Discrete Optimization 12 (2014) 129–146

Contents lists available at ScienceDirect

Discrete Optimization

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

A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints

Claudio Contardo [a,*], Rafael Martinelli [b]

[a] Département de management et technologie, ESG UQÀM and GERAD, Canada

[b] Departamento de Computação, Universidade Federal de Ouro Preto, Brazil

ARTICLE INFO

Article history:
Received 4 October 2012
Received in revised form 15 December 2013
Accepted 14 March 2014
Available online 31 March 2014

Keywords:
Multi-depot vehicle routing problem
Capacitated vehicle routing problem
Column generation
Exact algorithm

ABSTRACT

This article presents an exact algorithm for the multi-depot vehicle routing problem (MDVRP) under capacity and route length constraints. The MDVRP is formulated using a vehicle-flow and a set-partitioning formulation, both of which are exploited at different stages of the algorithm. The lower bound computed with the vehicle-flow formulation is used to eliminate non-promising edges, thus reducing the complexity of the pricing sub-problem used to solve the set-partitioning formulation. Several classes of valid inequalities are added to strengthen both formulations, including a new family of valid inequalities used to forbid cycles of an arbitrary length. To validate our approach, we also consider the capacitated vehicle routing problem (CVRP) as a particular case of the MDVRP, and conduct extensive computational experiments on several instances from the literature to show its effectiveness. The computational results show that the proposed algorithm is competitive against state-of-the-art methods for these two classes of vehicle routing problems, and is able to solve to optimality some previously open instances. Moreover, for the instances that cannot be solved by the proposed algorithm, the final lower bounds prove stronger than those obtained by earlier methods.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The multi-depot vehicle routing problem (MDVRP) is an important class of vehicle routing problem arising in freight distribution and can be defined as follows. We are given a set of depot locations $\mathcal{D}$ and a set of customer locations $\mathcal{C}$, which are assumed to be disjoint (even if two points share the same physical coordinates, they are still handled as different entities). With every customer $j \in \mathcal{C}$ is associated a demand d_j and a service time s_j. We consider a graph $G = (V, E)$ with $V = \mathcal{D} \cup \mathcal{C}$ and $E = \{\{i, j\} : i, j \in V, i \text{ and } j \text{ not both in } \mathcal{D}\}$. With every edge $e \in E$ is associated a traveling cost c_e. With every depot location $i \in \mathcal{D}$ is associated a fleet of size m_i. The fleet is assumed to be homogeneous with all vehicles having the same capacity Q and having to respect a maximum route length of T units. The route length of a route is the sum of the traveling costs along the edges used by the vehicle plus the service times of each visited customer. The objective is to select a subset of vehicles and to construct routes that respect the capacity and route length constraints, so as to visit each customer exactly once, at minimum traveling cost. The capacitated vehicle routing problem (CVRP) is a particular case of the MDVRP in which $|\mathcal{D}| = 1$, the number of vehicles to be used is exactly m and the maximum route length constraint is relaxed.

* Corresponding author.
E-mail addresses: contardo.claudio@uqam.ca, claudio.contardo@gerad.ca (C. Contardo), rafael.martinelli@iceb.ufop.br (R. Martinelli).

http://dx.doi.org/10.1016/j.disopt.2014.03.001
1572-5286/© 2014 Elsevier B.V. All rights reserved.

Both problems are $\mathcal{NP}$-hard since they are generalizations of the traveling salesman problem, therefore polynomial-time algorithms are unlikely to exist unless $\mathcal{P} = \mathcal{NP}$ [1]. Despite the computational complexity of these problems, state-of-the-art heuristic methods can find near-optimal solutions in a matter of seconds [2,3].

The literature on exact approaches for the MDVRP is sparse. In fact, most authors have focused on the development of heuristic methods to find good quality solutions quickly [4,2,3]. The most recent exact method reporting results on the MDVRP is that of Baldacci and Mingozzi [5]. The method is based on the additive bounding procedure of Christofides et al. [6] applied to several different relaxations of the problem. Ultimately, the set-partitioning formulation of the MDVRP is solved by means of column generation strengthened with the so-called strong capacity constraints and clique inequalities. They do not consider the inclusion of the route length constraint and so experiments are conducted only on those instances without such requirement. A problem closely related to the MDVRP is the periodic VRP (PVRP), in which the complete planning horizon is subdivided in periods, and vehicle routes cannot be longer than the length of one period. The MDVRP can be formulated as a PVRP by realizing that different depots can be modeled as multiple periods in the context of a PVRP. Therefore, any algorithm that solves the PVRP can also solve the MDVRP. Baldacci et al. [7] proposed an exact algorithm for the PVRP that generalizes their former method for the MDVRP but, as remarked by the authors, does not improve upon their previous results.

With respect to the CVRP, the literature on exact methods is broader. The most efficient exact algorithms for the CVRP are those of Lysgaard et al. [8], Fukasawa et al. [9], Baldacci et al. [10,11]. Lysgaard et al. [8] developed the most efficient branch-and-cut algorithm for the CVRP. They consider a compact two-index vehicle-flow formulation of the problem and use several classes of valid inequalities for which they provide novel and efficient separation algorithms. Their method is able to consistently solve problems with up to 50 customers. Fukasawa et al. [9] developed the first branch-and-cut-and-price algorithm for the CVRP. They consider a set partitioning formulation in which variables represent vehicle routes satisfying the capacity constraint. The routes are allowed to contain cycles, and the pricing sub-problem can be solved efficiently by using dynamic programming. This column generation approach is embedded into a branch-and-cut framework, using several of the valid inequalities introduced in [8]. Their computational results show that their method can scale and solve instances with twice as many customers as the previous exact method of Lysgaard et al. [8]. Baldacci et al. [10,11] introduced a new approach to solve the CVRP based on the additive bounding technique introduced by Christofides et al. [6] on which several different relaxations of the problem are solved sequentially in an additive manner. At last, the set-partitioning formulation of the problem is strengthened with strong capacity cuts and clique inequalities, and solved by column generation. The last part of the algorithms reduces to solve a pure integer linear problem with a reduced number of variables. The methods of Baldacci et al. [10,11] differ mainly in the SPPRC relaxation used. While the former relies on elementary routes, the latter relies now in the so-called *ng*-routes relaxation, a new pricing sub-problem that uses neighborhood structures to forbid cycles visiting nodes that are close to each other. While the method of Baldacci et al. [10] drastically reduced the computing times with respect to the former method of Fukasawa et al. [9], now the method based on the *ng*-routes relaxation proves even faster and is able to solve some instances that the previous method did not.

With respect to algorithmic enhancements for column generation methods for vehicle routing problems, they can be categorized into two main branches. On the one hand, the development of new pricing sub-problems aimed to achieve the so-called elementary bound (the lower bound obtained when the set of feasible routes is restricted to those not containing cycles) without solving the elementary shortest path problem with resource constraints (ESPPRC) at every iteration, which is known to be strongly $\mathcal{NP}$-hard [12]. On the other hand, the development of new families of valid inequalities aimed to strengthen the linear relaxations of the set-partitioning formulations of these problems.

In the first category, authors have focused in the development of relaxations of the original ESPPRC to consider routes with cycles. The shortest path problem under resource constraints and without cycles of length one or two (2-*cyc*-SPPRC), introduced by Houck et al. [13] and later used by Desrochers et al. [14] in the context of the vehicle routing problem with time windows (VRPTW) is an example of such relaxation. In the 2-*cyc*-SPPRC, routes are allowed to contain cycles as long as these cycles do not visit the same node twice with a separation of one intermediate node. The shortest path with resource constraints and without cycles of length k for $k \geq 3$ introduced by Irnich and Villeneuve [15] (k-cyc-SPPRC) is another example, in which routes are now allowed to contain cycles as long as no route visits a customer twice with less than k customers of separation. The decremental state-space relaxation (DSSR) introduced by Righini and Salani [16] achieves the elementary bound by first relaxing the elementarity constraint, and iteratively imposing elementarity on the nodes with cycles, until no more cycles are detected. The *ng*-routes relaxation (*ng*-SPPRC) introduced by Baldacci et al. [11] is a very efficient relaxation that achieves near-elementary routes in very short computing times. The intuition behind the *ng*-routes relaxation is as follows. During a labeling algorithm, cycles are allowed as long as they do not visit a set of forbidden nodes, which is constructed from predefined sets of neighbors of every node, and that intuitively forces cycles to contain nodes that are far from each other. The set of forbidden nodes is updated after every extension of the labels and is made in such a way that the efficiency of the dynamic programming algorithm is not compromised. As a result, very strong lower bounds can be obtained that in many cases coincide with the elementary bounds. Contardo et al. [17] introduced the so-called strong degree constraints (SDC), a new family of valid inequalities that are proved to impose partial elementarity. The addition of a strong degree constraint associated to a certain customer imposes that no variable associated to a route having a cycle on that node will be basic in the LP relaxation. The dual variable associated to such constraint can be used to derive a sharper rule than that of classic elementarity. More recently, Contardo et al. [18] have performed a computational study to assess the

efficiency of several strategies aimed at reaching the elementary lower bound in the context of the vehicle routing problem with time windows (VRPTW).

With respect to the development of new valid inequalities for the CVRP and the MDVRP, Fukasawa et al. [9] were the first to adapt the inequalities used by Lysgaard et al. [8] for the CVRP to the set-partitioning formulation. They showed that these inequalities can be included in the problem without breaking the structure of the pricing sub-problem. They used capacity constraints (CC), strengthened comb inequalities (SCI) and framed capacity inequalities (FCI) in their algorithm. Baldacci et al. [10] introduced the so-called strong capacity constraints (SCC), by realizing that the original capacity constraints could be lifted in the particular case of the set-partitioning formulation, yielding much stronger bounds. The addition of a SCC, however, imposes the addition of an extra resource in the labeling algorithm and therefore makes it harder. Jepsen et al. [19] introduced the subset-row inequalities (SRI), a particular family of rank-1 Chvátal–Gomory cuts [20]. They showed that these inequalities can be efficiently separated and included in the pricing sub-problems without compromising the overall performance of the algorithm. Contardo et al. [17] introduced the y-strong capacity constraints (y-SCC) and the strong framed capacity inequalities (SFCI) in the context of the capacitated location-routing problem (CLRP). These two inequalities are shown to dominate both the SCC and the FCI, respectively.

In this paper, we propose a new exact method for the MDVRP based on the solution of ad-hoc vehicle-flow and set-partitioning formulations strengthened with several classes of valid inequalities. The set-partitioning formulation is solved by column-and-cut generation, for which the column generation sub-problem is solved using the ng-SPPRC coupled with 2-cycle elimination, and we use the SDC to impose partial elementarity. Because the SDC may be hard to handle when their associated dual variables are large, we complement their use with the inclusion of a new family of valid inequalities to forbid cycles of an arbitrary length. As a result, our method is able to produce tight lower bounds and to solve to optimality some hard instances of the MDVRP and the CVRP in moderate computing times, including some previously unsolved instances. Moreover, for the instances that remain unsolved, our algorithm provides tighter lower bounds than existing methods.

The remaining of the article is organized as follows. In Section 2 we introduce the two formulations for the MDVRP used in this paper, namely a compact two-index vehicle-flow formulation and a set-partitioning formulation. In Section 3 we present the valid inequalities used in our method. In Section 4 we present the pricing algorithm used to solve the linear relaxation of the set-partitioning formulation of the problem. In Section 5 we present the exact method that solves the MDVRP and the CVRP to optimality. In Section 6 we report our computational results on selected instances from the literature for the CVRP and the MDVRP. We conclude the article in Section 7.

2. Mathematical formulations

In this section, we present the two formulations used in this article to model and solve the MDVRP, namely a vehicle-flow and a set-partitioning formulation. These two formulations of the problem are exploited at different stages of the algorithm. More precisely, the vehicle-flow formulation is used to derive a lower bound quickly and perform variable fixing. The set-partitioning formulation is then considered using the reduced network produced by the variable fixing procedure performed before.

2.1. Vehicle-flow formulation of the MDVRP

For every depot $i \in \mathcal{D}$ and customer $j \in \mathcal{C}$, let y_{ij} be a binary variable equal to 1 iff j is served by a single-vehicle route departing from depot i. For every $e \in E$, we let x_e be a binary variable equal to 1 iff edge e is used by a vehicle route visiting at least two customers. For a customer set $S \subseteq \mathcal{C}$, we let $r(S) = \lceil d(S)/Q \rceil$ be a lower bound on the number of vehicles needed to serve the customers in S due to the capacity constraint, and we let $\rho(S)$ be a lower bound on the number of vehicles needed to serve the customers in S due to the route length constraint. Note that while $r(S)$ can be computed in constant time, $\rho(S)$ can be difficult to compute as it involves the solution of a m-TSP with route length constraints, which is strongly $\mathcal{NP}$-hard. For a node subset $U \subset V$ we let $\delta(U)$ be the cutset of U, or equivalently the subset of edges with exactly one extremity in U. For an edge subset $F \subseteq E$ we also define $x(F) = \sum_{e \in F} x_e$, and if $F \subseteq \delta(\mathcal{D})$, $y(F) = \sum_{e \in F} y_e$. The vehicle-flow formulation of the MDVRP is as follows:

$$\min \sum_{e \in E} c_e x_e + 2 \sum_{i \in \mathcal{D}, j \in \mathcal{C}} c_{ij} y_{ij} \tag{1}$$

$$x(\delta\{j\}) + 2y(\delta(\{j\}) \cap \delta(\mathcal{D})) = 2 \quad j \in \mathcal{C} \tag{2}$$

$$x(\delta\{i\}) + 2y(\delta(\{i\})) \leq 2m_i \quad i \in \mathcal{D} \tag{3}$$

$$x(\delta\{S\}) + 2y(\delta(S) \cap \delta(\mathcal{D})) \geq 2\max\{r(S), \rho(S)\} \quad S \subseteq \mathcal{C} \tag{4}$$

$$x(\delta(S)) \geq 2[x(\delta(\{h\}) \cap \delta(\mathcal{D}')) + x(\delta(\{j\}) \cap \delta(\mathcal{D} \setminus \mathcal{D}'))] \quad S \subseteq \mathcal{C}, h, j \in S \quad \mathcal{D}' \subset \mathcal{D} \tag{5}$$

$$y_e \in \{0, 1\} \quad e \in \delta(\mathcal{D}) \tag{6}$$

$$x_e \in \{0, 1\} \quad e \in E. \tag{7}$$

The objective function aims to minimize the total traveling time. Constraints (2) are the degree constraints that impose each customer be visited exactly once. Constraints (3) is the fleet size constraint. They impose that at most m_i vehicles are used at each depot. Constraints (4) are the capacity and route length constraints. They impose that at least $\max\{r(S), \rho(S)\}$ vehicles are used to visit the customers of set S. Constraints (5) are the path constraints. They forbid routes to have the starting and ending points at two different depots. Finally, (6)–(7) impose that variables are indeed binary. This formulation is a particular case of the one introduced by Belenguer et al. [21] for the CLRP, with the only addition of the route length constraint represented by the constants ρ.

2.2. Set-partitioning formulation of the MDVRP

For each $i \in \mathcal{D}$ and $j \in \mathcal{C}$ we let y_{ij} be a binary variable equal to 1 iff customer j is served alone in a route departing from depot i, with cost equal to $2c_{ij}$. We now let Ω be the set of routes visiting at least two customers and respecting the capacity and route length constraints. For a depot $i \in \mathcal{D}$, we denote by Ω_i the subset of routes starting and ending i. For each $l \in \Omega$, we let θ_l be a binary variable equal to 1 iff route l is selected, and we denote by c_l its cost, which is equal to the sum of the traveling costs along the edges used by l. For each customer $j \in \mathcal{C}$ and route $l \in \Omega$ we let a_j^l be the number of times that customer j is visited by l. For each depot subset $D \subseteq \mathcal{D}$ and customer subset $C \subseteq \mathcal{C}$ we let $y(D : C) = \sum_{i \in D} \sum_{j \in C} y_{ij}$. The set-partitioning formulation of the MDVRP is as follows:

$$\min \sum_{l \in \Omega} c_l \theta_l + 2 \sum_{i \in \mathcal{D}, j \in \mathcal{C}} c_{ij} y_{ij} \tag{8}$$

$$\sum_{l \in \Omega} a_j^l \theta_l + y(\mathcal{D} : \{j\}) = 1 \quad j \in \mathcal{C} \tag{9}$$

$$\sum_{l \in \Omega_i} \theta_l + y(\{i\} : \mathcal{C}) \leq m_i \quad i \in \mathcal{D} \tag{10}$$

$$y_{ij} \in \{0, 1\} \quad \{i, j\} \in \delta(\mathcal{D}) \tag{11}$$

$$\theta_l \in \{0, 1\} \quad l \in \Omega. \tag{12}$$

Once again, the objective function aims to minimize the total traveling cost. Constraints (9) are the degree constraints that impose each customer be visited exactly once. Constraints (11)–(12) state the binary nature of the variables. Note that the capacity and the route length constraints are embedded into the definition of the route set Ω, and therefore do not need to be explicitly included in the formulation of the problem. For the same reason, the path constraints (5) are also not needed.

3. Valid inequalities

In this section we present the valid inequalities used to strengthen both formulations presented earlier. For the first families of inequalities, we refer to them as *robust* with respect to the pricing sub-problem because their inclusion does not impose the addition of extra resources in the labeling algorithm. For the last five classes of inequalities introduced, we refer to them as *non-robust*, because their addition imposes the use of additional resources during the labeling algorithm.

3.1. Robust valid inequalities

We call robust valid inequalities all those inequalities that are valid for formulation (1)–(7) and that can be used in formulation (8)–(12) by using the following identities that link vehicle-flow variables from the first formulation and route variables from the second formulation:

$$x_e = \sum_{l \in \Omega^m} \nu_l^e \theta_l \quad e \in E \tag{13}$$

$$y_e = \sum_{l \in \Omega^s} \nu_l^e \theta_l \quad e \in \delta(\mathcal{D}) \tag{14}$$

where ν_e^l is equal to the number of times that route l uses edge $e \in E$ along its path (with $\nu_e^l = 1$ for single-customer routes using edge e twice), $\Omega^m \subseteq \Omega$ is the set of routes servicing two or more customers, and $\Omega^s \subseteq \Omega$ is the subset of routes servicing a single customer.

Robust inequalities have the particularity that the contribution of their duals to the computation of the reduced costs of paths can be decomposed along the edges defining them, and therefore included in the pricing algorithm without compromising its performance. We consider some of the valid inequalities available in the CVRPSEP package [22], namely the framed capacity inequalities, strengthened comb inequalities, multistar inequalities and hypotour inequalities. We also use some of the inequalities introduced by Belenguer et al. [21], Contardo et al. [23], namely the y-capacity cuts, degree constraints and co-circuit inequalities using the separation algorithms of Contardo et al. [23].

Fig. 1. Route l with $a_1^l = 4$, $\nu_1^{1l} = 4$, $\nu_1^{2l} = 2$ and $\nu_1^{3l} = 1$.

3.2. Strong degree constraints

The strong degree constraints (SDC) were originally introduced by Contardo et al. [17] for the CLRP, and they are also valid for the CVRP and the MDVRP. Before presenting the inequality, let us define some notation. Given a customer $j \in \mathcal{C}$ and a route $l \in \Omega$, we let ξ_j^l be a binary constant equal to 1 iff route l visits node j. For a given customer $j \in \mathcal{C}$ we define $\xi^\theta(j) = \sum_{l\in\Omega} \xi_j^l \theta_l$. The SDC associated to node j is

$$\xi^\theta(j) + y(\mathcal{D} : \{j\}) \geq 1. \tag{15}$$

Contardo et al. [17] proved that this constraint imposes partial elementarity on node j, that is, no variable θ_l visiting node j twice or more will take a positive value in the solution of the linear relaxation of the set-partitioning formulation.

3.3. k-cycle elimination constraints

We now introduce a new family of valid inequalities that can be seen as a weaker form of the strong degree constraints. Let $k \geq 2$ be an integer constant. Let $j \in \mathcal{C}$ be a customer and let $l \in \Omega$ be a route. Let us define ν_j^{kl} as the number of times that route l visits customer j with at least k nodes between two consecutive appearances of j in the route. In Fig. 1 we illustrate by means of an example the behavior of the values ν_j^{kl} as k increases. The following k-cycle elimination constraint (k-CEC) is valid for the MDVRP:

$$\sum_{l\in\Omega} \nu_j^{kl}\theta_l + y(\mathcal{D} : \{j\}) \geq 1. \tag{16}$$

Because $\xi_j^l \leq \nu_j^{kl} \leq a_j^l$ it follows that the k-CEC is a strengthening of the degree constraint (9) but a weaker form of the SDC (15). The following proposition shows that the k-CEC for a given length k and customer j is effective to forbid a cycle of length k or less from visiting customer j twice.

Theorem 3.1. *Suppose that a k-CEC has been added to problem (8)–(12) for customer j and cycle length k. All routes $l \in \Omega$ visiting node j twice or more and such that $\nu_j^{kl} < a_j^l$ will be non-basic in the linear relaxation of problem (8)–(12). In particular, no route visiting customer j two consecutive times with $k-1$ or less intermediate nodes will take a positive value.*

Proof. If we consider the k-CEC and subtract from it the regular degree constraint (9) we obtain

$$\sum_{l\in\Omega} \left(\nu_j^{kl} - a_j^l\right)\theta_l \geq 0.$$

The summation above only has interest for those l such that $\nu_j^{kl} < a_i^l$. In that case, it becomes

$$\sum_{l\in\Omega,\, \nu_j^{kl} < a_j^l} \left(\nu_j^{kl} - a_j^l\right)\theta_l \geq 0.$$

This last summation implies that $\theta_l = 0$ for all l such that $\nu_j^{kl} < a_j^l$. □

3.4. y-strong capacity constraints

The y-strong capacity constraints (y-SCC) were also introduced by Contardo et al. [17]. Given a customer subset $S \subseteq \mathcal{C}$ and a route $l \in \Omega$, we let ξ_S^l be a binary constant equal to 1 iff route l visits at least one customer in S. We define $\xi^\theta(S) = \sum_{l\in\Omega} \xi_S^l \theta_l$. Also, let $r(S) = \lceil d(S)/Q \rceil$ be a lower bound on the number of vehicles that are needed to visit all customers in S, with $d(S) = \sum_{i\in S} d_i$. Let $S' \subseteq S$ be a subset satisfying $r(S \setminus S') = r(S)$. The following y-strong capacity constraint (y-SCC) is valid for the MDVRP:

$$\xi^\theta(S) + \sum_{i\in\mathcal{D}} \sum_{j\in S\setminus S'} y_{ij} \geq r(S). \tag{17}$$

This inequality dominates the SCC introduced by Baldacci et al. [10] and the y-capacity constraints (y-CC) introduced by Belenguer et al. [21]. Its addition, however, imposes a modification in the pricing algorithm to properly handle the associated dual variable.

3.5. Strong framed capacity inequalities

The strong framed capacity inequalities (SFCI) are a lifted form of the original FCI and were introduced by Contardo et al. [17]. Given a customer subset $S \subseteq \mathcal{C}$ (the frame) and a partition of it $(S_k)_{k\in K}$, let $r(S, (S_k)_{k\in K})$ be the solution of the following bin-packing problem. For each set S_k with accumulated demand $d(S_k) \leq Q$, consider one object of size $d(S_k)$. For each set S_k with accumulated demand $d(S_k) \geq Q$ consider $n_k = \lfloor d(S_k)/Q \rfloor$ objects of size Q, plus at most one object of size $d(S) - Q \times n_k$ (this last object does not appear if $d(S)$ divides Q). Set the bins to have capacity Q. If $r(S, (S_k)_{k\in K}) > r(S)$ then the following inequality is valid for the MDVRP:

$$\xi^{\theta}(S) + \sum_{k\in K} \xi^{\theta}(S_k) + 2 \sum_{i\in\mathcal{D}} \sum_{j\in S} y_{ij} \geq \sum_{k\in K} r(S_k) + r(S, (S_k)_{k\in K}). \tag{18}$$

Again, the addition of a SFCI requires a modification of the labeling algorithm during the recursion of the dynamic programming. Indeed, $|K| + 1$ additional resources are needed to properly handle the dual variable associated to a such constraint.

3.6. Subset-row inequalities

The subset-row inequalities were introduced by [19] with the purpose of replacing the clique inequalities valid for the set-partitioning formulation by some weaker form that however would be much easier to handle during a pricing sub-problem. In this article, we consider the subset-row inequalities for 3 customers. Given a customer subset C of size 3, for every route $l \in \Omega$ we let ψ^l be the number of times that l visits the customers in C. The following inequality is a valid subset-row inequality for MDVRP:

$$\sum_{l\in\Omega} \lfloor \psi^l/2 \rfloor \theta_l \leq 1. \tag{19}$$

The addition of a subset-row inequality of this form again forces the addition of an additional resource in the dynamic programming recursion.

3.7. Separation algorithms

For the robust constraints, we make use of the separation routines introduced in Lysgaard et al. [8] and Contardo et al. [23]. For the non-robust constraints, we use the same strategy as in Contardo et al. [17]. Constraints SDC and k-CEC are polynomial in number and can be easily separated by simple inspection. To find violated constraints y-SCC and SFCI, we verify for each robust constraint y-CC and FCI, if the non-robust version of such inequality is violated, and add it to the problem. Finally, for constraints SRI, we check for all possible triplets $\{i, j, k\}$ of three different customers if the corresponding SRI is violated.

4. The pricing sub-problem

In this section we present the pricing sub-problem used to generate columns of negative reduced cost when solving problem (8)–(12) by column generation. We first present the exact ng-SPPRC pricing algorithm. Then, we present a heuristic implementation of the algorithm used to speed-up the search for columns of negative reduced costs. Then, we present a decremental state-space relaxation (DSSR) framework used to solve the pricing sub-problem in an iterative fashion.

4.1. Exact pricing algorithm

Because of the multiple depots, the pricing sub-problem is performed independently for each depot $i \in \mathcal{D}$. We consider the ng-SPPRC pricing sub-problem, described as follows. At a pre-processing stage of the algorithm, we build ng-sets $(\mathcal{N}_j)_{j\in\mathcal{C}}$ such that $j \in \mathcal{N}_j$ for each j. Typically, the ng-set of node j contains the k closest customers to j. In our implementation, we use $k = 8$. The ng-SPPRC can provide very strong bounds in short computing times when compared to other relaxations of the ESPPRC. The key observation to support its success is that cycles are allowed on routes as long as these cycles are sufficiently long. Long cycles have a limited impact in the resulting lower bound of the problem, and that is why this pricing sub-problem provides a powerful relaxation. Irnich and Villeneuve [15] used this observation to develop the so-called SPPRC with k-cycle elimination (k-cyc-SPPRC), in which short cycles (in terms of the number of customers visited between two consecutive appearances of a given node) are forbidden whereas long cycles (in terms of the same number) are still allowed. The advantage of the ng-SPPRC with respect to k-cyc-SPPRC is how the length of a cycle is defined. Indeed, the number of customers between two appearances of a node may not be the right criterion, the length of the cycle in terms of distance being a better measure of the impact of a cycle in the linear relaxation lower bound. The ng-SPPRC thus forbids cycles that are short with respect to the time consumption regardless of the number of nodes between two appearances of a given customer.

When the restricted master problem of (8)–(12) is solved at any iteration, the degree constraints (9) and the fleet size constraints (10) provide dual variables $(\alpha_j)_{j\in\mathcal{C}}$ and $(\beta_i)_{i\in\mathcal{D}}$, respectively. If no other constraint has been added to the problem, one can define the following reduced costs on the edges of E, as follows:

$$\overline{c_{ij}} = \begin{cases} c_{ij} - \left(\frac{\alpha_j+\alpha_i}{2}\right) & \text{if } i,j \in \mathcal{C} \\ c_{ij} - \left(\frac{\beta_i+\alpha_j}{2}\right) & \text{if } i \in \mathcal{D}, j \in \mathcal{C}. \end{cases} \tag{20}$$

It is easy to see that the reduced cost of a route can be decomposed along the edges traversed by the route using the expressions in (20). Moreover, if cuts valid for the two-index vehicle-flow formulation (1)–(7) are translated into the set-partitioning formulation using identities (13)–(14), the contributions of their dual variables can also be decomposed along the edges traversed by a path and thus do not affect the complexity of the pricing sub-problem.

When one adds non-robust cuts, say cuts (15)–(19), the dynamic programming recursion needs to include additional resources. Let us define by Γ the set of all possible partial paths (potentially having cycles) starting at depot i, this is paths that visit a certain subset of customers without exceeding the capacity and route length requirements. Let us define a *cutset boolean resource* ν for a given set $S \subseteq \mathcal{C}$ as a binary function $\nu : \Gamma \to \{0, 1\}$, such that $\nu(L) = 1$ iff the partial path L visits at least one node in S. When one adds cuts from families (15), (17) and (18) to the restricted master problem, the contributions of the associated dual variables to the computation of the reduced costs on a path can be modeled using one or several cutset boolean resources, as follows. For each SCC cut (17) associated to set S one adds one cutset boolean resource to the label definition. For each SDC cut (15) associated to customer j one adds one cutset boolean resource associated to set $\{j\}$. For each SFCI (18) associated to a frame S and to partition $(S_k)_{k=1}^n$ one adds $n + 1$ cutset boolean variables: one for the frame S and one for each subset S_k, $k = 1, \ldots, n$. Let us denote by $\mathcal{R}_C$ the set of cutset boolean resources. For each $\nu \in \mathcal{R}_C$ we denote by $S(\nu)$ the cutset associated, and by $\sigma(\nu)$ the dual variable associated.

When adding cuts (16) or (18) one must include a different type of resource to model the contributions of the associated dual variables to the computation of the reduced costs on a path. For each k-CEC (16) associated to a customer j and a cycle length k we add an integer resource $\kappa : \Gamma \to \{0, 1, \ldots, k\}$. The quantity $\kappa(L)$ represents the number of customer nodes different from j visited by L since the last visit to j. Let us denote the set of resources associated to k-CEC as $\mathcal{R}_K$. For each $\kappa \in \mathcal{R}_K$, we denote by $j(\kappa)$ and $k(\kappa)$ the node and cycle length associated. Also we denote by $\sigma(\kappa)$ the dual variable associated. For each subset-row inequality (19) associated to a customer subset C we consider one binary resource $\pi : \Gamma \to \{0, 1\}$. The quantity $\pi(L)$ is 1 iff. L visits the nodes in C an odd number of times. Let us denote the set of resources associated to constraints (19) as $\mathcal{R}_S$, and by $C(\pi)$, $\sigma(\pi)$ the customer subset defining the cut and the dual variable associated, respectively.

For the dynamic programming recursion one needs to define first what a label is. A label is a tuple $L = (v, q, \tau, \overline{c}, \Pi, \mathcal{R}_C, \mathcal{R}_K, \mathcal{R}_S, p)$ encoding a partial path starting at depot i and satisfying the capacity and route length requirements. We will overload the notation already defined to denote the set of all labels as Γ. Indeed, a label represents a partial path and vice-versa. A label L is composed by the following information: the terminal node v representing the final node in the partial path encoded by L; the accumulated demand q serviced in the partial path; the accumulated travel time τ spent by the partial path, including the service times; the reduced cost $\overline{c}$ of the partial path; the set of forbidden label extensions Π, this is the set containing the customer nodes that cannot be visited immediately from extending L; the cutset boolean resources associated to the binding cuts (15), (17) and (18); the set of integer resources $\mathcal{R}_K$ associated to the binding k-CEC cuts (16); the set of binary resources $\mathcal{R}_S$ associated to the binding subset-row cuts; and the predecessor label p from which L was built after a label extension. The dynamic programming recursion starts from a label $L_0^i = (\{i\}, 0, 0, 0, \emptyset, (0)_{\nu\in\mathcal{R}_C}, (k(\kappa))_{\kappa\in\mathcal{R}_K}, (0)_{\pi\in\mathcal{R}_S}, nil)$. It is based on a label extension rule to create paths returning to the depot, and on a dominance rule to discard non-promising labels. Note that our dynamic programming recursion is unidirectional, unlike several recent algorithms that use bidirectional dynamic programming to save time in label extensions. Unfortunately, the time saved in extending labels may be lost later when full paths need to be built from joining partial paths pairwise. We have found that the fathoming rule, as it becomes sharper towards the end of the labeling algorithm, contributes to accelerate the unidirectional search that in practice resulted in a faster algorithm.

The label *extension rule* is as follows. If we are extending a label L with terminal node v to a customer $w \in \mathcal{C} \setminus \Pi(L)$, the new label L' is:

$$v(L') = w \tag{21}$$

$$q(L') = q(L) + d_w \tag{22}$$

$$\tau(L') = \tau(L) + \left(\tfrac{s_v+s_w}{2}\right) \tag{23}$$

$$\overline{c}(L') = \overline{c}(L) + \overline{c_{vw}} - \sum_{\nu\in\mathcal{R}_C(L,L')} \sigma(\nu) - \sum_{\kappa\in\mathcal{R}_K(L,L')} \sigma(\kappa) - \sum_{\pi\in\mathcal{R}_S(L,L')} \sigma(\pi) \tag{24}$$

$$\Pi(L') = (\Pi(L) \cap \mathcal{N}_w) \cup \{w\} \tag{25}$$

$$\nu(L') = \begin{cases} 1 & \text{if } w \in S(\nu) \\ \nu(L) & \text{otherwise} \end{cases} \quad \nu \in \mathcal{R}_C \tag{26}$$

$$\kappa(L') = \begin{cases} \min\{k(\kappa), \kappa(L) + 1\} & \text{if } w \neq j(\kappa) \\ 0 & \text{otherwise} \end{cases} \quad \kappa \in \mathcal{R}_K \tag{27}$$

$$\pi(L') = \begin{cases} \pi(L) + 1 (\text{mod } 2) & \text{if } w \in C(\pi) \\ \pi(L) & \text{otherwise} \end{cases} \quad \pi \in \mathcal{R}_S \tag{28}$$

$$p(L') = L, \tag{29}$$

with $\mathcal{R}_C(L, L') = \{\nu \in \mathcal{R}_C : \nu(L') \in S(\nu), \nu(L) = 0\}$, $\mathcal{R}_K(L, L') = \{\kappa \in \mathcal{R}_K : \nu(L') = j(\kappa), \kappa(L) \geq k(\kappa)\}$ and $\mathcal{R}_S(L, L') = \{\pi \in \mathcal{R}_S : \pi(L) = 1, \nu(L') \in C(\pi)\}$. If one extends a label L to the depot to create L', only the terminal node $\nu(L')$ and reduced cost $\overline{c}(L')$ need to be defined as $\nu(L') \leftarrow \{i\}$, $\overline{c}(L') \leftarrow \overline{c}(L) + \overline{c_{vi}}$.

Now, let us present the *dominance rule* used to discard non-promising labels. Let L and L' be two labels sharing the same terminal node v. We say that L dominates L' if:

$$q(L) \leq q(L') \tag{30}$$

$$\tau(L) \leq \tau(L') \tag{31}$$

$$\Pi(L) \subseteq \Pi(L') \tag{32}$$

$$\overline{c}(L) \leq \overline{c}(L') - \left(\sum_{\{\nu \in \mathcal{R}_C^{L>L'}\}} \sigma(\nu) + \sum_{\{\kappa \in \mathcal{R}_K^{L>L'}\}} \sigma(\kappa) \right) + \sum_{\{\pi \in \mathcal{R}_S^{L>L'}\}} \sigma(\pi), \tag{33}$$

with $\mathcal{R}_C^{L>L'} = \{\nu \in \mathcal{R}_C : \nu(L) > \nu(L')\}$, $\mathcal{R}_K^{L>L'} = \{\kappa \in \mathcal{R}_K : \kappa(L) > \kappa(L')\}$ and $\mathcal{R}_S^{L>L'} = \{\pi \in \mathcal{R}_S : \pi(L) > \pi(L')\}$. The first three conditions ensure that the feasible extensions of L' are also feasible extensions for L. The last inequality aims at considering all resources from which a common extension of L and L' would result in either increasing the reduced cost of L without increasing that of L', or decreasing that of L' without decreasing that of L. The inequality thus represents the impossibility for the reduced cost of L to exceed that of L'. It follows that L' can be safely discarded as it will never produce paths better than those that can be produced from extending L. If all relational conditions hold with equality, then both L and L' dominate each other, but one has to arbitrarily choose only one to be discarded. Algorithm 1 illustrates the dynamic programming algorithm used to solve the pricing sub-problem.

Algorithm 1 Dynamic programming algorithm for the solution of the pricing sub-problem

ng-SPPRC(i, $(\overline{c_e})_{e \in E}$, $(\mathcal{N}_j)_{j \in \mathcal{C}}$, $\mathcal{R}_C$, $\mathcal{R}_K$, $\mathcal{R}_S$)

$\mathcal{V} \leftarrow \{L_0^i\}$, $\mathcal{L} \leftarrow \emptyset$.
repeat
  Take label $L \in \mathcal{V}$ and set $\mathcal{V} \leftarrow \mathcal{V} \setminus \{L\}$.
  if $\nu(p(L)) \neq \{i\}$ **and** $\tau(L) + c_{\nu(L)i} + s_{\nu(L)}/2 \leq T$ **then**
    Extend L to depot i to create a new label L'.
    if $\overline{c}(L') < 0$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{L'\}$.
    end if
  end if
  for all $w \in \mathcal{C} \setminus \Pi(L)$ **s.t.** $q(L) + d_w \leq Q$ **and** $\tau(L) + c_{\nu(L)w} + (s_{\nu(L)} + 2s_w)/2 \leq T$ **do**
    Extend L to w to create a new label L'.
    Apply dominance rule and eventually discard L'.
    if L' has not been discarded **then**
      Apply dominance rule and eventually delete other labels in $\mathcal{L}$.
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{L'\}$.
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{L'\}$.
    end if
  end for
until $\mathcal{V} = \emptyset$
return $\{L \in \mathcal{L} : \nu(L) = i, \overline{c}(L) < 0\}$

Note that for the problems without route length constraints (like all the CVRP instances and some MDVRP instances), one can omit the time resource τ from the label definition, as well as from the label extension rule, feasibility checking and dominance rule. This allows a faster execution of the algorithm, mainly because of the less restrictive dominance rule.

4.2. Heuristic pricing algorithm

The exact solution of the pricing sub-problem can be extremely difficult in the presence of many non-robust cuts and/or large *ng*-sets. Therefore we perform a heuristic pricing algorithm based on a heuristic implementation of the dynamic programming. First, we consider reduced networks constructed as follows. Let us define, for a given node subset S, $E(S) = \{e = (u, v) \in E : u, v \in S\}$. We rank each edge $e \in E$ according to the following measure:

$$\rho(e) = \overline{c_e} - \frac{1}{2}\left(\sum_{\{v \in \mathcal{R}_C : e \in \delta(S(v))\}} \sigma(v) + \sum_{\{\kappa \in \mathcal{R}_K : e \in \delta(\{j(\kappa)\})\}} \sigma(\kappa) + \sum_{\{\pi \in \mathcal{R}_S : e \in E(C(\pi))\}} \sigma(\pi)\right). \tag{34}$$

We consider three settings of the pricing algorithm, each of which contains a 25%, 50% and 100% of the edges in the original graph, respectively, by selecting those edges with the smallest values of ρ and by discarding the remaining edges. The pricing sub-problem is solved sequentially for these three settings until detecting columns of negative reduced costs. In addition, we use constraints SDC (15) and k-CEC (16) to impose the related structural constraints.

4.3. Decremental state-space relaxation

The decremental state-space relaxation framework [16,24] used to solve the pricing sub-problem is based on two main observations. First, when the *ng*-sets $(\mathcal{N}_j)_{j \in \mathcal{C}}$ are partially or totally ignored, the resulting pricing sub-problem provides a lower bound on the solution of the original pricing sub-problem containing the complete sets of neighbors [25]. Second, it is possible to *underestimate* the non-robust valid inequalities so as to discard the use of some or all resources associated to non-robust cuts [17]. The resulting pricing sub-problem defined on the modified network and with the reduced number of resources and smaller *ng*-sets yields a lower bound on the solution of the original problem. In this article, we combine the two observations to enlarge the *ng*-sets and add resources associated to non-robust cuts in a dynamic fashion.

Let us define first what we understand by underestimation of a resource associated to a non-robust cut. Let $v \in \mathcal{R}_C$ be any cutset boolean resource. If we modify the reduced cost of every edge $e \in \delta(S(v))$ by subtracting $\sigma(v)/2$ to the reduced cost $\overline{c_e}$ of e, the resulting pricing sub-problem in the modified network and without considering v yields a lower bound on the original pricing sub-problem. Now, let $\kappa \in \mathcal{R}_K$ be a resource associated to a k-CEC. We can subtract $\sigma(\kappa)/2$ to the reduced cost $\overline{c_e}$ of every edge $e \in \delta(\{j(\kappa)\})$, which provides a valid lower bound of the pricing sub-problem if κ is in addition discarded. Finally, let $\pi \in \mathcal{R}_S$ be a resource associated with a subset-row inequality. It is possible to modify the reduced costs on the edges $E^C = \{e = (u, v) \in E : u, v \in C(\pi)\}$ by subtracting $\sigma(\pi)/2$ from the reduced costs. Again, this modification allows discarding the resource π to provide a lower bound on the solution of the original pricing sub-problem.

The pricing sub-problem is solved as follows. At first, the *ng*-sets are ignored as well as all resources associated to non-robust cuts. The resulting pricing sub-problem corresponds to a SPPRC with cycles and without additional resources. We solve this problem and check if the minimum reduced cost of a path is ≥ 0, in which case the algorithm is stopped and no column is returned. If the minimum cost is negative, we check if the associated column is feasible with respect to the original *ng*-sets, has no cycle of length one or two, and has a reduced cost that coincides with the original reduced cost associated to the inclusion of all additional resources. In that case, the algorithm is stopped and this column, together with all other columns such that the original reduced costs are negative, that are feasible with respect to the original *ng*-sets and that contain no cycles of lengths one or two, are returned. In the case in which the column reaching the minimum reduced cost is not feasible with respect to the original *ng*-sets, has a cycle of length 2, or a reduced cost that is lower than the one computed with the original resources, we consider the 10 columns with minimum reduced costs and extend the *ng*-sets so as to forbid all the infeasibilities with respect to the original *ng*-sets or to forbid cycles of length one or two for those columns. If the column reaching the minimum reduced cost is *ng*-feasible, we update the resources related to non-robust cuts to include those cuts whose underestimation produces a lower bound of the reduced cost rather than the reduced cost itself. To impose the elimination of cycles of length one or two, the *ng*-sets are potentially enlarged by adding customers that were not in the original sets $(\mathcal{N}_j)_{j \in \mathcal{C}}$. As a consequence, the bound computed using this relaxation is at least as strong as that of the *ng*-SPPRC plus 2-cycle elimination.

The decremental state-space relaxation used to solve this problem is complemented with a fathoming rule used to discard non-promising labels based on an additional optimality criterion. They make use of completion bounds [10,17,25] to speed-up each iteration of the algorithm. Indeed, the completion bounds used during a given iteration k are computed from the labels constructed at iteration $k-1$ [25]. Let $\mathcal{L}^{k-1}$ be the set of labels produced at iteration $k-1$ of the algorithm, and let $\mathcal{L}^k$ be the set of labels being produced at iteration k of the algorithm. Let $\mathcal{R}_C^{k-1}$, $\mathcal{R}_K^{k-1}$ and $\mathcal{R}_S^{k-1}$ the resources associated to non-robust cuts at iteration $k-1$ of the algorithm. Analogously, we use $\mathcal{R}_C^k$, $\mathcal{R}_K^k$ and $\mathcal{R}_S^k$ for the resources associated with non-robust cuts used at iteration k of the algorithm. Finally, we let $\overline{c}^{k-1}$, $\overline{c}^k$ be the reduced costs computed at iteration $k-1$ and k, respectively. For a given customer node v, load $q < Q$ and time $\tau < T$, we define $f(v, q, \tau) = \min\{\overline{c}^{k-1}(L) : L \in \mathcal{L}^{k-1}, v(L) = v, q(L) + q - d_v \leq Q, \tau + \tau(L) \leq T\}$. Also, we define, for every label $L \in \mathcal{L}^k$, quantities $(u_i(L))_{i=1}^6$ as follows:

$$u_1(L) = \sum_{\{v \in \mathcal{R}_C^{k-1} : v(L) \in S(v)\}} \sigma(v) \tag{35}$$

$$u_2(L) = \sum_{\{\nu \in \mathcal{R}_C^k \setminus \mathcal{R}_C^{k-1} : v(L) \in S(\nu)\}} \sigma(\nu) \tag{36}$$

$$u_3(L) = \sum_{\{\kappa \in \mathcal{R}_K^{k-1} : v(L) = j(\kappa)\}} \sigma(\kappa) \tag{37}$$

$$u_4(L) = \sum_{\{\kappa \in \mathcal{R}_K^{k} \setminus \mathcal{R}_K^{k-1} : v(L) = j(\kappa)\}} \sigma(\kappa) \tag{38}$$

$$u_5(L) = \sum_{\{\pi \in \mathcal{R}_S^{k-1} : v(L) \in C(\pi), \pi(L) = 0\}} \sigma(\pi) \tag{39}$$

$$u_6(L) = \sum_{\{\nu \in \mathcal{R}_S^{k} \setminus \mathcal{R}_S^{k-1} : v(L) \in C(\pi), \pi(L) = 0\}} \sigma(\pi). \tag{40}$$

The following result provides a fathoming rule for discarding non-promising labels.

Theorem 4.1. *For a given label $L \in \mathcal{L}^k$ with $v(L) = v$, $q(L) = q$ and $\tau(L) = \tau$, the quantity*

$$lb(L) = \begin{cases} \overline{c}^k(L) + f(v, q, \tau) + u_1(L) + u_3(L) + u_5(L) + \dfrac{u_2(L) + u_4(L) + u_6(L)}{2} & \text{if } k \geq 2 \\ -\infty & \text{if } k = 1 \end{cases} \tag{41}$$

represents a completion lower bound of L, this is a lower bound on the reduced cost achieved by any feasible path produced from extending L.

Proof. Let $\mathcal{P} = (L, L')$ be a feasible path (in terms of the current *ng*-sets at iteration k) extending L, with $L, L' \in \mathcal{L}^k$ and $v(L') = v(L)$. The first observation that we make is that L' may not appear at iteration $k-1$, for two reasons. First, label L' may have been deleted by dominance, in which case it was replaced by another label L'' of the same cost or less and that also has L as a feasible extension. Second, label L' could have been deleted by the fathoming rule, in which case we can deduce that path $\mathcal{P}$ has zero or positive reduced cost. Therefore we may assume, without loss of generality, that $L' \in \mathcal{L}^{k-1}$. In the absence of resources associated to non-robust cuts it is easy to verify that $\overline{c}^k(L') \geq \overline{c}^{k-1}(L') \geq f(v(L), q(L), \tau(L))$ and the result follows. Now, for each $\nu \in \mathcal{R}_C^{k-1}$ such that $v(L) \in S(\nu)$, the corresponding dual variable is being counted twice (once in L and once in L' from the previous iteration) when considering the reduced costs $\overline{c}^k(L)$ and $\overline{c}^{k-1}(L')$ separately, therefore the reduced cost associated with $\mathcal{P}$ can be increased by $\sigma(\nu)$ units. This explains the term $u_1(L)$ in the summation. If, however, $\nu \in \mathcal{R}_C^k \setminus \mathcal{R}_C^{k-1}$ and $v(L) \in S(\nu)$ then the corresponding dual variable is being counted at least 1.5 times (once for L and at least 0.5 times for L'), and so the reduced of $\mathcal{P}$ can be increased by $\sigma(\nu)/2$ units. This explains the term $u_2(L)/2$ in the summation. Similar reasonings lead to explain the appearance of the terms $u_3(L)$ and $u_4(L)/2$. For the resources associated with subset-row cuts $\mathcal{R}_S$, the reasoning is somehow the opposite. Let $\pi \in \mathcal{R}_S^{k-1}$ be such that $v(L) \in C(\pi)$ and $\pi(L) = 0$. The dual variable associated with resource π may be being counted at most one additional time. As to make sure that the quantity $lb(L)$ is still a lower bound on the reduced cost of $\mathcal{P}$, it is necessary to subtract $-\sigma(\pi)$. This explains the appearance of $u_5(L)$ in the summation. If $\pi \in \mathcal{R}_S^k \setminus \mathcal{R}_S^{k-1}$, then the dual variable associated with resource π may be being counted at most 0.5 additional times so is necessary to subtract $-\sigma(\pi)/2$ from the expression, which explains the appearance of $u_6(L)/2$. Note that $lb(L)$ is indeed a lower bound in the reduced cost associated with path $\mathcal{P}$ as there may be one or many other resources not associated with $v(L)$ whose associated dual variables would need to be added to $lb(L)$ for a more precise estimation of the reduced cost of $\mathcal{P}$. □

Using this result, it is possible to fathom a label L when $lb(L) \geq 0$. The fathoming rule is applied immediately before the dominance rule, as it is computationally less expensive. Let us denote by *ng*-SPPRC-DSSR(i, $(\overline{c_e})_{e \in E}$, $(\mathcal{N}_j)_{j \in \mathcal{C}}$, $\mathcal{R}_C$, $\mathcal{R}_K$, $\mathcal{R}_S$) the pricing sub-problem modified to include the DSSR with the fathoming of labels L such that $lb(L) \geq 0$ (see Algorithm 2). Note that because the domain of function f can be extremely large (even infinity if travel times are arbitrary real numbers), it is convenient to define it on a limited number of buckets of load and time. The finer the granularity used in these buckets, the more accurate that the resulting bounds will be, but also more memory space will be needed to store such quantities. In our code, we chose different settings for the granularities depending on the presence or absence of route length constraints. For the problems without such requirement (like all CVRP instances and some MDVRP ones), we consider one bucket for the time, and 100 for the load. For the problems including route length constraints (like some MDVRP instances) we first check which resource (load or time) is more constraining (in terms of the average number of customers that would fit in a route). We consider 50 buckets of the less constraining resource, and 5 for the more constraining one.

5. The exact method

In this section we describe the exact method used to solve the MDVRP to optimality. It is based on three main components: variable fixing, column-and-cut generation and column enumeration.

Algorithm 2 Pricing algorithm solved within a DSSR framework

ng-SPPRC-DSSR(i, $(\overline{c_e})_{e\in E}$, $(\mathcal{N}_j)_{j\in\mathcal{C}}$, $\mathcal{R}_C$, $\mathcal{R}_K$, $\mathcal{R}_S$)

$\widehat{\mathcal{N}_j} \leftarrow \emptyset, j \in \mathcal{C}$.
$\widehat{\mathcal{R}}_C \leftarrow \emptyset$, $\widehat{\mathcal{R}}_K \leftarrow \emptyset$, $\widehat{\mathcal{R}}_S \leftarrow \emptyset$.
Build edge reduced costs $(\widehat{c_e})_{e\in E}$ by underestimating all non-robust cuts.
$\widehat{f}(v, q, \tau) \leftarrow -\infty, v \in \mathcal{C}, 0 \le q \le Q, 0 \le \tau \le T$.
$k \leftarrow 0$.
repeat
  $k \leftarrow k + 1$.
  Solve problem ng-SPPRC(i, $(\widehat{c_e})_{e\in E}$, $(\widehat{\mathcal{N}_j})_{j\in\mathcal{C}}$, $\widehat{\mathcal{R}}_C$, $\widehat{\mathcal{R}}_K$, $\widehat{\mathcal{R}}_S$) using $\widehat{f}$ to fathom labels L such that $lb(L) \ge 0$.
  Let $\mathcal{U}$ be the 20 columns with most negative reduced costs at the current iteration.
  $L_{best} \leftarrow arg\ min\{\overline{c}^k(L) : L \in \mathcal{U}\}$.
  Enlarge ng-sets $(\widehat{\mathcal{N}_j})_{j\in\mathcal{C}}$ using the columns in $\mathcal{U}$.
  Enlarge sets $\widehat{\mathcal{R}}_C$, $\widehat{\mathcal{R}}_K$, $\widehat{\mathcal{R}}_S$ using the columns in $\mathcal{U}$.
  Re-compute the edge reduced costs $(\widehat{c_e})_{e\in E}$.
  Re-compute functions $\widehat{f}$ using the labels $\mathcal{L}^k$ built at the current iteration.
until L_{best} is ng-feasible w.r.t. $(\mathcal{N}_j)_{j\in\mathcal{C}}$ and 2-cycle elimination and $\overline{c}^k(L) = \overline{c}(L)$.
$\mathcal{S} \leftarrow \{L \in \mathcal{L}^k : v(L) = i, \overline{c}(L) < 0, L$ is feasible w.r.t. $(\mathcal{N}_j)_{j\in\mathcal{C}}$ and 2-cycle elimination$\}$.
return $\mathcal{S}$.

In the first part, the linear programming (LP) relaxation of the two-index vehicle-flow formulation is solved by means of the cutting planes method. Initially, the integrality constraints are dropped and we consider a restricted problem containing only a subset of constraints, namely (2)–(3). Constraints (4)–(5) are added as cutting planes as they are exponential in number. Other inequalities such as multistar inequalities, hypotour inequalities, framed capacity inequalities, strengthened comb inequalities, y-capacity cuts, degree constraints and co-circuit inequalities are also added as cutting planes. Note that the capacity and route length constraints (4) are relaxed by letting the right-hand side be equal to $r(S)$ (i.e., route length constraints are relaxed). The solution of the LP relaxation of formulation (1)–(7) is used to perform variable fixing based on the reduced costs of the edge variables $(x_e)_{e\in E}$. Let us call those reduced costs as $(\overline{c_e})_{e\in E}$. If the lower bound obtained from this procedure is denoted by z^1_{lb} and we denote by z_{ub} the value of a primal solution, we fix to zero all variables x_e (and thus discard them from the network) such that $z^1_{lb} + \overline{c_e} \ge z_{ub}$. The CPU time spent in this procedure is denoted by t^1 and the relative gap is denoted by gap^1, computed as $(z_{ub} - z^1_{lb})/z_{ub} \times 100$.

In the second part of the algorithm, a reduced network is considered by a priori discarding all edges that were fixed to zero in the first stage. The set-partitioning formulation (8)–(12) is thus considered with the reduced set of edges. The LP relaxation of this problem is solved by column-and-cut generation using the pricing algorithms described earlier. The problem is strengthened with the use of all inequalities introduced in this article. To control the addition of non-robust cuts and thus to bound the computational complexity of the resulting pricing sub-problem, we add the non-robust cuts in a escalated way. Initially, in what we call the first major iteration, we allow only the addition of k-CEC cuts (16) for $3 \le k \le 7$. In the second major iteration, we allow the addition of cuts (17) and (19) in addition to cuts (16). We add at most 100 cuts in total. In the third major iteration we allow, in addition to cuts (16), (17) and (19), the addition of cuts (15). We allow the addition of 150 more cuts. We perform at most 8 major iterations in total, each of which considers the addition of 50 more cuts than the previous one. The cuts are added in groups of 50 inequalities from the second to the eight major iteration.

Before the addition of a new group of valid inequalities, we perform a variable fixing procedure based on the solution of the exact pricing algorithm, as follows. The exact pricing algorithm is solved by including a maximum of 50% of non-robust inequalities, the rest being left underestimated. Let $(\overline{c_e})_{e\in E}$ be the reduced costs computed using identities (20), modified accordingly to include the non-robust cuts added so far as well as the constraints left underestimated. Let $\mathcal{L}$ be the set of labels produced by the resulting labeling algorithm. For every edge $e = (v, w) \in E$ (assuming that E does not include the removed edges) we compute the following quantity: $\gamma_e = \overline{c_e} + \sum_{\{v\in\mathcal{R}_C : v, w\in S(v)\}} \sigma(v) + \min_{0\le\tau\le T, 0\le q\le Q}\{\min\{\overline{c}(L) : v(L) = v, q(L) \le Q, \tau(L) \le \tau - s_v/2\} + \min\{\overline{c}(L) : v(L) = w, q(L) \le Q - q, \tau(L) \le T - \tau - c_e - s_w/2\}\}$. This quantity represents a lower bound on the reduced cost achieved by any route using edge e [26]. If $\gamma_e \ge z_{ub} - z_{lb}$ (with z_{lb} being the lower bound achieved before the addition of the next round of cuts) then edge e can be safely discarded from the network.

The third part of our algorithm is performed after each major iteration of the second part of the algorithm. Let us denote by z^{2e}_{lb} the lower bound obtained at the end of the current major iteration, and by gap^{2e} the associated relative gap. We enumerate all elementary columns whose reduced costs are less than or equal to $z_{ub} - z^{2e}_{lb}$ using the algorithm described in the next paragraphs. We set two hard limits that if exceeded result in an abortion of the enumeration process and trigger the execution of the next major iteration of the second part of the algorithm. First, we limit the number of columns generated $|\mathcal{P}^{2e}|$ to 5 millions. Second, we allow the enumeration procedure to consume at most 13 GB of RAM. If the enumeration procedure finishes with success, we consider a last major iteration in which the maximum number of separated inequalities is set to $+\infty$ for all families of non-robust cuts in an attempt to strengthen the final lower bound, denoted by z^{2f}_{lb}, and thus reduce the number of final columns in the problem, denoted by $|\mathcal{P}^{2f}|$. The associated relative gap is then denoted

by gap^{2f}. The total CPU time spent in column generation, including each of the major iterations performed, the variable fixing procedure and the column enumeration procedure is denoted by t^2. Roughly speaking, our computational experience suggests that this CPU time can be decomposed as follows: 70% of the CPU time is spent in standard column generation, 20% in variable fixing and 10% in column enumeration. The CPU time spent in producing the bound z_{lb}^{2f} after column enumeration is negligible. The reduced set of columns $\mathcal{P}^{2f}$ is passed to CPLEX and the optimal solution is obtained from solving the resulting IP by branch-and-cut. The final lower bound obtained with this procedure is denoted by z_{lb}^3 and the associated relative gap is denoted by gap^3. The total amount of CPU time spent in solving the IP is denoted by t^3.

To enumerate all columns whose reduced costs are less than or equal to $z_{ub} - z_{lb}^{2e}$, we use dynamic programming. In the new dynamic programming recursion, a label L is composed of the following quantities: the terminal node $v(L)$, the load $q(L)$, the accumulated travel time $\tau(L)$, the reduced cost $\overline{c}(L)$, the customers visited $V(L)$, the size of that set $|V(L)|$, the predecessor label $p(L)$ and the resources associated to non-robust cuts $\mathcal{R}_C$, $\mathcal{R}_K$ and $\mathcal{R}_S$. The resources are updated using the same *extension rule* as for the previous *ng*-SPPRC algorithm, with two exceptions: (1) sets $\Pi(L)$ are replaced by sets $V(L)$ that represent all nodes visited by L, and that are used to impose path elementarity; (2) we also include the length of the path $|V(L)|$ used to rapidly discard the dominance between two labels. The recursion is performed, just as for solving the *ng*-SPPRC, using unidirectional search. Unidirectional search may produce a much larger tree than bidirectional search. However, we have encountered two main advantages with respect to the bidirectional approach. First, the completion bounds become much sharper when the recursion reaches the end, and so does the fathoming rule to be described later. Second, we save the time spent in joining labels pairwise. Moreover, to control the explosion of the algorithm we always explore the unexplored label with smallest accumulated demand. This allows us, before every label extension, to discard all labels whose accumulated demands are strictly smaller than that of the label being explored.

Now, a modified *dominance rule* is used to discard labels. Indeed, because we enumerate all columns that may potentially appear in the optimal solution, dominance must be done using a much stronger criterion. Notably, we say that a label L dominates a label L' that share the same terminal node v if all of the following conditions hold:

i. $q(L) = q(L')$.
ii. $|V(L)| = |V(L')|$.
iii. $V(L) = V(L')$.
iv. $\tau(L) \leq \tau(L')$.

As said before, conditions (i)–(ii) are implied by (iii), but they are used to rapidly discard the dominance of one label with respect to another when any of these two conditions is not satisfied. Note also that because we want to enumerate all feasible routes, only elementary labels are generated. This dominance rule, because it is much more specific than the classical dominance used for *ng*-SPPRC, may not be sufficient to bound the computational complexity of the enumeration procedure. Therefore, we complement it with a fathoming rule which is based on the solution of the *ng*-SPPRC with resources associated to all non-robust constraints. We execute algorithm *ng*-SPPRC-DSSR with one modification: the fathoming rule is used to discard labels L such that $lb(L) \geq z_{ub} - z_{lb}^{2e}$. This modification is needed as otherwise the algorithm might fathom labels that are needed. Let $\mathcal{L}^{ng}$ be the set of all labels produced by this labeling procedure, and let $\overline{c}^{ng}$ denote the reduced costs. For every customer v, load q and travel time τ we let $F(v, q, \tau)$ be a list containing the 50 labels $L \in \mathcal{L}^{ng}$ with the smallest values of $\overline{c}^{ng}(L)$ and such that $v(L) = v$, $q(L) \leq Q - q + d_v$, $\tau(L) \leq T - \tau$. Moreover, if two labels have the same Π set, only the one with the smallest reduced cost is kept. The completion bound is computed using the following expression:

$$lb(L) = \overline{c}(L) + h(L) + u_1(L) + u_3(L) + u_5(L), \tag{42}$$

with u_1, u_3 and u_5 defined as before. If the elements of $F(v, q, \tau)$ for every $v \in \mathcal{C}, 0 \leq q \leq Q, 0 \leq \tau \leq T$, are sorted in non-decreasing order of reduced cost and denoted as $l_k, k = 1, \ldots, |F(v, q, \tau)|$, $h(L)$ is computed using the procedure described in Algorithm 3. This lower bound is used to fathom all labels L such that $lb(L) \geq z_{ub} - z_{lb}^{2e}$. Once again, the number of sets $F(\cdot, \cdot, \cdot)$ may be extremely large or even infinity if the times are arbitrary real numbers. To overcome this apparent flaw, we aggregate them in a limited number of buckets. The larger the granularity used in the aggregation, the less memory consumption that is needed to store the sets, but also less precise the completion bounds become. Depending on which resource (load or time) is more constraining, we consider 250 buckets of the less constraining resource and 25 of the more constraining one, for the instances with both capacity and route length constraints. For the instances including only the capacity constraints, we consider one bucket for the time and 500 for the load. Algorithm 4 illustrates the enumeration procedure used. Similar enumeration procedures have been used in [10,5,27,11,28,17]. For more details we refer to these articles.

6. Computational experiments

We have conducted a series of experiments on several sets of instances from the literature for both problems considered in this study. For the CVRP, we consider six classes of instances, namely classes A, B, E, F, M and P. Classes A, B and P were proposed by Augerat [29]. Class E was proposed by Christofides and Eilon [30]. Class F was proposed by Fisher [31]. Finally, class M was proposed by Christofides et al. [32]. All these instances can be found in the website http://www.branchandcut.org. For the MDVRP, we consider 23 instances from the dataset proposed by Cordeau et al. [33].

Algorithm 3 Procedure to compute $h(L)$

$h(L)$

$\gamma \leftarrow +\infty$
for all $k = 1$ **to** $|F(v(L), q(L), \tau(L))|$ **do**
  Let l_k be the k-th label in $F(v(L), q(L), \tau(L))$.
  $\gamma \leftarrow \overline{c}^{ng}(l_k)$.
  if $\Pi(l_k) \cap V(L) = \{v(L)\}$ **then**
    return γ
  end if
end for
return γ

Algorithm 4 Column enumeration algorithm

ENUM(i)

$\mathcal{V} \leftarrow \{L_0^i\}$, $\mathcal{L} \leftarrow \emptyset$.
Solve the ng-SPPRC(i, $z_{ub} - z_{lb}^{2e}$) and build sets $F(\cdot, \cdot, \cdot)$.
repeat
  $L \leftarrow arg\ min\{q(L) : L \in \mathcal{V}\}$
  $\mathcal{V} \leftarrow \mathcal{V} \setminus \{L\}$.
  Delete all labels $L' \in \mathcal{L}$ such that $v(L') \in \mathcal{C}, q(L') < q(L)$.
  if $v(p(L)) \neq \{i\}$ **and** $\tau(L) + c_{v(L)i} + \frac{s_{v(L)}}{2} \leq T$ **then**
    Extend L to depot i to create a new label L'.
    if $\overline{c}(L') < z_{ub} - z_{lb}^{2e}$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{L'\}$.
    end if
  end if
  for all $w \in \mathcal{C} \setminus V(L)$ **s.t.** $q(L) + d_w \leq Q$ **and** $\tau(L) + c_{v(L)w} + \left(\frac{s_{v(L)} + 2s_w}{2}\right) \leq T$ **do**
    Extend L to w to create a new label L'.
    Apply fathoming rule and eventually discard L'.
    Apply dominance rule and eventually discard L'.
    if L' has not been discarded **then**
      Apply dominance rule and eventually delete other labels in $\mathcal{L}$.
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{L'\}$.
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{L'\}$.
    end if
  end for
until $\mathcal{V} = \emptyset$
return $\{L \in \mathcal{L} : v(L) = i, \overline{c}(L) < z_{ub} - z_{lb}^{2e}\}$

The complete dataset can be found in http://neo.lcc.uma.es/radi-aeb/WebVRP. The algorithm has been coded in C++, compiled using the GNU g++ compiler v4.8 and run on an Intel Xeon E5462 2.8 GHz with 16 GB of RAM. Moreover, CPLEX 12.5 was used to both solve the linear relaxation of the set partition formulation for the column generation and its integer version for the last part of the proposed method.

In Tables 1–5 we present the computational results obtained with the proposed algorithm for both problems considered in this study. In these tables, column labeled "UB" represents the best known solution of the problem, as reported by previous methods. Under column labeled "Cutting Planes" we report the lower bound (under column "z_{lb}^1"), optimality gap in % (under column "gap^1"), the number of edges deleted by the reduced cost fixing in % (under column "%del"), and the CPU time (in seconds) taken to solve the LP relaxation of problem (1)–(7) (under column "t^1"). Under column labeled "Column and Cut Generation" we report the results obtained by the column-and-cut generation method. Columns labeled "z_{lb}^{2e}" and "gap^{2e}" represent the lower bound and the relative gap (in %) achieved at the end of the LP relaxation, just before the execution of a successful column enumeration. Column labeled "$|\mathcal{P}^{2e}|$" represents the number of routes generated by the enumeration algorithm. Columns labeled "z_{lb}^{2f}", "gap^{2f}" and "$|\mathcal{P}^{2f}|$" represent, respectively, the final lower bound, the final relative gap (in %) and the final number of columns kept after the last major iteration right after a successful execution of the column enumeration procedure. Column labeled "t^2" represents the overall CPU time (in seconds) spent in the column generation, variable fixing and column enumeration. Columns labeled "z_{lb}^3", "gap^3" and "t^3" represent the final lower bound, the final relative gap (in %) and the CPU time (in seconds) spent by CPLEX on the resolution of the final integer program, respectively. Finally, column labeled "t^{tot}" represents the total CPU time (in seconds) spent by our exact method. We use bold characters for the instances solved to optimality. As shown in these tables, our exact method is effective for solving all but two instances

Table 1
Detailed results on class A for the CVRP.

Instance	UB	Cutting planes				Column and cut generation										t^{tot}
		z^1_{lb}	gap^1	%del	t^1	z^{2e}_{lb}	gap^{2e}	$\lvert\mathcal{P}^{2e}\rvert$	z^{2f}_{lb}	gap^{2f}	$\lvert\mathcal{P}^{2f}\rvert$	t^2	z^3_{lb}	gap^3	t^3	
A-n37-k5	669	663.31	0.9	73.0	0.8	666.91	0.3	168	669	0.0	0	3.6	669	0.0	0.0	4.3
A-n37-k6	949	923.24	2.7	16.4	2.3	940.71	0.9	861	946.05	0.3	109	7.3	949	0.0	0.1	9.7
A-n38-k5	730	716.67	1.8	46.7	1.1	723.42	0.9	956	728.80	0.2	74	6.5	730	0.0	0.1	7.7
A-n39-k5	822	807.53	1.8	37.5	7.0	819.12	0.4	539	821.56	0.1	0	9.8	822	0.0	0.0	16.8
A-n39-k6	831	814.00	2.0	43.3	1.9	825.15	0.7	563	830.82	0.0	0	4.5	831	0.0	0.0	6.4
A-n44-k6	937	916.89	2.1	34.5	5.2	935.30	0.2	101	937	0.0	0	5.3	937	0.0	0.0	10.5
A-n45-k6	944	927.04	1.8	44.6	2.7	941.71	0.2	114	944	0.0	0	10.0	944	0.0	0.0	12.8
A-n45-k7	1146	1108.52	3.3	4.6	7.3	1142.35	0.3	597	1145.20	0.1	0	10.3	1146	0.0	0.0	17.6
A-n46-k7	914	908.26	0.6	72.1	2.5	914	0.0	0	914	0.0	0	3.4	914	0.0	0.0	5.8
A-n48-k7	1073	1051.20	2.0	28.5	3.2	1072.02	0.1	0	1072.02	0.1	0	8.3	1073	0.0	0.0	11.5
A-n53-k7	1010	996.65	1.3	54.1	4.4	1004.38	0.6	2 622	1010	0.0	0	18.1	1010	0.0	0.0	22.4
A-n54-k7	1167	1131.20	3.1	6.5	6.5	1156.71	0.9	33 633	1166.12	0.1	0	40.1	1167	0.0	0.0	46.7
A-n55-k9	1073	1056.33	1.6	47.3	2.7	1068.55	0.4	811	1073	0.0	0	12.3	1073	0.0	0.0	15.0
A-n60-k9	1354	1316.54	2.8	3.4	7.4	1345.68	0.6	14 674	1353.06	0.1	0	31.4	1354	0.0	0.0	38.8
A-n61-k9	1034	1007.33	2.6	31.9	6.5	1024.61	0.9	6 235	1032.00	0.2	192	26.2	1034	0.0	0.4	33.0
A-n62-k8	1288	1247.28	3.2	2.1	31.0	1282.05	0.5	17 431	1287.02	0.1	0	64.4	1288	0.0	0.0	95.4
A-n63-k10	1314	1263.93	3.8	0.9	9.8	1303.67	0.8	8 845	1309.71	0.3	731	574.6	1314	0.0	1.3	585.7
A-n63-k9	1616	1575.54	2.5	10.8	14.1	1610.82	0.3	1 798	1616	0.0	0	35.6	1616	0.0	0.0	49.7
A-n64-k9	1401	1348.55	3.7	2.5	20.9	1389.86	0.8	31 389	1396.12	0.3	1169	59.0	1401	0.0	3.3	83.2
A-n65-k9	1174	1153.12	1.8	49.3	8.1	1169.05	0.4	1 705	1174	0.0	0	27.4	1174	0.0	0.0	35.5
A-n69-k9	1159	1112.12	4.0	14.2	15.0	1145.26	1.2	53 546	1157.27	0.1	166	47.2	1159	0.0	0.4	62.6
A-n80-k10	1763	1707.21	3.2	2.6	39.1	1757.11	0.3	8 507	1762.08	0.1	0	96.4	1763	0.0	0.0	135.5
Average			2.4	28.5	9.1		0.5	8 413		0.1	111	50.1		0.0	0.2	59.4

Table 2
Detailed results on class B for the CVRP.

Instance	UB	Cutting planes				Column and cut generation										t^{tot}
		z^1_{lb}	gap^1	%del	t^1	z^{2e}_{lb}	gap^{2e}	$\lvert\mathcal{P}^{2e}\rvert$	z^{2f}_{lb}	gap^{2f}	$\lvert\mathcal{P}^{2f}\rvert$	t^2	z^3_{lb}	gap^3	t^3	
B-n38-k6	805	800.14	0.6	62.7	0.6	804.09	0.1	0	804.09	0.1	0	3.2	805	0.0	0.0	3.8
B-n39-k5	549	549	0.0	100.0	0.0	549	0.0	0	549	0.0	0	0.0	549	0.0	0.0	0.0
B-n41-k6	829	826.18	0.3	75.0	0.6	828.25	0.1	0	828.25	0.1	0	1.9	829	0.0	0.0	2.5
B-n43-k6	742	733.29	1.2	55.6	0.9	737.54	0.6	5 261	740.23	0.2	233	13.4	742	0.0	0.3	14.6
B-n44-k7	909	909	0.0	100.0	0.4	909	0.0	0	909	0.0	0	0.0	909	0.0	0.0	0.4
B-n45-k5	751	747.42	0.5	70.3	1.1	750.24	0.1	0	750.24	0.1	0	7.3	751	0.0	0.0	8.3
B-n45-k6	678	673.10	0.7	51.9	4.2	677.37	0.1	0	677.37	0.1	0	6.2	678	0.0	0.0	10.4
B-n50-k7	741	741	0.0	100.0	0.2	741	0.0	0	741	0.0	0	0.0	741	0.0	0.0	0.2
B-n50-k8	1312	1280.51	2.4	0.8	3.7	1303.72	0.6	141 084	1310.02	0.2	626	63.2	1312	0.0	0.9	67.9
B-n51-k7	1032	1024.90	0.7	59.8	1.2	1027.17	0.5	5 503	1032	0.0	0	21.4	1032	0.0	0.0	22.6
B-n52-k7	747	745.83	0.2	87.5	0.7	746.67	0.0	0	746.67	0.0	0	1.7	747	0.0	0.0	2.4
B-n56-k7	707	703.44	0.5	72.3	1.8	705.00	0.3	286	705.00	0.3	286	39.6	707	0.0	0.1	41.6
B-n57-k7	1153	1148.93	0.4	58.8	4.4	1152.05	0.1	0	1152.05	0.1	0	21.6	1153	0.0	0.0	26.0
B-n57-k9	1598	1587.00	0.7	36.0	5.3	1596.18	0.1	304	1598	0.0	0	22.7	1598	0.0	0.0	27.9
B-n63-k10	1496	1479.31	1.1	32.1	5.2	1487.42	0.6	22 654	1496	0.0	0	33.7	1496	0.0	0.0	38.9
B-n64-k9	861	859.24	0.2	86.3	3.0	860.70	0.0	0	860.70	0.0	0	7.1	861	0.0	0.0	10.0
B-n66-k9	1316	1297.01	1.4	32.2	13.8	1309.21	0.5	198 366	1315.07	0.1	0	72.0	1316	0.0	0.0	85.8
B-n67-k10	1032	1023.83	0.8	66.7	3.6	1027.94	0.4	3 270	1032	0.0	0	23.6	1032	0.0	0.0	27.2
B-n68-k9	1272	1256.67	1.2	36.7	5.0	1263.92	0.6	402 367	1267.98	0.3	5906	153.2	1272	0.0	19.8	178.0
B-n78-k10	1221	1203.87	1.4	24.3	12.4	1217.59	0.3	2 946	1221	0.0	0	96.1	1221	0.0	0.0	108.5
Average			0.7	60.4	3.4		0.3	39 102.1		0.1	352.6	29.4		0.0	1.1	33.8

for the CVRP, namely instances M-n200-k16 and M-n200-k17. These instances remain open. It is important to note that, to the best of our knowledge, this is the first time that a column generation approach was able to solve instance F-n135-k7. Furthermore, our algorithm has solved instance M-n151-k12 for the first time. These results show that our method takes advantage of the enhancements introduced with respect to previous column generation methods. For the MDVRP, our algorithm is able to solve 20 instances out of the 23 considered in our study, including 13 instances that have been solved for the first time.

In Tables 6–7 we present comparative results against state-of-the-art exact methods for the two classes of problems considered in this study. For the CVRP, we consider methods FLLPRUW'06 [9], BCM'08 [10], BBMR'10 [27] and BMR'11 [11]. For the proposed method, we report the results under column CM'14. For each of the methods, we report the number of instances solved to optimality, the average relative gap (in %) at the LP relaxation of the set-partitioning formulation and the average CPU time (in seconds) on the instances that are solved by all methods. For the MDVRP, we compare our results against method BM'09 [5] which is also based on a set-partitioning formulation of the problem. We provide a detailed comparison

Table 3
Detailed results on class P for the CVRP.

Instance	UB	Cutting planes				Column and cut generation										t^{tot}
		z^1_{lb}	gap^1	%del	t^1	z^{2e}_{lb}	gap^{2e}	$\lvert\mathcal{P}^{2e}\rvert$	z^{2f}_{lb}	gap^{2f}	$\lvert\mathcal{P}^{2f}\rvert$	t^2	z^3_{lb}	gap^3	t^3	
P-n40-k5	458	456.48	0.3	86.4	1.9	457.67	0.1	0	457.67	0.1	0	0.9	458	0.0	0.0	2.8
P-n45-k5	510	503.90	1.2	77.5	1.4	507.06	0.6	1 168	509.30	0.1	0	6.3	510	0.0	0.0	7.8
P-n50-k10	696	666.04	4.3	14.5	8.2	689.58	0.9	965	694.91	0.2	84	5.0	696	0.0	0.1	13.2
P-n50-k7	554	538.43	2.8	54.4	2.9	550.98	0.5	776	553.71	0.1	0	7.5	554	0.0	0.0	10.4
P-n50-k8	631	599.17	5.0	15.5	6.7	617.84	2.1	8 709	626.30	0.7	624	14.5	631	0.0	0.7	21.9
P-n51-k10	741	712.77	3.8	19.1	6.7	736.45	0.6	507	741	0.0	0	5.3	741	0.0	0.0	11.9
P-n55-k10	694	658.76	5.1	10.8	11.2	681.87	1.7	13 399	689.24	0.7	1011	13.8	694	0.0	1.3	26.3
P-n55-k15	989	904.23	8.6	0.0	64.7	972.85	1.6	1 362	984.64	0.4	216	12.6	989	0.0	0.3	77.6
P-n55-k7	568	546.56	3.8	38.7	6.6	559.35	1.5	31 334	565.67	0.4	448	19.7	568	0.0	0.4	26.7
P-n55-k8	588	572.13	2.7	52.1	6.4	581.30	1.1	9 621	586.34	0.3	148	12.1	588	0.0	0.2	18.6
P-n60-k10	744	715.81	3.8	25.3	11.0	739.68	0.6	901	743.93	0.0	0	9.4	744	0.0	0.0	20.4
P-n60-k15	968	925.80	4.4	2.5	16.3	963.59	0.5	365	967.54	0.0	0	4.4	968	0.0	0.0	20.7
P-n65-k10	792	764.56	3.5	29.6	19.3	788.25	0.5	782	791.39	0.1	0	11.9	792	0.0	0.0	31.2
P-n70-k10	827	793.46	4.1	18.4	22.9	815.35	1.4	96 160	823.19	0.5	1343	49.1	827	0.0	4.0	76.1
P-n76-k4	593	588.41	0.8	89.8	5.8	591.59	0.2	1 497	592.47	0.1	0	851.0	593	0.0	0.0	856.7
P-n76-k5	627	616.17	1.7	71.4	8.0	622.61	0.7	142 667	627	0.0	0	851.6	627	0.0	0.0	859.6
P-n101-k4	681	678.37	0.4	94.4	13.7	679.88	0.2	1 365	681	0.0	0	3675.4	681	0.0	0.0	3689.1
Average			3.3	41.2	12.6		0.9	18 328		0.2	228	326.5		0.0	0.4	339.5

Table 4
Detailed results on classes E–F–M for the CVRP.

Instance	UB	Cutting planes				Column and cut generation										t^{tot}
		z^1_{lb}	gap^1	%del	t^1	z^{2e}_{lb}	gap^{2e}	$\lvert\mathcal{P}^{2e}\rvert$	z^{2f}_{lb}	gap^{2f}	$\lvert\mathcal{P}^{2f}\rvert$	t^2	z^3_{lb}	gap^3	t^3	
E-n51-k5	521	517.87	0.6	86.7	2.6	519.27	0.3	824	521	0.0	0	8.6	521	0.0	0.0	11.2
E-n76-k7	682	664.92	2.5	56.0	15.2	677.21	0.7	28 672	681.28	0.1	0	751.2	682	0.0	0.0	766.4
E-n76-k8	735	715.23	2.7	49.5	24.9	727.23	1.1	234 919	734.14	0.1	0	112.7	735	0.0	0.0	137.6
E-n76-k10	830	796.38	4.1	20.8	38.1	817.41	1.5	292 104	826.56	0.4	1 608	90.5	830	0.0	5.7	134.3
E-n76-k14	1021	965.21	5.5	0.1	62.7	1007.54	1.3	39 091	1014.76	0.6	2 721	37.8	1021	0.0	6.7	107.2
E-n101-k8	815	799.79	1.9	65.1	45.3	809.91	0.6	1 328 532	814.64	0.0	0	1514.4	815	0.0	0.0	1559.7
E-n101-k14	1067	1022.44	4.2	6.6	166.1	1054.26	1.2	1 980 384	1063.51	0.3	4 126	404.0	1067	0.0	17.8	587.9
F-n45-k4	724	724	0.0	100.0	0.4	724	0.0	0	724	0.0	0	0.0	724	0.0	0.0	0.4
F-n72-k4	237	237	0.0	100.0	0.0	237	0.0	0	237	0.0	0	0.0	237	0.0	0.0	0.0
F-n135-k7[a]	1162	1159.51	0.2	91.9	87.3	1161.04	0.1	0	1161.04	0.1	0	83 078.5	1162	0.0	0.0	83 165.8
M-n101-k10	820	820	0.0	100.0	0.0	820	0.0	0	820	0.0	0	0.0	820	0.0	0.0	0.0
M-n121-k7	1034	1012.96	2.0	18.9	143.2	1032.54	0.1	21 840	1033.15	0.1	0	1921.7	1034	0.0	0.0	2064.9
M-n151-k12[b]	1015	972.88	4.1	10.5	810.9	1011.20	0.4	177 781	1012.55	0.2	11 509	9131.4	1015	0.0	167.7	10 110.0
M-n200-k16	1278	1202.06	5.9	0.0	2599.1	1266.90	0.9	0	1266.90	0.9	0	429 400.9	1266.90	0.9	0.0	432 000
M-n200-k17	1275	1196.21	6.2	0.0	1648.2	1269.19	0.5	0	1269.19	0.5	0	430 351.8	1269.19	0.5	0.0	432 000
Average			2.7	47.1	376.3		0.6	315 704[c]		0.2	1 536[c]	7465.4[c]		0.1	15.2[c]	7588.1[c]

[a] Optimality proven for the first time using column generation.
[b] Optimality proven for the first time.
[c] Average restricted to solved instances.

including only the instances considered in both studies. Moreover, the average CPU times are computed by taking into account only the instances solved to optimality by both methods. According to SPEC (http://www.spec.org/benchmarks.html), the machine used by us is about 4 times faster than the Pentium 4 2.6 GHz used by FLLPRUW'06 and the Pentium 4 2.4 GHz used by method BCM'08, 2 times faster than the AMD Athlon X2 4200 2.6 GHz used by method BM'09, 50% faster than the Intel Core 2 Duo P8400 2.26 GHz used by BBMR'10 and 25% faster than the Intel Xeon X7350 2.93 GHz used by method BMR'11. As the results show, our method is competitive against other exact methods for the CVRP, being able to obtain the tightest average gaps at the LP relaxation for all classes of instances. In terms of computing time, our method is comparable to methods BCM'08, BBMR'10 and BMR'11. However, it is more robust than these three methods as it solves more problems and achieves better bounds. When compared to method FLLPRUW'06, our method in general is much faster in most instances, however it spends a large amount of time to solve problem F-n135-k7 that is solved by the authors using a branch-and-cut method. For the MDVRP, our method is more robust than method BM'09. In fact, it solves all instances also solved by method BM'09 plus two instances that their algorithm fails to solve and 11 others which are not considered by the authors. For the commonly solved instances, the CPU times are comparable but our method in general achieves tighter lower bounds at the LP relaxation.

Finally, Table 8 shows the improvements obtained by the new family of valid inequalities, the *k*-CEC, introduced in Section 3.3. To isolate their impact, we have run additional computational experiments in which we do not include any class of non-robust valid inequality besides the *k*-CEC themselves. For each problem and each class of instances, we solve the linear relaxation with and without including the inequalities. We show a summary which considers only the instances which are not solved to optimality before the generation of any such inequalities. Therefore, the table shows for each class

Table 5
Detailed results on selected MDVRP instances.

Instance	UB	Cutting planes				Column and cut generation										t^{tot}
		z_{lb}^1	gap^1	%del	t^1	z_{lb}^{2e}	gap^{2e}	$\lvert\mathcal{P}^{2e}\rvert$	z_{lb}^{2f}	gap^{2f}	$\lvert\mathcal{P}^{2f}\rvert$	t^2	z_{lb}^3	gap^3	t^3	
p01	576.87	542.74	5.9	7.5	7.3	575.95	0.2	116	576.87	0.0	72	9.1	576.87	0.0	0.0	16.4
p02	473.54	452.20	4.5	43.4	3.1	468.64	1.0	3 621	472.55	0.2	157	33.0	473.53	0.0	0.1	36.1
p03	641.19	613.34	4.3	32.0	19.3	634.84	1.0	6 830	640.22	0.2	326	49.6	641.19	0.0	0.2	69.0
p04	1001.04	948.46	5.3	0.4	206.3	988.61	1.2	1 129 501	997.82	0.3	6 004	450.4	1001.04	0.0	35.7	692.4
p05[a]	750.03	731.20	2.5	57.4	20.4	744.85	0.7	1 540 616	748.97	0.1	3 096	2287.7	750.03	0.0	17.0	2325.1
p06	876.5	829.48	5.4	3.0	161.4	868.09	1.0	94 533	875.32	0.1	580	144.7	876.50	0.0	0.5	306.6
p07	881.97	830.26	5.9	0.2	85.6	872.56	1.1	146 724	881.87	0.0	154	222.6	881.97	0.0	0.4	308.5
p12	1318.95	1273.16	3.5	50.0	30.6	1309.61	0.7	14 002	1313.80	0.4	573	834.4	1318.95	0.0	0.5	865.5
p13[a]	1318.95	1273.16	3.5	50.0	30.5	1318.95	0.0	0	1318.95	0.0	0	20.2	1318.95	0.0	0.0	50.7
p14[a]	1360.12	1273.16	6.4	18.9	28.9	1360.12	0.0	63	1360.12	0.0	63	33.5	1360.12	0.0	0.0	62.3
p15[a]	2505.42	2382.16	4.9	17.2	416.9	2498.46	0.3	7 118	2505.42	0.0	244	1723.6	2505.42	0.0	0.0	2140.5
p16[a]	2572.23	2382.16	7.4	0.4	423.7	2572.23	0.0	0	2572.23	0.0	0	388.7	2572.23	0.0	0.0	812.4
p17[a]	2709.1	2382.16	12.1	0.0	440.2	2709.09	0.0	194	2709.09	0.0	194	339.0	2709.09	0.0	0.0	779.2
p18[a]	3702.85	3490.58	5.7	7.6	2498.2	3689.80	0.4	87 905	3699.78	0.1	1 019	11 010.5	3702.85	0.0	2.3	13 511.0
p19[a]	3827.06	3490.58	8.8	0.0	2491.3	3822.92	0.1	4 891	3827.06	0.0	227	1085.4	3827.06	0.0	0.0	3576.8
p20[a]	4058.07	3490.58	14.0	0.0	2737.7	4058.07	0.0	0	4058.07	0.0	0	986.5	4058.07	0.0	0.0	3724.1
pr01[a]	861.319	861.32	0.0	100.0	0.2	861.32	0.0	0	861.32	0.0	0	0.0	861.32	0.0	0.0	0.2
pr02	1307.34	1247.16	4.6	16.8	10.8	1300.02	0.6	0	1300.02	0.6	0	431 989.2	1300.02	0.6	0.0	432 000
pr03[a]	1803.8	1696.50	5.9	6.4	101.3	1790.32	0.7	2 359 688	1798.50	0.3	25 759	179 351.7	1803.80	0.0	224.0	179 677.0
pr04	2058.31	1928.11	6.3	0.1	886.7	2043.08	0.7	0	2043.08	0.7	0	431 113.3	2043.08	0.7	0.0	432 000
pr07[a]	1089.56	1056.69	3.0	61.4	3.2	1082.07	0.7	9 803	1089.56	0.0	105	1770.1	1089.56	0.0	0.0	1773.3
pr08[a]	1664.85	1560.51	6.3	3.5	104.5	1654.49	0.6	281 652	1662.61	0.1	1 492	68 411.1	1664.85	0.0	4.9	68 520.5
pr09	2133.2	1993.77	6.5	0.2	2102.1	2119.56	0.6	0	2119.56	0.6	0	429 897.9	2119.56	0.6	0.0	432 000
Average			5.8	20.7	557.0		0.5	284 363[b]		0.2	2 003[b]	13 458[b]		0.1	14[b]	13 962[b]

[a] Optimality proven for the first time.
[b] Average restricted to solved instances.

Table 6
Summary results for the CVRP.

Class	NP	FLLPRUW'06			BCM'08			BBMR'10			BMR'11			CM'14		
		opt	gap	t	opt	gap	t	opt	gap	t	opt	gap	t	opt	gap	t
A	22	**22**	0.81	1961.18	**22**	0.20	118.07	**22**	0.30	22.00	**22**	0.13	30.27	**22**	**0.09**	59.4
B	20	**20**	0.47	4763.00	**20**	0.16	417.17	**20**	0.10	66.00	**20**	**0.06**	66.75	**20**	0.08	33.8
E–M	12	9	1.19	42 614.50	8	0.69	1025.13	9	0.50	249.00	9	0.49	268.13	**10**	**0.27**	596.6
F	3	**3**	0.14	64.50							2	0.11	163.50	**3**	**0.03**	2.35
P	17	**17**	1.07	3572.53	15	0.39	272.07	**17**	0.20	54.00	**17**	0.23	39.80	**17**	**0.18**	60.09
Total	74	71			65			68			70			**72**		
Average			0.84	8198.62		0.36	357.28		0.28	70.86		0.21	72.97		**0.14**	114.4

Table 7
Comparison for the MDVRP.

Instance	BM'09			CM'14		
	gap_b	gap_f	t	gap_b	gap_f	t
p01	0.0	0.0	10.9	0.0	0.0	16.4
p02	0.3	0.0	54.4	0.2	0.0	36.1
p03	0.1	0.0	92.4	0.2	0.0	69.0
p04	0.7	0.0	5106.9	0.3	0.0	692.4
p05	2.5	2.5	1647.6	0.1	0.0	2325.1
p06	0.4	0.0	104.3	0.1	0.0	306.6
p07	0.5	0.0	294.4	0.0	0.0	308.5
p12	1.4	0.0	463.7	0.4	0.0	865.5
p15	0.8	0.8	2147.9	0.0	0.0	2140.5
Average	0.7	0.3	875.24	**0.1**	**0.0**	751.1
Opt		7/9			**9/9**	

the number of considered instances (column labeled "NP"), the average overall times for the column generation without generating the inequalities (column labeled "t_1") and with the generation of the inequalities (column labeled "t_2"), and on the last column (labeled "gap closed"), it shows the average percentage on which the relative gaps were closed (with respect to each respective upper bound) by the addition of the k-CEC. This improvement is computed as $\left(z_{lb}^1 - z_{lb}^0\right) / \left(z_{ub} - z_{lb}^0\right) \times 100$, where z_{lb}^0 and z_{lb}^1 are the lower bounds computed without and with the addition of constraints k-CEC, respectively; and z_{ub} is the value of the best known solution. Note that for all classes of instances there is a noticeable improvement, especially

Table 8
k-CEC improvement on CVRP and MDVRP instances.

Problem	Class	NP	t_1	t_2	gap closed
CVRP	A	21	16.7	47.8	13.5
	B	17	18.4	30.0	2.0
	E	7	39.6	263.6	3.1
	F	1	20 355.9	20 682.4	1.8
	M	4	1071.6	1371.7	2.4
	P	17	31.2	91.6	5.3
Average			389.8	463.9	6.6
MDVRP	p	14	348.3	1209.6	39.1
	pr	6	24 066.4	30 000.0	11.5
Average			7463.7	9846.7	30.8

for class A of the CVRP and both classes of the MDVRP. The inequalities were able to improve the gap in more than 30% on average for all the MDVRP instances included in our study.

7. Concluding remarks

In this article we have presented a new exact method for the multi-depot vehicle routing problem (MDVRP) under capacity and route length constraints. The MDVRP is modeled using ad-hoc vehicle-flow and set-partitioning formulations, and solved the first by the cutting planes method and the second by column-and-cut generation. Several families of valid inequalities are used to strengthen the lower bounds achieved by both formulations, including a new family of valid inequalities that is shown to forbid cycles of an arbitrary length. As a result, our method is able to produce the tightest lower bounds for two classes of problems, the MDVRP and the CVRP, on the instances considered in our study when compared to state-of-the-art methods, and is able to solve 14 open instances, one for the CVRP and 13 for the MDVRP. Besides, our method is able to solve to optimality the CVRP instance F-n135-k7 for the first time using a column generation approach. We can identify several avenues of future research. For instance, improving the cutting planes would result in an even faster and more robust pricing algorithm because of the initial variable fixing, which would be of great use for solving poorly constrained instances. Another avenue of further research would be the generalization of the *k*-CEC and SDC cuts to forbid other types of cycles, in the spirit of the *ng*-routes relaxation. Finally, let us remark that the SDC and *k*-CEC are two families of valid inequalities that can be adapted to several other classes of vehicle routing problems, like vehicle routing problems with time windows, multiple-echelon vehicle routing problems or multiple-period vehicle routing problems. Thus, future exact algorithms for these classes of problems should consider the inclusion of these cuts.

References

[1] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971.
[2] J.-F. Cordeau, M. Maischberger, A parallel iterated tabu search heuristic for vehicle routing problems, Comput. Oper. Res. 39 (2012) 2033–2050.
[3] T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems, Oper. Res. 60 (2013) 611–624.
[4] J. Renaud, G. Laporte, F.F. Boctor, A tabu search heuristic for the multi-depot vehicle routing problem, Comput. Oper. Res. 23 (1996) 229–235.
[5] R. Baldacci, A. Mingozzi, A unified exact method for solving different classes of vehicle routing problems, Math. Program. 120 (2009) 347–380.
[6] N. Christofides, A. Mingozzi, P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations, Math. Program. 20 (1981) 255–282.
[7] R. Baldacci, E. Bartolini, A. Mingozzi, A. Valletta, An exact algorithm for the period routing problem, Oper. Res. 59 (2011) 228–241.
[8] J. Lysgaard, A.N. Letchford, R.W. Eglese, A new branch-and-cut algorithm for the capacitated vehicle routing problem, Math. Program. 100 (2004) 423–445.
[9] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R.F. Werneck, Robust branch-and-cut-and-price for the capacitated vehicle routing problem, Math. Program. A 106 (2006) 491–511.
[10] R. Baldacci, N. Christofides, A. Mingozzi, An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts, Math. Program. 115 (2008) 351–385.
[11] R. Baldacci, A. Mingozzi, R. Roberti, New route relaxation and pricing strategies for the vehicle routing problem, Oper. Res. 59 (2011) 1269–1283.
[12] M. Dror, Note on the complexity of the shortest path models for column generation in VRPTW, Oper. Res. 42 (1994) 977–978.
[13] D. Houck, J. Picard, M. Queyranne, R. Vemuganti, The travelling salesman problem as a constrained shortest path problem: theory and computational experience, Opsearch 17 (1980) 93–109.
[14] M. Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Oper. Res. 40 (1992) 342–354.
[15] S. Irnich, D. Villeneuve, The shortest path problem with resource constraints and *k*-cycle elimination for $k \geq 3$, INFORMS J. Comput. 18 (2006) 391–406.
[16] G. Righini, M. Salani, New dynamic programming algorithms for the resource constrained elementary shortest path problem, Networks 51 (2008) 155–170.
[17] C. Contardo, J.-F. Cordeau, B. Gendron, An exact algorithm based on cut-and-column generation for the capacitated location-routing problem, INFORMS J. Comput. 26 (2014) 88–102.
[18] C. Contardo, G. Desaulniers, F. Lessard, Reaching the elementary lower bound in the vehicle routing problem with time windows, Technical Report G-2013–50, Cahiers du GERAD, 2013.
[19] M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger, Subset-row inequalities applied to the vehicle-routing problem with time windows, Oper. Res. 56 (2008) 497–511.
[20] R.E. Gomory, Outline of an algorithm for integer solutions to linear problems, Bull. Amer. Math. Soc. 64 (1958) 265–279.

[21] J.M. Belenguer, E. Benavent, C. Prins, C. Prodhon, R. Wolfler-Calvo, A branch-and-cut algorithm for the capacitated location routing problem, Comput. Oper. Res. 38 (2011) 931–941.
[22] J. Lysgaard, CVRPSEP: a package of separation routines for the capacitated vehicle routing problem, December 2003.
[23] C. Contardo, J.-F. Cordeau, B. Gendron, A computational comparison of flow formulations for the capacitated location-routing problem, Discrete Optim. 10 (2013) 263–295.
[24] N.L. Boland, J. Dethridge, I. Dumitrescu, Accelerated label setting algorithms for the elementary resource constrained shortest path problem, Oper. Res. Lett. 34 (2006) 58–68.
[25] R. Martinelli, Exact algorithms for arc and node routing problems, Ph.D. Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil, 2012.
[26] S. Irnich, G. Desaulniers, J. Desrosiers, A. Hadjar, Path-reduced costs for eliminating arcs in routing and scheduling, INFORMS J. Comput. 22 (2010) 297–313.
[27] R. Baldacci, E. Bartolini, A. Mingozzi, R. Roberti, An exact solution framework for a broad class of vehicle routing problems, Comput. Manag. Sci. 7 (2010) 229–268.
[28] R. Baldacci, A. Mingozzi, R. Wolfler-Calvo, An exact method for the capacitated location-routing problem, Oper. Res. 59 (2011) 1284–1296.
[29] P. Augerat, Approche polyhédrale du problème de tournées de véhicules, Ph.D. Thesis, Institut National Polytechnique de Grenoble, France, 1995.
[30] N. Christofides, S. Eilon, An algorithm for the vehicle dispatching problem, Oper. Res. Q. 20 (1969) 309–318.
[31] M.L. Fisher, Optimal solution of vehicle routing problems using minimum k-trees, Oper. Res. 42 (1994) 626–642.
[32] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, in: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (Eds.), Combinatorial Optimization, John Wiley, Chichester, UK, 1979.
[33] J.-F. Cordeau, M. Gendreau, G. Laporte, A tabu search heuristic for periodic and multi-depot vehicle routing problem, Networks 30 (1997) 105–119.