le \tau\}|}{|\mathcal{P}|},$$

and represents the percentage of instances that can be solved by s within a factor τ of time with respect to the fastest algorithm. Note that, since the computing times are very different, we plot the results on a logarithmic scale. Note that $\rho_s(0)$ is the percentage of optimally solved instances for which algorithm s is the fastest. Fig. 4 depicts the performance profiles of the BC (red dotted line) and the BP (solid black line) algorithms overall and for the instances grouped by number of vehicles.

Comparing the performance profile for all the instances (Fig. 4a) in $\tau = 0$, we can see that the BP algorithm is the fastest in 42.6% of the instances, while the BC algorithm only in 24.41%. It is interesting how the curve for the BP algorithm increases rapidly in the interval [2,4], reaching almost its maximum around 60%. This means that the BP algorithm solves another 15% of instances using between 4 and 16 times the computing time used by the BC algorithm. Conversely, the curve of the BC algorithm shows that this last is slower at reaching optimal solutions. At $\log_2(\tau) = 11$ we have already reached the maximum number of optimal solutions for both algorithms.

Table 3
Computational results for all the instances grouped by number of vehicles .

	# Veh	# Inst	# Opt	Gap 0(%)	Time 0	Gap(%)	Time	Nodes
BC	2	71	66	8.9	9.1	0.3	839.8	3739.1
	3	64	35	14.6	6.3	2.4	3487.9	12277.4
	4	46	16	13.6	28.6	6.2	4993.8	6522.2
	5	33	5	12.9	37.5	8.0	6335.2	9263.9
	M6	44	3	10.2	207.9	7.3	6762.4	3832.4
	Overall	**258**	**125**	**11.9**	**49.4**	**4.0**	**3950.3**	**7075.9**
BP	2	59	53	0.5	507.6	0.4	1083.2	1.7
	3	57	43	1.8	476.3	1.1	2074.2	5.9
	4	41	25	2.3	642.0	1.8	3020.2	22.6
	5	31	15	4.0	502.5	3.5	3822.0	10.9
	M6	42	21	3.4	162.4	2.5	4216.7	27.4
	Overall	**230**	**157**	**2.2**	**460.1**	**1.7**	**2615.4**	**13.5**

Table 4
Gap comparison on the instances with and without LB computed by the BP algorithm.

	Instances with LB							Instances without LB			
		BC			BP				BC		
	# Inst	# Opt	Time	Gap(%)	# Opt	Time	Gap(%)	# Inst	# Opt	Time	Gap(%)
2	59	59	245.0	0.0	53	1081.6	0.2	12	7	3764.1	3.8
3	57	35	3032.0	1.0	43	2064.0	1.0	7	0	7200.0	9.7
4	41	16	4724.6	4.0	25	2996.6	1.8	5	0	7200.0	13.5
5	31	5	6279.3	5.6	15	3770.0	3.5	2	0	7200.0	17.9
M6	42	3	6741.5	4.7	21	4101.6	2.5	2	0	7200.0	15.6
Overall	**230**	**118**	**3733.9**	**2.6**	**157**	**2580.2**	**1.5**	**28**	**7**	**5727.5**	**10.5**

Table 5
Generated cuts and separation time of the BC.

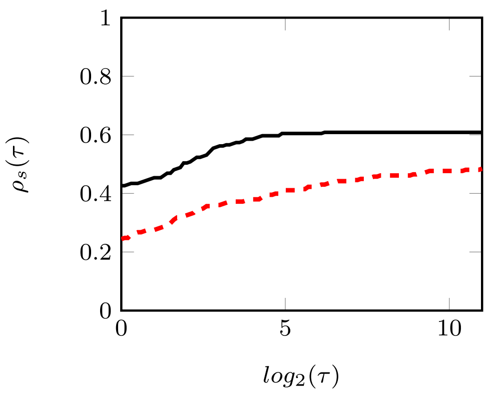
	# Cuts			Separation time		
# Veh	Connectivity	Parity	Total	Connectivity	Parity	Total
2	1866.2	92.7	1958.8	15.7	1.3	17.0
3	4307.9	100.2	4408.1	6.9	0.1	6.9
4	5173.2	107.7	5280.9	20.3	0.1	20.4
5	9155.2	132.1	9287.4	17.5	0.1	17.6
M6	11996.9	162.5	12159.5	24.4	0.2	24.5
Overall	**5721.5**	**114.2**	**5835.7**	**16.0**	**0.4**	**16.5**

Fig. 4 b shows the performance profile of both algorithms for the instances with 2 vehicles. The BC algorithm, as expected, is faster on this subset of instances, getting the shortest time in 77.46% of the cases, against a corresponding percentage of only 15.49% associated with the BP algorithm. However, the results are completely different at the increase of the number of vehicles. From Fig. 4c we can see that BP algorithm becomes the fastest in almost 60% of the instances while the BC algorithm only in just over 10%. Note that in this case, despite taking more time, the BC algorithm is able to find almost 55% of the optima. Fig. 4d–f show the performance profiles for 4, 5, and 6 or more vehicles, respectively. It can be seen that the BP algorithm completely dominates the comparison since, despite being below 60% in number of optima, it is the fastest algorithm in practically all the instances where the optimum is found, while the BC algorithm needs a lot more time to be able to obtain fewer optimal solutions (at most 3 optima for instances with 6 or more vehicles).

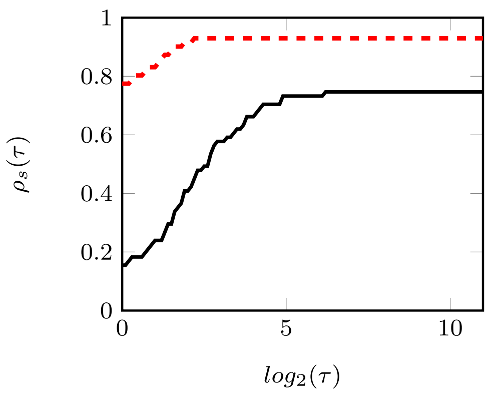
It is interesting to point out that, as mentioned before, in some instances the BP algorithm was unable to finish solving the linear master problem at the root node, so no lower bound has been computed by the algorithm for these instances. In Table 4, we first compare the gaps between the lower bounds computed by each algorithm with respect to the best known upper bound provided by any method for those instances for which the BP algorithm was able to find a lower bound.

The instances are grouped according to the number of vehicles. For each group, the second column gives the number of instances with a lower bound provided by both algorithms, columns three to five, and six to eight, report the number of optima found, the average computing time, and the average gap obtained by the BC and BP algorithms, respectively. Furthermore, the last four columns in Table 4 provide the results obtained with the BC algorithm for the instances in which no lower bound was computed by the BP algorithm in two hours of computing time. In particular, for each number of vehicles, columns 9–12 give the number of such instances, the number of those optimally solved, the average computing time, and the average gap for the unsolved instances obtained with the BC algorithm.

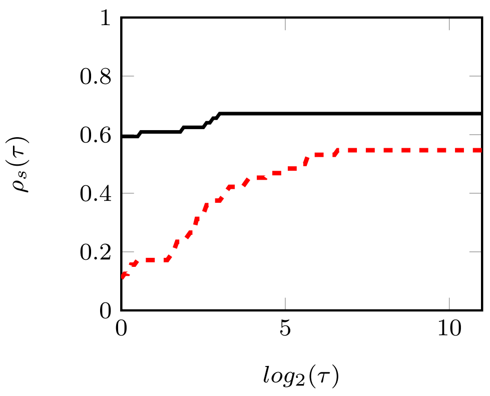
Again, we can see that the BC algorithm is effective on the instances with 2 vehicles, where all the 59 instances with LB in BP, and 7 out of 12 more instances, are optimally solved. Furthermore, the average gap for the 5 unsolved instances, 3.8%, is small. The effectiveness of the BC algorithm decreases for the instances with 3 vehicles, and it is inferior to that of the BP algorithm. Nevertheless, the BC algorithm is able to compute lower bounds, associated with an average gap of 9.7%, for the 7 instances with 3 vehicles for which the BP algorithm can not. For the instances with more



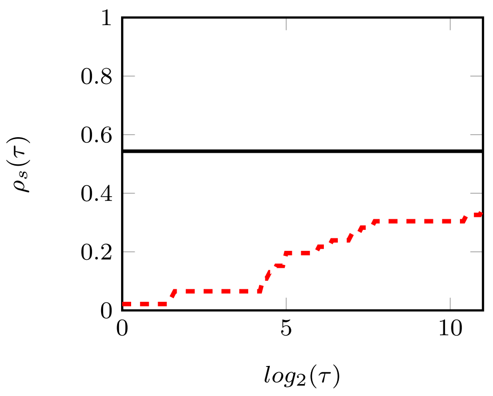
(a) *Performance profile - Overall*



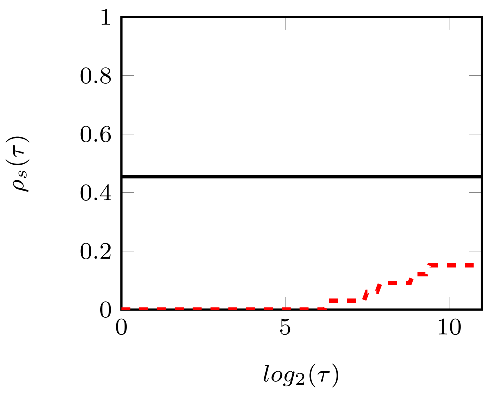
(b) *Performance profile - 2 vehicles*



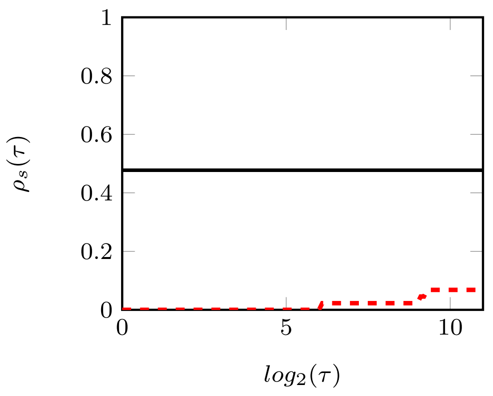
(c) *Performance profile - 3 vehicles*



(d) *Performance profile - 4 vehicles*



(e) *Performance profile - 5 vehicles*



(f) *Performance profile - M6 vehicles*

— Branch & Price   - - - Branch & Cut

Fig. 4. Performance profiles for different number of vehicles.

Table 6
Results of the heuristic algorithm.

# Veh	# Inst	# Opt	Time	Gap(%)
2	71	28	1.4	2.19
3	64	21	0.9	2.26
4	46	14	1.1	2.50
5	33	10	0.8	2.27
M6	44	6	1.1	2.30
Overall	**258**	**79**	**1.1**	**2.29**

vehicles, the performance of the BP algorithm is clearly superior to that of the BC algorithm. Note that with respect to the BC algorithm, both the number of unsolved instances and the average gap increase steadily as the number of vehicles increases, while this behavior is not so pronounced for the BP algorithm, whose reported average gaps do never exceed 3.5%. Moreover, the number of instances in which no lower bound is computed by the BP algorithm decreases as the number of vehicles grows. It is also worth noting that the gaps reported for the BC algorithm with respect to

Table 7
List of sets, parameters and variables.

$\mathbb{K} = \{1, \ldots, K\}$	set of vehicles
$\mathbb{H} = \{1, \ldots, L\}$	set of customers
$c \in \{1, \ldots, L\}$	a customer
$H_c \subseteq A$	set of arcs from which c can be served
$d_{ij} \geq 0$	length/distance associated with the traversal of arc $(i, j) \in A$
x_{ij}^k	number of times that the vehicle k traverses arc $(i, j) \in A$
z_c^k	takes value 1 if customer c is served by vehicle k
w	variable used to minimise the longest route
R^k	set of feasible routes for vehicle $k \in \mathbb{K}$
$\bar{R}$	$\cup R^k$
d^{kr}	length of the route $r \in R^k$
s_c^{kr}	parameter equal to 1 if the route $r \in R^k$ serves customer c
λ^{kr}	takes value 1 if the route $r \in R^k$ is assigned to the vehicle $k \in \mathbb{K}$
w^k	length of route assigned to vehicle $k \in \mathbb{K}$
$\mu_c \in \mathbb{R}^+$,	$\forall c \in \mathbb{H}$, dual variables associated with constraints (3b)
$\theta_k \in \mathbb{R}$,	$\forall k \in \mathbb{K}$, dual variables associated with constraints (3c)
$\rho_k \in \mathbb{R}^-$,	$\forall k \in \mathbb{K}$, dual variables associated with constraints (3d)
$\sigma_k \in \mathbb{R}^+$,	$\forall k = 1, \ldots, K-1$. dual variables associated with constraints (3e)
$\bar{c}^{kr}(\boldsymbol{\mu}, \boldsymbol{\theta}, \boldsymbol{\rho})$	reduced cost of route $r \in R^k$
$\mathcal{K} = \{v_1, \ldots, v_K\}$	set of vehicles sorted in non-ascending order for ρ_{v_k} values

these instances are very high, which seems to indicate that they are particularly difficult.

Finally, we present two tables with additional information regarding the performance of the heuristic separation procedures and the primal bound heuristic described in Section 5. Table 5 provides the average number of connectivity and parity cuts added by the BC in all instances grouped by number of vehicles, and the average computing time used to find them. As can be seen, the number of added violated connectivity cuts is very large and, as expected, increases very rapidly with the number of vehicles. The number of added parity inequalities is much smaller, but remember that its associated separation procedure is applied only at the root node, while the separation of connectivity inequalities is performed at every node of the tree search. Table 6 presents the number of optimal solutions found by the primal bound heuristic, the average computing time in seconds, and the average percentage gap respect to the best solution found by the exact procedures. The results show that this algorithm is robust and provides similar small gaps and computing times for all types of instances.

7. Conclusions

This paper addresses the Min-Max Close-Enough Arc Routing Problem, where a fleet of homogeneous vehicles has to serve a set of customers in such a way that the lengths of their routes are balanced. For this problem we have proposed two different models. The first one is an arc-based formulation, with arc and servicing variables, that has been used to develop a branch-and-cut algorithm, while the second one is a route-based set-covering formulation used to design a branch-and-price algorithm in which the pricing problems are solved by means of a branch-and-cut algorithm. Moreover, a heuristic to provide the exact algorithms with initial feasible solutions has been implemented. An extended computational analysis has been carried out, where we have studied the performance of the algorithms on 258 instances with up to 196 vertices, 544 arcs, 150 customers, and 11 vehicles. The results show that the branch-and-cut algorithm achieves the best results for the instances with 2 vehicles, while the performance of the branch-and-price algorithm is better for the instances with 3 or more vehicles. Overall, we have been able to optimally solve 174 out of these instances in two hours of computing time. The largest instance for which our algorithms have been able to find the optimal solution has 7 vehicles and 140 customers, which is a size that we think can be comparable to the size of real-life instances for some situations. If larger instances, such as the one reported in Shuttleworth et al. (2008), needed to be solved, the proposed heuristic algorithm, which presents a robust performance and short computing times for all instance sizes, could be applied.

As future lines of research, we plan on studying the Pricing Problem, because we think that it can be associated with a real-life problem in which not all the customers need to be served, but only those that are interesting from the economic point of view (that is, they provide some profit).

Acknowledgments

The work by Ángel Corberán, Isaac Plana, Miguel Reula, and José M. Sanchis was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades (MICIU) and Fondo Social Europeo (FSE) through project PGC2018-099428-B-I00. The authors want to thank the comments and suggestions done by three anonymous reviewers that have contributed to improve the content and readability of the article.

References

Ahr, D. (2004). *Contributions to multiple postmen problems*. University of Heidelberg, Germany Ph.D. thesis..
Ahr, D., & Reinelt, G. (2002). New heuristics and lower bounds for the min-max *K*-Chinese postman problem. In R. Möhring, & R. Raman (Eds.), *Algorithms-ESA 2002, 10th annual European symposium, Rome, Italy, September 2002. Proceedings, lecture notes in computer science: vol. 2461* (pp. 64–74). Springer.
Ahr, D., & Reinelt, G. (2006). A tabu search algorithm for the min-max *K*-Chinese postman problem. *Computers and Operations Research, 33*, 3404–3422.
Applegate, D., Cook, W., Dash, S., & Rohe, A. (2002). Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing, 14*, 97–189.
Aráoz, J., Fernández, E., & Franquesa, C. (2017). The generalized arc routing problem. *TOP, 25*, 497–525.
Ávila, T., Corberán, Á., Plana, I., & Sanchis, J. M. (2015). The stacker crane problem and the directed general routing problem. *Networks, 65*, 43–55.
Ávila, T., Corberán, Á., Plana, I., & Sanchis, J. M. (2016). A new branch-and-cut algorithm for the generalized directed rural postman problem. *Transportation Science, 50*, 750–761.
Ávila, T., Corberán, Á., Plana, I., & Sanchis, J. M. (2017). Formulations and exact algorithms for the distance-constrained generalized directed rural postman problem. *EURO Journal on Computational Optimization, 5*, 339–365.
Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*, 316–329.
Benavent, E., Corberán, A., Plana, I., & Sanchis, J. (2014). Arc routing problems with min-max objetives. In A. Corberán, & G. Laporte (Eds.), *Arc routing: Problems, methods and applications* (pp. 255–280). Philadelphia: MOS-SIAM Series on Optimization.
Bianchessi, N., Corberán, Á., Plana, I., Reula, M., & Sanchis, J. M. (2021). The profitable close enough arc routing problem. Submitted.

Cerrone, C., Cerulli, R., Golden, B., & Pentangelo, R. (2017). A flow formulation for the close-enough arc routing problem. In A. Sforza, & C. Sterle (Eds.), *Optimization and decision science: Methodologies and applications. ODS 2017.: vol. 217* (pp. 539–546). Springer Proceedings in Mathematics & Statistics.

Corberán, Á., Plana, I., Reula, M., & Sanchis, J. M. (2019). A matheuristic for the distance-constrained close-enough arc routing problem. *TOP, 27*, 312–326.

Corberán, Á., Plana, I., Reula, M., & Sanchis, J. M. (2021a). On the distance-constrained close enough arc routing problem. *European Journal of Operational Research, 291*, 32–51.

Corberán, Á., Plana, I., & Sanchis, J. (2021b). Arc routing problems: Data instances. http://www.uv.es/~corberan/instancias.htm,.

(2005). In G. Desaulniers, J. Desrosiers, & M. Solomon (Eds.), *Column generation*. New York: Springer.

Dolan, E., & Moré, J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*, 201–213.

Drexl, M. (2007). *On some generalized routing problems*. Rheinisch-Westfälische Technische Hochschule, Aachen University Ph.D. thesis..

Drexl, M. (2014). On the generalized directed rural postman problem. *Journal of the Operational Research Society, 65*, 1143–1154.

Duric, J. S., Jovanovic, S. Z., & Sibalija, T. (2018). Improving the efficiency of the warehouse storage process with the use of drones. *Advanced Quality, 46*, 46–51.

Eglese, R., Golden, B., & Wasil, E. (2014). Route optimization for meter reading and salt spreading. In A. Corberán, & G. Laporte (Eds.), *Arc routing: Problems, methods and applications* (pp. 303–320). Philadelphia: MOS-SIAM Series on Optimization.

Frederickson, G., Hecht, M., & Kim, C. (1978). Aproximation algorithms for some routing problems. *Journal on Computing, 7*, 178–193.

Gulczynski, D., Heath, J., & Price, C. (2006). The close enough traveling salesman problem: A discussion of several heuristics. In *Perspectives in operations research*. In *Operations research/computer science interfaces series: vol. 36* (pp. 217–283). Springer.

Hà, M.-H., Bostel, N., Langevin, A., & Rousseau, L.-M. (2012). An exact algorithm for close enough traveling salesman problem. In *Proceedings of the 1st international conference on operations research and enterprise systems* (pp. 233–238).

Hà, M.-H., Bostel, N., Langevin, A., & Rousseau, L.-M. (2014). Solving the close enough arc routing problem. *Networks, 63*, 107–118.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., & Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics, 36*, 695–702.

Renaud, A., Absi, N., & Feillet, D. (2017). The stochastic close-enough arc routing problem. *Networks, 69*, 205–221.

Ryan, D., & Foster, B. (1981). An integer programming approach to scheduling. *Computer Scheduling of Public Transport, 1*, 269–280.

Shuttleworth, R., Golden, B., Smith, S., & Wasil, E. (2008). Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Lastest advances and new challenges* (pp. 487–501). Springer.