A heuristic with a performance guarantee for the Commodity constrained Split Delivery Vehicle Routing Problem

Matteo Petris, Claudia Archetti, Diego Cattaruzza, Maxime Ogier, Frédéric Semet

A heuristic with a performance guarantee for the Commodity constrained Split Delivery Vehicle Routing Problem

Matteo Petris[a,*], Claudia Archetti[b], Diego Cattaruzza[c], Maxime Ogier[c], Frédéric Semet[c]

[a] *Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
[b] *Department of Information Systems, Decision Sciences and Statistics, ESSEC Business School, Cergy-Pontoise, France*
[c] *Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*

Abstract

The Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP) is a routing problem where customer demands are composed of multiple commodities. A fleet of capacitated vehicles must serve customer demands in a way that minimizes the total routing costs. Vehicles can transport any set of commodities and customers are allowed to be visited multiple times. However, the demand for a single commodity must be delivered by one vehicle only.

In this work, we developed a heuristic with a performance guarantee to solve the C-SDVRP. The proposed heuristic is based on a set covering formulation, where the exponentially-many variables correspond to routes. First, a subset of the variables is obtained by solving the linear relaxation of the formulation by means of a column generation approach which embeds a new pricing heuristic aimed to reduce the computational time. Solving the linear relaxation gives a valid lower bound used as a performance guarantee for the heuristic. Then, we devise a restricted master heuristic to provide good upper bounds: the formulation is restricted to the subset of variables found so far and solved as an integer program with a commercial solver. A local search based on a mathematical programming operator is applied to improve the solution.

We test the heuristic algorithm on benchmark instances from the literature. Several new (best-known) solutions are found in reasonable computational time. The comparison with the state of the art heuristics for solving C-SDVRP shows that our approach significantly improves the solution time, while keeping a comparable solution quality.

Keywords: `Vehicle routing problems, Multiple commodities, Split delivery, Column generation, Matheuristic, Pricing heuristic`

*Corresponding author
Email addresses: `matteo.petris@inria.fr` (Matteo Petris), `archetti@essec.edu` (Claudia Archetti), `diego.cattaruzza@centralelille.fr` (Diego Cattaruzza), `maxime.ogier@centralelille.fr` (Maxime Ogier), `frederic.semet@centralelille.fr` (Frédéric Semet)

1. Introduction

Splitting customer demands has proven to be beneficial in reducing the transportation costs and the number of vehicles [see 2, 6]. A first work in this direction is the article by [19]. The authors introduced the *Split Delivery Vehicle Routing Problem* (SDVRP), where customer demands are composed of a single commodity and can be split among any number of vehicles. This problem and its variants have been widely studied and exact [e.g. 9, 16, 4, 29] and heuristic algorithms [e.g. 3, 38, 14, 11] were proposed. Among these, [13] studies a particular case of the SDVRP, where customer demands are discretised a priori.

Although this delivery policy brings remarkable cost savings when compared with the policy where no splits are allowed, it is hardly applicable from a practical point of view. Indeed, customers are usually not keen to accept an unconstrained split delivery [6]. One step in the direction of making the split deliveries more adherent to real-world logistics is made in [24]. The authors proposed a variant of the SDVRP where the quantity delivered to each customer has to be greater than a preset minimum amount.

Finally, under a multi-commodity setting, a delivery policy that might reduce customer inconvenience due to split deliveries is to allow demands to be split by commodity. Whenever a vehicle delivers a commodity to a customer, the entire quantity associated with the commodity has to be provided. This policy was firstly studied in the *Discrete Split Delivery Vehicle Routing Problem* (DSDVRP) proposed in [31] to deal with a real-life case study. The problem was formally introduced in the literature under the name of *Commodity constrained Split Delivery Vehicle Routing Problem* (C-SDVRP) in [6]. In the C-SDVRP, a minimum cost set of routes have to be determined such that the customer demands, composed of multiple commodities, are met, and the capacity of the vehicles is respected. The authors showed that the C-SDVRP is a relaxation of the *Capacitated Vehicle Routing Problem* (CVRP) where all commodities of each customer are delivered with a single vehicle, and a restriction of the SDVRP. In addition, they proposed an in-depth analysis to assess the benefits of the C-SDVRP in terms of cost savings and applicability in comparison with the CVRP and the SDVRP. To do so, they introduced the first compact mathematical formulation and devised a branch-and-cut and a heuristic algorithm to solve it. Finally, they introduced a first set of benchmark instances characterised by 15, 20, 40, 60, 80 or 100 customers requiring 2 or 3 commodities.

Despite its practical relevance, the literature on the C-SDVRP and its variants is quite limited. An exact approach for the C-SDVRP was proposed in [5]. Specifically, the authors modelled the problem by means of a set covering formulation and devised a first branch-price-and-cut (BPC) algorithm. They formulate the pricing problem as an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC) and solve the *ng-path* relaxation by means of a label setting dynamic programming technique. [22] enhanced the performances of the BPC algorithm of [5] by embedding new procedures as the implicit bidirectional labelling search to solve the ESPPRC, the separation of non-robust valid inequalities to strengthen the lower bound, and the stabilization of the column generation procedure via dual-optimal inequalities. The authors extended the test-bed introduced in [6]

with 336 new instances with 4, 5 and 6 commodities. The enhanced BPC algorithm outperformed the one of [5], being faster and providing several new optima and better lower bounds.

Conversely, [23] focused on a heuristic algorithm for the C-SDVRP and proposed an adaptive large neighbourhood search (ALNS) that exploits the inherent characteristics of the problem. Specifically, several existing local search moves were adapted to better deal with the multi-commodity aspect, and a mathematical programming operator was developed to reassign commodities to routes. The authors assess the performance of their ALNS on the test-bed introduced in [6]. The ALNS found the optimal value for 81 out of the 84 instances with 15 and 20 customers, and provided 344 new best-known solutions for the 380 instances with more than 40 customers. In [39] the authors propose a small and large neighbourhood search (SLNS) which is capable of solving different variants of routing problems, among those the C-SDVRP. The SLNS is compared with the ALNS proposed in [23] on 320 instances with 100 customers and 2 and 3 commodities. The SLNS found 155 new best-known solutions, while the computational time is on average three times that of ALNS.

Finally, variants of the C-SDVRP (or of the DSDVRP) have been studied, see e.g. [37, 43, 1, 33, 28, 44, 25]. In [37], customer demands are composed of multiple items grouped in orders. Each order can be seen as a commodity required by a customer in the C-SDVRP. In addition to the C-SDVRP, this variant includes time windows for the customers and considers service times that depend on the order delivered. The authors proposed a branch-and-price approach.

An extension of the aforementioned problem in a pickup and delivery context is the *Vehicle Routing Problem with Discrete Split Deliveries and Pickups* proposed by [33]. [28] study the effect of splitting customer demands by commodity in a multi-compartment vehicle routing problem. The authors propose a branch-and-price to solve this multi-compartment C-SDVRP. [44] addressed another multi-compartment C-SDVRP in the context of a capacitated arc routing problem arising in the collection of recyclable waste. Vehicles with multiple compartments may make multiple visits to the same household to collect different recyclables, however, the amount of a single recyclable cannot be split.

In this paper, we consider a set covering formulation for the C-SDVRP, where the exponential number of variables are related to routes. Generating all such variables is intractable. Hence, we propose a *restricted master heuristic* [36] to solve the problem. This heuristic scheme consists in solving the formulation restricted to a subset of variables as a static integer program. Similar approaches have been successfully applied to deal with vehicle routing problems [see, e.g., 41, 30, 32, 12]. The main difference in the methodologies proposed in these works is the way the subset of variables is generated. [41] and [30] developed a tabu search heuristic to populate a subset of variables for the solution of a routing problem with a heterogeneous fleet of vehicles and the *Vehicle Routing Problem with Time Windows*, respectively. In [32], the authors devised a restricted master heuristic for the dial-a-ride problem: variables are generated by means of a hybrid column generation procedure where a variable neighbourhood search heuristic is employed to identify negative reduced cost columns. Finally, [12] deals with the *Joint Order Batching and Picker Routing Problem* (JOBPRP). The authors proposed formulation with exponentially many variables and

solve its linear relaxation by means of column generation. The objective is twofold: determining a subset of variables to use in a restricted master heuristic and calculating a lower bound on the optimal solution value.

The approach proposed in the current paper follows the strategy used in [12]. Unlike the existing literature on the C-SDVRP, our approach is a heuristic that provides lower and upper bounds even for large-scale instances of the problem within reasonable computation times. The lower bound serves as a performance guarantee for our heuristic. In the column generation phase, the pricing problem reduces to solve an ESPPRC. Efficient handling of the pricing problem is essential in a column generation procedure. Therefore, heuristics are commonly used to address the pricing problem before solving it exactly. In this respect, we devise a new pricing heuristic that exploits the multi-commodity aspect of the problem. More precisely, the heuristic articulates in two phases: **Phase 1** computes a set of promising customer sequences by solving the ESPPRC on a modified version of the pricing graph; **Phase 2** is called for each customer sequence produced by the first phase and determines all the negative reduced cost routes arising from the sequence by solving the *Shortest Path Problem with Resource Constraints* (SPPRC) on acyclic graphs. It is noteworthy that the first phase of our heuristic also provides a valid lower bound on the value of the pricing problem. After the column generation procedure, upper bounds are identified by the restricted master heuristic. Finally, a local search phase is applied to improve the upper bound. This phase uses the mathematical programming operator proposed in [23] to reassign commodities to the routes. Computational experiments proved that our approach successfully provides upper bounds of good quality in shorter computational times than the state-of-the-art heuristic approaches. More precisely, it is capable of solving large-size instances with four, five, and six commodities and improves a few best-known solution values from the literature. When compared against the state-of-the-art heuristics of [23] and [39], our approach improves the solution time with an average speedup ratio of 17.0, while keeping the percentage gap with respect to the upper bounds to 0.55% on average.

The remainder of the paper is organized as follows. In Section 2, we give a formal description of the C-SDVRP and introduce the notation. In Section 3, we present a set covering formulation for the problem. In Section 4 we describe the main components of the restricted master heuristic we devised to solve it. Finally, the computational results obtained on the benchmark instances are reported and discussed in Section 5.

2. Problem description

In the *Commodity constrained Split Delivery Vehicle Routing Problem* (C-SDVRP) the commodities of a set $\mathcal{K} = \{1, \ldots, \kappa\}$ have to be delivered from a depot 0 to a set of customers $\mathcal{N} = \{1, \ldots, n\}$. The request of a customer $j \in \mathcal{N}$ may be composed of multiple commodities and is identified by set $\mathcal{K}_j = \{k \in \mathcal{K} \colon D_{jk} > 0\}$, where $D_{jk} \geq 0$ is the demand of commodity $k \in \mathcal{K}$ to be delivered to customer j. An unlimited fleet performs the distribution of the commodities to the customers. Each vehicle has a capacity Q and is initially based at the depot. The vehicles can transport any subset of commodities provided that their capacity is not exceeded. We suppose without loss of generality that $Q \geq \max\{D_{jk} : j \in \mathcal{N}, k \in \mathcal{K}_j\}$. When a vehicle visits a customer $j \in \mathcal{N}$, a non-empty subset

$\mathcal{M}_j \subseteq \mathcal{K}_j$ of commodities is delivered to j. Hence, a customer request may be split, and a customer may be visited multiple times. However, when a vehicle visits customer j, the amount of each commodity $k \in \mathcal{M}_j$ delivered by the vehicle to j must be equal to D_{jk}. In other words, the demand for a single commodity cannot be split.

The C-SDVRP can be defined on a directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The vertex set $\mathcal{V} = \{0\} \cup \mathcal{N}$ contains a vertex 0 representing the depot, and the set $\mathcal{N}$ of vertices representing the customers. The arc set $\mathcal{A} = \{(i,j) : i,j \in \mathcal{V}, i \neq j\}$ contains arcs modelling each possible vehicle travel between two distinct vertices. Each arc $(i,j) \in \mathcal{A}$ is associated with a non-negative cost C_{ij} which corresponds to the cost of traversing arc (i,j). We suppose that the arc costs satisfy the triangular inequality. A route in graph $\mathcal{G}$ is a non-empty circuit starting and ending at the depot. A route is *feasible* if the total amount of commodities delivered to the customers visited along the route does not exceed the vehicle capacity Q. The set of feasible routes is denoted by $\mathcal{R}$. The cost of a route r is $C_r = \sum_{(i,j)\in\mathcal{A}(r)} C_{ij}$, where $\mathcal{A}(r)$ is the set of arcs traversed by the route.

The C-SDVRP aims to find a least-cost set of feasible routes such that all the customer requests are served.

3. Problem formulation

We consider the set covering formulation proposed in [5]. For each feasible route $r \in \mathcal{R}$, we introduce a binary coefficient a^r_{jk} with value one if commodity $k \in \mathcal{K}$ is delivered to customer $j \in \mathcal{N}$ by route r and zero otherwise. Then, for $r \in \mathcal{R}$, we introduce a binary variable λ_r taking value one if the route is selected in the solution and zero otherwise. Last, we define an auxiliary variable v to count the number of vehicles in the solution.

The Set Covering formulation [SC] reads as follows:

$$[\text{SC}] \quad \min \sum_{r\in\mathcal{R}} C_r \lambda_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r\in\mathcal{R}} a^r_{jk} \lambda_r \geq 1 \qquad \forall j \in \mathcal{N}, \forall k \in \mathcal{K}_j \tag{2}$$

$$\sum_{r\in\mathcal{R}} \lambda_r = v \tag{3}$$

$$\underline{v} \leq v \leq \bar{v} \tag{4}$$

$$\lambda_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{5}$$

Objective function (1) minimizes the total routing costs. Constraints (2), which we refer to as *covering constraints*, ensure that the customer demands are met. Constraints (3) and (4) define an auxiliary variable v and impose a lower bound and an upper bound on it, i.e., on the number of vehicles used in the solution. Finally, constraints (5) define variables λ_r as binary.

Bounds in constraints (4) can be obtained by solving an instance of the *Bin Packing Problem* (BPP), where bins represent vehicles associated with their capacity, and objects, with the respective weights, are the customer demands. The BPP is formulated as an integer program and solved with a commercial solver within a time limit.

Values $\underline{d}$ and $\bar{d}$ denote the obtained lower and upper bounds, respectively. Hence, the number of vehicles is bounded from below by the ceil function of $\underline{d}$ ($\underline{v} \coloneqq \lceil \underline{d} \rceil$) and from above by the minimum between twice value $\bar{d}$ [see 20] and the number of customers ($\bar{v} \coloneqq \min\{2\bar{d}, |\mathcal{N}|\}$).

4. A restricted master heuristic

This section describes the main components of the restricted master heuristic we designed to tackle [SC]. The heuristic scheme articulates in three phases.

In the first phase, the Master Problem (MP), i.e., the linear relaxation of the formulation [SC] is solved using a column generation procedure (see, e.g., [17]) to obtain a subset of variables $\mathcal{R}' \subseteq \mathcal{R}$ and a valid lower bound. Afterwards, if the solution of the MP is fractional, valid inequalities are possibly included to strengthen the lower bound and enrich the set $\mathcal{R}'$. The procedure is then repeated. In the second phase, an upper bound is obtained by solving formulation [SC] defined on the variables of $\mathcal{R}'$ generated in the first phase. Specifically, [SC] restricted to $\mathcal{R}'$ is solved as a static integer program with a commercial solver run within a time limit. Note that the set $\mathcal{R}'$ is preprocessed before solving [SC] to repair all the routes whose total amount of delivered commodities is not tight with respect to the vehicle capacity. For each of these routes, we randomly select commodities to be delivered to the customers they visit in order to generate new routes that fill the vehicle capacity. Finally, in the third phase, the mathematical programming operator proposed in [23] to reassign commodities to the routes is used to improve the upper bound determined in the second phase.

4.1. Column generation

As mentioned above, we developrd a column generation algorithm to solve the MP and populate a subset of variables $\mathcal{R}' \subseteq \mathcal{R}$. The restriction of the MP to $\mathcal{R}'$ is referred to as Restricted Master Problem (RMP). At each iteration of the procedure, the RMP and the pricing problem are solved sequentially. The pricing problem aims to either identify negative reduced cost variables (columns) to add to $\mathcal{R}'$ or to produce a certificate of optimality for the solution of the MP. We consider some heuristic approaches to quickly identify negative reduced cost variables when solving the pricing problem. Among others, we devised a novel pricing algorithm which exploits the multi-commodity aspect of the C-SDVRP. When the heuristic column generators do not yield negative reduced cost variables, we solve the pricing problem using an exact algorithm to produce a certificate of optimality. In addition, it allows us to compute the Lagrangian bound, a valid lower bound on the value of [SC] which we use to provide an optimality gap for the solution of the restricted master heuristic.

4.2. Pricing problem

In this section, we use the terms path and route interchangeably.

As in [5], at each iteration of the column generation procedure, we price out routing variables λ_r, $r \in \mathcal{R}$. The reduced cost of λ_r is given by:

$$\bar{C}_r = C_r - \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{K}_j} a^r_{jk} \rho_{jk} - \tau,$$

where ρ_{jk} and τ are the dual prices associated with constraints (2) and (3), respectively.

[5] showed that the pricing problem reduces to an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC), where the resource is associated with the vehicle capacity. Following to some extent [22], we formulate the ESPPRC on a directed multi-graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ defined over the original graph $\mathcal{G}$ as follows. We include in vertex set $\mathcal{V}'$ two replica i' and i'' of each vertex $i \in \mathcal{V}$. In arc set $\mathcal{A}'$, we include an arc (i'', j') for each arc $(i, j) \in \mathcal{A}$ to model the trip of a vehicle from vertex i to vertex j. Finally, for all non-empty subsets of commodities $\mathcal{M}_j \subseteq \mathcal{K}_j$, we introduce an arc $(j', j'')^{\mathcal{M}_j}$ to model the delivery of $\mathcal{M}_j$ to customer j. The resource consumption $\bar{D}$ is set to zero ($\bar{D}_{i''j'} \coloneqq 0$) on arcs (i'', j'), whereas on arcs $(j', j'')^{\mathcal{M}_j}$, it is equal to the demand associated with commodity subset $\mathcal{M}_j$, i.e., $\bar{D}^{\mathcal{M}_j}_{j'j''} \coloneqq \sum_{k \in \mathcal{M}_j} D_{jk}$. Finally, the cost on arcs (i'', j') is $\bar{C}_{i''j'} \coloneqq C_{ij}$ if $i'' \neq 0''$ and $j' \neq 0'$, and $\bar{C}_{i''j'} \coloneqq C_{ij} - \tau/2$, otherwise. The cost on arcs $(j', j'')^{\mathcal{M}_j}$ considers the dual prices of constraints (2) associated with customer $j \in \mathcal{N}$ and the commodities $k \in \mathcal{M}_j$, that is $\bar{C}^{\mathcal{M}_j}_{j'j''} \coloneqq -\sum_{k \in \mathcal{M}_j} \rho_{jk}$.

Solving the pricing problem results in searching for a negative reduced cost elementary path in $\mathcal{G}'$ from $0''$ to $0'$ such that the resource consumption does not exceed the vehicle capacity Q.

4.3. Solution of the pricing problem

Negative reduced cost paths are retrieved in the multi-graph $\mathcal{G}'$ by solving the ESPPRC by means of a label setting dynamic programming algorithm [see 21]. More precisely, labels identify partial paths in $\mathcal{G}'$ starting at $0''$ and are characterised by the following resources: reduced cost, accumulated demand, set of visited customers, and for each customer, the subset of commodities delivered. The starting point of the procedure is a label associated with vertex $0''$ with resources set to zero or empty. Then, labels are propagated from a vertex to another while satisfying the elementarity and capacity constraints: each customer is visited at most once along a partial path, and the accumulated demand cannot exceed the vehicle capacity Q. Dominance rules are applied to prune unpromising labels.

In order to accelerate the solution of the pricing problem, we implemented some state-of-the-art procedures. Specifically, the first one is the *ng-path* relaxation [7] which partially relaxes the elementarity constraint of the paths: a neighbourhood is pre-assigned to each customer, and cycles are allowed only if the customer visited more than once in a path is not in the neighbourhood of its predecessor in that path. In addition, we incorporate an implicit version of the bidirectional labelling search algorithm proposed in [35]. The labels are extended from vertex $0''$ to the other vertices of $\mathcal{G}'$ up to a value of the accumulated demand equal to $Q/2$. Then, the generated labels are merged to obtain complete paths [see 10, for more details].

However, even by embedding the two techniques mentioned above, the exact resolution of the pricing problem might be time consuming. We therefore proceed as follows. First, the pricing problem is solved with a new

heuristic coupled with two reduced graph heuristics similar to those presented in [22]. Then, the same reduced graph heuristics are also applied on the multi-graph $\mathcal{G}'$. Finally, the exact pricing method is invoked. We switch from one pricing algorithm to the next one when the first fails to produce negative reduced cost paths.

In the following, we give a detailed description the heuristics mentioned above, together with the description of the preprocessing phase to reduce the size of the multi-graph $\mathcal{G}'$.

4.3.1. Preprocessing phase

We perform the preprocessing procedure proposed in [5] and [22] to reduce the size of the multi-graph $\mathcal{G}'$. At each iteration of the column generation procedure, we only consider in $\mathcal{G}'$ arcs of type $(j', j'')^{\mathcal{M}_j}$ whose associated pair demand-cost $(\bar{D}^{\mathcal{M}_j}_{j'j''}, \bar{C}^{\mathcal{M}_j}_{j'j''})$ is Pareto-optimal. Since the number of commodities is small in the benchmark instances of the C-SDVRP ($|\mathcal{K}| \leq 6$), the Pareto-optimal commodity subsets can be computed by enumeration. The reader may refer to [22] for a general procedure, based on the solution of the *Shortest Path Problem with Resource Constraints* (SPPRC) on acyclic graphs, to determine such subsets when the enumeration strategy is not applicable.

4.3.2. A new two-phase pricing heuristic

The heuristic we propose to solve the pricing problem consists of two phases. Phase 1 aims to compute a set of promising customer sequences. To do so, we solve the ESPPRC on a modified graph of reduced size compared with the multi-graph $\mathcal{G}'$. Solving the ESPPRC on such a graph is not only faster than solving it on $\mathcal{G}'$, but Phase 1 also permits to derive a valid lower bound on the pricing problem value. Phase 2 aims to determine all negative reduced cost paths arising from each of the customer sequences provided by Phase 1. We solve the SPPRC on an acyclic graph for each customer sequence. The topology and size of graph $\mathcal{G}'$ allow to perform such operation in negligible time (see [26]).

The graph used in Phase 1, denoted by $\mathcal{G}'' = (\mathcal{V}', \mathcal{A}'')$, differs from the multi-graph $\mathcal{G}'$ in the arcs modelling the deliveries to the customers: in $\mathcal{G}''$ a unique subset of commodities can be delivered to each customer. Hence, $\mathcal{G}''$ is obtained from $\mathcal{G}'$ by removing, for each customer $j \in \mathcal{N}$, all the arcs of type $(j', j'')^{\mathcal{M}_j}$ but one, which we denote by (j', j''). The demand and cost are set on these arcs so that whenever a customer $j \in \mathcal{N}$ is visited the least consuming commodity is delivered and all profitable dual prices related to j are collected. Hence, they are defined as $\bar{D}_{j'j''} := \min\{D_{jk} : k \in \mathcal{K}_j\}$ and $\bar{C}_{j'j''} := -\sum_{k \in \mathcal{K}_j} \rho_{jk}$, respectively.

This definition of demand and cost permits to derive the following properties.

Proposition 1. *All feasible solutions of the ESPPRC on multi-graph $\mathcal{G}'$ are feasible solutions for the ESPPRC on graph $\mathcal{G}''$.*

Proposition 2. *The optimal solution of the ESPPRC on graph $\mathcal{G}''$ provides a lower bound on the optimal value of the ESPPRC on multi-graph $\mathcal{G}'$.*

Corollary 1. *If the optimal value of the ESPPRC on the reduced graph $\mathcal{G}''$ is positive then the optimal value of the ESPPRC on multi-graph $\mathcal{G}'$ is positive as well.*

In Phase 2, we determine all negative reduced cost routes arising from each customer sequence (path) generated in Phase 1. We do this by solving the SPPRC on an acyclic multi-graph for each path. Specifically, let $p = (j''_0 = 0'', j'_1, j''_1, \ldots, j'_{l(p)-1}, j''_{l(p)-1}, j'_{l(p)} = 0')$ be a path produced in Phase 1, where $l(p)$ denotes the length of p. The acyclic multi-graph $\mathcal{G}'(p) = (\mathcal{V}'(p), \mathcal{A}'(p))$ associated with p is defined as follows. $\mathcal{V}'(p)$ is the vertex st that includes only vertices visited along p, i.e., $\mathcal{V}'(p) = \{j''_0 = 0'', j'_1, j''_1, \ldots, j'_{l(p)-1}, j''_{l(p)-1}, j'_{l(p)} = 0'\}$. $\mathcal{A}'(p)$ contains the arcs of $\mathcal{G}'$ connecting each vertex in $\mathcal{V}'(p)$ to its successor in p, i.e. (i) arcs (j''_h, j'_{h+1}), $h = 0, \ldots, l(p) - 1$ to model the travel from j_h to j_{h+1}, and (ii) arcs $(j'_h, j''_h)^{\mathcal{M}_{j_h}}$, $h = 1, \ldots, l(p) - 1$, $\mathcal{M}_{j_h} \subseteq \mathcal{K}_j$ to model the deliveries of subsets of commodities $\mathcal{M}_{j_h}$ to customer j_h.

The negative reduced cost routes arising from path p correspond to the negative cost paths in $\mathcal{G}'(p)$ from $j_0 = 0''$ to $j_{l(p)} = 0'$, which satisfy the capacity constraint. These paths are determined by solving the SPPRC on the multi-graph $\mathcal{G}'(p)$. Although solving the SPPRC on acyclic graphs is NP-hard (see [18]), the size and particular topology of multi-graphs $\mathcal{G}'(p)$ allow to do this operation very efficiently in terms of computational time (see [26]).

In the following, we provide an example to illustrate how the proposed pricing heuristic works.

Example 1. *We consider a C-SDVRP instance with three customers $\mathcal{N} = \{1, 2, 3\}$ and three commodities $\mathcal{K} = \{1, 2, 3\}$: customer 1 requires the commodity of $\mathcal{K}_1 = \{1\}$, with $D_{11} = 2$; customer 2 requires the commodities of $\mathcal{K}_2 = \{1, 2, 3\}$, with $D_{21} = 2$, $D_{22} = 4$ and $D_{23} = 3$; customer 3 requires the commodities of $\mathcal{K}_3 = \{1, 3\}$, with $D_{31} = 2$ and $D_{33} = 1$. We assume the travelling cost from the depot to the customers and between customers to be unitary. The vehicle capacity is set to 10.*

Figure 1 shows the pricing multi-graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ arising from such instance at a certain iteration of the column generation procedure. The consumption and cost on arcs of type $(i'', j') \in \mathcal{A}'$ modelling the movement of the vehicle from one vertex to another are $(\bar{D}_{i''j'}, \bar{C}_{i''j'}) = (0, 1)$. Differently, the consumption and cost on arcs of type $(j', j'')^{\mathcal{M}_j} \in \mathcal{A}'$ modelling the delivery to the customers are reported in the figure with the following notation: $\mathcal{M}_j$:$(\bar{D}^{\mathcal{M}_j}_{j'j''}, \bar{C}^{\mathcal{M}_j}_{j'j''})$. We only consider the Pareto-optimal deliveries to the customers.

The graph $\mathcal{G}'' = (\mathcal{V}'', \mathcal{A}'')$ built at phase I is shown in Figure 2a. The consumption and cost on arcs of type $(i'', j') \in \mathcal{A}'$ are as in $\mathcal{G}'$ and those on arcs $(j', j'') \in \mathcal{A}'$ are displayed in the figure with the same convention as in Figure 1. In the first phase,we solve the ESPPRC on the graph $\mathcal{G}''$ to obtain all non-dominated negative cost paths (customer sequences) that respect the vehicle capacity. The second phase of the heuristic identifies the negative reduced cost routes arising from each of these sequences by solving the ESPPRC on acyclic multi-graphs. As an example, we show how this is done on the most negative path found in phase one (in red in Figure 2a), i.e., on $p = (0'', 2', 2'', 3', 3'', 0')$ with consumption and cost equal to 3 and -13, respectively. The acyclic multi-graph $\mathcal{G}'(p)$ associated with path p is shown in Figure 2b. In $\mathcal{G}'(p)$, all the possible deliveries to customers 2 and 3 are restored. Finally, the SPPRC is solved on $\mathcal{G}'(p)$ to obtain all non-dominated feasible routes with negative reduced costs. We

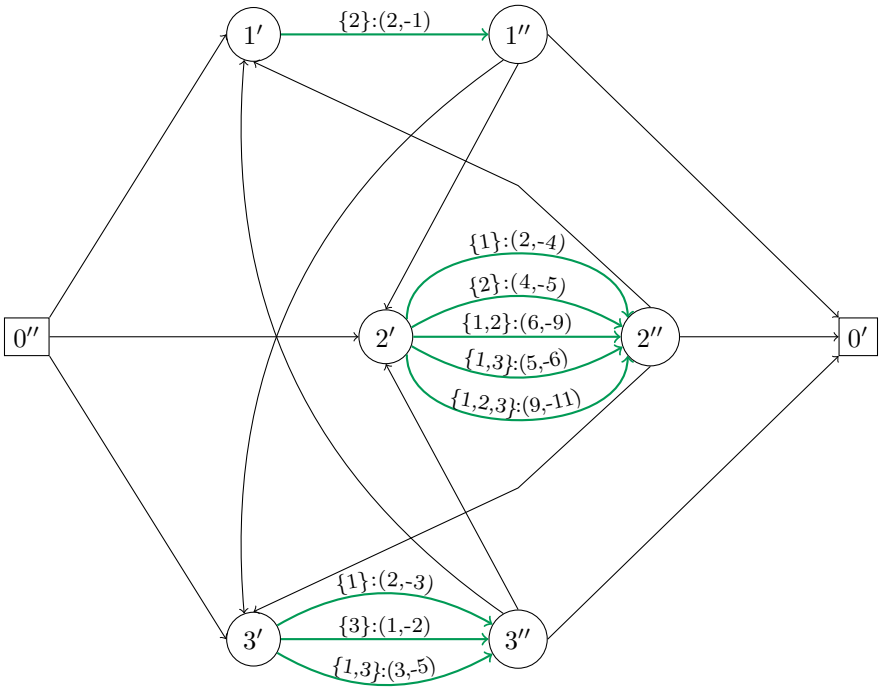


Figure 1: Pricing multi-graph $\mathcal{G}'$ for the C-SDVRP instance defined in Example 1.

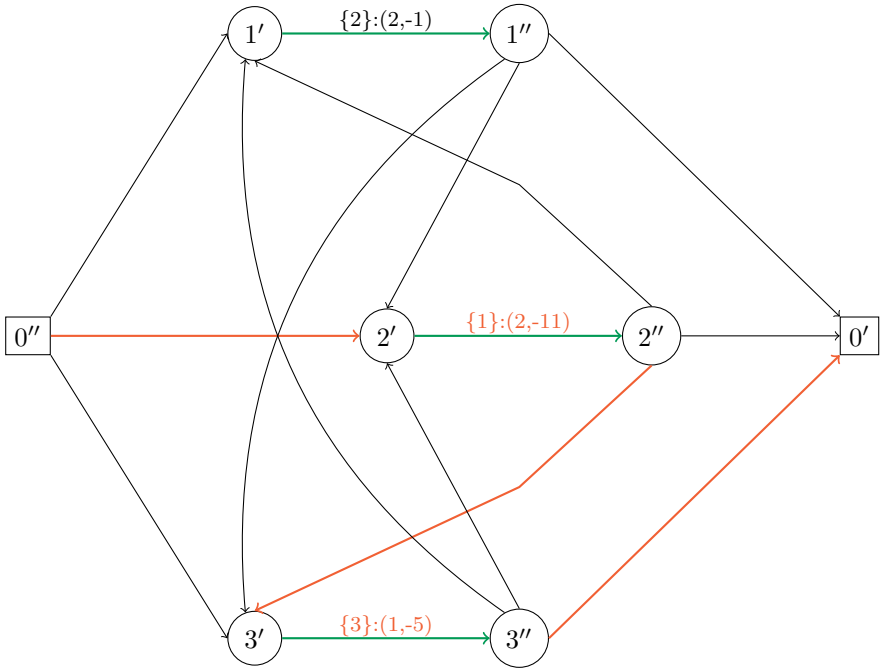
obtain six routes that visit customers 2 *and* 3 *in the order imposed by path* p *and deliver either subset of commodities* $\{1\}$ *or* $\{1,2\}$ *to customer* 2*, both combined with all the possible deliveries to customer* 3*. The route with the most negative reduced cost (in red in Figure 2b) delivers* $\{1,2\}$ *to customer* 2 *and* $\{1,3\}$ *to customer* 3*. Its consumption and reduced cost are* 9 *and* -11*, respectively.*

4.3.3. Reduced graph heuristics

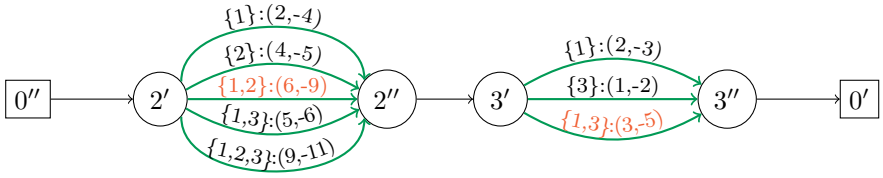
In this section, we present two classical heuristics to decrease the size of the pricing graph. They are applied to the multi-graph $\mathcal{G}'$ and to the graph $\mathcal{G}''$ of the first phase of heuristic, we just described. We discuss them for the case of $\mathcal{G}'$, knowing that the case of $\mathcal{G}''$ can be treated similarly.

The first heuristic is inspired by [42] and limits the possibilities of moving between customers. Specifically, a neighbourhood containing the g closest customers is built for each customer $j \in \mathcal{N}$. A partial path ending in j can only be extended to customers belonging to its neighbourhood. This is implemented by removing from $\mathcal{G}'$ all arcs of type (j'', l') such that l does not belong to the neighborhood of j. The pricing problem is solved considering a sequence of increasing neighbourhood sizes: $g = 3, 6, 10, |\mathcal{N}|$. The value of g is incremented when the associated pricing problem produces no negative reduced cost path.

The second heuristic is specifically designed to handle the multi-commodity aspect of the C-SDVRP. Indeed, it aims at reducing the delivery possibilities to customers. Specifically, we impose an upper bound b on the number of customers whose demand can be split per path. This strategy is motivated by the analysis carried out in [5]



(a) Graph $\mathcal{G}''$ arising from the first phase of the pricing heuristic in Example 1.



(b) Acyclic multi-graph $\mathcal{G}'(p)$ where $p = (0'', 2', 2'', 3', 3'', 0')$ arising from the second phase of the pricing heuristic in Example 1.

Figure 2: Graphs of the first and second phase of the novel pricing heuristic built in Example 1.

on optimal solutions of the C-SDVRP. They note that in most of them the number of split deliveries is less than three. To count the number of split customers in a path, we introduce an integer resource s in the label definition. The value of s is initially set to zero and is incremented by one unit along arcs of type $(j', j'')^{\mathcal{M}_j}$, if $\mathcal{M}_j$ does not correspond to the full delivery to j ($\mathcal{M}_j \subsetneqq \mathcal{K}_j$). On the other arcs, the value of s is simply propagated. Once s reaches the bound b, all the following customers visited in the path are delivered only with subset $\mathcal{K}_j$, i.e., only arcs $(j', j'')^{\mathcal{K}_j}$ are considered. As for the previous heuristic, we consider an incremental procedure relying on a sequence of increasing upper bounds: $b = 0, 1, 2, 3, \infty$.

4.4. Valid inequalities

We consider a family of robust valid inequalities, the so-called *capacity cuts*:

$$\sum_{r \in \mathcal{R}} \left(\sum_{(i,j) \in \delta^-(\mathcal{S})} b_{ij}^r \right) \lambda_r \geq \left\lceil \frac{\sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{K}_j} D_{jk}}{Q} \right\rceil \qquad \forall \mathcal{S} \subseteq \mathcal{N}, \tag{6}$$

where $\delta^-(\mathcal{S}) = \{(i,j) \in \mathcal{A} : i \notin \mathcal{S}, j \in \mathcal{S}\}$ is the set of arcs of graph $\mathcal{G}$ having their final extremity in $\mathcal{S}$ and b_{ij}^r is a binary coefficient taking value one if route $r \in \mathcal{R}$ traverses the arc $(i,j) \in \mathcal{A}$.

If the solution of the MP is fractional, we separate the capacity cuts (6). Since the separation of these inequalities is NP-hard, we do so by means of the heuristic algorithms presented in [34], namely the *extended shrinking heuristic* and the *greedy shrinking heuristic*. The violated cuts are included in the RMP, and the associated dual prices $\pi_{\mathcal{S}}$ are incorporated in the definition of the reduced cost of the variables λ_r, and then considered in the pricing problem solution.

4.5. Initialization of the set $\mathcal{R}'$

We initialize the set of routes $\mathcal{R}'$ to avoid starting the column generation procedure with large dual prices, which usually slows down the pricing problem resolution. Specifically, for each customer $j \in \mathcal{N}$, we include a round trip (0-j-0) delivering the commodities of each subset $\mathcal{M}_j \subseteq \mathcal{K}_j$ requested by j, feasible with respect to the capacity Q. Moreover, we insert in $\mathcal{R}'$ the routes obtained by applying a variant of the Clarke-Wright algorithm (CW) [15]. Precisely, we modified the randomized CW algorithm proposed in [8] to take into account the multi-commodity aspect of the C-SDVRP. We set a limit of 20 runs.

4.6. Local search

In this section, we present the local search we implement to improve the C-SDVRP solution provided by the restricted master heuristic. Specifically, we consider the Mathematical Programming Operator (MPO) proposed in [23] to reassign the commodities of a specific customer $j \in \mathcal{N}$ to the routes of the solution. We iteratively call the MPO for each customer in $\mathcal{N}$.

Let $j \in \mathcal{N}$ be a customer. We introduce the following notation. We denote by $\bar{\mathcal{R}}^{-j}$ the set of routes in the current C-SDVRP solution where all the visits to customer j are removed. More precisely, $\bar{\mathcal{R}}^{-j}$ contains the routes

of the current C-SDVRP solution which do not visit j and, for the ones that visit j, $\bar{\mathcal{R}}^{-j}$ contains a copy of those routes where j has been removed from the sequence of visited customers. For each $r \in \bar{\mathcal{R}}^{-j}$, Q_r^j denotes the residual capacity in route r. Finally, we indicate by C_r^j the cost of the cheapest insertion of customer j in route $r \in \bar{\mathcal{R}}^{-j}$.

The MPO consists in solving a *Capacitated Facility Location Problem* (CFLP) [see 27] where all commodities $k \in \mathcal{K}_j$ of customer j have to be assigned to the routes (facilities) of $\bar{\mathcal{R}}^{-j}$ at minimum insertion costs and such that residual capacities of the routes are not exceeded.

The integer program on which the MPO is based makes use of the following decision variables. For each $k \in \mathcal{K}_j$ and each $r \in \bar{\mathcal{R}}^{-j}$, we introduce a binary variable

$$y_{kr}^j = \begin{cases} 1 & \text{if commodity } k \text{ is delivered to customer } j \text{ by route } r \\ 0 & \text{otherwise.} \end{cases}$$

In addition, for each $r \in \bar{\mathcal{R}}^{-j}$ we include a binary variable z_r^j defined as follows

$$z_r^j = \begin{cases} 1 & \text{if route } r \text{ delivers to customer } j \text{ at least one commodity} \\ 0 & \text{otherwise.} \end{cases}$$

Note that j refers to a specific customer and is not used as an index for the variables.

The integer program is for customer j as follows:

$$\min \sum_{r \in \bar{\mathcal{R}}^{-j}} C_r^j z_r^j \tag{7}$$

$$\text{s.t.} \sum_{r \in \bar{\mathcal{R}}^{-j}} y_{kr}^j = 1 \qquad \forall k \in \mathcal{K}_j \tag{8}$$

$$\sum_{k \in \mathcal{K}_j} D_{jk} y_{kr}^j \leq Q_r^j z_r^j \qquad \forall r \in \bar{\mathcal{R}}^{-j} \tag{9}$$

$$y_{kr}^j \in \{0,1\} \qquad \forall k \in \mathcal{K}_j, \forall r \in \bar{\mathcal{R}}^{-j} \tag{10}$$

$$z_r^j \in \{0,1\} \qquad \forall r \in \bar{\mathcal{R}}^{-j} \tag{11}$$

The objective function (7) minimizes the total insertion cost. Constraints (8) guarantee that all commodities of customer j are covered by exactly one route. Constraints (9) ensure that if some commodities of $\mathcal{K}_j$ are added to a route, their demand do not exceed the remaining capacity of the route. Finally, Constraints (10) and (11) are the binary requirements.

5. Computational experiments

Our algorithm is implemented in C++ and compiled in release mode under a 64-bit version of MS Visual Studio 2019. CPLEX 12.9.0 (64-bit version) is used to solve the RMP in the column generation procedure and the restricted version of the formulation [SC]. All experiments are carried out on a 64-bit Windows machine equipped

with a Intel(R) Xeon(R) Silver 4214 processor with 24 cores hyper-threaded to 48 virtual cores, with a base clock frequency of 2.2 GHz, and 96 GB of RAM. A time limit of one hour and a single thread are imposed for each run of the algorithm.

In the following, we denote by LB and UB, respectively, the lower and upper bounds returned by our restricted master heuristic. The percentage optimality gap is defined as $100((UB - LB)/LB)$. The percentage gap with respect to the best-known solution value UB^{bk} from the literature is computed as $100((UB - UB^{bk})/UB^{bk})$. More precisely, values UB^{bk} are retrieved from [6, 23, 22] or [39]. Finally, all the solution times are expressed in seconds.

In this section, we first describe the benchmark instances, then, we measure the impact of the new two-phase pricing heuristic on the performance of the overall solution algorithm. Later, we present the results on the whole testbed. Finally, we compare our algorithm with the two existing heuristic approaches from the literature, providing the majority of the best-known solution values.

5.1. Benchmark instances

We tested our restricted master heuristic on the benchmark instances for the C-SDVRP proposed by [6] and [22]. The instances are divided in three groups: small ($|\mathcal{N}| = 15$), mid-size ($|\mathcal{N}| = 20, 40, 60, 80$) and large ($|\mathcal{N}| = 100$). In each small and mid-size instance, customers' locations are taken from the C101 and R101 Solomon's instance [40], whereas, in each large instance, locations are taken from the Solomon's RC101 instances. In addition, the following parameters define the instances: (i) number of commodities $|\mathcal{K}|$; (ii) probability p that a customer requires a commodity with a non-zero demand; (iii) interval Δ to select the non-zero demand of a commodity required by a customer, expressed as a percentage of vehicle capacity; (iv) percentage α of vehicle capacity with respect to the maximum demand ($Q = \alpha \max\{\sum_{k \in \mathcal{K}_j} D_{jk} : j \in \mathcal{N}\}$). Table 1 summarises the values of these parameters which characterise each group of instances.

Table 1: Characteristics of the small, mid-size and large instances.

Group	Number of instances	Values of the parameters					
		$\lvert\mathcal{N}\rvert$	customers' locations	$\lvert\mathcal{K}\rvert$	p	Δ	α
small	160	15	C101, R101	2, 3, 4, 5, 6	0.6, 1.0	[40, 60], [1, 100]	1.1, 1.5, 2.0, 2.5
mid-size	320	20, 40, 60, 80	C101, R101	3, 4, 5, 6	0.6, 1.0	[1, 100]	1.5
large	900	100	C101, R101, RC101	2, 3, 4, 5, 6	0.6, 1.0	[40, 60], [1, 100]	1.1, 1.5, 2.0, 2.5

Finally, we report a problem with the name of some of the instances in the benchmark with 100 customers, which are called C101, when the locations of the customers correspond to Solomon's RC101 instance. This problem has been corrected in the new database available at `https://hal.inria.fr/hal-03836982v1`. In addition, we mention that 40 instances with 100 customers with the locations of RC101, and 2 and 3 commodities are available at `https://logistik.bwl.uni-mainz.de/forschung/benchmarks/`, but have not been tested in any of the former

papers dealing with the problem, i.e, in [6, 23, 22] and [39]. Thus, we exclude this subset of instances when presenting the comparison with benchmark approaches.

5.2. Impact of the novel pricing heuristic

In this section, we measure the impact of the novel pricing heuristic presented in Section 4.3.2 on the performance of the overall solution algorithm. To do so, we define `RMH-2P` (`RMH-N2P`) to be the variant of the restricted master heuristic presented in Section 4 where we enable (disable) the two-phase pricing heuristic. We test the two variants on a subset of 180 instances characterised by 100 customers and 4 commodities, which is representative of the whole testbed.

The results obtained by comparing `RMH-2P` against `RMH-N2P` are shown in Table 2. We group the considered instances by the value of Δ, i.e., by the interval where the commodity demands are selected. Hence, for each group of instances, we have two columns associated with `RMH-2P` and `RMH-N2P`, respectively. The row headings of the table are: *avg. exact pricing it.*: average number of iterations of the exact pricing algorithm; *avg. UB*: average upper bound value; *avg. t[s]*: average solution time in seconds.

Table 2: Impact of the two-phase pricing heuristic on the instances with $|\mathcal{N}| = 100$ and $|\mathcal{K}| = 4$.

	$\Delta = [40, 60]$		$\Delta = [1, 100]$	
	`RMH-2P`	`RMH-N2P`	`RMH-2P`	`RMH-N2P`
avg. exact pricing it.	69.37	111.20	128.39	168.42
avg. UB	3 170.03	3 175.56	2 359.43	2 353.60
avg. t[s]	530.17	692.17	1900.58	1942.55

First, both variants can provide a lower bound for all instances. Variant `RMH-2P` yields better results on the instances characterised by $\Delta = [40, 60]$. Indeed, it allows us to reduce the iterations of the exact pricing algorithm by 38% and, consequently, the solution time on average by 23%. In addition, `RMH-2P` provides upper bounds of slightly better quality (see row *avg. UB*). Differently, both variants behave similarly on the instances with $\Delta = [1, 100]$. Although `RMH-2P` shows a good reduction of 24% of the iterations of the exact pricing algorithm, the reduction of the solution time of `RMH-2P` with respect to the one of `RMH-N2P` is rather limited (2%). This means that the two-phase heuristic in `RMH-2P` is not effective enough compared with the reduced graph heuristics and the exact pricing algorithm.

From these results, we infer that the performance of `RMH-2P` heavily depends on the interval of commodity demands Δ. The reason lies in how Phase 1 of the two-phase pricing heuristic is designed. The main idea of Phase 1 is to reduce the combinatorics in the solution of the ESPPRC due to the multi-commodity aspect of the C-SDVRP. Indeed, each customer is delivered with its least consuming commodity and all the profitable dual prices associated with the customer are collected. The quality of such an approximation is highly affected by the variability

of the commodity demands. If $\Delta = [40, 60]$, Phase 1 provides a reasonable approximation of the benefit of serving a customer, whereas this might not be the case if $\Delta = [1, 100]$. Indeed, in the latter case, when a customer is visited, we might collect all the profitable dual prices against a minimal consumption of the resource associated with capacity.

Based on the analysis conducted in this section, we apply variant `RMH-2P` to obtain results on the whole testbed. In the following, this variant will be referred to simply as `RMH`.

5.3. Results on the whole testbed

In this section, we discuss the results obtained by the restricted master heuristic (`RMH`) on the 1380 benchmark instances. Due to the large number of instances, the results are presented in an aggregated form. The detailed instance-wise version can be found at `https://hal.inria.fr/hal-03836982v1`. We compare the results with the best-known solution values from the state-of-the-art exact and heuristic methods for the C-SDVRP available in the literature, namely [6, 23, 22] and [39]. Note that in Section 5.4, we compare in more details `RMH` against [23] and [39].

Table 3 shows the results obtained by `RMH` on the small and mid-size instances. Each table row corresponds to a subset of instances with the same number of customers and commodities. The first four columns report some information regarding the instance subsets: $|\mathcal{N}|$: number of customers; $|\mathcal{K}|$: number of commodities; *avg.*$\#CC$: average number of customer-commodities ($\sum_{j \in \mathcal{N}} |\mathcal{K}_j|$) per instance; $\#$: number of instances in the subsets. The remaining eight columns of the table summarise the results of the `RMH`: $\#LB$: number of instances for which a LB is found; *opt. gap[%] avg./min./max.*: average/minimal/maximal optimality gap expressed as a percentage; *avg. t[s]*: average solution time in seconds; $\#opt$: number of optima identified by `RMH` with respect to the ones identified by [22]; $\#equal$: number of times `RMH` returned the best-known solution values from the literature; $\#impr.$: number of times `RMH` improved the best-known solution values from the literature (considering also the *new solutions*, i.e., the cases where no solution was available in the literature); *avg. gap[%].*: average percentage gap with respect to the best-known solution values; the character '-' indicates that no best-known solution value is available.

First, note that `RMH` provides a lower bound for all the small and mid-size instances. For the instances with $|\mathcal{N}| = 15$ and $|\mathcal{N}| = 20$, the branch-price-and-cut algorithm of [22] provides 158 and 71 optima over 160 and 80 instances, respectively. `RMH` is able to identify 119 of them if $|\mathcal{N}| = 15$ and 41 of them if $|\mathcal{N}| = 20$. In respectively 50 and 6 of such cases, `RMH` proves the optimality of the obtained solutions (opt. gap = 0). The identified solutions are globally good, the optimality gap being on average equal to 0.69% and larger than 2.5% only for nine instances out of 240. Moreover, the average gap with respect to the best-known solution is 0.18%. Six new best solutions are found (see column $\#impr.$) and the optimality gap referred to these six new best solutions is, on average, 2.09%.

For the instances with $|\mathcal{N}| = 40, 60, 80$, the number of commodities has a significant impact on the measure of the quality of the solutions found by `RMH`: the greater the number of commodities, the larger the average optimality gap (see columns *opt. gap[%]*). However, the number of instances for which the optimality gap exceeds 5% is limited

Table 3: Results on the small and mid-size instances.

Instances				RMH results								
					opt. gap[%]					best known		
$\|\mathcal{N}\|$	$\|\mathcal{K}\|$	avg.#CC	#inst.	#LB	avg.	min.	max.	avg.t[s]	#opt	#equal	#impr.	avg. gap[%]
15	2	26.00	32	32	0.29	0.00	2.16	0.32	30	30	0	0.02
	3	36.50	32	32	0.68	0.00	5.89	3.98	22	22	0	0.23
	4	48.56	32	32	0.43	0.00	2.19	14.59	29	29	0	0.04
	5	60.13	32	32	0.84	0.00	6.93	57.30	18	18	0	0.41
	6	72.66	32	32	0.63	0.00	2.51	87.25	20	20	1	0.12
20	3	48.70	20	20	0.81	0.00	2.45	4.52	10	10	0	0.19
	4	64.10	20	20	0.94	0.00	2.70	33.54	9	9	1	0.24
	5	80.50	20	20	0.97	0.00	4.70	137.27	11	12	2	0.12
	6	96.10	20	20	0.98	0.04	2.76	467.27	11	11	2	0.28
40	3	98.10	20	20	1.69	0.24	2.79	58.76	2	2	0	0.41
	4	129.10	20	20	1.90	0.79	4.16	243.24	0	0	9	0.76
	5	160.40	20	20	2.35	0.58	5.27	749.61	1	1	14	0.71
	6	192.45	20	20	2.74	0.73	5.91	1627.37	0	0	17	0.27
60	3	145.00	20	20	2.27	1.14	3.38	170.76	0	0	0	0.76
	4	193.70	20	20	2.66	0.62	4.89	556.42	0	0	18	0.51
	5	240.10	20	20	3.46	0.92	6.21	1848.11	0	0	20	-
	6	287.70	20	20	4.66	1.71	7.70	2467.48	0	0	20	-
80	3	195.20	20	20	2.75	1.47	5.02	354.51	0	0	2	0.98
	4	256.40	20	20	3.74	1.52	8.34	1045.47	0	0	20	-
	5	320.80	20	20	4.31	2.31	7.46	2268.34	0	0	20	-
	6	386.75	20	20	6.16	2.37	13.28	3044.60	0	0	20	-

(35 out of 240), and most of them correspond to instances with 80 customers and a number of commodities larger than four. For these instances, no further insight can be drawn regarding solution quality. Indeed, no solution was reported in the literature. `RMH` equals three best-known solutions and improves 160 of them (see columns *#equal.* and *#impr.*). Note that, out of those 160, `RMH` provides the first known solution for 39, 58 and 60 instances with 40, 60 and 80 customers, respectively. The average optimality gap of such solutions is 2.72%, 3.68%, and 4.73%. The gap with respect to the best-known solution values is on average equal to 0.70% and larger than 2.5% only in

four cases.

In general, RMH runs in relatively short computational times on small and mid-size instances. Moreover, Table 3 shows that the average solution time grows with the number of commodities and the number of customers.

In Table 4, we report the results obtained on the large instances ($|\mathcal{N}| = 100$) which are grouped by the number of commodities $|\mathcal{K}|$, the interval Δ for customer demand and the probability p that a customer requires a commodity. The remaining columns reporting the characteristics of the instance are as in Table 3. The columns which summarise the results of the RMH are: *#LB*: number of instances for which a LB is found; *opt. gap[%] avg./min./max.*: average/minimal/maximal optimality gap expressed in percentage; *avg.t[s]*: average solution time in seconds; *#impr.*: number of times RMH improved the best-known solution values from the literature (counting also the new solutions); *gap[%] avg./min./max.*: average/minimal/maximal percentage gap with respect to the best-known solution values. A character '-' indicates that no solution value was previously known.

First, we note that RMH manages to provide a lower bound for 874 of the 900 instances. The 26 instances where it does not succeed are characterised by $|\mathcal{K}| = 5, 6$ and 25 of them also by $p = 1$ and $\Delta = [1, 100]$, i.e., they have 500 or 600 non-zero customer demands. When a lower bound is found, the optimality gap mirrors the behaviour observed in Table 3: it deteriorates as the number of commodities increases. More precisely, it is generally larger when $p = 1$, and especially when $\Delta = [1, 100]$. In addition, the optimality gap is larger than 5% for 241 instances and, among those, 191 are characterised by five and six commodities. For the instances with fewer commodities, the average optimality gap is 2.71%, meaning that RMH offers a good performance guarantee. For the ones with $|\mathcal{K}| = 2, 3$, RMH shows a behaviour comparable with the approaches from the literature identifying the best-known solution values, namely [6], [23] and [39]. Indeed, the gap against the best-known solution values is on average 0.79% and it is zero for five instances with two commodities. Our approach provides a new solution for the 540 instances with $|\mathcal{K}| = 4, 5, 6$. In addition, RMH manages to improve the value of 55 instances with $|\mathcal{K}| = 2, 3$, 40 of these are new solutions. More insights about the comparison against the existing heuristic methods in the literature are drawn in Section 5.4.

We conclude this section with a discussion of the price to be paid to have a performance guarantee, i.e., to compute good lower bounds. In the case of RMH, providing lower bounds involve invoking the exact pricing algorithm in the column generation procedure and the separation of the capacity cuts (6).

To analyse the impact of the lower bound computation, we record the upper bound and the computational time before the first iteration of the exact pricing algorithm. Note that such upper bounds are obtained by solving the formulation [SC] restricted to the subset of columns found so far and applying the mathematical programming operator. Such results are reported in Table 5 in the columns with the heading RMH-NG. The rows of the table represent subsets of instances with the same number of customers. The columns are: $|\mathcal{N}|$: number of customers; *#inst.*: number of instances; *avg.UB*: average upper bounds of RMH-NG and RMH; *avg.gap[%]*: average percentage gap between the upper bounds of RMH with respect to the ones of RMH-NG; *avg.t[s]*: average solution time in seconds of RMH-NG and RMH; *avg.ratio*: ratio between the solution time of RMH with respect to the one of RMH-NG;

Table 4: Results on the large instances.

Instances					RMH results								
										best known			
						opt. gap[%]					gap[%]		
$\|\mathcal{K}\|$	Δ	p	avg.#CC	#inst.	#LB	avg.	min.	max.	avg.t[s]	#impr.	avg.	min.	max.
2	[40,60]	0.6	136.40	40	40	1.36	0.03	3.39	30.49	0	0.43	0.00	1.74
		1	200.00	50	50	1.39	0.09	3.12	68.28	11	0.39	-0.05	1.10
	[1,100]	0.6	136.40	40	40	2.40	1.06	5.24	183.20	0	0.95	0.04	2.87
		1	200.00	50	50	1.94	0.54	3.95	250.62	16	0.48	-0.23	2.27
3	[40,60]	0.6	188.40	40	40	1.77	0.70	3.22	155.42	1	0.45	-0.15	1.79
		1	300.00	50	50	2.46	0.74	7.22	247.15	12	1.09	-0.60	5.17
	[1,100]	0.6	188.40	40	40	2.46	0.67	5.54	459.32	2	0.89	-0.11	3.49
		1	300.00	50	50	3.51	1.50	6.71	995.53	13	1.62	-0.45	4.48
4	[40,60]	0.6	242.95	40	40	2.42	1.00	5.00	379.13	40	-	-	-
		1	400.00	50	50	3.91	0.93	9.11	651.01	50	-	-	-
	[1,100]	0.6	243.30	40	40	3.29	0.80	7.33	1100.49	40	-	-	-
		1	400.00	50	50	5.10	2.05	9.61	2540.64	50	-	-	-
5	[40,60]	0.6	299.13	40	40	3.66	1.65	6.19	957.39	40	-	-	-
		1	500.00	50	50	5.19	2.54	10.08	1609.24	50	-	-	-
	[1,100]	0.6	301.48	40	40	4.82	1.66	13.01	2196.31	40	-	-	-
		1	500.00	50	45	5.85	2.99	11.50	3479.58	50	-	-	-
6	[40,60]	0.6	359.65	40	40	4.78	2.03	8.05	1764.76	40	-	-	-
		1	600.00	50	50	6.43	2.95	11.98	2641.67	50	-	-	-
	[1,100]	0.6	357.73	40	39	6.04	3.14	15.24	2972.40	40	-	-	-
		1	600.00	50	30	7.37	4.99	11.89	3622.95	50	-	-	-

As expected, the price to be paid for having a performance guarantee is paid in terms of computational time: that of `RMH` is on average 3.9 times that of `RMH-NG`. The extra time spent in `RMH` serves not only to compute good lower bounds but also to improve the quality of the upper bounds. Indeed, several additional columns are generated in `RMH` and the improvement of the upper bounds of `RMH` compared with respect to those of `RMH-NG` is on average 0.6%. These results show that `RMH` can be easily adapted according to the needs: in the case where a solution is needed in a short computing time, the `RMH-NG` can be used without sacrificing solution quality to a large extent.

Table 5: Impact of the performance guarantee.

Instances		RMH-NG vs. RMH					
		avg. UB			avg.t[s]		
$\|\mathcal{N}\|$	#inst.	RMH-NG	RMH	avg. gap[%]	RMH-NG	RMH	avg. ratio
15	160	389.35	386.69	-0.65	9.92	32.69	3.93
20	80	703.78	693.76	-1.41	41.47	160.65	4.07
40	80	1152.61	1139.68	-1.14	135.62	669.74	5.67
60	80	1659.90	1644.11	-0.96	254.19	1260.69	4.72
80	80	2126.85	2113.77	-0.66	361.21	1678.23	4.45
100	900	2793.71	2780.10	-0.38	382.29	1348.10	3.64

5.4. Comparison with [23] and [39]

In this section, we analyse how RMH performs against the two existing heuristic algorithms available in the literature providing the majority of the best-known solution values, namely [23] and [39]. [23] consider instances with two and three commodities, as those with more commodities were not yet available. Among the instances used in [23], [39] considers only the ones with 100 customers. Hence, in this section, the testbed is restricted accordingly.

Tables 6 and 7 report the comparison against [23] and [39], respectively. The rows of the tables correspond to subsets of instances with the same number of customers and commodities. The first three columns report the number of customers ($|\mathcal{N}|$), commodities ($|\mathcal{K}|$) and instances (*#inst.*) in each subset. The next five columns compare the upper bounds reporting the number of instances where RMH matches (*#equal*) or improves (*#impr.*) the upper bounds of the competing algorithm and the average/minimal/maximal gap (*gap[%] avg./min./max.*). The last three columns indicate the average solution times (*avg.t[s]*) of RMH, that of the compared heuristic, and the average ratio between the solution times of the compared heuristic with respect to those of RMH (*avg.ratio*).

The RMH runs much faster than the other heuristics: the speedup ratio is on average 9.4 and 28.0 against [23] and [39], respectively. In spite of the significant decrease in the solution time, the quality of the solutions provided by RMH remains comparable with that of the competing algorithms. Indeed, when compared with [23], RMH matches 71 upper bounds, improves 70 of them by an average of 0.26% and the percentage gap of the remaining ones is on average 0.83%. The results show a similar trend in the comparison with [39]: RMH matches five upper bounds, improves 36 of them by an average of0.35% and the percentage gap of the remaining ones is on average 0.69%. Recall that our heuristic also provides a performance guarantee contrary to [23] and [39].

Table 6: Comparison with [23] on the instances with $|\mathcal{K}| = 2, 3$ and customers' locations from C101 and R101.

Instances			RMH vs. [23] UB							
					gap[%]			avg.t[s]		
$\|\mathcal{N}\|$	$\|\mathcal{K}\|$	#inst.	#equal	#impr.	avg.	min.	max.	RMH	[23]	avg.ratio
15	2	32	30	0	0.02	0.00	0.28	0.32	8.91	35.12
15	3	32	22	0	0.17	0.00	2.47	3.98	15.63	21.05
20	3	20	12	0	0.18	0.00	0.98	4.52	56.78	14.28
40	3	20	2	2	0.36	-0.09	1.01	58.76	117.14	2.72
60	3	20	0	0	0.70	0.05	2.15	170.76	281.83	1.94
80	3	20	0	2	0.98	-0.26	2.83	354.51	511.00	1.47
100	2	160	5	42	0.34	-0.67	2.66	136.07	445.09	9.46
	3	160	0	24	0.91	-0.96	5.17	481.80	822.60	4.02

Table 7: Comparison with [39] on the instances with $|\mathcal{N}| = 100$ and $|\mathcal{K}| = 2, 3$ and customers' locations from C101 and R101.

Instances			RMH vs. [39] UB							
					gap[%]			avg.t[s]		
$\|\mathcal{N}\|$	$\|\mathcal{K}\|$	#inst.	#equal	#impr.	avg.	min.	max.	RMH	[39]	avg.ratio
100	2	160	5	12	0.49	-0.34	2.84	136.07	1800	44.62
	3	160	0	24	0.63	-1.45	4.03	481.80	1800	11.47

6. Conclusions

In this paper, we considered the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP), a routing problem where customer demands may be composed of multiple commodities, and split deliveries are allowed as long as the demand of a single commodity is delivered all at once. We presented a heuristic with a performance guarantee to solve the problem. Our heuristic is based on a column generation approach which embeds a new pricing heuristic that exploits the multi-commodity aspect of the problem. Such contribution allowed us to reduce the computational time on instances where the variability of the customer demands is not large. We performed a thorough computational analysis on the 1380 benchmark instances available in the literature. We provide an upper bound for all the considered instances, the majority of those are guaranteed to be of good quality (optimality gap less than 5%). Some new best-known solutions are found. Finally, our approach outperforms the state-of-the-art metaheuristic for the C-SDVRP ([23] and [39]) in terms of computational time while maintaining the quality of the

upper bounds comparable.

Future research may be devoted to the inclusion of additional families of valid inequalities (possibly robust) to improve both the lower and upper bounds. Finally, our approach may be adapted to solve variants of the problem where the additional constraints can be easily handled in the pricing problem (e.g. a multi-compartment C-SDVRP).

References

[1] M. Alinaghian and N. Shokouhi. Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, 76:85–99, apr 2018. doi: 10.1016/j.omega.2017.05.002.

[2] C. Archetti, M. W. P. Savelsbergh, and M. G. Speranza. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, 40(2):226–234, may 2006. doi: 10.1287/trsc.1050.0117.

[3] C. Archetti, M. G. Speranza, and A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73, feb 2006. doi: 10.1287/trsc.1040.0103.

[4] C. Archetti, N. Bianchessi, and M. G. Speranza. Branch-and-cut algorithms for the split delivery vehicle routing problem. *European Journal of Operational Research*, 238(3):685–698, 2014. doi: https://doi.org/10.1016/j.ejor.2014.04.026.

[5] C. Archetti, N. Bianchessi, and M. G. Speranza. A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, 64:1 – 10, 2015. doi: https://doi.org/10.1016/j.cor.2015.04.023.

[6] C. Archetti, A. M. Campbell, and M. G. Speranza. Multicommodity vs. single-commodity routing. *Transportation Science*, 50(2):461–472, may 2016. doi: 10.1287/trsc.2014.0528.

[7] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, oct 2011. doi: 10.1287/opre.1110.0975.

[8] M. Battarra, B. Golden, and D. Vigo. Tuning a parametric clarke–wright heuristic via a genetic algorithm. *Journal of the Operational Research Society*, 59(11):1568–1572, nov 2008. doi: 10.1057/palgrave.jors.2602488.

[9] J. M. Belenguer, M. C. Martinez, and E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48(5):801–810, oct 2000. doi: 10.1287/opre.48.5.801.12407.

[10] C. Bode and S. Irnich. Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, 60(5):1167–1182, oct 2012. doi: 10.1287/opre.1120.1079.

[11] A. Bortfeldt and J. Yi. The split delivery vehicle routing problem with three-dimensional loading constraints. *European Journal of Operational Research*, 282(2):545–558, apr 2020. doi: 10.1016/j.ejor.2019.09.024.

[12] O. Briant, H. Cambazard, D. Cattaruzza, N. Catusse, A.-L. Ladier, and M. Ogier. An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, 285(2):497–512, sep 2020. doi: 10.1016/j.ejor.2020.01.059.

[13] M.-C. Chen, Y.-H. Hsiao, R. H. Reddy, and M. K. Tiwari. The self-learning particle swarm optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks. *Transportation Research Part E: Logistics and Transportation Review*, 91:208 – 226, 2016. doi: https://doi.org/10.1016/j.tre.2016.04.003.

[14] P. Chen, B. Golden, X. Wang, and E. Wasil. A novel approach to solve the split delivery vehicle routing problem. *International Transactions in Operational Research*, 24(1-2):27–41, jan 2016. doi: 10.1111/itor.12250.

[15] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, aug 1964. doi: 10.1287/opre.12.4.568.

[16] G. Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192, feb 2010. doi: 10.1287/opre.1090.0713.

[17] J. Desrosiers and M. E. Lübbecke. A primer in column generation. In *Column Generation*, pages 1–32. Springer-Verlag, 2005. doi: 10.1007/0-387-25486-2_1.

[18] L. Di Puglia Pugliese and F. Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, jun 2013. doi: 10.1002/net.21511.

[19] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, may 1989. doi: 10.1287/trsc.23.2.141.

[20] A. Federgruen and D. Simchi-Levi. Analysis of vehicle routing and inventory-routing problems. *Handbooks in operations research and management science*, 8:297–373, 1995.

[21] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.20033.

[22] T. Gschwind, N. Bianchessi, and S. Irnich. Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, 278(1):91–104, oct 2019. doi: 10.1016/j.ejor.2019.04.008.

[23] W. Gu, D. Cattaruzza, M. Ogier, and F. Semet. Adaptive large neighborhood search for the commodity constrained split delivery VRP. *Computers & Operations Research*, 112:104761, dec 2019. doi: 10.1016/j.cor.2019.07.019.

[24] D. Gulczynski, B. Golden, and E. Wasil. The split delivery vehicle routing problem with minimum delivery amounts. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):612–626, sep 2010. doi: 10.1016/j.tre.2009.12.007.

[25] F. Guo, Z. Huang, and W. Huang. Heuristic approaches for a vehicle routing problem with an incompatible loading constraint and splitting deliveries by order. *Computers & Operations Research*, 134:105379, oct 2021. doi: 10.1016/j.cor.2021.105379.

[26] B. Jaumard, F. Semet, and T. Vovor. A two-phase resource constrained shortest path algorithm for acyclic graphs. *Cahiers du GERAD - G9648*, 1999.

[27] P. B. Mirchandani and R. L. Francis. *Discrete location theory*. 1990.

[28] S. Mirzaei and S. Wøhlk. A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, 8(1):1–33, mar 2019. doi: 10.1007/s13676-016-0096-x.

[29] P. Munari and M. Savelsbergh. Compact formulations for split delivery routing problems. *Transportation Science*, jan 2022. doi: 10.1287/trsc.2021.1106.

[30] İ. Muter, Ş. İ. Birbil, and G. Şahin. Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems. *INFORMS Journal on Computing*, 22(4):603–619, nov 2010. doi: 10.1287/ijoc.1090.0376.

[31] Y. Nakao and H. Nagamochi. A DP-based heuristic algorithm for the discrete split delivery vehicle routing problem. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 1(2):217–226, 2007. doi: 10.1299/jamdsm.1.217.

[32] S. N. Parragh and V. Schmid. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 40(1):490–497, jan 2013. doi: 10.1016/j.cor.2012.08.004.

[33] M. Qiu, Z. Fu, R. Eglese, and Q. Tang. A tabu search algorithm for the vehicle routing problem with discrete split deliveries and pickups. *Computers & Operations Research*, 100:102–116, dec 2018. doi: 10.1016/j.cor.2018.07.021.

[34] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter. On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2-3):343–359, jan 2003. doi: 10.1007/s10107-002-0323-0.

[35] G. Righini and M. Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, sep 2006. doi: 10.1016/j.disopt.2006.05.007.

[36] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, and E. Uchoa. Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing*, 31(2):251–267, apr 2019. doi: 10.1287/ijoc.2018.0822.

[37] M. Salani and I. Vacca. Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research*, 213(3):470–477, sep 2011. doi: 10.1016/j.ejor.2011.03.023.

[38] M. M. Silva, A. Subramanian, and L. S. Ochi. An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research*, 53:234–249, 2015. doi: https://doi.org/10.1016/j.cor.2014.08.005.

[39] G. Soleilhac. *Optimisation de la distribution de marchandises avec sous-traitance du transport: une problématique chargeur*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique, 2022.

[40] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, apr 1987. doi: 10.1287/opre.35.2.254.

[41] E. D. Taillard. A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO - Operations Research*, 33(1):1–14, jan 1999. doi: 10.1051/ro:1999101.

[42] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346, nov 2003. doi: 10.1287/ijoc.15.4.333.24890.

[43] Z. Wang, Y. Li, and X. Hu. A heuristic approach and a tabu search for the heterogeneous multi-type fleet vehicle routing problem with time windows and an incompatible loading constraint. *Computers & Industrial Engineering*, 89:162–176, nov 2015. doi: 10.1016/j.cie.2014.11.004.

[44] H. Zbib and G. Laporte. The commodity-split multi-compartment capacitated arc routing problem. *Computers & Operations Research*, 122:104994, oct 2020. doi: 10.1016/j.cor.2020.104994.