CADERNOS DO LOGIS

Volume 2019, Number 3

On the exact solution of vehicle routing problems with backhauls

Eduardo Queiroga, Yuri Frota, Ruslan Sadykov, Anand Subramanian, Eduardo Uchoa, Thibaut Vidal

August, 2019

On the exact solution of vehicle routing problems with backhauls

Eduardo Queiroga[a,*], Yuri Frota[a], Ruslan Sadykov[b], Anand Subramanian[c], Eduardo Uchoa[d], Thibaut Vidal[e]

[a] *Instituto de Computação, Universidade Federal Fluminense, Av. Gal. Milton Tavares de Souza, s/n, São Domingos, 24210-346, Niterói, Brazil*
[b] *INRIA Bordeaux, Sud-Ouest, 200 Avenue de la Veille Tour, 33405 Talence, France*
[c] *Universidade Federal da Paraíba, Departamento de Sistemas de Computação, Centro de Informática, Rua dos Escoteiros s/n, Mangabeira, 58055-000, João Pessoa, Brazil*
[d] *Departamento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria 156, Niterói, Brazil*
[e] *Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rua Marquês de São Vicente, 225 - Gávea, 22451-900, Rio de Janeiro, Brazil*

Abstract

In this paper, we are interested in the exact solution of the vehicle routing problem with backhauls (VRPB), a classical vehicle routing variant with two types of customers: linehaul (delivery) and backhaul (pickup) ones. We propose two branch-cut-and-price (BCP) algorithms for the VRPB. The first of them follows the traditional approach with one pricing subproblem, whereas the second one exploits the linehaul/backhaul customer partitioning and defines two pricing subproblems. The methods incorporate elements of state-of-the-art BCP algorithms, such as rounded capacity cuts, limited-memory rank-1 cuts, strong branching, route enumeration, arc elimination using reduced costs and dual stabilization. Computational experiments show that the proposed algorithms are capable of obtaining optimal solutions for all existing benchmark instances with up to 200 customers, many of them for the first time. It is observed that the approach involving two pricing subproblems is more efficient computationally than the traditional one. Moreover, new instances are also proposed for which we provide tight bounds. Also, we provide results for benchmark instances of the heterogeneous fixed fleet VRPB and the VRPB with time windows.

Keywords: Routing, Backhauls, Branch-cut-and-price, Integer programming

1. Introduction

In the classical *capacitated vehicle routing problem* (CVRP), a homogeneous fleet of vehicles is considered to build a set of least-cost routes such that: (i) all customers are visited once by exactly one route, (ii) the capacity of the vehicles is respected, and (iii) each route starts and ends

*Corresponding author
Email addresses: eduardoqueiroga@id.uff.br, eduardovqueiroga@gmail.com (Eduardo Queiroga), yuri@ic.uff.br (Yuri Frota), ruslan.sadykov@inria.fr (Ruslan Sadykov), anand@ci.ufpb.br (Anand Subramanian), uchoa@producao.uff.br (Eduardo Uchoa), vidalt@inf.puc-rio.br (Thibaut Vidal)

at the depot. Although some applications in distribution can be modeled as a VRP, there are many applications with their own particularities such as those where customers require different types of services. This paper approaches the *VRP with backhauls* (VRPB) (Deif & Bodin, 1984), a well-known variant which considers two types of customers: *linehaul* and *backhaul*.

Linehaul customers have a delivery demand which is loaded at the depot and the backhaul customers have a pickup demand that is transported to the depot. In the VRPB, a route must visit linehaul customers before backhaul customers. At least one linehaul customer must be visited before possible backhaul customers, but a route may only be composed of linehauls. This kind of route is desirable to avoid en-route load rearrangements. For example, in beverage distribution, the collection of empty bottles should usually be performed after delivering full ones. Jacobs-Blecha & Goetschalckx (1992) discussed how the grocery industry could save millions of dollars by exploiting backhauls. As in the CVRP, the objective is to minimize the total travel cost.

Koç & Laporte (2018) presented a recent literature review of the VRPB, including variants such as the *mixed VRPB* (MVRPB), *VRPB with time windows* (VRPBTW) and the *heterogeneous fixed fleet VRPB* (HFFVRPB). Many studies have proposed (meta)heuristics for the VRPB. Constructive procedures were suggested in Deif & Bodin (1984); Goetschalckx & Jacobs-Blecha (1989); Jacobs-Blecha & Goetschalckx (1992); Toth & Vigo (1996), whereas metaheuristic approaches were developed in Osman & Wassan (2002); Wassan (2007); Brandão (2006); Gajpal & Abad (2009); Zachariadis & Kiranoudis (2012); Cuervo et al. (2014); Brandão (2016). Ropke & Pisinger (2006) and Vidal et al. (2014) proposed unified metaheuristics capable of solving a variety of problems including among others the VRPB and VRPBTW.

On the other hand, there are relatively few exact methods for the VRPB. Yano et al. (1987) introduced an exact algorithm for a particular case of the problem in which there should be at most four customers in a route. Goetschalckx & Jacobs-Blecha (1989) proposed an integer linear programming (ILP) formulation which extends the model by Fisher & Jaikumar (1981) for the CVRP. Toth & Vigo (1997) proposed an ILP for the VRPB which is similar to the two index vehicle flow formulation for the asymmetric VRP by Laporte et al. (1986). The authors also devised a Lagrangian relaxation scheme which is strengthened by cutting planes. This relaxation is combined with another one obtained by disregarding the capacity constraints of the model, producing an overall dual bounding procedure. Such procedure is used on a branch-and-bound algorithm to solve the VRPB to optimality. Mingozzi et al. (1999) proposed a set partitioning (SP) formulation that makes use of variables for elementary paths over two subgraphs induced by the linehaul and backhaul customers, respectively. Two heuristics were combined to solve the dual problem and, through the resulting bound, they reduced the number of paths (variables) of the model without loss of optimality. Since the number of routes remained very large, an additional reduction was applied so that the resulting ILP could be solved using a MIP solver. Recently, an alternative mixed ILP for the VRPB was put forward by Granada-Echeverri et al. (2019).

Koç & Laporte (2018) pointed out the following future research perspective:

> *"The standard VRPB instances of Goetschalckx & Jacobs-Blecha (1989) and Toth & Vigo (1997) have been effectively solved by heuristics. However, it is our belief that further studies should focus on developing effective and powerful exact methods, such as branch-and-cut-and-price, to solve all available standard VRPB instances to optimality (see Poggi & Uchoa, 2014)."*

In view of this, the present work proposes two branch-cut-and-price (BCP) approaches for the VRPB. The algorithms incorporate elements of state-of-the-art BCP algorithms, such as rounded capacity cuts, limited-memory rank-1 cuts, strong branching, route enumeration, arc elimination using reduced costs and dual stabilization. The exact methods are found capable of solving all instances from the literature to optimality, many of them for the first time. As a result, we decided to generate a novel and more challenging benchmark dataset with instances involving up to 1000 customers. Furthermore, we also report results for the VRPBTW and HFFVRPB thanks to a simple extension of one of our algorithms.

The remainder of this paper is organized as follows. Section 2 formally defines the problems considered in this work. Section 3 presents the set partitioning formulations used. Section 4 presents the proposed BCP algorithms. Section 5 discusses the results of our extensive computational experiments on different benchmark instances. Finally, Section 6 concludes.

2. Problem definitions

In this section, the VRPB variants approached in this paper are formally defined.

2.1. VRPB

Let $G = (V, A)$ be a directed graph and $V = \{0\} \cup L \cup B$, where vertex 0 represents the depot, while $L = \{1, 2, \ldots, n\}$ and $B = \{n+1, n+2, \ldots, n+m\}$ are the set of linehaul and backhaul vertices, respectively. Moreover, define $L_0 = L \cup \{0\}$ and $B_0 = B \cup \{0\}$, thus $A = A_L \cup A_{LB} \cup A_B$, such that:

- $A_L = \{(i, j) : i \in L_0, j \in L, i \neq j\}$,
- $A_{LB} = \{(i, j) : i \in L, j \in B_0\}$,
- $A_B = \{(i, j) : i \in B, j \in B_0, i \neq j\}$.

Graph G is not complete, since there are no arcs from B to L and no arcs from 0 to B. For each arc $a \in A$ there is a nonnegative traveling cost c_a. Let $V^+ = V \setminus \{0\}$ be the set of customers. Each vertex $j \in V^+$ has a nonnegative d_j demand delivery (when $j \in L$) or pickup (when $j \in B$). Given a homogeneous fleet of K vehicles with capacity Q, the VRPB aims at finding K routes (elementary cycles in G passing by the depot) that minimize the total travel cost and satisfy the following constraints:

a) Each vertex $j \in V^+$ must be visited by exactly one route.

b) A route has to visit linehaul customers before backhaul customers, i.e., after visiting a backhaul customer it is forbidden to visit a linehaul customer (implicit in the definition of G).

c) A route may only be composed by linehaul customers, but it cannot only be composed by backhaul customers (also implicit in the definition of G).

d) The sum of the delivery demands does not exceed the vehicle capacity.

e) The sum of the pickup demands does not exceed the vehicle capacity.

2.2. VRPBTW

The VRPBTW generalizes the VRPB by considering a time window $[a_i, b_i]$ and a service time s_i for each customer $i \in V^+$. In the VRPBTW, the travel cost c_a of an arc a is interpreted as the travel time. A service can start to be performed from a_i until b_i, thus vehicles that arrive early must wait. Unlike in the VRPB, previous VRPBTW studies allowed routes containing only backhaul customers. Moreover, the number of vehicles is not specified *a priori*. The primary objective is to minimize the number of vehicles, whereas the secondary objective is to minimize the total travel time.

2.3. HFFVRPB

The HFFVRPB extends the VRPB by considering a finite set of vehicle types T, where each type $k \in T$ has u^k available vehicles with capacity Q^k and cost c_a^k, $\forall a \in A$. The composition of the heterogeneous fleet must respect the availability of each type of vehicle, but without necessarily using all vehicle types. The objective is to minimize the total travel cost.

3. Set partitioning formulations

Before introducing the SP-based formulations, we first present formulation $\mathcal{F}0$ by Toth & Vigo (1997), in Equations (1)–(7). Each variable x_a indicates whether an arc $a \in A$ is traversed by some vehicle. Given a subset S of L or B, let $r(S) = \left\lceil \sum_{i \in S} d_i / Q \right\rceil$ be a lower bound on the minimum number of vehicles necessary to serve all customers in S. Also, let $\delta^-(S) = \{(i,j) \in A : i \in V \setminus S, j \in S\}$ and $\delta^+(S) = \{(i,j) \in A : i \in S, j \in V \setminus S\}$. For simplicity, let $\delta^-(\{i\}) = \delta^-(i)$ and $\delta^+(\{i\}) = \delta^+(i)$, $\forall i \in V$.

$$(\mathcal{F}0) \text{ Min } \sum_{a \in A} c_a x_a \tag{1}$$

$$\text{s.t. } \sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in V^+, \tag{2}$$

$$\sum_{a\in\delta^+(i)} x_a = 1 \qquad \forall i \in V^+, \tag{3}$$

$$\sum_{a\in\delta^+(0)} x_a = K, \tag{4}$$

$$\sum_{a\in\delta^-(S)} x_a \geq r(S) \qquad \forall S \subseteq L, \tag{5}$$

$$\sum_{a\in\delta^-(S)} x_a \geq r(S) \qquad \forall S \subseteq B, \tag{6}$$

$$x_a \in \{0,1\} \qquad a \in A \tag{7}$$

Constraints (2)–(3) ensure that each customer is visited exactly once, while constraint (4) imposes that K vehicles must leave the depot. Constraints (5)–(6) are the *rounded capacity constraints* (RCC) and also guarantee the subtour elimination. They are separated on demand in a cutting plane fashion. Constraints (7) define the domain of the variables.

In what follows, we describe two SP formulations for the VRPB by extending $\mathcal{F}0$. Both formulations are compared in terms of linear relaxation and effectiveness of the application of rank-1 cuts.

3.1. Formulation $\mathcal{F}1$

Let Ω be the set of all q-routes in G, which are walks (paths that may be not elementary) starting and ending at the depot and that do not violate the capacity constraints for both linehaul and backhaul customers. A customer $i \in V^+$ visited k times consumes $k \times d_i$ load units. Let h_a^p be the number of times a path $p \in \Omega$ traverses the arc $a \in A$ and λ_p a binary variable that indicates that p is used. $\mathcal{F}0$ can be extended by adding variables λ and constraints (8)–(9):

$$x_a = \sum_{p\in\Omega} h_a^p \lambda_p \qquad \forall a \in A, \tag{8}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in \Omega \tag{9}$$

Formulation $\mathcal{F}1$ is then given by (1)–(9). By eliminating the x variables using (8) and relaxing the integrality constraints, one obtains the linear relaxation of $\mathcal{F}1$ as follows:

$$\text{Min} \sum_{p\in\Omega} \left(\sum_{a\in A} c_a h_a^p \right) \lambda_p \tag{10}$$

$$\text{s.t.} \sum_{a\in\delta^-(i)} \sum_{p\in\Omega} h_a^p \lambda_p = 1 \qquad \forall i \in V^+, \tag{11}$$

$$\sum_{a\in\delta^+(0)} \sum_{p\in\Omega} h_a^p \lambda_p = K, \tag{12}$$

$$\sum_{a\in\delta^-(S)} \sum_{p\in\Omega} h_a^p \lambda_p \geq r(S) \qquad \forall S \subseteq L, \tag{13}$$

$$\sum_{a\in\delta^-(S)}\sum_{p\in\Omega} h_a^p\lambda_p \geq r(S) \qquad \forall S\subseteq B, \tag{14}$$

$$\lambda_p \geq 0 \qquad \forall p\in\Omega \tag{15}$$

Constraints (13)–(14) are not necessary for correctness because any integer solution satisfying (11)-(12) corresponds to K feasible elementary routes. Nevertheless, they can cut fractional solutions and are important to strengthen the formulation. Such constraints are added on demand in a cutting plane fashion. On the other hand, the constraints that would be obtained from (3) are now completely redundant and can be dropped. In this kind of SP-based formulation, it is common to use relaxations such as q-routes instead of elementary routes, because the pricing subproblem becomes weakly $\mathcal{NP}$-hard and thus more computationally tractable (Poggi & Uchoa, 2014). The disadvantage, on the other hand, is that this worsens the linear relaxation.

3.2. Formulation $\mathcal{F}2$

In the SP-based formulation by Mingozzi et al. (1999), there are variables associated to paths with only linehaul and backhaul customers. There are additional binary variables, one for each arc in A_{LB}, used in constraints that ensure that linehaul and backhaul paths should be connected to form a complete feasible route. We now describe a new formulation $\mathcal{F}2$ which follows a similar principle but does not use additional variables.

Let $G_L = (L_0, A_L)$ and $G_B = (L\cup B_0, A_{LB}\cup A_B)$ be subgraphs of G and let Ω_L and Ω_B be the set of q-paths over G_L and G_B, respectively. For G_L, the q-paths are walks that start at the depot and end at some customer in L, not violating the linehaul capacity constraint. For G_B, the q-paths are walks that start at a linehaul customer and end at the depot, not violating the backhaul capacity constraint. The q-paths in Ω_B contain exactly one linehaul customer, which will be interpreted as *connecting vertices*. Given $i\in L$, the subset $\Omega_L^i\subseteq\Omega_L$ is composed by paths ending at i and $\Omega_B^i\subseteq\Omega_B$ by paths starting at i. A binary variable λ_p^L (λ_p^B) defines the use of a q-path $p\in\Omega_L$ ($p\in\Omega_B$). The constant h_a^p indicates how may times arc a appears in q-path p (it is necessarily zero when a and p are associated with distinct graphs). Formulation $\mathcal{F}0$ can be extended by including variables λ^L and λ^B, as well as constraints (16)–(19). Constraints (17), in particular, ensures that the chosen paths are properly connected.

$$x_a = \sum_{p\in\Omega_L} h_a^p\lambda_p^L + \sum_{p\in\Omega_B} h_a^p\lambda_p^B \qquad \forall a\in A, \tag{16}$$

$$\sum_{p\in\Omega_L^i}\lambda_p^L = \sum_{p\in\Omega_B^i}\lambda_p^B \qquad \forall i\in L, \tag{17}$$

$$\lambda_p^L\in\{0,1\} \qquad \forall p\in\Omega_L, \tag{18}$$

$$\lambda_p^B\in\{0,1\} \qquad \forall p\in\Omega_B \tag{19}$$

Hence, $\mathcal{F}2$ is defined by (1)–(7) and (16)–(19). By eliminating the x variables using (16),

relaxing the integrality constraints and performing some simplifications, it is possible to write the linear relaxation of $\mathcal{F}2$ as follows:

$$\text{Min} \sum_{p\in\Omega_L}\left(\sum_{a\in A} c_a h_a^p\right)\lambda_p^L + \sum_{p\in\Omega_B}\left(\sum_{a\in A} c_a h_a^p\right)\lambda_p^B \tag{20}$$

$$\text{s.t.} \sum_{a\in\delta^-(i)}\left(\sum_{p\in\Omega_L} h_a^p\lambda_p^L + \sum_{p\in\Omega_B} h_a^p\lambda_p^B\right) = 1 \qquad \forall i \in V^+, \tag{21}$$

$$\sum_{a\in\delta^+(0)}\sum_{p\in\Omega_L} h_a^p\lambda_p^L = K, \tag{22}$$

$$\sum_{p\in\Omega_L^i}\lambda_p^L = \sum_{p\in\Omega_B^i}\lambda_p^B \qquad \forall i \in L, \tag{23}$$

$$\sum_{a\in\delta^-(S)}\sum_{p\in\Omega_L} h_a^p\lambda_p^L \geq r(S) \qquad \forall S \subseteq L, \tag{24}$$

$$\sum_{a\in\delta^-(S)}\sum_{p\in\Omega_B} h_a^p\lambda_p^B \geq r(S) \qquad \forall S \subseteq B, \tag{25}$$

$$\lambda_p^L \geq 0 \qquad \forall p \in \Omega_L, \tag{26}$$

$$\lambda_p^B \geq 0 \qquad \forall p \in \Omega_B \tag{27}$$

As in formulation $\mathcal{F}1$, constraints (24)–(25) should be added on demand as cutting planes, and the constraints that would be derived from (3) becomes redundant and can be dropped.

3.3. Strengthening the formulations

3.3.1. ng-routes

Strengthening the route relaxation without significantly affecting the complexity of the pricing subproblem is a challenging task. One of the most successful route relaxation schemes is the so-called *ng*-routes and *ng*-paths, introduced by Baldacci et al. (2011) as an alternative to *q*-routes. For each customer $i \in V^+$, let $N_i \subseteq V^+$ be the neighborhood of $i \in V^+$ (a.k.a. *ng*-set), where N_i is typically composed by the closest customers to i. In a *ng*-route (or *ng*-path), a customer i can be revisited only after visiting a customer j such that $i \notin N_j$.

The size of the *ng*-sets controls the level of elementarity obtained, since larger sets allows fewer non-elementary routes. In one extreme, if *ng*-sets are empty, *ng*-routes are *q*-routes. On the other extreme, if all *ng*-sets are equal to V^+, then *ng*-routes are elementary. In practice, *ng*-sets of size around 8-10 provide a good trade-off between formulation strength and complexity of the column generation.

In the VRPB, it only makes sense to define *ng*-sets with customers of the same type: if $i \in L$ then $N_i \subseteq L$, while if $i \in B$ then $N_i \subseteq B$. Formulation $\mathcal{F}1$ can be strengthened by restricting Ω to *ng*-routes. Similarly, $\mathcal{F}2$ can be strengthened by restricting Ω_L and Ω_B to *ng*-paths.

3.3.2. Rank-1 cuts

By applying the Chvátal-Gomory rounding over the sum of inequalities (11) multiplied by $\rho \in \mathbb{R}_{\geq 0}^{|V^+|}$, we can to obtain the rank-1 cut (28), which is valid for $\mathcal{F}1$.

$$\sum_{p \in \Omega} \left\lfloor \sum_{i \in V^+} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in V^+} \rho_i \right\rfloor \tag{28}$$

Analogously, the rank-1 cut (29), which is valid for $\mathcal{F}2$, can be derived from (21).

$$\sum_{p \in \Omega_L} \left\lfloor \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p^L + \sum_{p \in \Omega_B} \left\lfloor \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p^B \leq \left\lfloor \sum_{i \in V^+} \rho_i \right\rfloor \tag{29}$$

Rank-1 cuts are a generalization of the Subset Row Cuts (Jepsen et al., 2008) and are known to be very strong, but separating them makes the pricing subproblems significantly more difficult. Hence, we use the limited memory technique proposed by Pecin et al. (2017a) for mitigating the negative impact in the pricing.

3.4. Comparing $\mathcal{F}1$ and $\mathcal{F}2$

In this subsection, we assume that $\mathcal{F}1$ and $\mathcal{F}2$ use *ng*-routes and *ng*-paths defined over the same *ng*-sets.

Proposition 1. *The linear relaxations of $\mathcal{F}1$ and $\mathcal{F}2$ are equally strong.*

Proof. Let P_1 and P_2 be the polyhedra defined by the linear relaxations of $\mathcal{F}1$ and $\mathcal{F}2$, respectively. We show that for any solution of P_1 there is a solution of P_2 with the same objective value, and vice versa.

Given a solution $\bar{\lambda} \in P_1$, the function described in Algorithm 1 returns a solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ in $\mathcal{F}2$ space. It is clear from lines 7–8 that constraints (17) are satisfied by that solution. It can be verified through inequalities (8) and (16) that both $\bar{\lambda}$ and $\mathtt{P}_2(\bar{\lambda})$ induce the same values for the arc variables x. This is true because an arc $a \in A$ can be part of paths either from Ω_L or Ω_B, but never from both sets. As $\bar{\lambda} \in P_1$, then x solution satisfies (2)–(6). So, $\mathtt{P}_2(\bar{\lambda})$ should satisfy the corresponding constraints (21)–(24) and belongs to P_2. Moreover, $\bar{\lambda}$ and $\mathtt{P}_2(\bar{\lambda})$ have the same cost.

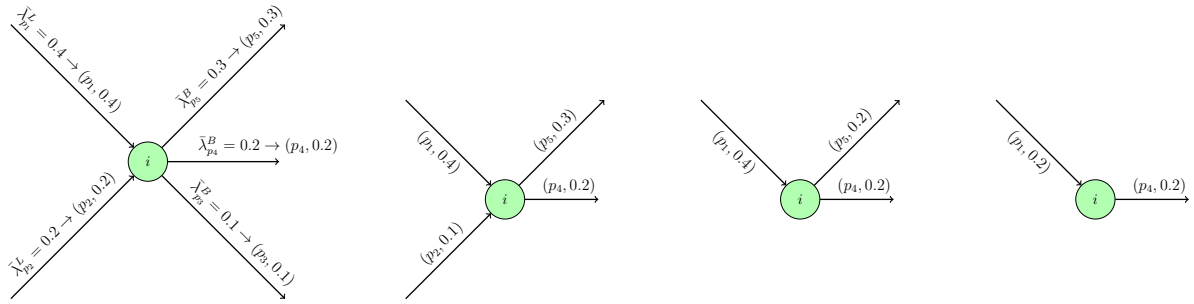
Let $(\bar{\lambda}^L, \bar{\lambda}^B)$ be a solution in P_2. The function described in Algorithm 2 returns a solution $\mathtt{P}_1(\bar{\lambda}^L, \bar{\lambda}^B) = \bar{\lambda}$ in $\mathcal{F}1$ space (Figure 1 illustrates how the algorithm works for a certain connecting vertex i). Note that lines 9 and 12 (that assume the existence of a suitable path p^2 to complete path p^1) are only correct because constraints (17) are satisfied by $(\bar{\lambda}^L, \bar{\lambda}^B)$. Again, it can be verified through inequalities (8) and (16) that both solutions $(\bar{\lambda}^L, \bar{\lambda}^B)$ and $\mathtt{P}_1(\bar{\lambda}^L, \bar{\lambda}^B)$ yield the same values for the arc variables x, so the latter solution belongs to P_1 and they have the same cost. □

Algorithm 1: Obtains the solution $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$ corresponding to $\bar{\lambda} \in P_1$

Function $\texttt{P}_2(\bar{\lambda})$
  Let $\gamma = \{(p, \bar{\lambda}_p) : p \in \Omega,\ \bar{\lambda}_p > 0\}$ be the set that maps the routes to their values
  Let $L(p) \in \Omega_L$ and $B(p) \in \Omega_B$ be the paths obtained by splitting route $p \in \Omega$ in its connecting vertex (the last linehaul customer)
  Let $(\bar{\lambda}^L, \bar{\lambda}^B)$ be the solution to be built for P_2, such that $\bar{\lambda}^L_p$ is initially zero $\forall p \in \Omega_L$ and $\bar{\lambda}^B_p$ is initially zero $\forall p \in \Omega_B$
  while $\gamma \neq \emptyset$ **do**
    Let (p, ζ) be a pair in γ
    $\bar{\lambda}^L_{L(p)} = \bar{\lambda}^L_{L(p)} + \zeta$
    $\bar{\lambda}^B_{B(p)} = \bar{\lambda}^B_{B(p)} + \zeta$
    $\gamma = \gamma \setminus \{(p, \zeta)\}$ `// Remove` p
  return $(\bar{\lambda}^L, \bar{\lambda}^B)$

Algorithm 2: Obtains the solution $\bar{\lambda} \in P_1$ corresponding to $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$

Function $\texttt{P}_1(\bar{\lambda}^L, \bar{\lambda}^B)$
  Let $\gamma^i = \{(p, \bar{\lambda}^L) : p \in \Omega^i_L,\ \bar{\lambda}^L > 0\} \cup \{(p, \bar{\lambda}^B) : p \in \Omega^i_B,\ \bar{\lambda}^B > 0\}$, $i \in L$, be the sets that maps the paths related to each connecting vertex i to their values
  Let $p_l \oplus p_b$ be the route in Ω obtained by concatenating the paths $p_l \in \Omega_L$ and $p_b \in \Omega_B$
  Let $\bar{\lambda}$ be the solution to be built for P_1, such that $\bar{\lambda}_p$ is initially zero $\forall p \in \Omega$
  for $i \in L$ **do**
    while $\gamma^i \neq \emptyset$ **do**
      Let (p^1, ζ^1) be a pair in γ^i whose ζ^1 is minimum
      if $p^1 \in \Omega^i_L$ **then**
        Let (p^2, ζ^2) be any pair in γ^i such that $p^2 \in \Omega^i_B$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^1 \oplus p^2$
      else `//` $p^1 \in \Omega^i_B$
        Let (p^2, ζ^2) be any pair in γ^i such that $p^2 \in \Omega^i_L$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^2 \oplus p^1$
      $\gamma^i = \gamma^i \setminus \{(p^1, \zeta^1), (p^2, \zeta^2)\}$ `// Remove` p^1 `and` p^2
      if $\zeta^2 - \zeta^1 > 0$ **then**
        $\gamma^i = \gamma^i \cup \{(p^2, \zeta^2 - \zeta^1)\}$ `// Reinsert` p^2 `with updated value`
  return $\bar{\lambda}$



(a) $\bar{\lambda}_{p_2 \oplus p_3} = 0.1$ (b) $\bar{\lambda}_{p_2 \oplus p_5} = 0.1$ (c) $\bar{\lambda}_{p_1 \oplus p_5} = 0.2$ (d) $\bar{\lambda}_{p_1 \oplus p_4} = 0.2$

Figure 1: Illustration of Algorithm 2. Obtaining the values for $\bar{\lambda}_p$, such that $p \in \Omega$ and the connecting vertex of p is $i \in L$. In Figure 1a, paths p_3 and p_2 are chosen according to the lines 7 and 12, respectively. Next, the value for $\lambda_{p_2 \oplus p_3} = 0.1$ is defined, the pair $(p_3, 0.1)$ is removed from γ^i and $(p_2, 0.2)$ is updated to $(p_2, 0.1)$. Figures 1b, 1c and 1d illustrate the continuation of the algorithm, until γ^i is empty. The algorithm performs this process for every vertex $i \in L$ as connecting vertex.

The functions defined in Algorithm 1 and Algorithm 2 define a one-to-one correspondence between solutions in P_1 and P_2. In fact, for all $\bar{\lambda} \in P_1$, $\mathtt{P}_1(\mathtt{P}_2(\bar{\lambda})) = \bar{\lambda}$; for all $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$, $\mathtt{P}_2(\mathtt{P}_1((\bar{\lambda}^L, \bar{\lambda}^B))) = (\bar{\lambda}^L, \bar{\lambda}^B)$. That correspondence will also be used in the proof of the following result.

Proposition 2. *Rank-1 cuts* (28) *are at least as strong as* (29) *and may be strictly stronger.*

Proof. Consider the rank-1 cuts (28) and (29) corresponding to the same vector of multipliers ρ. Consider a path $p \in \Omega$ and its split paths $L(p) \in \Omega_L$ and $B(p) \in \Omega_B$. It is always true that:

$$\sum_{i \in V^+} \sum_{a \in \delta^-(i)} \rho_i h^p_a = \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h^{L(p)}_a + \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h^{B(p)}_a. \tag{30}$$

If the condition

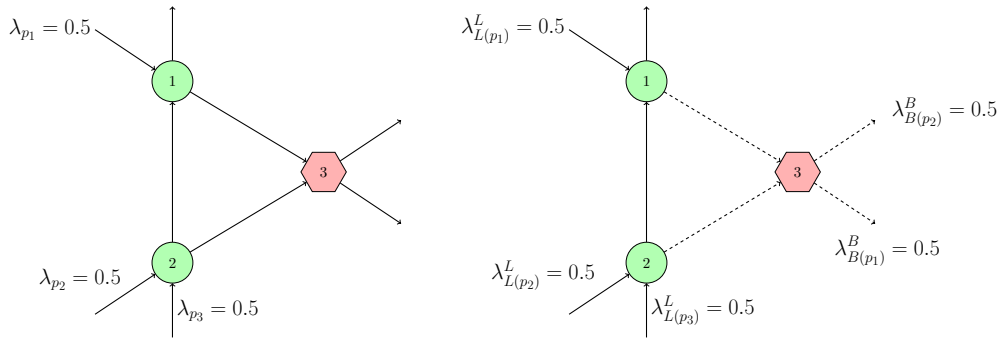
$$\left\lfloor \sum_{i \in V^+} \sum_{a \in \delta^-(i)} \rho_i h^p_a \right\rfloor = \left\lfloor \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h^{L(p)}_a \right\rfloor + \left\lfloor \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h^{B(p)}_a \right\rfloor \tag{31}$$

is true for all $p \in \Omega$, then rank-1 cuts (28) and (29) are equally strong, in the sense that a solution $\bar{\lambda} \in P_1$ is cut by (28) if and only if the corresponding solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ is cut by (29). Otherwise, if for some $p \in \Omega$ the left-hand-side of (31) is strictly larger than its right-hand-side, then (28) is strictly stronger than (29).

Let $C = \{i \in V^+ : \rho_i > 0\}$. If $C \subseteq L$, the second term in the right-hand-side of (30) is zero. So (31) is true and (28) and (29) are equally strong. The coefficient of λ_p in (28) will be identical to the coefficient of $\lambda^L_{L(p)}$ in (29), while the coefficient of $\lambda^B_{B(p)}$ will be zero. A similar reasoning

shows that when $C \subseteq B$, rank-1 cuts (28) and (29) are also equally strong.

On the other hand, when C has customers of both types, (31) may not be true. Figure 2 illustrates an example of rank-1 cut (a 3-Subset Row Cut), when $\rho_i = 1/2$ for $i \in C$, where C is composed by linehaul customers 1 and 2 and by backhaul customer 3. In this example, (28) cuts the fractional solution $\bar{\lambda} \in P_1$ (route p_1 passes by customers 1 and 3, p_2 by 2 and 3, and route p_3 by 2 and 1) but the corresponding solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ is not cut by (29). □



(a) $\lambda_{p_1} + \lambda_{p_2} + \lambda_{p_3} \leq 1$ (b) $\lambda^L_{L(p_3)} \leq 1$

Figure 2: Example of rank-1 cut with both types of customers, where the hexagon represents the backhaul customer. Note that the cut in 2a is effective, but 2b is not.

4. Branch-cut-and-price algorithms

This section describes BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$, two BCP algorithms for the VRPB based on $\mathcal{F}1$ and $\mathcal{F}2$, respectively. More precisely, we discuss elements related to pricing, cut generation, branching and path enumeration. Furthermore, we also describe how BCP$_{\mathcal{F}1}$ can be adapted to solve the HFFVRPB and VRPBTW.

4.1. Pricing subproblem

In both BCP algorithms, the pricing subproblems are modeled as a *resource constrained shortest path problem* (RCSP), which is defined as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph, where $\mathcal{V}$ is the set of vertices, $\mathcal{A}$ the set of arcs and $\bar{c}_a \in \mathbb{R}$ is the cost of the arc $a \in \mathcal{A}$. $\mathcal{V}$ has special nodes v_{source} and v_{sink}, they can be the same vertex or two distinct vertices. For each arc $a \in \mathcal{A}$, there exists a resource consumption $q_a \in \mathbb{R}_+$. Also, an interval $[l_i, u_i]$ is associated to each vertex $i \in \mathcal{V}$. A resource constrained path $p = (v_{source} = v_0, v_1, \ldots, v_{k-1}, v_{sink} = v_k)$ over $\mathcal{G}$ is feasible if $k \geq 1$, $v_j \neq v_{source}, v_j \neq v_{sink}$, $1 \leq j \leq k-1$, and the accumulated resource consumption S_j at visit j, $0 \leq j \leq k$, where $S_0 = 0$ and $S_j = \max\{l_{v_j}, S_{j-1} + q_{(v_{j-1}, v_j)}\}$, does not exceed u_{v_j}. Note that this definition allows "dropping resources", if needed to satisfy the lower limit l_i at a

vertex i. On the other hand, the upper limits on accumulated resource consumption are strict. The RCSP objective is to find a resource-constrained path with minimum cost.

4.1.1. RCSP graph for $BCP_{\mathcal{F}1}$

For $\text{BCP}_{\mathcal{F}1}$, the RCSP graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}) = (V, A) = G$; $v_{source} = v_{sink} = 0$. Each arc $a = (i, j) \in \mathcal{A}$ has a capacity resource consumption given by $q_a = d_j$ and each vertex $i \in \mathcal{V}$ has a resource interval defined as:

$$[l_i, u_i] = \begin{cases} [0, 2Q], i = 0 \\ [0, Q], i \in L \\ [Q + d_i, 2Q], i \in B \end{cases}$$

Figure 3 illustrates the RCSP graph for $\text{BCP}_{\mathcal{F}1}$. It can be seen that a resource constrained path in that graph can visit customers in L until the capacity limit Q is reached. However, when the path visits the first backhaul customer, the values of l_i for $i \in B$ force any unused linehaul capacity to be dropped. Therefore, the total backhaul capacity is also limited by Q. The cost of an arc $\bar{c}_a$ is the reduced cost calculated through the dual variables associated with constraints (11)–(14).

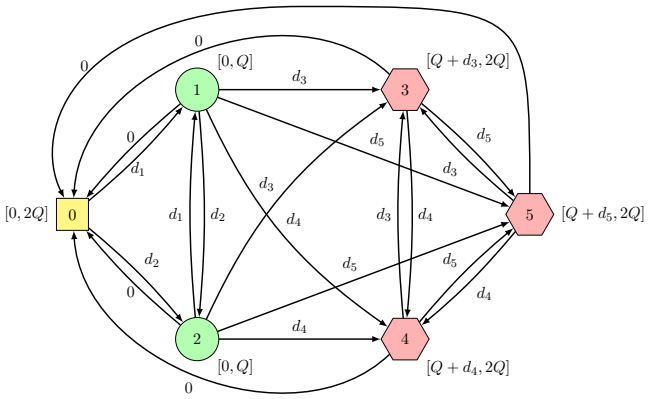


Figure 3: RCSP graph for $\text{BCP}_{\mathcal{F}1}$.

4.1.2. RCSP graph for $BCP_{\mathcal{F}2}$

For $\text{BCP}_{\mathcal{F}2}$ there are two RCSP graphs. The first RCSP graph is $\mathcal{G}_L = (\mathcal{V}_L, \mathcal{A}_L)$, where $\mathcal{V}_L = L_0 \cup \{0'\}$ and $\mathcal{A}_L = A_L \cup \{(i, 0') : i \in L\}$; $v_{source} = 0$ and $v_{sink} = 0'$. Each arc $a = (i, j) \in \mathcal{A}_L$ has a capacity resource consumption given by $q_a = d_j$ (assuming that $d_{0'} = 0$) and each vertex $i \in \mathcal{V}_L$ has resource consumption interval $[0, Q]$.

The second RCSP graph is $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{A}_B)$, where $\mathcal{V}_B = \{0'\} \cup L \cup B_0$ and $\mathcal{A}_B = A_{LB} \cup A_B \cup \{(0', i) : i \in V^+\}$; $v_{source} = 0'$ and $v_{sink} = 0$. Each arc $a = (i, j) \in \mathcal{A}_B$ has a capacity resource

consumption given by $q_a = d_j$ (assuming that $d_0 = 0$) and each vertex $i \in \mathcal{V}_B$ has a resource interval $[0, Q]$.

Figure 4 illustrates the two RCSP graphs for BCP$_{\mathcal{F}2}$. The cost of an arc $\bar{c}_a$ for both graphs is the reduced cost calculated through the dual variables associated with constraints (21)–(25).

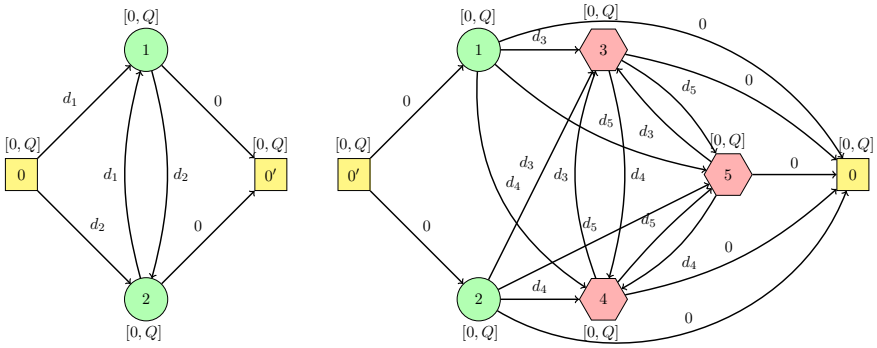


Figure 4: RCSP graphs for BCP$_{\mathcal{F}2}$

4.1.3. Solving the pricing subproblems

The RCSP problems above defined are solved by a labeling algorithm, using the bucket graph based variant proposed by Sadykov et al. (2017). Such algorithm also handles *ng*-routes (for BCP$_{\mathcal{F}1}$) and *ng*-paths (for BCP$_{\mathcal{F}2}$). In both cases the *ng*-sets have cardinality 8. As mentioned before, the *ng*-set of a linehaul customer only has linehaul customers, and the *ng*-set of a backhaul customer only has backhaul customers. Moreover, the labeling algorithm also considers the modification in the reduced costs induced by the dual variables of the limited memory rank-1 cuts added.

The big advantage of $\mathcal{F}2$ over $\mathcal{F}1$ is that it reduces the time spent solving pricing problems, the usual bottleneck of the BCP algorithms. In a labeling algorithm for the RCSP, the number of undominated labels grows more than linearly with the size of the paths (in fact, exponentially in the worse case). Therefore, solving two RCSPs with capacity limit Q (associated to the paths in Ω^L and Ω^B) is typically much faster than solving a single RCSP with limit $2Q$ (associated to the longer routes in Ω).

4.2. Cut generation, branching and path enumeration

In both BCP algorithms, rounded capacity cuts are separated by the heuristic procedure available in CVRPSEP (Lysgaard, 2003). Fractional solutions of $\mathcal{F}1$ and $\mathcal{F}2$ are first converted to arc variables x in order to perform that separation.

Limited memory rank-1 cuts are separated for sets C, such that $|C| \leq 5$, using the optimal multipliers given in Pecin et al. (2017b). As shown in Proposition 2, rank-1 cuts for $\mathcal{F}2$ where

C has both linehaul and backhaul customers are weak and not likely to be violated. This is the main potential disadvantage of $\mathcal{F}2$ over $\mathcal{F}1$.

In both BCP algorithms, branching is performed over aggregations of arc variables. For a pair of vertices i and j in V, $i < j$, $y_{ij} = x_{ij} + x_{ji}$ (if $(j,i) \notin A$, $y_{ij} = x_{ij}$) should be integer. A fractional y_{ij} is chosen by a strong branching procedure similar to the one in Pecin et al. (2017b).

Both BCP algorithms may also perform route enumeration, as in Baldacci et al. (2008) and Contardo & Martinelli (2014), when the gap between a node lower bound and the upper bound is sufficiently small. This means that all elementary routes in Ω (for BCP$_{\mathcal{F}1}$) or all elementary paths in Ω^L and Ω^B (for BCP$_{\mathcal{F}1}$) with reduced cost not higher than the gap are enumerated to a pool. After that, the pricing is performed by inspection, which can save a lot of time. As the lower bounds increase, fixing by reduced cost reduces the size of the pools. Eventually, the pool size becomes small enough so that the restricted $\mathcal{F}1$ (or $\mathcal{F}2$) can be solved using a MIP solver, thus finishing the node. The enumeration is another significant potential advantage of $\mathcal{F}2$ over $\mathcal{F}1$. As there are much fewer paths in Ω^L and Ω^B than in Ω, it is possible to perform enumeration in $\mathcal{F}2$ earlier, with a larger gap.

4.3. VRPBTW and HFFVRPB

The BCP$_{\mathcal{F}1}$ approach can be directly adapted to solve the VRPBTW. This only requires an additional time resource. For a given arc $a = (i,j)$ in the RCSP graph, the consumption of this resource is $c_{ij} + s_j$. The resource consumption interval for that resource in each vertex is the associated customer time window. The hierarchical objective of the VRPBTW can be handled by running the algorithm for different values of K. The initial value of K is defined by the best known solution. The value of K is then iteratively decremented until the problem becomes infeasible (the last feasible solution found is the optimal one).

On the other hand, $\mathcal{F}2$ cannot be adapted to solve the VRPBTW. This is due to the fact that the time resource is global, in the sense that it can not be split *a priori* between linehaul and backhaul customers (in contrast, there are separated capacities Q for linehaul and backhaul customers).

In order to adapt BCP$_{\mathcal{F}1}$ for HFFVRPB, it is necessary to define a distinct RCSP graph for each type of vehicle, where each graph has specific arc costs. Moreover, constraints (12) should now limit the number of available vehicles for each vehicle type, as specified in the problem instance.

Adapting BCP$_{\mathcal{F}2}$ for HFFVRPB would require a larger number of connecting constraints, like (23), to ensure that only linehaul and backhaul paths corresponding to the same vehicle type are connected. We therefore decided not to test BCP$_{\mathcal{F}2}$ for this variant.

5. Computational experiments

The BCP algorithms were coded in Julia 0.6 interface for the generic VRPSolver (Pessoa et al., 2019a) which makes use of JuMP (Dunning et al., 2017) and LightGraphs packages. The models used in the implementation are given in the Appendix B. The solver utilizes the BaPCod C++ library (Vanderbeck et al., 2018) as BCP framework combined with the C++ implementations by Sadykov et al. (2017) which contain: (i) a labeling algorithm for solving the pricing subproblems based on bucket graphs; (ii) path enumeration; (iii) a bucket arc elimination routine; (iv) a routine for separating limited-memory rank-1 cuts; and (v) dual price smoothing stabilization (Pessoa et al., 2018). Moreover, CVRPSEP package (Lysgaard, 2003) is used in the RCC separators and CPLEX 12.8 is used to solve the LP relaxations and the MIPs over the enumerated paths.

The experiments were executed on a 2 Deca-core Haswell Intel Xeon E5-2680 v3 server with 2.50 GHz and 128 GB of RAM. Each algorithm was run on a single thread for each instance. To reduce the testing time, multiple runs (64) for different instances were performed simultaneously on the same machine, effectively reducing the amount of RAM allocated to each process. A time limit of 60 hours was imposed for the algorithms.

5.1. Benchmark instances

All experiments were performed over symmetric instances. Their description is presented in the following.

5.1.1. VRPB instances

We considered three sets of VRPB instances. The first two are classical small and medium size datasets, whereas the third one is introduced in this work to test the limits of our methods on instances of larger scale.

- **GJB**. This dataset consists of 68 instances proposed by Goetschalckx & Jacobs-Blecha (1989) including between 25 and 200 customers. The fleet size K is fixed and any feasible solution should have exactly K non-empty routes. We use double precision for the distance matrix and the upper bounds provided in the work by Cuervo et al. (2014).

- **TV**. This group is composed of 33 instances suggested by Toth & Vigo (1997) varying between 21 and 100 customers. The convention regarding the number of vehicles is the same as in the previous dataset. In this case, the values of the distance matrix were rounded to the nearest integer. Moreover, we also use the upper bounds presented in Cuervo et al. (2014).

- **X**. This new benchmark dataset contains 300 instances varying between 100 and 1000 customers. They were generated based on the CVRP instances proposed by Uchoa et al. (2017).

For each CVRP instance, we created 3 VRPB ones with 50%, 66% and 80% of linehaul customers, respectively, following the same scheme as Toth & Vigo (1997). For example, we used the CVRP instance X-n101-k25 to generate the VRPB instances X-n101-50-k13, X-n101-66-k17, X-n101-80-k21. It is important to emphasize that the fleet size is not fixed for this dataset. We adopted the nearest integer precision convention for the distance matrix. The upper bounds for these instances were obtained by running the algorithms developed in Vidal et al. (2014) and Subramanian et al. (2013). This newly proposed benchmark is available at `http://www.vrp-rep.org/datasets/download/queiroga-et-al-2019.zip`.

5.1.2. VRPBTW and HFFVRPB instances

The experiments on the VRPBTW and HFFVRPB were conducted with the following benchmarks:

- **GDDS**. This dataset contains 15 instances proposed by Gélinas et al. (1995) for the VRPBTW, all of them with 100 customers. All distances are calculated with double precision. The upper bounds were obtained from Vidal et al. (2014).
- **T**. This benchmark is composed of 18 instances proposed by Tütüncü (2010) and contain between 50 to 100 customers. The double precision convention for the distance matrix was also adopted. For the instances HFFVRPB3, HFFVRPB6, HFFVRPB8, HFFVRPB12, HFFVRPB14, HFFVRPB17 and HFFVRPB18, we used the upper bounds provided in Tütüncü (2010). For the remaining ones, we considered those reported in Penna et al. (2019), where the authors claim that the first five aforementioned instances have no feasible solution.

5.2. Results for the VRPB

In the tables presented hereafter, UB refers to the upper bound provided to the exact algorithms, $z(IP)$ indicates the value of the optimal solution or an improved upper bound, LB^f_{root} corresponds to the final lower bound found at the root node; t_{total} is the total CPU time, $t_{pricing}$ is the total pricing time, and *nodes* represents the number of nodes in the tree.

Table 1 presents the results obtained by BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$ for the GJB instances. All instances were solved to optimality by both algorithms. Note that almost all instances were solved to optimality at the root node, including most of the 200-customer ones. Furthermore, we were able to improve the best-known solution the instance O1. Regarding the CPU time, BCP$_{\mathcal{F}2}$ is clearly faster than BCP$_{\mathcal{F}1}$, except for very few cases (instances G4 and G5). BCP$_{\mathcal{F}2}$ can be around 7 times faster as it happened on instance O1. Hence, although the LB^f_{root} obtained by BCP$_{\mathcal{F}2}$ can be occasionally slightly weaker than the one achieved by BCP$_{\mathcal{F}1}$, it appears that the first has a better overall performance than the latter. Nonetheless, in practice, the bound LB^f_{root} obtained by F2 can be better than the one obtained by F1 because the cut generation may be interrupted when the CPU time required to solve the pricing subproblems is high.

Table 1: Results obtained for the GJB instances

Problem data						$z(IP)$	$BCP_{\mathcal{F}1}$			$BCP_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB		LB^f_{root}	t_{total}	$nodes$	LB^f_{root}	t_{total}	$nodes$
A1	25	20	5	8	229,885.65	229,885.65	229,885.65	< 1	1	229,885.65	< 1	1
A2	25	20	5	5	180,119.21	180,119.21	180,119.21	< 1	1	180,119.21	< 1	1
A3	25	20	5	4	163,405.38	163,405.38	163,405.38	< 1	1	163,405.38	< 1	1
A4	25	20	5	3	155,796.41	155,796.41	155,796.41	< 1	1	155,796.41	< 1	1
B1	30	20	10	7	239,080.15	$239,080.16^a$	239,080.16	< 1	1	239,080.16	< 1	1
B2	30	20	10	5	198,047.77	198,047.77	198,047.77	< 1	1	198,047.77	< 1	1
B3	30	20	10	3	169,372.29	169,372.29	169,372.29	< 1	1	169,372.29	< 1	1
C1	40	20	20	7	250,556.77	250,556.77	250,556.77	< 1	1	250,556.77	< 1	1
C2	40	20	20	5	215,020.23	215,020.23	215,020.23	2	1	215,020.23	$< \mathbf{1}$	1
C3	40	20	20	5	199,345.96	199,345.96	199,345.96	< 1	1	199,345.96	< 1	1
C4	40	20	20	4	195,366.63	195,366.63	195,366.63	< 1	1	195,366.63	< 1	1
D1	38	30	8	12	322,530.13	322,530.13	322,530.13	< 1	1	322,530.13	< 1	1
D2	38	30	8	11	316,708.86	316,708.86	316,708.86	< 1	1	316,708.86	< 1	1
D3	38	30	8	7	239,478.63	239,478.63	239,478.63	< 1	1	239,478.63	< 1	1
D4	38	30	8	5	205,831.94	205,831.94	205,831.94	6	1	205,831.94	**2**	1
E1	45	30	15	7	238,879.58	238,879.58	238,879.58	< 1	1	238,879.58	< 1	1
E2	45	30	15	4	212,263.11	212,263.11	212,263.11	< 1	1	212,263.11	< 1	1
E3	45	30	15	4	206,659.17	206,659.17	206,659.17	1	1	206,659.17	$< \mathbf{1}$	1
F1	60	30	30	6	263,173.96	263,173.96	263,173.96	5	1	263,173.96	**3**	1
F2	60	30	30	7	265,214.16	265,214.16	265,214.16	2	1	265,214.16	$< \mathbf{1}$	1
F3	60	30	30	5	241,120.77	$241,120.78^a$	241,120.78	2	1	241,120.78	**1**	1
F4	60	30	30	4	233,861.84	$233,861.85^a$	233,861.85	3	1	233,861.85	**2**	1
G1	57	45	12	10	306,305.40	306,305.40	306,305.40	5	1	306,305.40	5	1
G2	57	45	12	6	245,440.99	245,440.99	245,440.99	3	1	245,440.99	3	1
G3	57	45	12	5	229,507.48	229,507.48	229,507.48	3	1	229,507.48	**2**	1
G4	57	45	12	6	232,521.25	232,521.25	232,521.25	**3**	1	232,521.25	5	1
G5	57	45	12	5	221,730.35	221,730.35	221,730.35	**3**	1	221,730.35	4	1
G6	57	45	12	4	213,457.45	213,457.45	213,457.45	3	1	213,457.45	**2**	1
H1	68	45	23	6	268,933.06	268,933.06	268,933.06	8	1	268,933.06	**7**	1
H2	68	45	23	5	253,365.50	253,365.50	253,365.50	5	1	253,365.50	**2**	1
H3	68	45	23	4	247,449.04	247,449.04	247,449.04	4	1	247,449.04	**3**	1
H4	68	45	23	5	250,220.77	250,220.77	250,220.77	4	1	250,220.77	**3**	1
H5	68	45	23	4	246,121.31	246,121.31	246,121.31	5	1	246,121.31	**3**	1
H6	68	45	23	5	249,135.32	249,135.32	249,135.32	5	1	249,135.32	**3**	1
I1	90	45	45	10	350,245.28	350,245.28	350,245.28	19	1	350,245.28	**5**	1
I2	90	45	45	7	309,943.84	309,943.84	309,943.84	16	1	309,943.84	**3**	1
I3	90	45	45	5	294,507.38	294,507.38	294,507.38	37	1	294,507.38	**12**	1
I4	90	45	45	6	295,988.44	$295,988.45^a$	295,988.45	22	1	293,840.10	**9**	3
I5	90	45	45	7	301,236.00	$301,236.01^a$	301,236.01	12	1	301,236.01	**4**	1
J1	94	75	19	10	335,006.68	335,006.68	335,006.68	13	1	335,006.68	**12**	1
J2	94	75	19	8	310,417.21	310,417.21	310,417.21	48	1	310,417.21	**33**	1

(*Continues on the next page)*

Problem data						$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB		LB^f_{root}	t_{total}	$nodes$	LB^f_{root}	t_{total}	$nodes$
J3	94	75	19	6	279,219.21	279,219.21	279,219.21	19	1	279,219.21	**12**	1
J4	94	75	19	7	296,533.16	296,533.16	294,480.85	367	5	294,168.05	**309**	7
K1	113	75	38	10	394,071.16	394, 071.17[a]	394,071.17	52	1	394,071.17	**23**	1
K2	113	75	38	8	362,130.00	362,130.00	362,130.00	36	1	362,130.00	**14**	1
K3	113	75	38	9	365,694.08	365,694.08	365,694.08	26	1	365,694.08	**12**	1
K4	113	75	38	7	348,949.39	348,949.39	348,949.39	67	1	348,949.39	**29**	1
L1	150	75	75	10	417,896.72	417,896.71	417,896.71	82	1	417,896.71	**44**	1
L2	150	75	75	8	401,228.80	401,228.80	401,228.80	110	1	401,228.80	**57**	1
L3	150	75	75	9	402,677.72	402,677.72	402,677.72	76	1	402,677.72	**35**	1
L4	150	75	75	7	384,636.33	384,636.33	384,636.33	67	1	384,636.33	**28**	1
L5	150	75	75	8	387,564.55	387,564.55	387,564.55	55	1	387,564.55	**23**	1
M1	125	100	25	11	398,593.19	398,593.19	398,593.19	95	1	398,593.19	**56**	1
M2	125	100	25	10	396,916.97	396,916.97	396,916.97	112	1	395,706.60	**85**	3
M3	125	100	25	9	375,695.41	375, 695.42[a]	373,010.93	6210	41	372,016.21	**4139**	39
M4	125	100	25	7	348,140.16	348,140.16	348,140.16	181	1	347,010.67	**160**	3
N1	150	100	50	11	408,100.62	408,100.62	408,100.62	112	1	406,628.97	**56**	3
N2	150	100	50	10	408,065.44	408,065.44	408,065.44	124	1	406,269.57	**77**	3
N3	150	100	50	9	394,337.86	394,337.86	394,337.86	169	1	394,337.86	**46**	1
N4	150	100	50	10	394,788.36	394,788.36	394,788.36	193	1	394,788.36	**50**	1
N5	150	100	50	7	373,476.30	373,476.30	373,476.30	247	1	373,476.30	**80**	1
N6	150	100	50	8	373,758.65	373,758.65	373,758.65	189	1	373,758.65	**65**	1
O1	200	100	100	10	478,347.72	**478,126.75**	475,839.08	9078	23	476,239.68	**1173**	9
O2	200	100	100	11	477,256.15	477,256.15	477,256.15	285	1	477,256.15	**77**	1
O3	200	100	100	9	457,294.48	457,294.48	457,294.48	207	1	457,294.48	**80**	1
O4	200	100	100	10	458,874.87	458,874.87	458,874.87	130	1	458,874.87	**39**	1
O5	200	100	100	7	436,974.20	436,974.20	436,974.20	524	1	436,974.20	**168**	1
O6	200	100	100	8	438,004.69	438,004.69	438,004.69	269	1	438,004.69	**108**	1
Mean								284.2			105.59	
Geometric mean								14.3			8.4	

[a]Difference between optimal solution and BKS possible due to the rounding.

Table 2 reports the results obtained by BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$ for the TV instances. Once again, all instances were solved to optimality by both algorithms, where 9 of them were proven optimal for the first time. Furthermore, we were able to improve the best-known solution of instance E-n101-B-66. Incidentally, except for this particular instance, all other cases were solved at the at root node.

Table 2: Results obtained for the TV instances

Problem data						$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB		LB^f_{root}	t_{total}	$nodes$	LB^f_{root}	t_{total}	$nodes$
E-n22-50	21	10	11	3	371	371	371.00	2	1	371.00	2	1
E-n22-66	21	14	7	3	366	366	366.00	2	1	366.00	2	1

(*Continues on the next page)*

Problem data						$z(IP)$	$BCP_{\mathcal{F}1}$			$BCP_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB		LB^f_{root}	t_{total}	$nodes$	LB^f_{root}	t_{total}	$nodes$
E-n22-80	21	17	4	3	375	375	375.00	**2**	1	375.00	3	1
E-n23-50	22	11	11	2	682	682	682.00	3	1	682.00	3	1
E-n23-66	22	15	7	2	649	649	649.00	3	1	649.00	3	1
E-n23-80	22	18	4	2	623	623	623.00	3	1	623.00	3	1
E-n30-50	29	14	15	2	501	501	501.00	4	1	501.00	**3**	1
E-n30-66	29	19	10	3	537	537	537.00	3	1	537.00	3	1
E-n30-80	29	23	6	3	514	514	514.00	**3**	1	514.00	5	1
E-n33-50	32	16	16	3	738	738	738.00	3	1	738.00	3	1
E-n33-66	32	21	11	3	750	750	750.00	3	1	750.00	3	1
E-n33-80	32	26	6	3	736	736	736.00	3	1	736.00	3	1
E-n51-50	50	25	25	3	559	559	559.00	4	1	559.00	**3**	1
E-n51-66	50	33	17	4	548	548	548.00	4	1	548.00	**3**	1
E-n51-80	50	40	10	4	565	565	565.00	**4**	1	565.00	6	1
E-n76-A-50	75	38	37	6	739	739	739.00	8	1	739.00	**6**	1
E-n76-A-66	75	50	25	7	768	768	768.00	6	1	768.00	**5**	1
E-n76-A-80	75	60	15	8	781	781*	781.00	5	1	781.00	**4**	1
E-n76-B-50	75	38	37	8	801	801	801.00	4	1	801.00	4	1
E-n76-B-66	75	50	25	10	873	873	873.00	**6**	1	873.00	7	1
E-n76-B-80	75	60	15	12	919	919	919.00	4	1	919.00	4	1
E-n76-C-50	75	38	37	5	713	713	713.00	13	1	713.00	**8**	1
E-n76-C-66	75	50	25	6	734	734	734.00	11	1	734.00	**9**	1
E-n76-C-80	75	60	15	7	733	733*	733.00	**23**	1	733.00	26	1
E-n76-D-50	75	38	37	4	690	690	690.00	6	1	690.00	**4**	1
E-n76-D-66	75	50	25	5	715	715*	715.00	26	1	715.00	**15**	1
E-n76-D-80	75	60	15	6	694	694*	694.00	14	1	694.00	**10**	1
E-n101-A-50	100	50	50	4	831	831*	831.00	39	1	831.00	**15**	1
E-n101-A-66	100	66	34	6	846	846	846.00	14	1	846.00	**9**	1
E-n101-A-80	100	80	20	6	856	856*	856.00	114	1	856.00	**72**	1
E-n101-B-50	100	50	50	7	923	923*	923.00	28	1	923.00	**22**	1
E-n101-B-66	100	66	34	9	983	**982***	976.22	1020	6	973.95	**243**	7
E-n101-B-80	100	80	20	11	1008	1008*	1008.00	**44**	1	1008.00	45	1
Average								43.3			16.8	
Geometric mean								7.7			6.5	

New optimal solutions found by BCP algorithms are marked with an asterisk.

Table 3 provides a comparison between $BCP_{\mathcal{F}1}$ and $BCP_{\mathcal{F}2}$ for the first 45 instances of the X set. They were solved to optimality by both methods. Note that instances X-n125-80-k23 and X-n162-66-k8 are particularly difficult and required more than 10000 and 135000 seconds to be solved, respectively, regardless of the method. Overall, $BCP_{\mathcal{F}2}$ visibly had a superior runtime performance than $BCP_{\mathcal{F}1}$, more specifically, the former was, on average, approximately 4 times faster than the latter.

Table 3: Comparison between the two BCP algorithms for the X instances. Only the first 45 instances of X were considered.

Instance	UB	$z(IP)$	BCP$_{\mathcal{F}1}$				BCP$_{\mathcal{F}2}$			
			LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n101-50-k13	19033	19033	18943.90	246	19	11	18925.33	**73**	3	5
X-n101-66-k17	20490	20490	20366.60	465	39	23	20356.54	**162**	4	5
X-n101-80-k21	23305	23305	23305.00	63	7	1	23305.00	**33**	2	1
X-n106-50-k7	15413	15413	15413.00	81	27	1	15413.00	**20**	4	1
X-n106-66-k9	18984	18984	18984.00	146	37	1	18984.00	**40**	8	1
X-n106-80-k11	22131	22131	22102.84	1242	239	11	22098.35	**397**	28	7
X-n110-50-k7	13103	13103	13103.00	22	7	1	13103.00	**10**	2	1
X-n110-66-k9	13598	13598	13598.00	23	11	1	13598.00	**9**	3	1
X-n110-80-k11	14302	14302	14225.50	414	42	5	14215.00	**281**	21	7
X-n115-50-k8	13927	13927	13927.00	35	16	1	13927.00	**22**	7	1
X-n115-66-k8	14032	14032	14032.00	48	20	1	14032.00	**25**	9	1
X-n115-80-k9	13536	13536	13536.00	50	19	1	13536.00	**31**	13	1
X-n120-50-k3	12416	12416	12416.00	243	81	1	12416.00	**73**	17	1
X-n120-66-k4	13145	13145	13099.07	1377	545	3	13145.00	**325**	137	1
X-n120-80-k5	13528	13528	13475.60	3052	1707	15	13464.19	**2737**	1575	17
X-n125-50-k16	32224	32224	32078.30	3688	310	79	32064.62	**915**	56	39
X-n125-66-k19	36400	36400	36350.86	1098	362	9	36348.25	**271**	34	3
X-n125-80-k23	43960	43960	43824.25	**10323**	2341	129	43822.41	11877	1306	245
X-n129-50-k10	19468	19468	19428.45	1358	143	9	19408.88	**335**	29	7
X-n129-66-k12	22606	22606	22555.75	946	141	11	22553.90	**226**	19	7
X-n129-80-k14	24575	24575	24561.70	308	51	3	24552.18	**108**	22	3
X-n134-50-k7	8369	8369	8270.35	15713	10160	105	8315.24	**868**	390	5
X-n134-66-k9	8974	8974	8912.48	5796	3749	65	8890.39	**621**	353	15
X-n134-80-k11	9699	9699	9636.49	4606	2478	65	9636.66	**1222**	714	27
X-n139-50-k5	13281	13281	13228.16	1639	656	5	13236.76	**290**	41	3
X-n139-66-k7	13512	13512	13512.00	153	63	1	13512.00	**51**	15	1
X-n139-80-k8	13662	13662	13662.00	65	29	1	13662.00	**40**	19	1
X-n143-50-k4	14539	14539	14539.00	1592	941	1	14539.00	**214**	76	1
X-n143-66-k4	14310	14310	14310.00	233	128	1	14310.00	**82**	41	1
X-n143-80-k5	14447	14447	14395.42	3148	2244	5	14396.43	**2822**	1714	13
X-n148-50-k25	28210	28210	28173.04	112	13	3	28210.00	**30**	4	1
X-n148-66-k29	30482	30482	30403.78	421	40	13	30391.98	**112**	6	3
X-n148-80-k36	35430	35430	35333.16	394	22	13	35332.44	**318**	5	3
X-n153-50-k19	20536	20536	20536.00	53	32	1	20536.00	**23**	11	1
X-n153-66-k20	20613	20613	20613.00	68	34	1	20613.00	**31**	12	1
X-n153-80-k21	20819	20819	20813.00	77	40	3	20810.50	**57**	24	3
X-n157-50-k7	11727	11727	11727.00	333	150	1	11727.00	**37**	12	1
X-n157-66-k9	13651	13651	13651.00	123	49	1	13651.00	**43**	14	1
X-n157-80-k11	15264	15264	15256.18	1186	252	3	15245.48	**733**	164	7
X-n162-50-k6	12812	12812	12784.51	1310	762	3	12812.00	**157**	55	1
X-n162-66-k8	13450	**13417**	13289.78	137067	80154	607	13300.60	**19365**	8164	85

(*Continues on the next page)*

Instance	UB	$z(IP)$	BCP$_{\mathcal{F}1}$				BCP$_{\mathcal{F}2}$			
			LB^f_{root}	t_{total}	$t_{pricing}$	nodes	LB^f_{root}	t_{total}	$t_{pricing}$	*nodes*
X-n162-80-k9	13854	13854	13819.09	2294	1016	3	13854.00	**812**	329	1
X-n167-50-k5	16489	16489	16489.00	1989	1058	1	16489.00	**336**	91	1
X-n167-66-k7	17827	17827	17735.36	11480	6049	21	17716.79	**3411**	1813	17
X-n167-80-k8	19415	19415	19374.16	1554	740	3	19382.21	**770**	348	3
Average				4814.1	2600.5	27.6		**1120.3**	393.6	12.2
Geometric mean				540.2	163.5	4.5		**177.8**	38.1	3.0

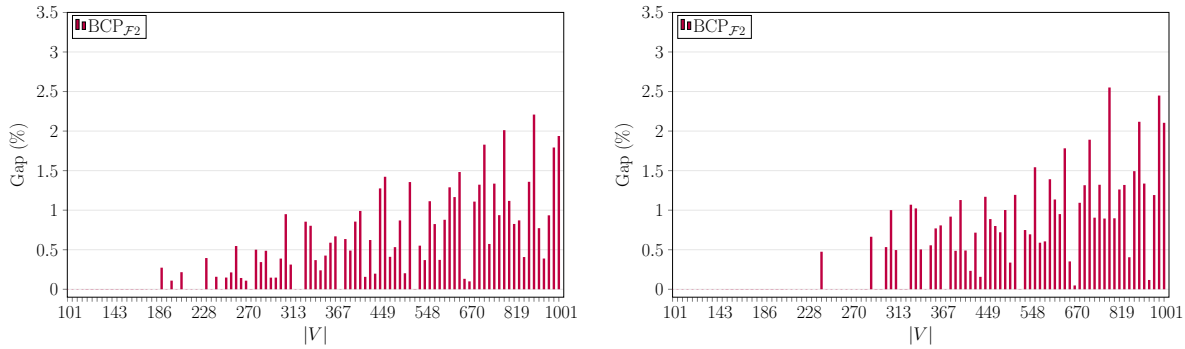
Because of the overall superior performance of BCP$_{\mathcal{F}2}$, we decided to run only this algorithm for the remaining X instances. Table 4 presents a summary of the results obtained by this method considering all instances of set X, while the table provided in the Appendix A shows the detailed results (except for those already reported in Table 3). On average, the results suggest the average gap does not seem to substantially vary according to the percentage of linehaul customers, but the CPU time clearly increases with the number of linehaul customers. On the other hand, the more the instance is balanced, the higher the number of proven optimal solutions. Finally, one can observe that 14 best-known solutions were improved, considering the cases where their optimality was proven or not.

Table 4: Summary of the results obtained by BCP$_{\mathcal{F}2}$ for the X instances, considering the percentage of linehaul customers.

	50%	66%	80%	All
Average gap (%)	0.53	0.46	0.50	0.50
Average time (min)	2110.3	2244.0	2359.7	2238.0
#Optima	46	40	37	123
#BKS improvements	6	3	5	14

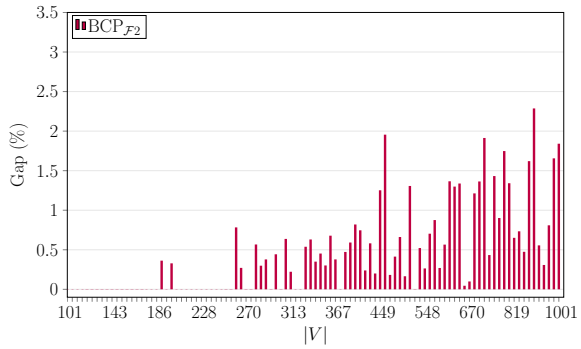
Figure 5 shows the gaps for each instance, according to the percentage of linehaul customers. It is possible to verify that all instances involving up to 237 customers were solved to optimality for 50%, whereas this number decreases to 186 customers for 66% and 80%. Furthermore, one can observe that the average gaps were generally below 2.5%, even for the larger instances, but in the vast majority of the cases they were below 2.0%, thus ratifying the high quality of the bounds reported.

Figure 6 illustrates the behavior of the average gaps as the estimated size of the routes increases. In this case, we used the same criterion adopted in Uchoa et al. (2017) to classify the groups of instances in "very small", "small", "medium", "long", "very long". The box plots suggest that the smaller the size of the routes, the smaller the gaps and the higher the robustness obtained. In addition, note that at least 25% of the instances of each group were solved to optimality.

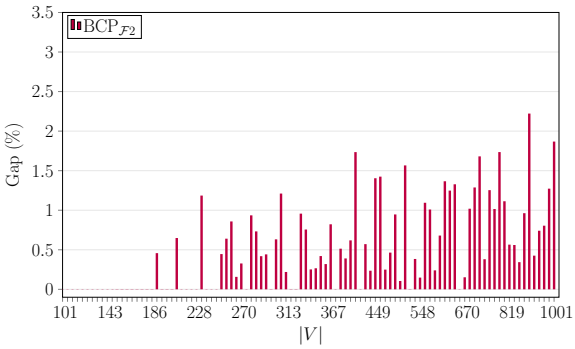


(a) All

(b) 50%



(c) 66%



(d) 80%

Figure 5: Average gaps for the X instances. In Figure 5a, the value reported is given for each value of $|V|$ as the average gap of the three related instances. The other figures show the gap of the instances associated with the corresponding percentage of linehauls.

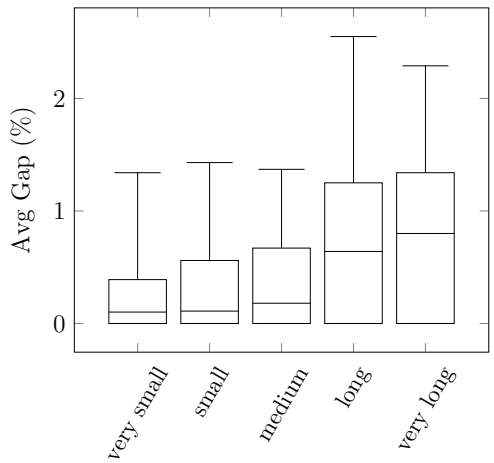


Figure 6: Average gaps with respect to the size of the route

5.3. Results for the HFFVRPB and VRPBTW

Table 5 shows the results obtained for the HFFVRPB instances. All optimal solutions were found by the proposed algorithm. The instances with up to 75 customers were solved to optimality at the root node in a matter of seconds, whereas the 100-customer instances were solved in at most 2123 seconds. The proposed algorithm was capable of improving best-known solution of 6 instances, including all the 100-customer ones. Moreover, we also confirmed the observation made by Penna et al. (2019) and proved that instances HFFVRPB3, HFFVRPB6, HFFVRPB8, HFFVRPB12 and HFFVRPB14 are indeed infeasible.

Table 5: Results for the HFFVRPB

Problem data					BCP			
Instance	$n+m$	n	m	UB	LB^t_{root}	$z(IP)$	t_{total}	$nodes$
HFFVRPB1	50	25	25	874.60	874.60	874.60	4	1
HFFVRPB2	50	34	16	911.20	911.20	911.20	4	1
HFFVRPB3	50	40	10	998.22	–	–	–	–
HFFVRPB4	50	25	25	1050.60	1050.60	1050.60	6	1
HFFVRPB5	50	34	16	1051.30	1051.30	1051.30	5	1
HFFVRPB6	50	40	10	1183.36	–	–	–	–
HFFVRPB7	75	37	38	1073.90	1070.00	**1070.00**	25	1
HFFVRPB8	75	50	25	1182.66	–	–	–	
HFFVRPB9	75	60	15	1003.20	1003.20	1003.20	8	1
HFFVRPB10	75	37	38	1553.00	1553.00	1553.00	7	1
HFFVRPB11	75	50	25	1659.80	1659.80	1659.80	11	1
HFFVRPB12	75	60	15	1917.54	–	–	–	–
HFFVRPB13	100	50	50	1181.70	1167.43	**1180.30**	2123	5
HFFVRPB14	100	67	33	1109.02	–	–	–	–
HFFVRPB15	100	80	20	1114.90	1097.36	**1105.10**	1443	8
HFFVRPB16	100	50	50	1314.50	1305.98	**1312.80**	941	2
HFFVRPB17	100	67	33	1585.30	1210.77	**1211.70**	269	1
HFFVRPB18	100	80	20	1615.08	1279.36	**1282.00**	479	2

The results obtained for the VRPBTW instances can be found in Table 6. The optimality of all instances was proven, where new improved solutions were found for instances BHR104A, BHR104B and BHR104C. Almost all instances were solved to optimality at the root node, most of them in a matter of seconds. Instance BHR104A appears to be the most challenging one, where the algorithm required more than 1400 seconds to solve it.

6. Conclusions

In this paper, we proposed two branch-cud-and-price (BCP) approaches based on different mathematical formulations for the vehicle routing problem with backhauls (VRPB). While in one formulation the columns are based on complete routes ($\mathcal{F}_1$), in the other one the columns are based on separate linehaul and backhauls paths ($\mathcal{F}_2$). The BCP algorithms were implemented using the VRPSolver and they contain several successful methodological ingredients such as *ng*-routes/paths, limited memory rank-1 cuts, rounded capacity cuts, strong branching, route enumeration, arc elimination using reduced costs and dual stabilization.

Although it was proven that the linear relaxations of the formulations are equally strong, we demonstrated that rank-1 cuts for $\mathcal{F}_1$ may be stronger than the same type of cuts for $\mathcal{F}_2$.

Table 6: Results for VRPBTW

Problem data			BCP			
Instance	%BH	*UB*	LB^f_{root}	$z(IP)$	t_{total}	*nodes*
BHR101A	10	22/1818.86	1819	22/1818.86	1	1
BHR101B	30	23/1959.52	1960	23/1959.52	1	1
BHR101C	50	24/1939.10	1939	24/1939.10	1	1
BHR102A	10	19/1653.18	1653	19/1653.18	2	1
BHR102B	30	22/1750.70	1751	22/1750.70	1	1
BHR102C	50	22/1775.76	1776	22/1775.76	1	1
BHR103A	10	15/1385.38	1385	15/1385.38	2	1
BHR103B	30	15/1390.32	1390	15/1390.32	3	1
BHR103C	50	17/1456.48	1456	17/1456.48	2	1
BHR104A	10	10/1203.44	1183	**10/1202.53**	1437	11
BHR104B	30	11/1154.84	1258	**10/1258.48**	55	1
BHR104C	50	11/1191.38	1189	**11/1188.78**	11	1
BHR105A	10	15/1560.15	1547	15/1560.15	86	3
BHR105B	30	16/1583.30	1583	16/1583.30	1	1
BHR105C	50	16/1709.66	1688	16/1709.66	51	1

However, computational experiments on well-known benchmark instances revealed that the BCP algorithm over $\mathcal{F}_2$ has a better overall performance in practice. Nevertheless, both algorithms were capable of finding the optimal solutions for all instances, some of them for the first time. We also performed tests on a newly proposed set of instances that were derived from the X dataset of Uchoa et al. (2017). The BCP implementation based on $\mathcal{F}_2$ yielded better results than the one based on $\mathcal{F}_1$, confirming the efficiency of using separate variables for linehaul and backhaul paths. Finally, we conducted experiments on benchmark instances for the HFFVRPB, and of the VRPBTW. For these two problems, all benchmark instances were solved to optimality.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), grants 428549/2016-0, 307843/2018-1, and 308528/2018-2, and by the Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), grant E-26/203.310/2016. The experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr/).

Appendix A. Detailed results for the X instances

Table A.7: Results for X instances by the BCP$_{\mathcal{F}2}$ with a time limit of 60 hours. The results which were already reported in Table 3 were omitted. The final lower bound is denoted by LB_f.

Instance	*UB*	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	*nodes*
X-n172-50-k27	30634	30634.00	**30634**	30534.07	270	41	19
X-n172-66-k31	31864	31864.00	**31864**	31807.62	161	20	7

(*Continues on the next page*)

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n172-80-k39	36803	36803.00	**36803**	36745.58	560	37	15
X-n176-50-k23	45239	45239.00	**45239**	45161.05	437	60	37
X-n176-66-k24	46416	46416.00	**46416**	46336.75	736	122	65
X-n176-80-k25	47033	47033.00	**47033**	46986.46	341	61	11
X-n181-50-k12	16549	16549.00	**16549**	16549.00	57	15	1
X-n181-66-k15	18832	18832.00	**18832**	18832.00	41	14	1
X-n181-80-k18	21241	21241.00	**21241**	21241.00	54	15	1
X-n186-50-k8	17978	17978.00	**17978**	17867.86	9088	1721	41
X-n186-66-k10	19751	19751.00	**19751**	19751.00	302	118	1
X-n186-80-k12	21754	21754.00	**21754**	21630.52	21953	9826	113
X-n190-50-k4	11552	11552.00	**11552**	11492.09	9096	4850	29
X-n190-66-k5	12784	12737.65	–	12718.64	–	177686	231
X-n190-80-k6	14410	14344.08	–	14339.48	–	191111	237
X-n195-50-k27	29470	29470.00	**29470**	29375.37	698	46	25
X-n195-66-k34	33137	33137.00	**33137**	33076.72	166	20	7
X-n195-80-k42	38629	38629.00	**38629**	38629.00	186	27	1
X-n200-50-k18	34416	34408.00	**34408**	34290.22	81125	4131	1269
X-n200-66-k24	40474	40341.02	–	40320.87	–	21542	4435
X-n200-80-k29	47741	47741.00	**47741**	47713.57	585	76	5
X-n204-50-k10	15877	15877.00	**15877**	15840.93	1349	269	7
X-n204-66-k12	16703	16703.00	**16703**	16563.38	24018	4998	179
X-n204-80-k15	17832	17832.00	**17832**	17790.30	1394	411	5
X-n209-50-k8	21837	21837.00	**21837**	21648.76	52058	6891	205
X-n209-66-k11	24378	24378.00	**24378**	24208.76	75385	38246	327
X-n209-80-k13	27177	27000.70	–	26982.38	–	115022	661
X-n214-50-k6	9574	9574.00	**9574**	9549.86	3422	1118	9
X-n214-66-k8	10001	10001.00	**10001**	9963.86	2356	1490	9
X-n214-80-k9	10457	10457.00	**10457**	10405.89	18255	10668	63
X-n219-50-k37	64691	64691.00	**64691**	64619.03	50	10	3
X-n219-66-k48	80405	80405.00	**80405**	80405.00	40	14	1
X-n219-80-k59	95845	95845.00	**95845**	95845.00	36	16	1
X-n223-50-k18	27449	27442.00	**27442**	27326.24	11835	1069	139
X-n223-66-k23	30717	30717.00	**30717**	30567.40	35662	3976	407
X-n223-80-k27	34440	34440.00	**34440**	34335.92	15291	1752	161
X-n228-50-k19	23128	23128.00	**23128**	23078.40	1022	327	23
X-n228-66-k20	24114	24113.00	**24113**	24051.49	5511	2057	39
X-n228-80-k21	24592	24592.00	**24592**	24592.00	724	319	1
X-n233-50-k10	17186	17186.00	**17186**	17052.96	93406	71427	159
X-n233-66-k12	18026	18026.00	**18026**	17965.02	2328	1394	9
X-n233-80-k14	18885	18661.17	–	18641.11	–	174564	793
X-n237-50-k7	20745	20745.00	**20745**	20675.23	23322	13874	33
X-n237-66-k9	22471	22471.00	**22471**	22379.42	83074	66372	105
X-n237-80-k11	24357	24357.00	**24357**	24307.49	2593	1530	7
X-n242-50-k25	47949	47721.34	–	47671.06	–	19577	2899
X-n242-66-k32	57197	57197.00	**57197**	57043.85	84515	9241	1559

(*Continues on the next page*)

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n242-80-k39	68969	68965.00	**68965**	68827.21	63924	8640	895
X-n247-50-k42	36701	36701.00	**36701**	36701.00	76	42	1
X-n247-66-k43	36994	36994.00	**36994**	36994.00	84	41	1
X-n247-80-k45	37220	37205.00	**37205**	37199.14	293	130	3
X-n251-50-k14	24968	24968.00	**24968**	24875.60	25557	3127	127
X-n251-66-k18	27817	27817.00	**27817**	27712.41	72575	8566	355
X-n251-80-k22	32170	32026.60	–	32006.18	–	50325	965
X-n256-50-k8	15922	15922.00	**15922**	15922.00	404	189	1
X-n256-66-k11	17250	17250.00	**17250**	17250.00	521	290	1
X-n256-80-k13	18189	18072.54	–	18040.41	–	142662	819
X-n261-50-k7	21555	21555.00	**21555**	21456.47	63665	40112	37
X-n261-66-k9	23065	22884.42	–	22855.49	–	187116	61
X-n261-80-k11	25128	24912.36	–	24866.24	–	180864	281
X-n266-50-k30	47815	47783.00	**47783**	47648.94	101191	13933	1461
X-n266-66-k39	55962	55793.31	55945	55781.93	–	34684	3563
X-n266-80-k47	63880	63779.26	–	63730.98	–	24218	3147
X-n270-50-k18	24776	24751.00	**24751**	24653.46	45333	6267	237
X-n270-66-k24	26377	26377.00	**26377**	26328.74	1941	272	21
X-n270-80-k29	29789	29691.59	–	29658.37	–	30397	1377
X-n275-50-k14	15561	15561.00	**15561**	15514.61	5294	832	21
X-n275-66-k19	16944	16944.00	**16944**	16929.03	513	113	3
X-n275-80-k22	18690	18688.00	**18688**	18658.37	3719	681	29
X-n280-50-k13	29132	29132.00	**29132**	29004.13	128359	93994	71
X-n280-66-k15	31315	31137.44	–	31110.77	–	158023	407
X-n280-80-k16	32332	32029.64	–	32012.29	–	192361	147
X-n284-50-k8	15944	15944.00	**15944**	15833.57	93999	43893	209
X-n284-66-k10	17277	17225.38	–	17195.71	–	190019	21
X-n284-80-k12	18830	18692.09	–	18675.05	–	184435	127
X-n289-50-k34	57957	57572.16	–	57529.63	–	33748	1775
X-n289-66-k38	63446	63206.09	–	63186.76	–	42435	2423
X-n289-80-k47	75963	75644.77	–	75627.89	–	40762	1739
X-n294-50-k26	30859	30859.00	**30859**	30746.57	4905	468	45
X-n294-66-k33	34636	34636.00	**34636**	34542.41	12903	1056	143
X-n294-80-k40	39269	39095.51	–	39077.00	–	22714	1951
X-n298-50-k16	25081	25081.00	**25081**	24958.25	35865	3878	173
X-n298-66-k21	27643	27520.57	–	27470.73	–	54914	899
X-n298-80-k25	30222	30222.00	**30222**	30107.65	85792	22231	305
X-n303-50-k11	17763	17668.28	–	17646.72	–	128405	101
X-n303-66-k13	18120	18120.00	**18120**	18047.09	54891	34827	91
X-n303-80-k16	19603	19479.14	–	19456.65	–	172877	249
X-n308-50-k9	22544	22318.43	–	22304.65	–	194203	5
X-n308-66-k11	24154	24000.04	–	23990.67	–	194915	29
X-n308-80-k12	25164	24859.34	–	24844.42	–	209695	13
X-n313-50-k39	57762	57475.59	–	57444.76	–	41082	2765
X-n313-66-k44	60089	59935.56	60069	59914.94	–	28091	2901

(*Continues on the next page*)

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n313-80-k56	73834	73671.96	–	73654.22	–	37558	2661
X-n317-50-k27	43396	43391.00	**43391**	43367.32	1290	236	17
X-n317-66-k35	54502	54502.00	**54502**	54485.52	1018	233	15
X-n317-80-k43	63683	63683.00	**63683**	63665.51	626	250	9
X-n322-50-k14	23309	23309.00	**23309**	23139.98	132954	23529	349
X-n322-66-k19	25034	25034.00	**25034**	24951.63	6100	1703	21
X-n322-80-k23	27500	27500.00	**27500**	27375.81	184907	63083	453
X-n327-50-k10	21610	21378.95	–	21346.84	–	107246	145
X-n327-66-k13	23322	23196.47	–	23185.82	–	159555	69
X-n327-80-k16	24990	24750.89	–	24728.67	–	186367	221
X-n331-50-k8	24152	23905.00	–	23854.86	–	156722	103
X-n331-66-k10	26247	26081.55	–	26056.27	–	162636	81
X-n331-80-k12	28265	28051.29	–	28038.31	–	185486	35
X-n336-50-k45	81760	81348.01	–	81323.04	–	34124	2331
X-n336-66-k57	99226	98878.51	–	98861.47	–	40186	2861
X-n336-80-k68	116185	115892.61	–	115870.08	–	36471	2555
X-n344-50-k22	28527	28527.00	**28527**	28408.88	65642	10809	245
X-n344-66-k29	31845	31700.96	–	31675.50	–	33287	1215
X-n344-80-k35	35743	35647.85	–	35632.86	–	40381	921
X-n351-50-k21	18584	18480.67	–	18443.06	–	51270	937
X-n351-66-k26	19758	19698.37	–	19681.73	–	71277	859
X-n351-80-k32	22158	22064.80	–	22053.53	–	99086	703
X-n359-50-k15	33255	32999.06	–	32957.10	–	102315	183
X-n359-66-k19	37695	37439.36	–	37418.44	–	160781	161
X-n359-80-k23	43412	43273.41	–	43260.73	–	143353	193
X-n367-50-k12	20526	20360.22	–	20344.03	–	184200	9
X-n367-66-k14	21479	21397.74	–	21397.74	–	192713	3
X-n367-80-k15	22386	22201.94	–	22179.72	–	201875	17
X-n376-50-k47	80736	80736.00	**80736**	80684.10	705	139	11
X-n376-66-k62	100613	100613.00	**100613**	100573.61	2125	510	33
X-n376-80-k75	119581	119581.00	**119581**	119581.00	363	213	1
X-n384-50-k27	41206	40827.49	–	40802.84	–	43174	1045
X-n384-66-k35	47373	47149.06	–	47102.51	–	37375	1199
X-n384-80-k42	55386	55101.38	–	55085.96	–	50940	871
X-n393-50-k19	30005	29859.00	–	29847.13	–	72360	235
X-n393-66-k25	29340	29166.36	–	29143.75	–	84039	307
X-n393-80-k31	32619	32491.64	–	32485.70	–	106245	289
X-n401-50-k15	39746	39297.54	–	39263.14	–	199911	47
X-n401-66-k20	47658	47267.32	–	47253.12	–	202207	43
X-n401-80-k23	54270	53934.17	–	53919.99	–	205753	41
X-n411-50-k14	17959	17870.92	–	17870.92	–	99554	3
X-n411-66-k15	18785	18644.85	–	18629.82	–	135572	7
X-n411-80-k17	19496	19157.82	–	19150.91	–	206181	19
X-n420-50-k67	75527	75350.81	–	75327.16	–	57441	2107
X-n420-66-k86	76079	75897.40	–	75879.08	–	57456	1719

(*Continues on the next page*)

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n420-80-k105	89381	89356.00	**89356**	89268.90	72223	10370	707
X-n429-50-k31	41284	40988.41	–	40966.92	–	37302	953
X-n429-66-k40	47793	47515.13	–	47492.82	–	42041	987
X-n429-80-k48	54835	54521.75	–	54504.94	–	51458	749
X-n439-50-k19	27011	26968.47	–	26942.30	–	107470	25
X-n439-66-k25	28883	28825.18	–	28803.40	–	103337	235
X-n439-80-k30	32074	31998.55	–	31985.70	–	135395	165
X-n449-50-k15	36929	36497.20	–	36468.68	–	137845	125
X-n449-66-k20	41846	41322.08	–	41312.22	–	192684	93
X-n449-80-k23	46738	46081.67	–	46060.93	–	203178	49
X-n459-50-k14	18891	18723.33	–	18690.69	–	181996	11
X-n459-66-k18	20561	20158.99	–	20131.38	–	182592	49
X-n459-80-k21	22047	21732.92	–	21718.78	–	197129	55
X-n469-50-k70	123817	122782.66	123773	122729.09	–	45176	2703
X-n469-66-k90	148455	148184.44	–	148163.44	–	27860	2595
X-n469-80-k109	178511	178066.20	–	178047.53	–	30321	2083
X-n480-50-k36	52309	51932.05	–	51896.70	–	51958	507
X-n480-66-k47	63577	63313.99	–	63296.34	–	76292	563
X-n480-80-k56	73993	73649.48	–	73631.99	–	105667	313
X-n491-50-k30	43952	43511.61	–	43488.98	–	132730	121
X-n491-66-k39	49627	49298.76	–	49150.99	–	91111	339
X-n491-80-k47	56141	55609.55	–	55565.97	–	137699	303
X-n502-50-k20	40591	40453.96	–	40443.78	–	122851	11
X-n502-66-k26	49285	49203.55	–	49193.95	–	107574	27
X-n502-80-k31	56997	56936.89	–	56921.43	–	177770	11
X-n513-50-k11	21675	21416.30	–	21416.30	–	132874	3
X-n513-66-k14	22426	22132.96	–	22132.96	–	145374	3
X-n513-80-k17	23448	23080.70	–	23080.70	–	206960	3
X-n524-50-k125	154137	154137.00	**154137**	154079.33	1829	778	39
X-n524-66-k129	154416	154416.00	**154416**	154359.10	12384	4118	255
X-n524-80-k132	154497	154446.00	**154446**	154412.50	3549	1782	45
X-n536-50-k49	54658	54248.59	–	54192.49	–	104940	303
X-n536-66-k64	66032	65687.22	–	65667.46	–	114173	385
X-n536-80-k77	77811	77512.33	–	77494.63	–	149699	297
X-n548-50-k25	53049	52680.43	–	52648.24	–	122148	73
X-n548-66-k33	61421	61258.80	–	61242.89	–	142637	89
X-n548-80-k40	71867	71760.30	–	71748.57	–	155396	139
X-n561-50-k22	31826	31335.02	–	31306.93	–	190355	43
X-n561-66-k28	34370	34128.27	–	34099.20	–	190425	43
X-n561-80-k34	38053	37636.59	–	37587.65	–	187865	87
X-n573-50-k22	40239	40002.04	–	40002.04	–	117244	1
X-n573-66-k25	44151	43764.44	–	43764.44	–	181946	1
X-n573-80-k27	47054	46579.21	–	46579.21	–	200243	3
X-n586-50-k80	122632	121889.49	–	121856.57	–	128653	585
X-n586-66-k105	140396	140016.87	–	139990.62	–	71338	821

(*Continues on the next page)*

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n586-80-k127	160390	160005.48	–	159981.89	–	79723	613
X-n599-50-k47	65292	64383.64	–	64333.73	–	137453	197
X-n599-66-k61	76472	76039.68	–	76017.14	–	94481	339
X-n599-80-k74	89844	89233.37	–	89219.48	–	135692	243
X-n613-50-k32	40838	40374.44	–	40355.38	–	141694	75
X-n613-66-k41	46074	45444.70	–	45357.42	–	188735	95
X-n613-80-k50	52096	51384.31	–	51375.06	–	192914	151
X-n627-50-k22	38096	37733.89	–	37716.87	–	154012	71
X-n627-66-k29	44782	44200.41	–	44185.52	–	194269	61
X-n627-80-k35	52429	51774.60	–	51767.12	–	198419	53
X-n641-50-k18	42333	41578.58	–	41564.73	–	183134	39
X-n641-66-k23	47501	46865.58	–	46857.22	–	199661	19
X-n641-80-k28	54116	53397.37	–	53389.09	–	204268	13
X-n655-50-k66	59442	59232.54	–	59216.15	–	124641	515
X-n655-66-k87	72456	72423.25	–	72417.83	–	91208	787
X-n655-80-k105	86564	86564.00	**86564**	86541.56	50660	23748	165
X-n670-50-k112	144707	144637.00	–	144627.00	–	122379	45
X-n670-66-k117	144990	144845.79	–	144818.98	–	136613	27
X-n670-80-k120	145275	145053.66	–	145035.64	–	193090	17
X-n685-50-k43	48023	47497.89	–	47478.54	–	170949	73
X-n685-66-k54	53240	52594.33	–	52579.88	–	196578	55
X-n685-80-k62	59301	58696.62	–	58691.03	–	202085	51
X-n701-50-k23	51390	50713.99	–	50657.00	–	187874	39
X-n701-66-k30	58844	58041.55	–	58032.92	–	200435	9
X-n701-80-k36	68618	67734.38	–	67721.67	–	207724	7
X-n716-50-k18	29757	29194.35	–	29188.27	–	162933	7
X-n716-66-k23	32527	31904.64	–	31904.64	–	200782	1
X-n716-80-k28	37976	37337.69	–	37337.69	–	191495	1
X-n733-50-k83	80585	79855.45	–	79820.80	–	116949	267
X-n733-66-k102	92156	91756.77	–	91722.70	–	74442	455
X-n733-80-k125	110659	110237.94	–	110222.30	–	128547	197
X-n749-50-k49	47740	47109.17	–	47081.33	–	150513	91
X-n749-66-k63	55560	54764.41	–	54753.89	–	196196	75
X-n749-80-k78	63991	63188.68	–	63181.67	–	200628	95
X-n766-50-k58	95674	94818.31	–	94818.31	–	165150	1
X-n766-66-k62	101566	100650.37	–	100632.43	–	196948	5
X-n766-80-k65	106758	105674.28	–	105664.59	–	205395	15
X-n783-50-k24	49027	47776.33	–	47757.98	–	198844	19
X-n783-66-k31	53429	52495.08	–	52495.08	–	177559	3
X-n783-80-k38	60937	59879.72	–	59872.22	–	196223	3
X-n801-50-k20	48459	48023.99	–	48014.68	–	180780	5
X-n801-66-k27	54929	54192.27	–	54192.27	–	184590	3
X-n801-80-k32	62698	61999.99	–	61999.99	–	180045	3
X-n819-50-k86	89296	88169.26	–	88143.14	–	128254	189
X-n819-66-k112	108431	107725.03	–	107687.51	–	132300	187

(*Continues on the next page*)

Instance	UB	LB_f	$z(IP)$	LB^f_{root}	t_{total}	$t_{pricing}$	$nodes$
X-n819-80-k136	128617	127890.37	–	127873.51	–	144472	151
X-n837-50-k71	116553	115015.71	–	114974.20	–	124162	111
X-n837-66-k94	129183	128235.55	–	128202.08	–	157542	137
X-n837-80-k114	154966	154097.53	–	154083.03	–	173868	177
X-n856-50-k48	57777	57543.00	–	57528.60	–	128258	53
X-n856-66-k63	63542	63240.52	–	63221.75	–	157856	37
X-n856-80-k76	73802	73548.57	–	73533.10	–	168100	57
X-n876-50-k30	58780	57902.28	–	57890.13	–	186246	35
X-n876-66-k38	69617	68488.90	–	68479.07	–	203526	11
X-n876-80-k46	80983	80203.60	–	80203.60	–	207511	3
X-n895-50-k19	40668	39806.81	–	39806.81	–	161965	3
X-n895-66-k25	44059	43051.50	–	43051.50	–	132947	1
X-n895-80-k30	48451	47374.51	–	47374.51	–	188132	1
X-n916-50-k105	190108	187567.37	–	187526.31	–	107633	165
X-n916-66-k136	222807	221569.19	–	221516.85	–	122939	143
X-n916-80-k165	263885	262761.35	–	262719.81	–	138277	123
X-n936-50-k132	127497	127346.87	–	127322.08	–	164181	25
X-n936-66-k138	128871	128474.08	–	128443.05	–	199818	33
X-n936-80-k143	130808	129838.93	–	129817.96	–	204051	61
X-n957-50-k44	57019	56339.78	–	56297.23	–	162896	23
X-n957-66-k58	62593	62086.46	–	62069.11	–	190143	29
X-n957-80-k70	71855	71276.90	–	71239.16	–	203048	29
X-n979-50-k30	69739	68031.10	–	68010.80	–	183342	5
X-n979-66-k39	84499	83099.97	–	83099.97	–	212646	1
X-n979-80-k47	99605	98337.39	–	98337.39	–	195325	1
X-n1001-50-k22	49978	48926.15	–	48926.15	–	152333	1
X-n1001-66-k28	56126	55092.82	–	55092.82	–	174370	1
X-n1001-80-k34	63278	62096.05	–	62096.05	–	161585	1

Appendix B. VRPSolver models

VRPSolver is a framework for building BCP algorithms for VRP and related problems, available at `vrpsolver.math.u-bordeaux.fr`. It was used to implement our proposed VRPB algorithms, BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$. We present here the VRPSolver models used. This appendix is not self-contained, important concepts used in these models, such as main resources, packing sets, mapping between variables and arcs, and Rounded Capacity Cuts (RCC) separators, should be found in Pessoa et al. (2019b). The parameterization of the solver for all problems and formulations is the same: $\tau^{\text{soft}} = 5$ sec., $\tau^{\text{soft}} = 10$ sec., $\phi^{\text{bidir}} = 1$, $\omega^{\text{labels}} = 2 \cdot 10^5$, $\omega^{\text{routes}} = 2 \cdot 10^6$, $\eta^{\max} = 20$, $\delta^{\text{gap}} = 1.5\%$, $\zeta_1^{\text{num}} = 50$, $\zeta_1^{\text{estim}} = 1.0$. The meaning of these parameters is also explained in Pessoa et al. (2019b).

Appendix B.1. Formulation $\mathcal{F}1$

We first give the model corresponding to formulation $\mathcal{F}1$ for the VRPB. The RCSP graph is exactly the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}) = (V, A) = G$, together with the consumption and intervals defined in Section 4.1.1. The capacity resource is defined as a main resource. Define an integer variable x_a for each $a \in A$ (exactly the same arc variables defined in formulation $\mathcal{F}0$). The formulation is:

$$\text{Min} \sum_{a \in A} c_a x_a \tag{B.1}$$

$$\text{S.t.} \sum_{a \in \delta^-(i)} x_a = 1, \quad i \in V^+. \tag{B.2}$$

The number of paths in the solution is fixed to K ($L = U = K$). Each variable x_a is mapped to arc a ($M(x_a) = \{a\}$, $a \in A$). Packing sets are defined on vertices $\mathcal{B}^{\mathcal{V}} = \cup_{i \in V^+}\{\{i\}\}$. There are two RCC separators, the first is defined on $(\cup_{i \in L}\{(\{i\}, d_i)\}, Q)$, and the second is defined on $(\cup_{i \in B}\{(\{i\}, d_i)\}, Q)$. Branching is performed on the aggregation of x variables corresponding to opposite arcs. Enumeration is activated.

To adapt the model to the VRPBTW, we add a second main resource corresponding to the time, so $R = R_M = \{1, 2\}$. Arc resource consumption for the second resource equals the travelling time plus the service time: $q_{a=(i,j),2} = c_{ij} + s_j$, $a \in A$, where $s_0 = 0$. The resource consumption intervals for the second resource are equal to customer time windows (or to the time horizon for the depot). Otherwise, the model is the same as for the VRPB.

The model for the HFFVRPB, considering a set T of vehicle types, is the following. We define graphs $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{A}^k)$, $k \in T$, all of them isomorphic to G. However, the intervals are defined using the capacity Q^k of each type. We denote vertex i in graph $\mathcal{G}^k$ as i^k. Define $\delta^-(i^k)$ as the set of arcs in $\mathcal{A}^k$ entering i^k. Define an integer variable x_a^k per vehicle type $k \in T$ and per arc $a \in A^k$. The formulation is:

$$\text{Min} \sum_{k \in T} \sum_{a \in A^k} c_a^k x_a^k \tag{B.3}$$

$$\text{S.t.} \sum_{k \in T} \sum_{e \in \delta^-(i^k)} x_a^k = 1, \quad i \in V^+, \tag{B.4}$$

where c_a^k are the type dependent costs. The bounds for the number of paths from graph $\mathcal{G}^k$, $k \in T$, in the solution are $[0, U^k]$. Each variable $x^k_{a=(i^k,j^k)}$ is mapped to arc (i^k, j^k). Packing sets are defined on vertices $\mathcal{B}^{\mathcal{V}} = \cup_{i \in V^+}\{\{i^k : k \in T\}\}$. We have two RCC separators, the first is defined on $(\cup_{i \in L}\{(\{i^k : k \in T\}, d_i)\}, \max_{k \in T} Q^k)$, and the second is defined on $(\cup_{i \in B}\{(\{i^k : k \in T\}, d_i)\}, \max_{k \in T} Q^k)$. Branching is performed on variable expressions: i) on the number of used vehicles of each type $\sum_{i \in L} x^k_{(0,i)}$, $k \in T$; ii) on assignment of customers to vehicle types $\sum_{a \in \delta^-(i^k)} x_a^k$, $k \in T$, $i \in L \cup B$; iii) on aggregated edges $\sum_{k \in T} x^k_{(i^k,j^k)} + x^k_{(j^k,i^k)}$, $i, j \in V$, $i < j$.

Appendix B.2. Formulation $\mathcal{F}2$

We now give the model corresponding to formulation $\mathcal{F}2$ for the VRPB. It is less direct than the model for $\mathcal{F}1$, some tricks are needed for using VRPSolver in that case.

There are two RCSP graphs. The backhaul graph $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{A}_B)$ is exactly the one defined in Section 4.1.2. However, the linehaul graph $\mathcal{G}'_L = (\mathcal{V}'_L, \mathcal{A}'_L)$ is a bit different: $\mathcal{V}'_L = L_0 \cup \{i' : i \in L_0\}$ and $\mathcal{A}'_L = A_L \cup \{(i, i') : i \in L\} \cup \{(i', 0') : i \in L\}$; $v_{source} = 0$ and $v_{sink} = 0'$. Each arc $a = (i, j) \in A_L$ has a capacity resource consumption given by $q_a = d_j$, the other arcs in $\mathcal{A}'_L$ have zero consumption. Each vertex $i \in \mathcal{V}'_L$ has resource interval $[0, Q]$. The additional copies of the linehaul vertices in $\mathcal{G}'_L$ are introduced in order to be able to use path enumeration in it. Without them, the necessary condition to use enumeration defined in Pessoa et al. (2019b) would not be satisfied.

Define an integer variable x_a for each $a \in A$ (again, exactly the same arc variables defined in formulation $\mathcal{F}0$). In addition, there are two integer variables z_i, w_i for every linehaul customer $i \in L$. The formulation is:

$$\text{Min} \sum_{a \in A} c_a x_a \tag{B.5}$$

$$\text{S.t.} \sum_{a \in \delta^-(i)} x_a = 1, \quad i \in V^+, \tag{B.6}$$

$$z_i = w_i, \quad i \in L. \tag{B.7}$$

The number of paths from both $\mathcal{G}'_L$ and $\mathcal{G}_B$ in the solution is fixed to K. Each variable x_a, $a = (i, j) \in A_L$, is mapped to arc (i, j) in $\mathcal{A}'_L$. Each variable x_a, $a = (i, j) \in A_{LB}$, is mapped to arc (i', j) in $\mathcal{A}'_B$. Each variable x_a, $a = (i, j) \in A_B$, is mapped to arc (i, j) in $\mathcal{A}'_B$. A variable z_i, $i \in L$, is mapped to arc (i, i') in $\mathcal{A}'_L$. Finally, variables w_i, $i \in L$, is mapped to arc $(0', i')$ in $\mathcal{A}'_B$. Packing sets are defined on vertices, one packing set is defined for each vertex in the both graphs, except for the depot vertices. Branching is performed on the aggregation of x variables corresponding to opposite arcs and z variables. Enumeration is activated. Figure B.7 illustrates RCSP graphs $\mathcal{G}_L$ and $\mathcal{G}_B$, the consumptions and variables mapped to each arc are also depicted.

The Julia code corresponding to the above VRPB model is available on the VRPSolver webpage `vrpsolver.math.u-bordeaux.fr`.

References

Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, *115*, 351–385.

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

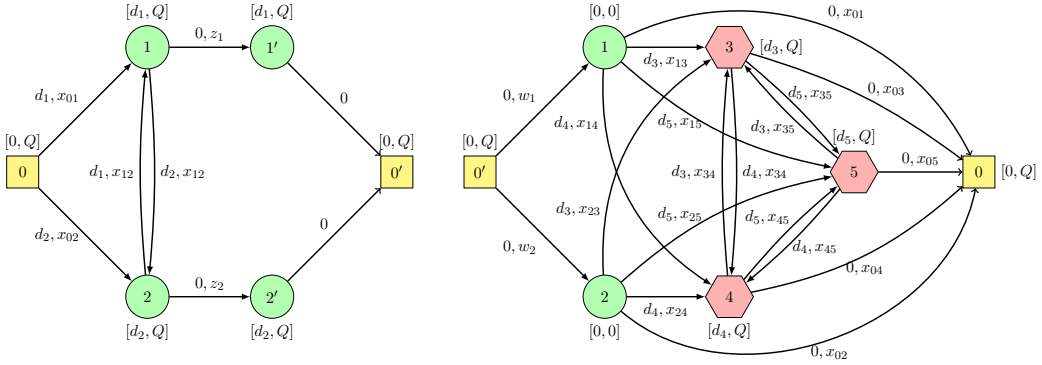


Figure B.7: RCSP graphs for the VRPSolver model of $\mathcal{F}2$

.

Brandão, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, *173*, 540 – 555.

Brandão, J. (2016). A deterministic iterated local search algorithm for the vehicle routing problem with backhauls. *TOP*, *24*, 445–465.

Contardo, C., & Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization*, *12*, 129 – 146.

Cuervo, D. P., Goos, P., Sörensen, K., & Arráiz, E. (2014). An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, *237*, 454 – 464.

Deif, I., & Bodin, L. (1984). Extension of the clarke and wright algorithm for solving the vehicle routing problem with backhauling. In *Proceedings of the Babson conference on software uses in transportation and logistics management* (pp. 75–96). Babson Park, MA.

Dunning, I., Huchette, J., & Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM Review*, *59*, 295–320.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*, 109–124.

Gajpal, Y., & Abad, P. (2009). Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, *196*, 102 – 117.

Gélinas, S., Desrochers, M., Desrosiers, J., & Solomon, M. M. (1995). A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, *61*, 91–109.

Goetschalckx, M., & Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, *42*, 39 – 51.

Granada-Echeverri, M., Toro, E., & Santa, J. (2019). A mixed integer linear programming formulation for the vehicle routing problem with backhauls. *International Journal of Industrial Engineering Computations*, *10*, 295–308.

Jacobs-Blecha, C., & Goetschalckx, M. (1992). *The Vehicle Routing Problem with Backhauls: Properties and Solution Algorithms*. Technical Report Georgia Tech Research Corporation.

Jepsen, M., Petersen, B., Spoorendonk, S., & Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, *56*, 497–511.

Koç, Ç., & Laporte, G. (2018). Vehicle routing with backhauls: Review and research perspectives. *Computers & Operations Research*, *91*, 79 – 91.

Laporte, G., Mercure, H., & Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, *16*, 33–46.

Lysgaard, J. (2003). *CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem*. Tech. Report Aarhus University, Denmark.

Mingozzi, A., Giorgi, S., & Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science*, *33*, 315–329.

Osman, I. H., & Wassan, N. A. (2002). A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, *5*, 263–285.

Pecin, D., Pessoa, A., Poggi, M., & Uchoa, E. (2017a). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, *9*, 61–100.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., & Santos, H. (2017b). Limited memory rank-1 cuts for vehicle routing problems. *Operations Research Letters*, *45*, 206 – 209.

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., & Prins, C. (2019). A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. *Annals of Operations Research*, *273*, 5–74.

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2018). Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, *30*, 339–360.

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2019a). A generic exact solver for vehicle routing and related problems. In A. Lodi, & V. Nagarajan (Eds.), *Integer Programming and Combinatorial Optimization* (pp. 354–369). Cham: Springer International Publishing.

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2019b). *A Generic Exact Solver for Vehicle Routing and Related Problems*. Cadernos do LOGIS 2019/2 Universidade Federal Fluminense.

Poggi, M., & Uchoa, E. (2014). Chapter 3: New exact algorithms for the capacitated vehicle routing problem. In P. Toth, & D. Vigo (Eds.), *Vehicle routing: problems, methods, and applications* chapter 3. (pp. 59–86). SIAM.

Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *171*, 750 – 775.

Sadykov, R., Uchoa, E., & Pessoa, A. (2017). *A bucket graph based labeling algorithm with application to vehicle routing*. Technical Report Rep. L-2017-7, Cadernos do LOGIS-UFF, Niterói, Brazil.

Subramanian, A., Uchoa, E., & Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, *40*, 2519 – 2531.

Toth, P., & Vigo, D. (1996). A heuristic algorithm for the vehicle routing problem with backhauls. In L. Bianco, & P. Toth (Eds.), *Advanced Methods in Transportation Analysis* (pp. 585–608). Berlin, Heidelberg: Springer Berlin Heidelberg.

Toth, P., & Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, *31*, 372–385.

Tütüncü, G. Y. (2010). An interactive GRAMPS algorithm for the heterogeneous fixed fleet vehicle routing problem with and without backhauls. *European Journal of Operational Research*, *201*, 593 – 600.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., & Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, *257*, 845 – 858.

Vanderbeck, F., Sadykov, R., & Tahiri, I. (2018). BaPCod – a generic branch-and-price code. Available at https://realopt.bordeaux.inria.fr/?page_id=2.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, *234*, 658 – 673.

Wassan, N. (2007). Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. *Journal of the Operational Research Society*, *58*, 1630–1641.

Yano, C. A., Chan, T. J., Richter, L. K., Cutler, T., Murty, K. G., & McGettigan, D. (1987). Vehicle routing at quality stores. *INFORMS Journal on Applied Analytics*, *17*, 52–63.

Zachariadis, E. E., & Kiranoudis, C. T. (2012). An effective local search approach for the vehicle routing problem with backhauls. *Expert Systems with Applications*, *39*, 3174 – 3184.