A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem

Claudia Archetti, Nicola Bianchessi, M. Grazia Speranza
Department of Economics and Management,
University of Brescia, Brescia, Italy
Email: {claudia.archetti,nicola.bianchessi,grazia.speranza}@unibs.it

May 5, 2015

Abstract

We consider the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP), a routing problem where customers may request multiple commodities. The vehicles can deliver any set of commodities, and multiple visits to a customer are allowed only if the customer requests multiple commodities. If the customer is visited more than once, the different vehicles will deliver different sets of commodities. Allowing the splitting of the demand of a customer only for different commodities may be more costly than allowing also the splitting of each individual commodity, but at the same time it is easier to organize and more acceptable to customers. We model the C-SDVRP by means of a set partitioning formulation and present a branch-price-and-cut algorithm. In the pricing phase, the ng-path relaxation of a constrained elementary shortest path problem is solved with a label setting dynamic programming algorithm. Capacity cuts are added in order to strengthen the lower bound. We solve to optimality within 2 hours instances with up to 40 customers and 3 commodities per customer.

1 Introduction

The class of vehicle routing problems (VRPs) is one of the largest and most studied classes of combinatorial optimization problems. It is also one of the most computationally challenging classes. In VRPs capacitated vehicles are used to distribute products and satisfy the demand of a set of customers. With very few exceptions, vehicle routing problems model the distribution of a single product. The underlying assumption is that only the volume or the weight of products matters. Thus, even if several products are demanded by customers, the demand of a customer is expressed with a single number, the total weight or volume of the demanded products. Accordingly, the capacity of the vehicles is expressed in weight or volume, depending on which constraint is more binding. We refer to Toth and Vigo (2014) for a recent collection of chapters on different VRPs.

There exist distribution problems where different commodities, or groups of commodities, have to be treated individually. When different commodities require different temperatures, such as in the case of frozen, fresh and dry food, vehicles with compartments may be used. In waste management different kinds of waste cannot be mixed up and must be kept separated at any stage of the collection problem. If a single vehicle is dedicated to the collection, then the vehicle container must be divided in compartments, one for each kind of waste. We refer to Derigs et al. (2011) for a recent work on a multi-compartment routing problem, with fixed size of the compartments, and to references therein. Also for organizational purposes, different commodities may be handled by means of dedicated vehicles. Having an individual product loaded on a vehicle makes the loading and the unloading of a vehicle much simpler and avoids the need for any reshuffling of the load during the distribution process.

In this paper we consider the Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP) introduced in Archetti et al. (2014), where different commodities are distributed to customers with capacitated vehicles. The vehicles are flexible and can deliver any set of commodities. A customer may be served by more than one vehicle but a single commodity can be delivered to each customer by one vehicle only. Each vehicle starts from a depot, visits a set of customers and returns to the depot at the end of the tour. Any customer may request any set of commodities. A vehicle that carries multiple commodities is totally flexible, that is it can carry any amount of any commodity, provided the constraint on the vehicle capacity is satisfied. We assume that the demand of a commodity of each customer does not exceed the capacity of a vehicle. The problem is a relaxation of the classical Capacitated Vehicle Routing Problem, as multiple visits to customers are allowed. On the other hand, it is more constrained than the Split Delivery Vehicle Routing Problem as the entire demand of a commodity must be delivered by the same vehicle. The concept is that, while allowing split deliveries may be unacceptable to customers, allowing different commodities to be delivered by different vehicles may be more acceptable. In addition to the applications mentioned above, the C-SDVRP models the situation where different companies share a warehouse and a fleet of vehicles for the distribution of their products.

One commodity is associated with one company. A vehicle may deliver products of different companies to different customers or to the same customers. Different vehicles may serve the same customer only if they deliver products of different companies.

In Archetti et al. (2014) the C-SDVRP is introduced and compared with alternative ways of distributing multiple commodities, such as by allowing the splitting of individual commodities or by using vehicles dedicated to individual commodities. The tests were performed on 64 small instances, with 15 customers and up to 3 commodities, 80 mid-size instances with 20, 40, 60, 80 customers and up to 3 commodities, and large instances with 100 customers. A branch-and-cut algorithm was able to solve 25 out of 64 small instances to optimality within 30 minutes. All the remaining instances were heuristically solved by creating multiple copies of each customer, one for each commodity required, and using a heuristic for the Capacitated Vehicle Routing Problem.

In this paper we focus on the exact solution of the C-SDVRP. We formulate the problem through a set partitioning formulation and adopt a branch-price-and-cut approach. In the pricing phase, the ng-path relaxation of a constrained elementary shortest path problem is solved by means of a label setting dynamic programming algorithm. Capacity cuts are added in order to strengthen the lower bound. We tested the algorithm on the small and mid-size instances tested in Archetti et al. (2014), with a time limit of 2 hours. We solved to optimality all small instances within a few seconds, except for three instances for which the computing time is larger than 100 seconds. For the mid-size instances, we could solve to optimality 19 out of 20 instances with 20 customers and 5 out of 20 instances with 40 customers. For all instances with up to 60 customers, except one, we found a lower and an upper bound. The average and maximum optimality gap are 0.63% and 2.41%, respectively. Over all the mid-size instances, we improved 20 out of 80 best known solutions.

In Section 2 the C-SDVRP is described and in Section 3 the set partitioning formulation is presented. The structure of the branch-price-and-cut algorithm is described in Section 4, together with the formulation and solution of the pricing problem, and the branching scheme. The computational results are presented in Section 5.

2 Problem definition

The Commodity constrained Split Delivery Vehicle Routing Problem (C-SDVRP) can be defined on a directed graph $G = (V, A)$ with vertex set $V = \{0, \dots, n\}$ and arc set $A = \{(i, j) : i \neq j; i, j \in V\}$. The vertex set V contains vertex 0, representing the depot, and the set $N = \{1, \dots, n\}$, representing the n customers. A cost c_{ij} is associated with each arc $(i, j) \in A$ and represents the non-negative cost of traversing arc (i, j). Travel costs satisfy the triangle inequality. Let $K = \{1, \dots, m\}$ be the set of commodities that have to be distributed from the depot to the customers. The demand of commodity $k \in K$ to be delivered to customer $i \in N$ is denoted by d_{ik}. The set $K_i = \{k \in K \mid d_{ik} > 0\}$ contains the commodities to be delivered to customer $i \in N$. We define as F the fleet of identical vehicles that is available to serve the customers and as Q the vehicle capacity. Vehicles are flexible and can deliver any subset of commodities. Each customer may be visited more than once. When a commodity is delivered by a vehicle to a customer, the entire amount requested by the customer is provided. Thus, multiple visits to a customer are allowed only if the customer requests multiple commodities. If a customer receives multiple visits, it means that the different vehicles will deliver different commodities. The objective is to find a set of routes serving all the customers in such a way that the total traveling cost is minimized.

3 Formulation

We model the problem by means of a set partitioning formulation making use of an exponential number of variables, each associated with a different feasible route. We adopt the following notation. We define as R the set of all feasible routes. A route corresponds to a non empty cycle in graph G starting from and ending at the depot. For each route $r \in R$, let $c^r = \sum_{(i,j) \in r} c_{ij}$ be the cost associated with the route. Then, let a^r_{ik}, e^r_i and b^r_{ij} be binary coefficients equal to 1 if commodity k is delivered to customer $i \in N$, customer $i \in N$ is visited, and arc $(i, j) \in A$ is traversed in route r, respectively, and 0 otherwise. The C-SDVRP can be formulated as follows:

$$\min \sum_{r\in R} c^r \lambda^r \tag{1}$$

$$\text{s.t.:} \sum_{r\in R} a^r_{ik}\lambda^r \geq 1 \qquad i \in N, \quad k \in K_i \tag{2}$$

$$\sum_{r\in R} \lambda^r = \phi \tag{3}$$

$$\sum_{r\in R} e^r_i \lambda^r = z_i \qquad i \in N \tag{4}$$

$$\sum_{r\in R} b^r_{ij}\lambda^r = x_{ij} \qquad (i,j) \in A \tag{5}$$

$$1 \leq z_i \leq \max\{|K_i|, |F|\} \text{ and integer} \qquad i \in N \tag{6}$$

$$\left\lceil \frac{\sum_{i\in N}\sum_{k\in K_i} d_{ik}}{Q} \right\rceil \leq \phi \leq |F| \text{ and integer} \tag{7}$$

$$0 \leq x_{ij} \leq |F| \text{ and integer} \qquad (i,j) \in A \tag{8}$$

$$\lambda^r \in \{0,1\} \qquad r \in R, \tag{9}$$

where x_{ij} and z_i are integer variables representing the number of times the vehicles traverse arc $(i,j) \in A$ and visit customer $i \in N$, respectively, ϕ is an integer variable representing the number of vehicles used, and λ^r is a binary variable equal to 1 if route $r \in R$ is assigned to a vehicle. The objective function (1) aims at minimizing the total traveling cost. Constraints (2) ensure that all commodities requested by each customer will be delivered. Constraints (3) and (7) bound the number of vehicles that can be used. (4) and (6) bound the number of times a customer $i \in N$ can be visited. (5) and (8) bound the number of times an arc $(i,j) \in A$ can be traversed. Finally, (9) state that λ^r are binary variables. Note that x_{ij} can take any integer value between 0 and $|F|$ contrary to what happens in the Split Delivery Vehicle Routing Problem (SDVRP) where x_{ij} is a binary variable for $i,j \in N$. In fact, in Archetti et al. (2014) the authors show that in the C-SDVRP arcs joining two customers can be traversed more than once.

4 Branch-price-and-cut algorithm

In the following we describe the algorithm designed to solve the set partitioning formulation (1)–(9) which will be referred to as the Master Problem (MP).

In order to solve the MP we design a branch-price-and-cut algorithm (Barnhart et al. (1998); Desaulniers et al. (2005)), that is a branch-and-bound algorithm where, at each node of the branch-and-bound tree, λ^r variables are generated by means of column generation while addressing the linear relaxation of the problem associated with the node. We denote the linear relaxation of the MP, restricted to a subset of columns, as Restricted Linear Master Problem (RLMP). At each column generation iteration, a pricing problem is solved to generate negative reduced cost variables to be added to the RLMP. When no negative reduced cost variable is found it means that the Linear Master Problem (LMP), that is the linear relaxation of the MP, has been solved to optimality and the column generation algorithm ends. If the LMP solution is fractional, before applying branching rules, we possibly insert violated valid inequalities in the current RLMP and iterate its solution process in order to strengthen the bound.

In the following subsections the main components of the algorithm are described in details. The term *path* is used as a synonymous for *route*.

4.1 Pricing problem formulation

Given the solution values of the dual variables μ_{ik}, ρ, θ_i, σ_{ij} associated with constraints (2), (3), (4), and (5) of the RLMP, respectively, the pricing problem can be modeled as follows:

$$\min \sum_{(i,j)\in A} \bar{c}_{ij} X_{ij} - \sum_{i\in N}\sum_{k\in K_i} \mu_{ik} Y_{ik} \tag{10}$$

$$\text{s.t.:} \sum_{j\in N} X_{0j} = 1 \tag{11}$$

$$\sum_{(i,j)\in\delta^+(i)} X_{ij} = \sum_{(j,i)\in\delta^-(i)} X_{ji} = Z_i \qquad i \in N \tag{12}$$

$$\sum_{(i,j)\in\delta^+(S)} X_{ij} \geq Z_s \qquad S \subseteq N, |S| \geq 2, s \in S \tag{13}$$

$$Y_{ik} \leq Z_i \qquad i \in N,\ k \in K_i \tag{14}$$

$$\sum_{k\in K_i} Y_{ik} \geq Z_i \qquad i \in N \tag{15}$$

$$\sum_{i\in N}\sum_{k\in K_i} d_{ik} Y_{ik} \leq Q \tag{16}$$

$$Y_{ik} \in \{0,1\} \qquad i \in N,\ k \in K_i \tag{17}$$

$$Z_i \in \{0,1\} \qquad i \in N \tag{18}$$

$$X_{ij} \in \{0,1\} \qquad (i,j) \in A, \tag{19}$$

where $\bar{c}_{ij}$ is the reduced cost of arc $(i,j) \in A$ ($\bar{c}_{ij} = c_{ij} - \sigma_{ij} - \theta_j$ if $j \neq 0$, $\bar{c}_{ij} = c_{ij} - \sigma_{ij} - \rho$ if $j = 0$), $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$. For the ease of notation, we write $\delta^+(i)$ and $\delta^-(i)$ when $S = \{i\}$. Binary variables X_{ij}, Z_i and Y_{ik} are equal to 1 if arc $(i,j) \in A$ is traversed, customer $i \in N$ is visited, and commodity $k \in K_i$ is delivered to customer $i \in N$, respectively, and 0 otherwise.

The objective function (10) aims at minimizing the reduced cost of the route. Constraints (11) and (12) are the degree constraints, whereas constraints (13) are subtour elimination constraints. Consistency between the Y_{ik} and Z_i variables is imposed through constraints (14) and (15). In particular, constraints (15) ensure that a delivery will take place at each customer visited. Finally, inequalities (16) impose the capacity constraint for the route.

The pricing problem combines an Elementary Shortest Path Problem with a 0-1 Knapsack Problem augmented by additional constraints. Actually, for a given path, determining the optimal values of the Y_{ik} variables corresponds to solving the following problem:

$$\max \sum_{i \in \overline{N}} \sum_{k \in K_i} \mu_{ik} Y_{ik} \tag{20}$$

$$\text{s.t.:} \sum_{i \in \overline{N}} \sum_{k \in K_i} d_{ik} Y_{ik} \leq Q \tag{21}$$

$$\sum_{k \in K_i} Y_{ik} \geq 1 \qquad i \in \overline{N} \tag{22}$$

$$Y_{ik} \in \{0,1\} \qquad i \in \overline{N},\ k \in K_i \tag{23}$$

where $\overline{N}$ is the set of customers visited along the path. While addressing the SDVRP by means of column generation, the resulting pricing problem is such that in each solution path at most one customer will receive a split delivery greater than 1 and less than his demand (see Archetti et al. (2011)). Here, the structure of (20)-(23) is such that any customer visited along the solution path of (10)-(19) may receive any subset of the commodities requested.

In order to address the pricing problem (10)-(19), we formulate it as an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC) defined over an expanded graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$.

We define as $\mathcal{G}(i) = (\mathcal{V}_i, \mathcal{A}_i)$ the subgraph induced by the vertices corresponding to customer i. The set of vertices $\mathcal{V}_i$ is composed of vertices v_s^i, v_t^i and v_k^i for each $k \in K_i$. The arc set $\mathcal{A}_i$ includes the following arcs:

- arcs (v_s^i, v_k^i), $k \in K_i$, with an associated cost equal to $-\mu_{ik}$;

- arcs (v^i_r, v^i_q), $r < q$, $r, q \in K_i$, with an associated cost equal to $-\mu_{iq}$;
- arcs (v^i_k, v^i_t), $k \in K_i$, with an associated zero cost.

Moreover, a quantity equal to d_{ik} is associated with each arc entering in v^i_k, $i \neq 0$ and $k \in K_i$, meaning that when vertex v^i_k is visited, a quantity equal to d_{ik} is delivered to i.

The vertex set $\mathcal{V}$ of the expanded graph includes two different vertices, s and t, associated with the starting and ending depot, and the set $\mathcal{V}_i$ of $|K_i| + 2$ vertices associated with each customer $i \in N$. The arc set $\mathcal{A}$ includes the sets of arcs $\mathcal{A}_i$, $i \in N$, and an arc (v^i_t, v^j_s) for each arc $(i, j) \in A$, with cost equal to $\bar{c}_{ij}$, $v^0_t = s$ and $v^0_s = t$. Thus, each subgraph $\mathcal{G}(i)$ corresponds to an acyclic network (see Figure 1), where v^i_s is the only vertex with incoming arcs originating from vertices not belonging to $\mathcal{V}_i$ and v^i_t is the only vertex with outgoing arcs to vertices not belonging to $\mathcal{V}_i$.

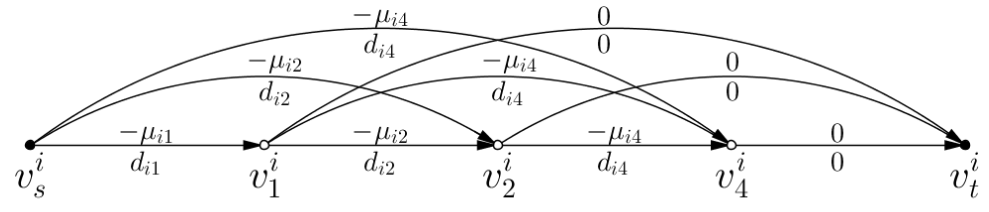

Figure 1: Acyclic network $\mathcal{G}(i)$ for customer $i \in N$, with $K_i = \{1, 2, 4\}$.

The pricing problem consists in finding the least (reduced) cost elementary path from s to t in $\mathcal{G}$, such that the quantity delivered along the path is less than or equal to Q. The ESPPRC defined over graph $\mathcal{G}$ is equivalent to (10)-(19).

The ESPPRC is NP-hard in the strong sense (see Dror (1994)). When the number of vertices is relatively large, the pricing problem may be very difficult to solve. We thus apply the ng-path relaxation of the problem that allows the generation of paths that may contain cycles (see Baldacci et al. (2011)).

4.2 Pricing problem solution

We solve the pricing problem by means of a label setting dynamic programming algorithm (see Irnich and Desaulniers (2005)). The main idea of this technique is to build feasible paths, starting from the source vertex of the graph, and extending them from vertex to vertex in all feasible directions. A label (R, C, i) is associated with each partial path from the source of the graph to vertex i. Each component of the vector R represents the consumption of a resource along the path until vertex i, whereas C is the cost of the path. When (R, C, i) is extended to another label (R', C', j) associated with vertex j, the components of R and the cost C must be updated according to the resource consumption and the cost associated with arc (i, j). Extensions must be feasible with respect to the consumptions of all the resources. To the sake of efficiency, dominance rules are applied to discard dominated paths reaching the same vertex. The solution is the path represented by the minimum cost label associated with vertex t.

Let us first consider the non-elementary version of the pricing problem. We solve it by using labels of the form (q, C, v_b^i), where $b \in \{s, t\} \cup \{k | k \in K_i\}$ and q is the quantity loaded on the vehicle when leaving vertex v_b^i. The initial label at vertex $v_t^0 = s$ is $(0, 0, v_t^0)$. At a given vertex v_b^i, a partial path associated with label (q, C, v_b^i) is feasible if $q \leq Q$. Let us define $\tilde{c}_{i_b j_{b'}}$ as the cost of arc $(v_b^i, v_{b'}^j) \in \mathcal{A}$. When arc $(v_b^i, v_{b'}^j) \in \mathcal{A}$ is traversed, a label (q, C, v_b^i) is extended to the label $(q', C', v_{b'}^j)$, where $C' = C + \bar{c}_{i_b j_{b'}}$ and the extension rule for the q component of the label is defined as follows. Along arcs (v_t^i, v_s^j) and (v_k^i, v_t^i), $i, j \in N$, $k \in K_i$, the resource consumption does not vary and its value is simply reported in the new label. Along arcs (v_s^i, v_k^i) and (v_r^i, v_q^i), $i \in N$, $k, r, q \in K_i$, the resource consumption is increased by d_{ik} and d_{iq}, respectively. Dominance rules are then applied in order to discard paths that are dominated by other paths. Label (q', C', v_b^i) dominates label (q'', C'', v_b^i) if $q' \leq q''$, $C' \leq C''$, and one of the two conditions is strictly satisfied.

The ng-path relaxation of the pricing problem is obtained considering a further component in the label definition, that is a set B of $|N|$ binary resources. A subset of customers $N_i \subseteq N$ is defined for each customer $i \in N$. N_i includes i and its closest ν neighbors, where ν is a positive integer value. Each element of B is set to 0 in the initial label at vertex s. Let us consider then a partial path ending at vertex $v_t^i \in \mathcal{V}$. The extension of the path along arc $(v_t^i, v_s^j) \in \mathcal{A}$ is feasible with respect to component B if one of the following conditions holds:

- $v_s^j = t$, or equivalently $j = 0$;
- $j \notin N_i$;
- $j \in N_i$ and $B(j) = 0$.

When arc (v_t^i, v_s^j) is traversed, component B is extended as follows. If $v_s^j = t$, $B'(u)$ is set to 0 for each $u \in N$; otherwise $B'(u)$ is set to 0 for all $u \notin N_i \cap N_j$, $B'(u)$ becomes equal to $B(u)$ for all $u \in N_i \cap N_j$, and finally $B'(j)$ is set to 1. When one arc joining two vertices associated with the same customer is traversed, i.e., one of the arcs (v_s^i, v_k^i), (v_k^i, v_q^i) or (v_k^i, v_t^i) with $k, r \in K_i$, component B does not change.

Label (q', B', C', v_b^i) dominates label (q'', B'', C'', v_b^i) if $q' \leq q''$, $C' \leq C''$, $B'(u) \leq B''(u)$ for each $u \in N_i$, and one of the conditions is strictly satisfied.

An ng-path in graph $\mathcal{G}$ may contain a cycle $v_s^i = i_1, i_2, \dots, i_r = v_s^i$, iff $i \notin N_{i_q}$ for some $q \in \{2, \dots, r-1\}$ (see Baldacci et al. (2011) for more details). Note that, when we serve customer i, vertices v_s^i and v_t^i are visited together with at least one vertex v_k^i with $k \in K_i$. Since graph $\mathcal{G}(i)$ is acyclic, a cycle in $\mathcal{G}$ is obtained when at least two customers are considered. Thus, a cycle $v_s^i = i_1, i_2, \dots, i_r = v_s^i$ visits at least 7 vertices, i.e., $r \geq 7$: three vertices correspond to the service of the first customers, 3 vertices correspond to the service of the second vertex and a further vertex is visited to return to the first customer.

Preprocessing. For a given set of dual variable values, each partial path reaching vertex v_s^i, $i \in N$, gives rise to at most β_i non-dominated partial paths ending at node v_t^i, where β_i is the number of Pareto-optimal solutions of the following bi-objective problem:

$$\max \sum_{k \in K_i} \mu_{ik} \mathcal{Y}_k \tag{24}$$

$$\min \sum_{k \in K_i} d_{ik} \mathcal{Y}_k \tag{25}$$

$$\text{s.t.:} \sum_{k \in K_i} d_{ik} \mathcal{Y}_k \leq Q \tag{26}$$

$$1 \leq \sum_{k \in K_i} \mathcal{Y}_k \leq |K_i| \tag{27}$$

$$\mathcal{Y}_k \in \{0,1\} \qquad k \in K_i, \tag{28}$$

where $\mathcal{Y}_k$ is a binary variable equal to 1 if commodity $k \in K_i$ is delivered to customer i, and 0 otherwise. The objective function (24) aims at maximizing the values of the dual variables μ_{ik} associated with commodities $k \in K_i$, while (25) aims at minimizing the capacity consumption. Constraints (26) and (27) bound the capacity consumption and the number of commodities that are delivered to customer i, respectively. The role of constraint (27) will be explained in Section 4.4.

For each $i \in N$, we define $\mathcal{S}_i$, with $|\mathcal{S}_i| = \beta_i$, as the set of Pareto-optimal solutions of the corresponding problem (24)-(28). The Pareto-optimal solutions in set $\mathcal{S}_i$ can be represented by an acyclic network $\overline{\mathcal{G}}(i)$ with source and destination vertices v_s^i and v_t^i, respectively, and a path between v_s^i and v_t^i for each solution in $\mathcal{S}_i$. Each path consists of a sequence of vertices v_k^i, $k \in K_i$, those corresponding to the commodities identified by the solution (see Figure 2). Set $\mathcal{S}_i$ is found by solving a SPPRC on the acyclic network $\mathcal{G}(i)$ (see Frieze (1976)) and keeping the non-dominated solutions.

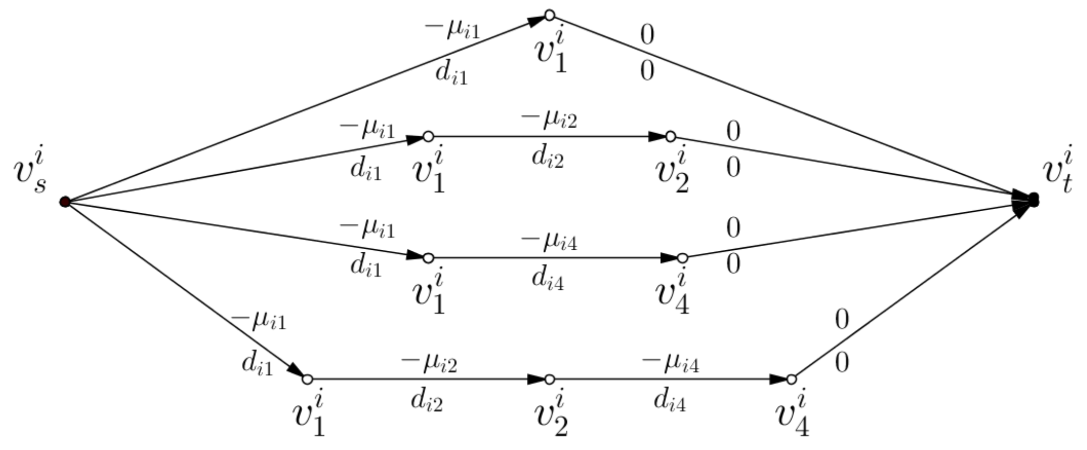

Figure 2: Acyclic network $\overline{\mathcal{G}}(i)$ for customer $i \in N$, with $K_i = \{1, 2, 4\}$, $d_{i1} = 2$, $d_{i2} = 3$, $d_{i4} = 5$, $\mu_{i1} = 4$, $\mu_{i2} = 2$, $\mu_{i4} = 5$, and $\mathcal{S}_i = \{\{1\}, \{1, 2\}, \{1, 4\}, \{1, 2, 4\}\}$.

Replacing networks $\mathcal{G}(i)$ with networks $\overline{\mathcal{G}}(i)$ helps in avoiding redundant computations. The replacement is performed at every column generation iteration as the value of the dual variables changes at each iteration.

Acceleration strategies. In order to speed up the label setting algorithm we consider two different acceleration techniques. The first one is the decremental state space relaxation technique (Righini and Salani (2008)). The idea of this technique is to iteratively solve the pricing problem by enlarging, at each iteration, the set of vertices for which the elementarity is required. When combined with ng-paths, the procedure is such that sets N_i are initially empty and are enlarged at each iteration. In particular, each customer $i \in N$ is associated with a set M_i of binary values with $|M_i| = |N|$. At the first iteration, $M_i(i) = 1$ and $M_i(u) = 0$, $u \neq i$, $u \in N$. Then, the pricing problem is solved with a modified extension rule for component B when arc $(v_t^i, v_s^j) \in \mathcal{A}$ is traversed:

- if $v_s^j = t$, $B'(u) = 0$, $u \in N$;
- otherwise, $B'(u) = 0$, $u \notin \tilde{N}_i \cap \tilde{N}_j$; $B'(u) = B(u)$, $u \in \tilde{N}_i \cap \tilde{N}_j$; and $B'(j) = 1$,

where $\tilde{N}_i = \{u \in N_i | M_i(u) = 1\}$. Once vertex v_0^t is reached and an $s - t$ path is constructed, the procedure checks whether this path is a feasible ng-path, i.e., it contains no cycle with respect to sets N_i, $i \in N$. If the path is a feasible ng-path, then the algorithm terminates. Otherwise, the first vertex v_s^u in the path that defines a forbidden cycle is selected, $M_i(u)$ is set to 1 for each $i \in N$ such that $u \in N_i$, and the solution process is iterated. In practice, the elementarity is not required at the first iteration. The problem is solved and the first vertex u which generates a cycle that is forbidden in the ng-path relaxation is added to all sets $\tilde{N}_i$ such that $u \in N_i$.

The second technique we implemented is the 2-cycle elimination proposed in Christofides et al. (1981). The predecessor of the last vertex visited is considered as a further component in the label definition. The predecessor is updated whenever arc $(v_t^i, v_s^j) \in \mathcal{A}$ is traversed and remains unchanged when the path is extended through the other arcs of the expanded graph.

Heuristic column generation. To accelerate the solution of the LMP, at each column generation iteration heuristic versions of the label setting algorithm are applied before solving the pricing problem to optimality. While solving the pricing problem heuristically, the decremental state space relaxation technique is not applied and the dominance among labels is checked without considering the values of component B.

At each column generation iteration we derive from graph $\mathcal{G}$ a graph $\mathcal{G}'$ where, for each $i \in N$, a subset of the outgoing arcs $(v_t^i, v_s^j) \in \mathcal{A}$, $j \in N$, is considered. This subset includes the $\bar{n}_a$ outgoing arcs with lowest reduced cost values. Different heuristic algorithms are considered. The first two heuristics look for negative reduced cost paths in graphs $\mathcal{G}'$ and $\mathcal{G}$, respectively, not allowing the possibility to perform split deliveries. The last two heuristics work again on graphs $\mathcal{G}'$ and $\mathcal{G}$, respectively, but allow the possibility to perform split deliveries.

At each iteration of the column generation algorithm the heuristics are sequentially applied. The iteration finishes as soon as a heuristic succeeds in finding negative cost paths.

4.3 Valid inequalities

When the LMP solution is fractional and the node cannot be pruned, before starting the branching phase we look for violated capacity constraints which are formulated as follows:

$$\sum_{r \in R} \sum_{(i,j) \in \delta^+(S)} b_{ij}^r \lambda^r \geq \left\lceil \frac{\sum_{i \in S} \sum_{k \in K_i} d_{ik}}{Q} \right\rceil \quad S \subseteq N. \tag{29}$$

In order to identify violated inequalities (29), we implemented the *shrinking heuristic* presented in Ralphs et al. (2003). In particular, two variants of the *shrinking heuristic* are implemented, namely the *extended shrinking heuristic* and the *greedy shrinking heuristic*. The reader is referred to Ralphs et al. (2003) for more details on these algorithms.

4.4 Branching

When the optimal solution of the current LMP $(\tilde{\boldsymbol{\lambda}}, \tilde{\phi}, \tilde{\mathbf{z}}, \tilde{\mathbf{x}})$ is fractional, different branching rules are hierarchically applied. The rules are presented in the following, in order of priority.

The first branching rule is on the fractional number of vehicles used. Then, the number of visits to each customer is considered. If $\tilde{z}_i$ is fractional for some $i \in N$, we branch on it. We give priority to $\tilde{z}_i$ lower than 2. This allows us to forbid split deliveries to customer i when z_i is fixed to 1. Moreover, when z_i is constrained to be greater than or equal to $\alpha > 1$, then the right-hand side term of constraint (27) is set equal to $|K_i| - (\alpha - 1)$. For each class of priority, we choose the customer i with fractional part of $\tilde{z}_i$ closest to 0.5.

The third branching rule is related to the arc use, i.e., to variables $\tilde{\mathbf{x}}$. If $\tilde{x}_{ij}$ is fractional for some $(i, j) \in A$, then we branch on the use of arc (i, j). As before, we choose the arc (i, j) with fractional part of $\tilde{x}_{ij}$ closest to 0.5.

Finally, if none of the above branching decisions can be imposed, we consider each pair of different commodities p and q, $p \in K_i$, $q \in K_j$, $i, j \in N$, and compute $\kappa_{pq} = \sum_{r \in R} a^r_{pi} a^r_{qj} \tilde{\lambda}^r$, i.e. κ_{pq} is the sum of the λ variables associated with routes delivering commodities p and q to customers i and j, respectively. If κ_{pq} is fractional for some pair p and q, we branch on it. On one branch we set $\kappa_{pq} = 0$, meaning that commodities p and q must be delivered in different routes, while on the other branch we set $\kappa_{pq} = 1$, meaning that commodities p and q must be delivered in the same route. The subproblem is modified accordingly. When κ_{pq} is set to 0, a new binary resource B_{pq} must be introduced in the label definition in order to prevent the delivery of both commodities. B_{pq} is set to 0 in the initial label and is increased by one whenever commodity p or q is delivered. Label L' is feasible w.r.t. resource B_{pq} if $L'(B_{pq}) \leq 1$, and may dominate label L'' if $L'(B_{pq}) \leq L''(B_{pq})$. When κ_{pq} is set to 1, two new binary resources, B_p and B_q, are required. B_p (B_q) is set to 0 in the initial label and is increased by one whenever commodity p (q) is delivered. Label L' is feasible w.r.t. resource B_p (B_q) if $L'(B_p) \leq 1$ ($L'(B_q) \leq 1$). In particular, label L' may dominate label L'' if $L'(B_p) = L''(B_p)$ and $L'(B_q) = L''(B_q)$, and is extended to vertex t if $L'(B_p) = L'(B_q)$. We give priority to κ_{pq} values where commodities p and q are associated with the same customer. For each class of priority, we choose the κ_{pq} value closest to 0.5. This branching rule implements the Ryan and Foster (2003) rule and guarantees the correctness of the solution algorithm (see Barnhart et al. (1998)).

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

5 Experimental results

The branch-price-and-cut algorithm was implemented in C++ and compiled in release mode under MS Visual Studio Express 2013 for Windows Desktop (64-bit version). The experiments were carried out on a 64-bit Windows machine, with Intel Xeon processor W3680, 3.33 GHz, and 12 GB of RAM. CPLEX 12.5 (64 bit version) was used to solve the linear relaxation of the MPs at each column generation iteration. The overall execution time limit for each run was set to 2 hours. A single thread was used in all experiments.

After a preliminary testing phase we fixed the parameters of the branch-price-and-cut algorithm as follows. The number of neighbors ν used while defining the ng-path relaxation of the pricing problem is set to $\lceil \frac{n}{2} \rceil$, provided that $10 \leq \lceil \frac{n}{2} \rceil \leq 15$. If $\lceil \frac{n}{2} \rceil < 10$ then $\nu = 10$, while $\nu = 15$ when $\lceil \frac{n}{2} \rceil > 15$. The parameter $\bar{n}_a$ used in the heuristic column generation phase is set to $min\{10, (n+1) * 2^{-4}\}$. Finally, violated inequalities are identified up to the fifth level of the branch-and-bound tree and, then, only when the branching has no impact on the lower bound, i.e., when the lower bound of the child node is equal to the lower bound of the father node.

5.1 Test Instances

We tested the branch-price-and-cut algorithm on benchmark instances for the C-SDVRP proposed in Archetti et al. (2014). Three sets of instances with up to 3 commodities were generated: small instances with $n = 15$, mid-size instances with $n = 20, 40, 60, 80$, and large instances with $n = 100$. We consider only the small and the mid-size instances, as no significant result could be obtained on large instances.

Small instances have the following characteristics:

- Customer locations: The first 15 locations in the R101 and C101 Solomon's instances are considered;
- Number of commodities: Two and three;
- Probability that a customer requires a commodity: Two cases are considered. Probability set to 100% means that each customer requires each commodity. Probability set to 60% means that the probability that a customer requires a commodity is 60%;
- Demand range: This data represents the interval for the generation of the demand of each customer for each commodity. Two intervals are considered: [1;100] and [40;60];

- Vehicle capacity: Denoting as $d_{max} = max_{i \in C} \sum_{j=1}^{m} d_{ij}$, the vehicle capacity is generated as $Q = \alpha d_{max}$. Four values of α are considered: 1.1, 1.5, 2 and 2.5.

One instance for each of the 64 combinations of characteristics was generated.

Mid-size instances have a number of commodities equal to three, demand range [1;100], and $\alpha = 1.5$. For each of the 16 combinations of value of n, type of instance (R101 and C101), and value of the probability (60% and 100%), 5 different instances were generated by randomly choosing customers from the original instance. The total number of mid-size instances is 80.

5.2 Computational results on small instances

Detailed results on the set of small instances are reported in Table 1. The first four columns report data on the instance: name of the original instance, number of commodities (m), probability that a customer requires a commodity (p), demand range (Δ) and value of α. The following 4+4 columns refer to the computational results. The first four columns report statistics on the solution obtained at the root node of the branch-and-bound tree: the value of the LMP as a percentage of the final lower bound value ($\underline{z}^*_{LMP}$), the time spent to compute $\underline{z}^*_{LMP}$ in seconds (t_{LMP}), the value of the LMP once the capacity cuts are added as a percentage of the final lower bound value ($\underline{z}^*_{LMP+cuts}$) and the time spent to compute $\underline{z}^*_{LMP+cuts}$ ($t_{LMP+cuts}$). The last four columns report statistics on the final solution obtained by the branch-price-and-cut algorithm: the lower bound value ($\underline{z}^*$), the upper bound value ($\overline{z}^*$), the optimality percentage gap (gap %) and the total execution time in seconds (t).

The algorithm is able to solve to optimality all instances. In particular, it solves 41 instances out of the 64 in less than 1 second and takes more than 10 seconds for only 6 instances. As mentioned in the introduction, Archetti et al. (2014) proposed a branch-and-cut algorithm for the C-SDVRP which was able to solve 25 out of the 64 instances within a time limit of 30 minutes.

Table 1: Small instances

Instance					root				$\underline{z}^*$	$\overline{z}^*$	gap	t
	m	p	Δ	α	$\underline{z}^*_{LMP}$	t_{LMP} (sec.)	$\underline{z}^*_{LMP+cuts}$	$t_{LMP+cuts}$ (sec.)			(%)	(sec.)
C101	2	0.6	[1;100]	1.1	100.00	0.168	-	-	283.3404	283.3404	0	0.169
		0.6	[40;60]	1.1	100.00	0.014	-	-	480.4342	480.4342	0	0.015
		1	[1;100]	1.1	99.63	0.099	99.76	0.124	422.4965	422.4965	0	0.261
		1	[40;60]	1.1	98.59	0.024	98.77	0.029	685.1662	685.1662	0	0.051
		0.6	[1;100]	1.5	99.81	0.189	100.00	0.221	241.0386	241.0386	0	0.222
		0.6	[40;60]	1.5	97.50	0.042	100.00	0.079	348.2731	348.2731	0	0.079
		1	[1;100]	1.5	95.77	0.148	99.61	0.222	340.5474	340.5474	0	1.070
		1	[40;60]	1.5	97.39	0.082	97.84	0.131	490.2973	490.2973	0	0.248
		0.6	[1;100]	1.1	100.00	0.287	-	-	200.9092	200.9092	0	0.287
		0.6	[40;60]	1.1	100.00	0.147	-	-	239.6829	239.6829	0	0.147
		1	[1;100]	1.1	99.61	0.212	99.97	0.426	239.9424	239.9424	0	0.688
		1	[40;60]	1.1	97.62	0.182	99.50	0.381	357.5929	357.5929	0	1.407
		0.6	[1;100]	1.5	99.36	0.453	100.00	1.102	170.0453	170.0453	0	1.103
		0.6	[40;60]	1.5	99.30	0.264	99.97	0.436	207.8259	207.8259	0	0.710
		1	[1;100]	1.5	99.53	0.654	100.00	0.962	205.7830	205.7830	0	0.963
		1	[40;60]	1.5	97.35	0.261	99.97	0.436	302.1813	302.1813	0	0.639
R101	2	0.6	[1;100]	1.1	97.82	0.076	99.68	0.140	408.8971	408.8971	0	0.321
		0.6	[40;60]	1.1	95.94	0.018	98.25	0.025	565.3383	565.3383	0	0.087
		1	[1;100]	1.1	99.45	0.086	99.67	0.119	537.2029	537.2029	0	0.310
		1	[40;60]	1.1	100.00	0.018	-	-	679.0232	679.0232	0	0.018
		0.6	[1;100]	1.5	98.28	0.116	99.89	0.246	353.0119	353.0119	0	0.324
		0.6	[40;60]	1.5	97.34	0.039	99.48	0.069	455.9724	455.9724	0	0.116
		1	[1;100]	1.5	97.50	0.105	99.23	0.163	443.0829	443.0829	0	0.259
		1	[40;60]	1.5	99.57	0.050	100.00	0.063	558.9565	558.9565	0	0.063
		0.6	[1;100]	1.1	100.00	0.230	-	-	301.4316	301.4316	0	0.230
		0.6	[40;60]	1.1	97.28	0.057	98.77	0.109	379.2848	379.2848	0	0.299
		1	[1;100]	1.1	97.24	0.245	100.00	0.481	370.5561	370.5561	0	0.482
		1	[40;60]	1.1	98.53	0.078	99.63	0.096	444.1868	444.1868	0	0.236
		0.6	[1;100]	1.5	99.73	0.331	100.00	0.403	280.7646	280.7646	0	0.403
		0.6	[40;60]	1.5	96.38	0.139	97.65	0.190	342.6854	342.6854	0	0.581
		1	[1;100]	1.5	98.45	0.509	99.98	0.887	323.5761	323.5761	0	1.459
		1	[40;60]	1.5	97.64	0.161	99.71	0.277	401.6087	401.6087	0	0.412
C101	3	0.6	[1;100]	1.1	97.76	0.184	100.00	0.312	333.4709	333.4709	0	0.312
		0.6	[40;60]	1.1	93.08	0.100	95.47	0.196	440.2241	440.2241	0	17.776
		1	[1;100]	1.1	93.96	1.827	98.42	5.359	428.5164	428.5164	0	740.205
		1	[40;60]	1.1	98.18	0.264	99.45	0.426	638.0919	638.0919	0	3.354
		0.6	[1;100]	1.5	93.79	0.292	100.00	0.637	262.8069	262.8069	0	0.638
		0.6	[40;60]	1.5	98.27	0.256	99.99	0.417	306.6363	306.6363	0	0.572
		1	[1;100]	1.5	96.43	2.694	99.96	5.137	315.9600	315.9600	0	6.394
		1	[40;60]	1.5	96.25	0.867	100.00	1.658	457.9430	457.9430	0	1.659
		0.6	[1;100]	1.1	99.26	0.392	-	-	204.9380	204.9380	0	7.360
		0.6	[40;60]	1.1	93.54	0.190	99.63	0.361	263.2896	263.2896	0	1.365
		1	[1;100]	1.1	93.20	1.945	99.93	4.343	265.0623	265.0623	0	7.256
		1	[40;60]	1.1	99.66	1.475	100.00	2.336	347.3580	347.3580	0	2.337
		0.6	[1;100]	1.5	99.43	0.967	99.94	1.300	168.2958	168.2958	0	3.075
		0.6	[40;60]	1.5	100.00	0.345	-	-	202.9044	202.9044	0	0.345
		1	[1;100]	1.5	98.64	35.196	99.86	62.526	206.6970	206.6970	0	116.820
		1	[40;60]	1.5	95.12	2.048	99.69	4.515	310.7978	310.7978	0	103.281
R101	3	0.6	[1;100]	1.1	97.54	0.095	100.00	0.212	401.7502	401.7502	0	0.212
		0.6	[40;60]	1.1	97.73	0.036	99.63	0.097	497.1385	497.1385	0	0.262
		1	[1;100]	1.1	97.90	0.586	99.41	1.129	491.0411	491.0411	0	49.043
		1	[40;60]	1.1	98.67	0.123	100.00	0.199	679.0232	679.0232	0	0.200
		0.6	[1;100]	1.5	98.32	0.241	98.62	0.328	347.3693	347.3693	0	1.230
		0.6	[40;60]	1.5	97.57	0.115	99.09	0.191	410.8130	410.8130	0	0.824
		1	[1;100]	1.5	96.64	1.408	97.75	1.931	409.2905	409.2905	0	8.755
		1	[40;60]	1.5	97.04	0.374	100.00	0.598	541.0336	541.0336	0	0.598
		0.6	[1;100]	1.1	100.00	0.404	-	-	303.1439	303.1439	0	0.405
		0.6	[40;60]	1.1	99.76	0.227	99.76	0.247	343.7159	343.7159	0	0.380
		1	[1;100]	1.1	98.89	1.709	100.00	2.811	345.8351	345.8351	0	2.813
		1	[40;60]	1.1	97.25	1.029	99.19	1.732	444.1868	444.1868	0	3.033
		0.6	[1;100]	1.5	100.00	0.886	-	-	278.2234	278.2234	0	0.887
		0.6	[40;60]	1.5	99.05	0.380	100.00	0.485	312.3100	312.3100	0	0.485
		1	[1;100]	1.5	97.77	25.233	99.03	33.867	320.3490	320.3490	0	43.270
		1	[40;60]	1.5	96.78	1.458	99.81	2.634	393.8357	393.8357	0	3.275

5.3 Computational results on mid-size instances

Detailed results on the set of mid-size instances are reported in Table 2. The first four columns report statistics on the instance: name of the original instance, number of customers (n), probability that a customer requires a commodity (p), instance id (id). We recall that, in the set of mid-size instances, 5 instances are generated for each combination of characteristics and that the number of commodities is set to three, the demand range is [1;100] and $\alpha = 1.5$. The following 4+4 columns refer to the computational results and have the same meaning as in Table 1. The column 'nodes explored' gives the number of nodes explored in the branch-and-bound tree. Finally, the last two columns report the value of the best known feasible solution, taken from Archetti et al. (2014), and the percentage gap between this value and the lower bound reported in column $\underline{z}^*$, respectively.

The algorithm is able to solve to optimality all, except one, instances with 20 customers and 5 instances with 40 customers. The number of nodes explored is less than 100 for all instances except 4 with 20 customers. The low number of nodes explored for the instances with 60 and 80 customers is due to the fact that in many cases the computing time is entirely spent to compute the lower bound at the root node.

Table 2: Mid-size instances

Instance				root									best	
	n	p	id	$\underline{z}^*_{LMP}$	t_{LMP} (sec.)	$\underline{z}^*_{LMP+cuts}$	$t_{LMP+cuts}$ (sec.)	$\underline{z}^*$	$\overline{z}^*$	gap (%)	t (sec.)	nodes explored	known value	gap (%)
C101	20	0.6	1	99.91	1.007	100.00	1.119	573.8617	573.8617	0	1.120	1	573.8617	0
			2	95.78	0.791	100.00	1.475	592.0651	592.0651	0	1.476	1	592.0651	0
			3	89.70	0.537	98.09	1.761	595.5289	595.5289	0	58.239	387	596.4489	0.15
			4	97.92	0.489	98.96	0.986	617.8838	617.8838	0	5.369	25	617.8838	0
			5	96.77	0.571	98.31	1.060	628.2804	628.2804	0	13.763	77	628.2804	0
		1	1	98.33	3.952	99.69	6.976	750.6251	750.6251	0	13.456	5	784.1952	4.47
			2	96.56	2.988	99.96	8.139	714.6453	714.6453	0	12.114	3	726.9382	1.72
			3	94.74	3.108	98.18	6.025	626.1553	626.1553	0	1614.536	2907	631.5932	0.87
			4	94.54	2.106	99.21	5.328	747.7008	747.7008	0	109.229	49	749.9016	0.29
			5	97.26	2.650	99.21	4.497	768.5189	768.5189	0	21.574	43	775.7170	0.94
R101	20	0.6	1	97.41	0.919	98.54	1.288	457.8598	457.8598	0	5.952	17	457.8598	0
			2	98.87	1.031	99.96	2.015	667.0131	667.0131	0	2.848	3	667.0131	0
			3	98.07	0.705	100.00	1.331	455.0534	455.0534	0	1.332	1	455.0534	0
			4	96.96	0.681	99.67	1.602	589.9082	589.9082	0	3.687	9	589.9082	0
			5	97.85	0.256	99.68	0.578	663.2159	663.2159	0	1.095	3	663.2159	0
		1	1	97.55	2.586	99.63	4.799	599.8380	599.8380	0	6.595	3	600.5648	0.12
			2	97.32	4.239	99.89	11.135	863.8829	863.8829	0	30.048	15	865.5480	0.19
			3	95.92	3.532	98.89	6.316	615.3454	617.9120	0.42		7182	617.9662	0.43
			4	96.55	2.314	99.66	6.236	712.0175	712.0175	0	24.305	51	712.7199	0.1
			5	94.74	2.452	98.43	5.141	794.4068	794.4068	0	4119.999	11703	797.2678	0.36
C101	40	0.6	1	93.42	51.170	98.71	201.460	838.4410	-	-		570	844.5669	0.73
			2	95.75	9.604	99.04	27.242	990.2575	-	-		1093	1005.8069	1.57
			3	96.52	29.481	99.73	743.449	879.2568	879.2568	0	1538.354	9	904.5190	2.87
			4	96.10	11.562	99.35	39.544	920.9192	921.0554	0.01		1364	924.0024	0.33
			5	96.00	13.405	99.23	59.715	868.7426	868.7426	0	2939.939	677	868.7426	0
		1	1	93.06	31.576	99.52	118.267	1300.3077	-	-		864	1330.0617	2.29
			2	94.15	15.233	99.20	60.052	1354.5182	-	-		1401	1357.7864	0.24
			3	93.59	29.739	99.57	131.467	1298.3122	-	-		341	1309.3540	0.85
			4	95.18	23.035	99.22	75.314	1231.9142	-	-		931	1238.8599	0.56
			5	91.42	32.488	99.45	135.468	1263.3306	-	-		971	1287.4121	1.91
R101	40	0.6	1	97.84	15.839	98.63	29.925	759.2221	761.7828	0.34		1821	765.8648	0.87
			2	97.57	7.898	98.32	16.601	895.9290	896.0147	0.01		3927	897.5237	0.18
			3	97.49	8.675	99.12	29.198	851.0276	851.0276	0	2469.400	867	870.6020	2.3
			4	96.91	11.112	98.67	24.438	970.6249	-	-		2464	973.9919	0.35
			5	95.72	9.743	99.23	41.537	854.3462	854.346200	0	2152.453	555	855.1124	0.09
		1	1	98.05	24.756	98.62	53.271	1232.6644	-	-		1359	1246.4958	1.12
			2	97.05	27.297	99.13	70.848	1234.0506	-	-		1384	1244.9154	0.88
			3	98.50	34.065	99.41	81.282	1056.1320	1056.132000	0	2411.205	241	1060.3880	0.4
			4	97.59	29.096	99.30	62.261	1242.0752	-	-		1625	1255.4495	1.08
			5	97.33	26.470	99.11	57.155	1096.0153	-	-		747	1101.1214	0.47

Table 2: Mid-size instances

Instance				root										
	n	p	id	$\underline{z}^*_{LMP}$	t_{LMP} (sec.)	$\underline{z}^*_{LMP+cuts}$	$t_{LMP+cuts}$ (sec.)	$\underline{z}^*$	$\overline{z}^*$	gap (%)	t (sec.)	nodes explored	best known value	gap (%)
C101	60	0.6	1	94.36	78.844	99.37	1936.216	1216.3381	-	-		140	1241.1029	2.04
			2	96.11	48.177	99.46	189.640	1330.1278	1331.7718	0.12		277	1342.5981	0.94
			3	96.78	328.642	99.51	808.366	1180.2855	-	-		123	1184.7776	0.38
			4	96.23	38.862	99.50	151.149	1281.4176	-	-		143	1303.1604	1.7
			5	94.64	76.209	99.09	279.632	1294.5609	-	-		291	1303.7277	0.71
		1	1	96.15	75.338	99.57	271.348	1966.8749	-	-		284	2003.0431	1.84
			2	95.81	135.069	99.51	669.144	1657.6147	-	-		174	1667.6461	0.61
			3	95.58	133.520	99.85	605.378	1782.6547	-	-		215	1816.1236	1.88
			4	96.44	84.016	99.43	286.563	1890.5147	-	-		257	1906.5617	0.85
			5	97.21	1557.079	100.00	-	1618.6416	-	-		0	1650.5799	1.97
R101	60	0.6	1	98.69	50.263	99.14	102.376	1279.2011	-	-		662	1293.2293	1.1
			2	98.19	24.074	99.18	68.894	1306.2143	-	-		1057	1326.5733	1.56
			3	98.45	37.274	98.91	65.243	1028.3194	1028.5158	0.02		624	1034.5583	0.61
			4	98.83	35.127	99.24	93.056	1215.9017	-	-		435	1235.5766	1.62
			5	98.16	31.550	99.04	88.417	1146.7673	-	-		372	1149.5550	0.24
		1	1	-	-	-	-	-	-	-		0	2086.6870	-
			2	98.52	3964.079	99.77	4416.453	1633.5296	-	-		96	1662.5753	1.78
			3	98.88	1723.508	99.65	1915.697	1544.3313	-	-		275	1581.5715	2.41
			4	99.21	6528.569	100.00	-	1694.8091	-	-		0	1732.8742	2.25
			5	98.67	3021.241	99.54	3483.092	1487.2549	-	-		96	1507.7615	1.38
C101	80	0.6	1	95.25	142.396	99.45	487.901	1629.5895	-	-		93	1648.0955	1.14
			2	96.17	258.235	99.86	717.439	1588.9384	-	-		26	1616.9601	1.76
			3	96.18	124.481	99.46	483.473	1726.6793	-	-		216	1750.6889	1.39
			4	96.17	259.836	99.67	1178.101	1430.9602	-	-		5	1468.4500	2.62
			5	94.67	298.103	99.51	922.496	1651.3305	-	-		75	1702.9460	3.13
		1	1	-	-	-	-	-	-	-		0	2277.9557	-
			2	100.00	-	-	-	1992.4920	-	-		0	2133.9879	7.1
			3	-	-	-	-	-	-	-		0	2623.9684	-
			4	-	-	-	-	-	-	-		0	2389.1831	-
			5	-	-	-	-	-	-	-		0	2410.9786	-
R101	80	0.6	1	98.72	80.866	99.58	187.489	1430.7070	-	-		371	1467.6251	2.58
			2	99.25	142.698	99.69	288.063	1459.2404	-	-		205	1482.3639	1.58
			3	99.25	85.498	99.56	175.094	1583.1556	-	-		468	1618.6780	2.24
			4	99.35	190.747	99.57	309.440	1400.0232	-	-		153	1432.0982	2.29
			5	-	-	-	-	-	-	-		0	1482.7288	-
		1	1	-	-	-	-	-	-	-		0	2137.3158	-
			2	-	-	-	-	-	-	-		0	1955.9811	-
			3	-	-	-	-	-	-	-		0	2302.7431	-
			4	-	-	-	-	-	-	-		0	2123.9847	-
			5	-	-	-	-	-	-	-		0	2155.4925	-

Table 3 reports a summary of the computational results on the mid-size instances. Instances are clustered on the basis of the values of n and p. Each row corresponds to 10 instances. Column '# opt.' reports the number of instances solved to optimality, '# LB' is the number of times a lower bound was computed, 'av. gap' and 'max gap' report, in percentage, the average and maximum gap, respectively, between the computed lower bound and the value of the best known solution (the best between the solution computed by the branch-price-and-cut algorithm and the one reported in Archetti et al. (2014)), and '# best known' is the number of times the branch-price-and-cut found or improved the best feasible solution. In parentheses we report the number of improved feasible solutions with respect to the best known reported in Archetti et al. (2014).

As expected, instances of larger size are more difficult to solve. Moreover, when $p = 1$ the difficulty of solving the problem increases. This is due to the fact that, when $p = 1$, all customers require all the commodities while a smaller number of commodities are required when $p = 0.6$. In the latter case, the expanded graph used for the solution of the pricing problem is smaller. Out of the 30 instances for which the algorithm finds a solution, this solution is better than the best known solution found in Archetti et al. (2014) on 20 cases. For all instances, except one, with up to 60 customers a lower bound is found and the maximum optimality gap is 2.41%.

Table 3: Summary of results on mid-size instances

				av.	max	# best
n	p	# opt.	# LB	gap	gap	known
20	0.6	10	10	0.00	0.00	10(1)
	1	9	10	0.04	0.42	10(10)
40	0.6	4	10	0.30	1.57	7(6)
	1	1	10	0.94	2.29	1(1)
60	0.6	0	10	0.95	2.04	2(2)
	1	0	9	1.66	2.41	0
80	0.6	0	9	2.08	3.13	0
	1	0	1	7.10	7.10	0

Finally, in Table 4 we report some information concerning the solution of the mid-size instances for which the branch-price-and-cut algorithm was able to compute a feasible solution. These are the 30 instances for which an upper bound ($\overline{z}^*$) is provided in Table 2. The first 4 columns have the same meaning as in Table 2. We then report the number of routes performed and the number of customers served in each route. When the number of customers visited in a route is underlined, the route contains at least one split customer. In the following column we report the number of split deliveries performed in each route that visits at least one split customer. Finally, the last two columns report the total number of customers that are served by two and three routes, respectively. In all except one instance at least one customer receives a split delivery while in only 4 instances we have customers which are served by 3 routes. Moreover, in 20 cases over 30 the number of routes containing at least one split customer is greater than or equal to half of the total number of routes. The number of split customers in a single route is in most cases 1 or 2 but is as large as 5 in one case.

Table 4: Mid-size solutions

Instance	n	p	id	# of routes	# of customers per route	# of split deliveries	2-split	3-split
C101	20	0.6	1	6	$(3, \underline{6}, \underline{4}, \underline{3}, \underline{3}, 3)$	(1, 1, 1, 1)	2	0
			2	7	$(2, \underline{3}, 2, \underline{3}, 5, 3, 3)$	(1, 1)	1	0
			3	7	$(3, \underline{4}, \underline{3}, \underline{3}, \underline{5}, 1, \underline{5})$	(2, 2, 2, 1, 1)	4	0
			4	8	$(2, \underline{2}, 2, 3, \underline{4}, \underline{5}, 1, 3)$	(1, 1, 2)	2	0
			5	8	$(1, 3, \underline{6}, 2, \underline{3}, 2, 2, 2)$	(1, 1)	1	0
		1	1	9	$(1, \underline{3}, \underline{3}, \underline{3}, \underline{3}, \underline{4}, 2, 2, 2)$	(1, 1, 1, 2, 1)	3	0
			2	9	$(1, \underline{3}, \underline{2}, \underline{5}, \underline{4}, \underline{3}, \underline{3}, \underline{3}, 2)$	(2, 1, 3, 2, 1, 1, 1)	4	1
			3	9	$(1, \underline{4}, 3, \underline{3}, \underline{3}, \underline{3}, 2, \underline{3}, \underline{2})$	(2, 1, 1, 2, 1, 1)	4	0
			4	10	$(2, \underline{3}, \underline{2}, \underline{4}, \underline{3}, \underline{3}, 2, 1, 2, 2)$	(2, 1, 1, 2, 2)	4	0
			5	10	$(2, \underline{3}, \underline{2}, \underline{4}, 2, 2, 1, 2, 2, 2)$	(1, 1, 2)	2	0
R101	20	0.6	1	6	$(\underline{3}, 2, \underline{3}, \underline{6}, \underline{5}, 4)$	(2, 1, 2, 1)	3	0
			2	7	$(1, \underline{5}, \underline{2}, 3, 4, 3, 3)$	(1, 1)	1	0
			3	7	$(1, 4, 4, 3, 2, 4, 2)$		0	0
			4	8	$(2, \underline{3}, \underline{5}, 4, \underline{4}, 1, \underline{2}, 2)$	(2, 2, 1, 1)	3	0
			5	8	$(2, 3, \underline{4}, \underline{3}, 2, 2, 4, 1)$	(1, 1)	1	0
		1	1	9	$(2, \underline{2}, \underline{4}, \underline{3}, 1, \underline{3}, \underline{3}, \underline{3}, \underline{3})$	(1, 1, 1, 1, 1, 2, 1)	4	0
			2	9	$(1, \underline{4}, \underline{4}, \underline{5}, \underline{3}, 2, \underline{2}, \underline{3}, \underline{2})$	(3, 1, 4, 1, 1, 1, 1)	5	0
			3	9	$(2, \underline{3}, \underline{6}, 2, 2, \underline{3}, \underline{3}, \underline{3}, \underline{5})$	(1, 5, 2, 2, 2, 4)	5	2
			4	10	$(\underline{2}, \underline{3}, \underline{4}, \underline{2}, 1, \underline{3}, \underline{3}, \underline{3}, 2, \underline{3})$	(1, 2, 2, 1, 2, 1, 1, 2)	6	0
			5	10	$(1, \underline{3}, 3, 2, \underline{2}, 1, 3, \underline{3}, \underline{3}, 2)$	(2, 1, 1, 2)	3	0
C101	40	0.6	3	11	$(3, 2, \underline{6}, 3, 3, \underline{5}, 8, 2, 3, 3, 3)$	(1, 1)	1	0
			4	12	$(2, \underline{3}, \underline{5}, \underline{4}, \underline{3}, \underline{5}, \underline{6}, 5, 4, \underline{4}, 3, 2)$	(1, 2, 1, 2, 2, 2, 2)	6	0
			5	11	$(2, \underline{5}, 6, \underline{6}, \underline{4}, \underline{4}, 3, \underline{5}, \underline{3}, \underline{3}, \underline{3})$	(1, 1, 1, 1, 1, 1, 1, 1)	4	0
R101	40	0.6	1	9	$(\underline{6}, \underline{4}, 7, 4, \underline{7}, 4, \underline{4}, 3, \underline{5})$	(1, 2, 3, 1, 1)	4	0
			2	12	$(2, 3, \underline{4}, 5, \underline{4}, \underline{4}, \underline{4}, \underline{3}, \underline{6}, 4, 2, \underline{3})$	(1, 1, 1, 1, 1, 1, 2)	4	0
			3	11	$(3, \underline{4}, \underline{7}, \underline{6}, 3, \underline{4}, \underline{4}, \underline{5}, \underline{2}, 2, 5)$	(1, 1, 3, 1, 1, 2, 1)	5	0
			5	11	$(\underline{3}, \underline{5}, \underline{6}, \underline{5}, 6, 4, 1, 3, \underline{4}, 3, \underline{7})$	(1, 3, 3, 2, 3, 1)	4	1
		1	3	16	$(2, \underline{4}, \underline{2}, \underline{4}, \underline{4}, \underline{3}, \underline{4}, \underline{3}, 3, \underline{3}, 2, 3, \underline{3}, 2, \underline{3}, \underline{3})$	(3, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1)	6	1
C101	60	0.6	2	17	$(2, \underline{4}, \underline{5}, \underline{3}, \underline{3}, \underline{5}, \underline{3}, \underline{3}, \underline{5}, \underline{4}, \underline{9}, 3, \underline{4}, \underline{6}, \underline{3}, \underline{3}, 2)$	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)	7	0
R101	60	0.6	3	14	$(2, 3, \underline{6}, \underline{6}, 7, \underline{3}, \underline{3}, \underline{5}, 4, 5, 4, 7, 3, \underline{5})$	(1, 1, 1, 1, 1, 1)	3	0

6 Conclusions

We presented in this paper a branch-price-and-cut algorithm for the solution of a routing problem that explicitly considers the presence of multiple commodities in a distribution problem and allows multiple visits to a customer as long as a commodity is not split between vehicles. The algorithm solves instances with up to 40 customers and 3 commodities, more than doubling the size 15 of the instances previously solved to optimality. In particular, if we consider the 60 medium size instances with at most 60 customers, the algorithm is able to compute or improve the best known upper bound values in 50% of cases (30/60), and only in 5/59 cases the lower bound computed is far from the best known upper bound by more than 2%. The algorithm was not able to compute the lower bound for one instance, but for the remaining 59/60 instances the average and maximum optimality gaps are 0.63% and 2.41%, respectively. Also, for the instances with at most 60 customers we improved 20 of the best known solutions.

Future research should be devoted to the development of a heuristic for the solution of the C-SDVRP. In fact, the problem was solved heuristically by Archetti et al. (2014) by transforming it into a VRP while no ad-hoc heuristic has been proposed. Several variants of the C-SDVRP that consider multiple commodities would also deserve to be studied.

References

C. Archetti, N. Bianchessi, and M. G. Speranza. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58:241–254, 2011.

C. Archetti, A. M. Campbell, and M.G. Speranza. Multi-commodity vs. single-commodity routing. *Transportation Science*, 2014. doi: 10.1287/trsc.2014.0528.

R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59:1269–1283, 2011.

C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

U. Derigs, J. Gottlieb, J. Kalkoff, M. Piesche, F. Rothlauf, and U. Vogel. Vehicle routing with compartments: applications, modelling and heuristics. *OR Spectrum*, 33:885–914, 2011.

G. Desaulniers, J. Desrosiers, and M. Solomon, editors. *Column generation.* Springer, 2005.

M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994.

A.M. Frieze. Shortest path algorithms for knapsack type problems. *Mathematical Programming*, 11:150–157, 1976.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, pages 33–65. Springer, 2005.

T.K. Ralphs, L. Kopman, W.R. Pulleyblank, and L.E. Trotter. On the capacitated vehicle routing problem. *Mathematical Programmming*, 94:343–359, 2003.

G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51:155–209, 2008.

D.M. Ryan and B.A. Foster. On the capacitated vehicle routing problem. *Mathematical Programmming*, 94:343–359, 2003.

P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods and Applications.* Series on Optimization. MOS-SIAM, Philadephia, Second edition, 2014.