Enhanced Branch-Cut-and-Price algorithm for Heterogeneous Fleet Vehicle Routing Problems

Artur Pessoa[a], Ruslan Sadykov[b], Eduardo Uchoa[a,*]

[a] *Universidade Federal Fluminense - Departamento de Engenharia de Produção*
Rua Passo da Pátria 156, Niterói - RJ - Brasil - 24210-240
[b] *INRIA Bordeaux – Sud-Ouest*
200 Avenue de la Veille Tour, 33405 Talence, France

Abstract

This paper considers a family of Vehicle Routing Problem (VRP) variants that generalize the classical Capacitated VRP by taking into account the possibility that vehicles differ by capacity, costs, depot allocation, or even by the subset of customers that they can visit. This work proposes a branch-cut-and-price algorithm that adapts advanced features found in the best performing exact algorithms for homogeneous fleet VRPs. The original contributions include: (i) the use of Extended Capacity Cuts, defined over a pseudo-polynomially large extended formulation, together with Rank-1 Cuts, defined over the Set Partitioning Formulation; (ii) the concept of vehicle-type dependent memory for Rank-1 Cuts; and (iii) a new family of lifted Extended Capacity Cuts that takes advantage of the vehicle-type dependent route enumeration. The algorithm was extensively tested in instances of the literature and was shown to be significantly better than previous exact algorithms, finding optimal solutions for many instances with up to 200 customers and also for some larger instances. A new set of set of benchmark instances is also proposed.

Keywords: Routing; Column Generation; Cutting Planes

[*] Corresponding author: Tel. +55 21 2629-5709.
Email addresses: pessoa@producao.uff.br (Artur Pessoa), ruslan.Sadykov@inria.fr (Ruslan Sadykov), uchoa@producao.uff.br (Eduardo Uchoa)

1. Introduction

This paper deals with the Heterogeneous Fleet Vehicle Routing Problem (HFVRP), which can be defined as follows. Let $G = (V, A)$ be a complete directed graph where $V = \{0, 1, \ldots, n\}$ is a set composed of $n + 1$ vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. The vertex 0 denotes the depot, where the vehicle fleet is located; while the set $V' = V \setminus \{0\}$ is composed of the remaining vertices, they represent the n customers. Each customer $i \in V'$ has a positive integer demand q_i. The fleet is composed of m different types of vehicles, with $M = \{1, \ldots, m\}$. For each $u \in M$, there are K_u available vehicles, each with an integer capacity Q_u. Assume w.l.o.g. that $Q_1 \leq Q_2 \leq \ldots \leq Q_m$. Every vehicle type is also associated with a fixed cost denoted by f_u. Finally, for each arc $(i, j) \in A$ and $u \in M$ there are associated costs c^u_{ij}. In many cases, c^u_{ij} is defined as $r_u d_{ij}$, where d_{ij} is the distance between the vertices (i, j) and r_u is a type-dependent travel cost per distance unit. The objective is to determine the set of routes that minimize the sum of fixed and travel costs in such a way that: (i) every route starts and ends at the depot and is associated to a vehicle type such that the sum of the demands of the clients in the route does not exceed the vehicle's capacity; (ii) each customer is visited by exactly one route; (iii) the number of routes associated to a vehicle type does not exceed its availability. This problem is $\mathcal{NP}$-hard since the classical CVRP corresponds to the special case with $m = 1$. Some authors reserve the name HFVRP for the cases where there are potentially constraining limits for the number of vehicles available (see Baldacci et al. (2008a)). For those authors, when $K_u = \infty$ for all $u \in M$, the problem is known as Fleet Size and Mix (FSM). In this paper we use the name HFVRP in the broad sense, including FSM as a special case.

A closely related variant is the Multi-Depot VRP (MDVRP), where vehicles have identical capacity and cost, but differ by being attached to different depots. It can be easily modeled as a HFVRP by associating each depot to a vehicle type and setting costs c^u_{0j} for leaving and c^u_{i0} for entering the depot that depend on $u \in M$. The Site Dependent VRP (SDVRP), a variant

where vehicles differ only by the subset of the customers that they can visit, is also easily modeled as a HFVRP by setting infinity costs for c_{ij}^u if vehicle type u can not visit either i or j. Therefore, those problems can be solved as special cases of the HFVRP.

The HFVRP is a very important problem, since fleets are likely to be heterogeneous in most practical situations. According to Hoff et al. (2010), even when an acquired fleet is homogeneous, it usually become heterogeneous over the time, as newer vehicles are incorporated. Moreover, from both tactical and operational point of view, it is usually advantageous to have a mixed vehicle fleet. This increases the flexibility in terms of distribution planning and can bring significant economies. For example, it can avoid the waste of having to send a truck to perform a set of deliveries that could also be performed by a much cheaper van.

Many heuristics have been proposed for HFVRP (see Penna et al. (2017) for recent references). On the other hand, the literature on exact methods is not large. The first lower bounds for the HFVRP were introduced by Golden et al. (1984). Other bounding schemes were proposed later. Westerlund et al. (2003) give an extended formulation and some valid cuts. Yaman (2006) performed a deep theoretical analysis of several different formulations and proposed new valid cuts. Choi and Tcha (2007) produced better lower bounds using column generation. Baldacci et al. (2008a) proposed some valid inequalities as well as a two-commodity Mixed Integer Programming (MIP) formulation. All those bounds and formulations were not good enough for producing efficient exact algorithms, at that moment instances with only 20 customers proposed in Golden et al. (1984) (and those modified by Taillard (1999)) were listed as open.

The first reasonably efficient exact algorithm for HFVRP was proposed in Pessoa et al. (2007) and Pessoa et al. (2009) and could solve almost all Golden-Taillard instances with up to 75 customers. That algorithm was a Branch-Cut-and-Price (BCP) that combined the techniques used in a previous CVRP algorithm (Fukasawa et al. (2006)) with new cuts over a pseudo-polynomially large extended formulation. Finally, the algorithm in Baldacci and Mingozzi (2009) adapts ideas from the CVRP algorithm in Baldacci

et al. (2008b), including cuts over the set partitioning formulation and route enumeration instead of branching. It could solve most Golden-Taillard instances with up to 100 customers. Those are still the best published results for an exact HFVRP algorithm. Good results on MDVRP and SDVRP instances were also obtained by that algorithm.

Recently, there were very significant improvements in the exact algorithms for solving homogeneous fleet VRPs. For the classical CVRP, a new BCP by Pecin et al. (2017b) combining and enhancing previous techniques from several authors with a powerful new idea, the concept of limited-memory cuts over the set partitioning formulation, could more than double the size where instances are expected to be solved. In fact, as can be checked in the updated results kept in CVRPLIB (Uchoa et al. (2017)), classical instances with up to 420 customers are already solved. Before that work, the largest solved CVRP instance had 150 customers. A related BCP algorithm, using the improved concept of limited-memory over arcs (Pecin et al. (2017a)), could also obtain very good results on the solution of the VRP with Time Windows (VRPTW).

The goal of this work is to adapt the techniques used in those recent works in order to propose a new state-of-the-art algorithm for the HFVRP. This is not a simple task. Problems with heterogeneous fleet are inherently more difficult, so those techniques need to be sharpened to take full advantage of the new context. This is the first BCP algorithm that combines cuts over the set partitioning formulation with cuts over the pseudo-polynomial extended formulation. We could verify experimentally that the separation of Extended Capacity Cuts (ECCs) was decisive for solving some hard instances.

The remainder of this paper is organized as follows. Section 2 presents our pseudo-polynomial original formulation and the corresponding set partitioning formulation that can be obtained from it by a Dantzig-Wolfe decomposition. Section 3 describes the Extended Capacity Cuts and the limited-memory Rank 1 Cuts, introducing the concept of vehicle-type dependent memory. Section 4 presents the overall BCP algorithm, describing the proposed hybridization of route enumeration with strong branching. Section

5 presents the computational results obtained, making comparisons with those reported in the literature. Additional experiments on newly created instances are also reported. Section 6 presents the concluding remarks of this work. The appendix presents instance-by-instance detailed computational results.

2. Formulation

We introduce a HFVRP formulation that has a pseudo-polynomially large number of variables and constraints. For each vehicle type $u \in M$, define an acyclic graph $\mathcal{N}_u = (\mathcal{V}_u = T_u \cup O_u, \mathcal{A}_u = A_u^1 \cup A_u^2 \cup A_u^3)$ with node-sets $T_u = \{(i,l,u) : i \in V', l = q_i, \ldots, Q_u\}$ and $O_u = \{(0,l,u) : l = 0, \ldots, Q_u\}$. The arc-set A_u^1 is composed of all possible arcs that go from a node (i,l,u) in T_u to a node $(j, l+q_j, u)$ also in T_u, they are represented as tuples of format (i,j,l,u). More formally, $A_u^1 = \{(i,j,l,u) = ((i,l,u),(j,l+q_j,u)) : (i,l,u) \in T_u; j \in V', l+q_j \leq Q_u\}$. The arcs in arc-set A_u^2 go from node $(0,0,u)$ to a node in $\{(i,q_i,u)) : i \in V'\}$ and are represented as tuples of format $(0,i,0,u)$. Finally, the arcs in A_u^3 go from a node $(i,l,u) \in T_u$ to node $(0,l,u)$, they are represented as $(i,0,l,u)$. For any set $S \subseteq \mathcal{V}_u$, $\delta^-(S) \subseteq \mathcal{A}_u$ represents the set of arcs entering S; $\delta^+(S) \subset \mathcal{A}_u$ represents the leaving arcs. For each $u \in M$ and each $i \in V'$, define the subset of nodes associated to i as $T_u^i = \{(i',l,u) \in T_u : i' = i\}$. For each $u \in M$ and for each $a = (i,j,l,u) \in \mathcal{A}_u$ define a binary variable x_a (also notated as x_{ijlu}) to indicate whether arc $(i,j) \in A$ belongs to a route of vehicle type u such that the total demand of i and of the vertices preceding it in the route is exactly l. Under the interpretation that the demands are being collected from the customers, $x_{ijlu} = 1$ would mean that a vehicle of type u that departed from the depot empty goes from i to j carrying a load of l units. Define $\hat{c}_{ij}^u = c_{ij}^u + f_u$, if (i,j,l,u) belongs to A_u^2, $\hat{c}_{ij}^u = c_{ij}^u$ otherwise. The formulation F1 follows:

$$\text{(F1) min} \quad \sum_{u \in M} \sum_{a=(i,j,l,u) \in \mathcal{A}_u} \hat{c}_{ij}^u x_a \tag{1}$$

subject to

$$\sum_{u \in M} \sum_{a \in \delta^-(T_u^i)} x_a = 1, \qquad \forall i \in V', \tag{2}$$

$$\sum_{a \in A_u^2} x_a \leq K_u, \qquad \forall u \in M, \tag{3}$$

$$\sum_{a \in \delta^-(\{v\})} x_a - \sum_{a \in \delta^+(\{v\})} x_a = 0, \qquad \forall u \in M, \forall v \in T_u, \tag{4}$$

$$x \geq 0, \tag{5}$$

$$x \text{ integer}. \tag{6}$$

Equations (2) state that each customer should be visited exactly once. Equations (3) limit the maximum number of routes for each vehicle type. Under the demand collection interpretation, flow-conservation Equations (4) mean that if a vehicle leaves a customer i with load l, the same vehicle must have arrived in i with load $l - q_i$. Formulation F1 has $O(mn^2Q_m)$ variables and $O(mnQ_m)$ constraints, so it can not be directly used unless Q_m is very small.

For each $u \in M$, let Ω_u be the set of all paths in $\mathcal{N}_u$ from vertex $(0, 0, u)$ to another vertex in O_u. Those paths are called q-routes (Christofides et al. (1981)). When projected into the original graph G, the q-routes correspond to walks starting and ending at the depot with total demand at most Q_u. A customer may be visited more than once in a q-route, but its demand is counted again in each additional visit. For each $u \in M$, let b_a^p be a binary coefficient indicating whether arc $a \in \mathcal{A}_u$ appears or not in route $p \in \Omega_u$. For each $u \in M$ and each $p \in \Omega_u$ define a non-negative variable λ_p. The following extended formulation (containing both x and λ variables) is equivalent to F1:

$$\text{(F1')} \ \min \qquad \sum_{u \in M} \sum_{a=(i,j,l,u) \in \mathcal{A}_u} \hat{c}_{ij}^u x_a \tag{7}$$

subject to

$$\sum_{u \in M} \sum_{a \in \delta^-(T_u^i)} x_a = 1, \qquad \forall i \in V', \tag{8}$$

$$\sum_{a \in A_u^2} x_a \leq K_u, \qquad \forall u \in M, \tag{9}$$

$$\sum_{p \in \Omega_u} b_a^p \lambda_p - x_a = 0, \qquad \forall u \in M, \forall a \in \mathcal{A}_u, \tag{10}$$

$$x, \lambda \geq 0, \tag{11}$$

$$x \text{ integer.} \tag{12}$$

Relaxing the integrality and eliminating the x variables using Equations (10), we obtain the following linear relaxation of a Set Partitioning Formulation, having an exponential number of variables but only $n+m$ contraints:

$$\text{(SPF) min} \quad \sum_{u \in M} \sum_{p \in \Omega_u} \Big(\sum_{a=(i,j,l,u) \in \mathcal{A}_u} b_a^p \hat{c}_{ij}^u \Big) \lambda_p = \sum_{p \in \Omega} \hat{c}_p \lambda_p \tag{13}$$

subject to

$$\sum_{u \in M} \sum_{p \in \Omega_u} \Big(\sum_{a \in \delta^-(T_u^i)} b_a^p \Big) \lambda_p = \sum_{p \in \Omega} h_i^p \lambda_p = 1 \quad \forall i \in V', \tag{14}$$

$$\sum_{p \in \Omega_u} \Big(\sum_{a \in A_u^2} b_a^p \Big) \lambda_p = \sum_{p \in \Omega_u} \lambda_p \leq K_u, \qquad \forall u \in M, \tag{15}$$

$$\lambda \geq 0. \tag{16}$$

The cost $\hat{c}_p$ of a variable λ_p in (13) is given by the sum of the cost of the arcs in p, the coefficients h_i^p in (14) count how many times arcs of p enter in each vertex i and $\Omega = \cup_{u \in M} \Omega_u$. SPF could be obtained directly by a Dantzig-Wolfe decomposition of F1, by noting that the q-routes correspond to the extreme rays of the unbounded polyhedron defined by (4) and (5). Anyway, SPF can be solved by column generation. The pricing subproblems, one for each $u \in M$, require finding the most negative q-route in each $\mathcal{N}_u$, with respect to arc reduced costs defined as $\bar{c}_{ijlu} = c_{ijlu} - \pi_j - \nu_u$, where π_j is the dual variable of the constraint from (14) corresponding to j (π_0 is defined as 0) and ν_u is the dual variable of the constraint from (15) corresponding to u. This can be done by dynamic programming, in $O(|\mathcal{A}_u|) = O(n^2.Q_u)$ time.

Any additional cut over the x variables can be translated into an equivalent cut over the λ variables using Equations (10) and introduced in the SPF. Those cuts do not change the complexity of the pricing subproblems, the effect of their dual variables is only changing the calculation of the arc reduced costs. Therefore, according the classification proposed in Poggi de Aragão and Uchoa (2003) those cuts are *robust*.

A stronger relaxation would be obtained if the Ω sets are redefined in order to only contain elementary paths, those where a customer can not be visited more than once. However, that makes the pricing subproblems strongly $\mathcal{NP}$-hard and indeed hard to be solved in practice. The known relaxation with best trade-off between pricing time and lower bound quality is based in *ng*-routes, as proposed in Baldacci et al. (2011). Let $N_i \subseteq V'$ be the neighborhood of i, typically defined in order to contain the closest customers to i. An *ng*-route can only revisit a customer i if it passes first by another customer j such that $i \notin N_j$. When compared to q-routes, *ng*-routes increase the number of states in the dynamic programming used in the pricing by a factor that is bounded by $2^{N_{max}-1}$, where N_{max} is the size of the larger neighborhood. In many instances, reasonably small neighborhoods (say, $N_{max} = 8$) already provide bounds that are not much worse than those that would be obtained by pricing elementary routes.

3. Cuts

The lower bounds obtained by SPF, even if Ω only contains near-elementary paths, are not good enough for building an efficient branch-and-price algorithm. The formulation should be strengthened by cuts. In this work we use two types of cuts.

3.1. Extended Capacity Cuts

For each arc $(i, j) \in A$, define an aggregated binary arc variable $x_{ij} = \sum_{u \in M} \sum_{l=1}^{Q_u} x_{ijlu}$. Let $S \subseteq V'$ be a subset of the customers, define $q(S) = \sum_{i \in S} q_i$ as its total demand. The value $r(S) = \lceil q(S)/Q_m \rceil$ is a valid lower bound on the number of vehicles that must visit S. Therefore, the following

Rounded Capacity Cut (RCC) (Laporte and Norbert (1983)) is valid:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq r(S), \tag{17}$$

RCCs are known to be very effective on homogeneous fleet CVRP. However, they are not much effective on typical HFVRP instances. This happens because $r(S)$, calculated considering the largest vehicle capacity Q_m, can be a poor bound on the actual number of vehicles visiting S.

For each arc $(i,j) \in A$ and $l = 1, \ldots, Q_m$, define an aggregated binary arc-load variable $x_{ijl} = \sum_{u\in M:Q_u\leq l} x_{ijlu}$. For each set $S \subseteq V'$, the sum of the load of the vehicles leaving S minus the sum of the load of the vehicles entering S is equal to the total demand collected from S:

$$\sum_{(i,j)\in\delta^+(S)} \sum_{l=1}^{Q_m} l\, x_{ijl} - \sum_{(i,j)\in\delta^-(S)} \sum_{l=1}^{Q_m-1} l\, x_{ij} = l(S). \tag{18}$$

Those equalities are always satisfied by the linear relaxation of SPF (a solution over its λ variables is translated into an equivalent solution over the x variables using (10)). However, Equalities (18) can be used as a rich source of cuts, the so-called Extended Capacity Cuts (ECCs). As done in Pessoa et al. (2009), in this work we only separate ECCs obtainable by integer rounding, those of format:

$$\sum_{(i,j)\in\delta^+(S)} \sum_{l=1}^{Q_m} \lceil sl \rceil\, x_{ijl} - \sum_{(i,j)\in\delta^-(S)} \sum_{l=1}^{Q_m-1} \lfloor sl \rfloor\, x_{ijl} \geq \lceil sq(S) \rceil, \tag{19}$$

where s, $0 < s < 1$, is a rational multiplier. As discussed in (Uchoa et al., 2008), at most $0.3Q_m^2$ distinct multipliers need to be tried. By the way, an RCC is the particular ECC obtained with multiplier $s = 1/Q_m$. General ECCs can be effective in typical HFVRP instances, even when RCCs are not.

The separation of ECCs used in this work is built upon the heuristic procedure available in CVRPSEP (Lysgaard (2003)) for separating RCCs.

For each candidate set S identified by that procedure, but such that the RCC is not violated, we test whether the ECCs obtained with other multipliers are violated.

3.2. Rank-1 Cuts with Subproblem (Vehicle-type) Dependent Memory

Jepsen et al. (2008) introduced the Subset Row Cuts (SRCs), a family of cuts defined over the SPF. Those cuts were generalized in Petersen et al. (2008) as follows: Given $S \subseteq V'$ and a positive multiplier s_i for each $i \in S$, the following Rank-1 Cut (R1C):

$$\sum_{p \in \Omega} \left\lfloor \sum_{i \in S} s_i h_i^p \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in S} s_i \right\rfloor \tag{20}$$

can be obtained by a Chvátal-Gomory rounding of the corresponding rows in (14). The original SRCs correspond to the R1Cs obtained with identical multipliers of format $s_i = 1/Z$, where Z is a positive integer, for all $i \in S$. Recently, Pecin et al. (2017c) performed an investigation of the Set Partitioning polyhedron to determine the best possible sets of multipliers for R1Cs with up to 5 rows.

R1Cs are strong cuts and can reduce substantially the integrality gaps of the SPF. However, since they are defined directly over the SPF variables, those cuts are non-robust: each added cut changes the pricing subproblem and makes it harder. Essentially, the dynamic programming labeling algorithms used to solve the subproblem should have an additional dimension for each active cut. As a consequence, cut separation should be stopped before the pricing becomes far too expensive and the full potential gap reductions cannot be achieved.

In order to mitigate the negative impact of those cuts in the pricing subproblem, Pecin et al. (2017b) proposed the concept of cuts with limited-memory, based on the following observation:

- A cut can be weakened without changing its effect (on the bounds), as long as the variables with weakened coefficients take zero value.

More specifically, a R1C over a customer subset $S \subseteq V'$ and with multiplier vector s, is associated to a memory node set $M(S, s)$, $S \subseteq M(S, s) \subseteq V'$. The

idea (somehow inspired by the ng-route relaxation) is that, when a route $p \in \Omega$ leaves the memory set $M(S, s)$, it may "forget" previous visits to nodes in S, yielding a coefficient for λ_p in the cut that may be smaller than the original $\lfloor \sum_{i \in S} s_i h_i^p \rfloor$ coefficient. The memory set is defined, after the separation of a violated R1C, as a minimal set that preserves the coefficients of the variables λ_p with positive value in the current linear relaxation solution. Of course, variables priced later may not have the best possible coefficients in previous cuts, but this can be corrected in the next separation rounds. Eventually, limited-memory R1Cs achieves the same bounds that would be obtained by the original R1Cs. Yet, those limited-memory R1Cs, while still non-robust, are much better handled by the labeling algorithm used for solving the pricing subproblem: an additional dimension is needed only for the labels corresponding to a partial path ending in $M(S, s)$. This may be very advantageous computationally if $|M(S, s)| << n$, as usually happens. The limited-memory concept was the main innovation that enabled major improvements upon previous algorithms, and ultimately the exact solution of classical CVRP instances with up to 360 customers in Pecin et al. (2017b) and 420 customers in Pecin et al. (2017c).

In order to further reduce the impact of R1Cs, Pecin et al. (2017a) introduced the generalized concept of memory arc sets. In that case, a R1C is associated with a memory set $AM(S, s) \subseteq A$; a route that uses an arc not contained in $AM(S, s)$ may "forget" previous visits to nodes in S. Memory arc sets allow a sharper definition of which λ variables need to keep their original coefficients in a R1C, leading to even less impact in the pricing. The drawback of arc memory (with respect to node memory) is poorer convergence, in the sense that more separation rounds may be required until the correct memory sets are found. Extensive experiments on VRPTW made clear that node memory is better for the easier instances, but arc memory can be much better for the hardest ones, those where avoiding an exponential explosion in the pricing time is critical.

In this paper we take advantage of the HFVRP context for an additional generalization and sharpening of the R1C memory definition. Consider a R1C over a customer subset $S \subseteq V'$ and with an $|S|$-dimensional vector of

Algorithm 1 Computing coefficient $\alpha(S, s, AM(S, s, u), p)$

$\alpha \leftarrow 0$, $state \leftarrow 0$
for every arc (i, j) in route p (in order) **do**
    if $(i, j) \notin AM(S, s, u)$ **then**
        $state \leftarrow 0$
    if $j \in S$ **then**
        $state \leftarrow state + s_j$
        if $state \geq 1$ **then**
            $\alpha \leftarrow \alpha + 1$, $state \leftarrow state - 1$
return α

multipliers s. The corresponding Rank-1 Cut with Subproblem Dependent Memory, with associated memory sets $AM(S, s, u)$, for each $u \in M$, is defined as:

$$\sum_{u \in M} \sum_{p \in \Omega_u} \alpha(S, s, AM(S, s, u), p) \lambda_p \leq \left\lfloor \sum_{i \in S} s_i \right\rfloor, \tag{21}$$

where coefficient $\alpha(S, s, AM(S, s, u), p)$ of variable λ_p, $p \in \Omega_u$, is computed as described in Algorithm 1. This algorithm can be explained as follows. Whenever a route p visits a node $j \in S$, the multiplier s_j is added to the *state* variable. When $state \geq 1$, *state* is decremented and α is incremented. If $AM(S, s, u) = A$, the procedure would always return $\lfloor \sum_{i \in S} s_i h_i^p \rfloor$ and the limited-memory cut would be equivalent to the original cut. On the other hand, if $AM(S, s, u) \subset A$, it may happen that p uses an arc $(i, j) \notin AM(S, s, u)$ when $state > 0$, causing *state* to be reset to zero and "forgetting" some previous visits to nodes in S. In this case, the returned coefficient may be less than the original coefficient.

The potential advantage of using m different memories for each R1C is reducing pricing time. Suppose that the minimal sets found after the separation of a certain R1C are $AM(S, s, u)$, $u \in M$. The impact of that R1C in the solution of the pricing subproblem corresponding to vehicle type u depends only of $AM(S, s, u)$, the smaller that set is, the better. Suppose that a single-memory-per-cut scheme is used instead. The minimal set $AM(S, s)$ found after the separation would be equal to $\cup_{u \in M} AM(S, s, u)$. In that

case, all the subproblems would be impacted by the same, probably larger, set $AM(S, s)$. The drawback of subproblem dependent memory is poorer convergence, more separation rounds to achieve the same final bound.

Limited-memory R1C separation was implemented as follows. All subsets of customers with $|S| = 3$ are candidate sets. There would be too many such candidates for $|S| \in \{4, 5\}$. In those cases, the procedure considers only compact sets S, in the sense that the distance between any pair of customers in S does not exceed a predefined maximum distance. Moreover, a local search procedure is used for selecting candidate sets among the compact sets. For each candidate set S and multiplier vector s (among those listed in Pecin et al. (2017c)), we check if the R1C corresponding to S and s, without memory limit, is violated. If so, arc memories $AM(S, s, u)$ of minimal size (actually near-minimal, since the arcs inside S are always included in the memories) such that the cut violation remains the same are determined. If there already exists a cut defined for the same subset S and vector s but with arc-memories $AM'(S, s, u)$, this cut is simply enhanced by enlarging its memory to $AM'(S, s, u) \cup AM(S, s, u)$, for $u \in M$. Otherwise, a new cut is added for S and s, with memories $AM(S, s, u)$.

4. Branch-Cut-and-Price Algorithm

The full Branch-Cut-and-Price (BCP) algorithm implemented solves each node by a combination of column and cut generation. The column generation for each subproblem $u \in M$ is performed by a bidirectional labeling algorithm similar to the one described in Pecin et al. (2017b), a truncated version of that algorithm is used as a heuristic in the first iterations, when finding routes with negative reduced cost is still easy. Labeling algorithms (like those described in that paper) are also used for fixing arcs in $\mathcal{A}_u$ and for trying to enumerate elementary routes in Ω_u with small reduced costs. Column generation convergence is improved with the dual stabilization techniques described in Pessoa et al. (2013). The search for the optimal integer solution hybridizes traditional branching with route enumeration, as will be described next.

4.1. Subproblem Enumeration and Lifting of Cuts

The column and cut generation algorithm in Baldacci and Mingozzi (2009) does not perform branching. Instead, after solving the root node, it tries to enumerate the sets of elementary routes $R_u \subseteq \Omega_u$, for each $u \in M$, with reduced cost smaller than the duality gap, the difference between the lower bound and the value of the best known feasible solution. It can be shown that routes not in $R = \cup_{u \in M} R_u$ can not appear in any improving solution. So, if R is not too large, those routes are used to produce a set partitioning problem that is solved by a generic MIP solver. Otherwise, when the limit on $|R|$ is exceeded, enumeration is aborted and the algorithm halts.

Contardo and Martinelli (2014) proposed another strategy in order to better profit from the route enumeration. The enumeration of the root node can be performed earlier, as soon as the resulting R has up to a few million routes, which are stored in a pool. It is not practical to produce a set partitioning problem with so many routes, since its solution would be far beyond the capability of current MIP solvers. So, the column and cut generation proceeds. However, instead of using a labeling algorithm, the pricing starts to be performed by inspection in the pool, which can be faster. Moreover, additional non-robust cuts have little impact in the inspection pricing time. Therefore, those cuts may be separated in a very aggressive way. This usually increases substantially the lower bound, allowing reductions in the pool size by fixing variables (that now are routes) by reduced costs. For example, they reported that the enumeration of a hard CVRP instance with a gap of 5 units produced a pool with 4M routes. The aggressive separation of SRCs then reduced the gap to 1.5 and the final pool only had 13K routes. The resulting set partitioning was easily solved, finishing the instance.

In this work we used a similar enumeration-to-pools scheme, but take advantage of the particular HFVRP characteristics. The size of each set R_u depends substantially on the current gap, but also depends a lot on the value of Q_u. This means that subproblems with smaller values of Q_u can be successfully enumerated much earlier, when gaps are only reasonable. On the other hand, subproblems with larger values of Q_u can only be enumerated later, when the gaps are very small. Therefore, at a given point of the

solution of a BPC node, a subproblem $u \in M$ can be in one of the following possible modes:

1. Enumerated - The set R_u could already be computed and is stored in a pool. The pricing corresponding to u is performed by inspection. Every time the gap decreases, reduced cost fixing is used for eliminating routes from R_u .

2. Non-enumerated - The set R_u is too large and could not be computed. The pricing corresponding to u is performed by the labeling algorithm. The arc fixing algorithm is called every time the gap decreases by 15%, with respect to the gap in the last arc fixing call. Fixing eliminates arcs from $\mathcal{A}_u$ in order to speedup subsequent calls to the pricing.

A node is only finished by the MIP solver when all subproblems are enumerated and $|R_u| < 5000$, for all $u \in M$. However, even when part of the subproblems are enumerated it is possible to strengthen the cuts:

- Limited-memory Rank-1 Cuts - When subproblem u goes to the enumerated mode, the memories $AM(S, s, u)$ for all sets S and multiplier vectors s, can be augmented to A.

- Extended Capacity Cuts - Let $EM \subseteq M$ be the subset of subproblems in enumerated mode. An Extended Capacity Cut with format (19), with set S and multiplier s, can be lifted to:

$$\sum_{u \in EM} \sum_{p \in R_u} \lceil s\, q(S,p) \rceil \lambda_p + \sum_{u \in M \setminus EM} \Big(\sum_{(i,j) \in \delta^+(S)} \sum_{l=1}^{Q_u} \lceil sl \rceil\, x_{ijlu} - \sum_{(i,j) \in \delta^-(S)} \sum_{l=1}^{Q_u - 1} \lfloor sl \rfloor\, x_{ijlu} \Big) \geq \lceil s\, q(S) \rceil, \tag{22}$$

where $q(S,p)$ is the total demand actually delivered by route p to customers in set S. Those lifted Extended Capacity Cuts are a rich family of strong cuts. For example, the particular case where $EM =$

M and $s = Q_m$ yields cuts equivalent to the Strong Capacity Cuts proposed in Baldacci et al. (2008b).

4.2. Strong Branching

Three different kinds of branching can be performed, all of them can be expressed as constraints over aggregations of original variables x, so they do not affect the pricing:

1. Branching on the total number of routes used by a vehicle type $u \in M$, $\sum_{a \in A_u^2} x_a$;
2. Branching on the assignment of a customer $i \in V'$ to a vehicle type $u \in M$, $\sum_{a \in \delta^-(R_u^i)} x_a$;
3. Branching on the "edges" of the original graph, $\sum_{u \in M} \sum_{l=1}^{Q_u} (x_{ijlu} + x_{jilu})$.

There is no predefined priority, we let the strong branching mechanism choose among branching candidates from those three kinds.

The hierarchical strong branching procedure, inspired by those found in Røpke (2012) and Pecin et al. (2017b), has the following phases:

- The Phase Zero performs the first selection of $\min\{100, TS(v)\}$ branching candidates, where $TS(v)$ is an estimative of the size of the subtree rooted in v based on the node gap and the average bound improvements obtained in previous branchings, $TS(v) = \infty$ for the root node. Up to half of the candidates are taken from the history of previous calls to the strong branching procedure, favoring candidates that already obtained good scores.

- The Phase One performs a quick evaluation of each candidate by solving the current restricted Master LP twice, adding the constraint corresponding to each child node. Column and cut generation are not performed. The candidates are scored by the product rule Achterberg (2007) and the $\min\{3, \lceil TS(v)/10 \rceil\}$ best candidates go to Phase Two.

- Phase Two performs more precise evaluations of each candidate, doing heuristic column generation (but no cut generation) on both child nodes. The scores are also based on the product rule.

The evaluations of Phase 1 and 2 are not only used to select the best candidate for the current branching, they are stored in tables (the branching history) for subsequent use in the Phase Zero. The whole procedure is guided by the principle that the strong branching effort in a node should depend on the expected subtree size. The rationale is the following. If $TS(v)$ is large, even a small improvement in that branching will compensate the cost of a more precise evaluation of several candidates. On the other hand, if $TS(v)$ is small, the branching should be fast, relying on the historical data and on the rough evaluations of Phase One. The estimation of $TS(v)$ is done by the model proposed in Kullmann (2009).

As mentioned before, our BCP uses a hybrid strategy. After each call to the arc fixing by reduced cost, it tries to enumerate each subproblem not yet in the enumerated mode. However, the strong branching can still be performed after all subproblems are successfully enumerated. This happens when some set of routes R_u, $u \in M$, is too large (we use 5000 routes as the limit) to allow a quick solution by the MIP solver. Of course, in both children of an enumerated node the pricing will continue to be done by inspection in the pools.

In spite of the sophisticated limited-memory mechanisms used, R1Cs are still non-robust and may make the pricing by labeling too expensive. Therefore, cut separation is stopped in any node if the running time of a single call to the labeling exceeds 6 seconds (unless all subproblems are in the enumerated mode). Cut separation may also be stopped in any node by tailing-off, if the last 3 separation rounds did not reduce the gap by at least 2%. In some rare cases, the separation of an additional round of R1Cs makes a single call to the labeling algorithm to exceed 12 seconds. In that situation, fearing that a combinatorial explosion may be happening, the labeling algorithm is aborted and the node *rolls-back to the state before the last round of cuts* and branching is performed.

5. Computational results

The algorithm described in this paper was coded in C++ and compiled with GCC 5.3.0. BaPCod package by Vanderbeck et al. (2017) was used to handle the Branch-Cut-and-Price framework. IBM CPLEX Optimizer version 12.6.0 was used as the LP solver in column generation and as the IP solver to solve the set partitioning problem with enumerated columns. The labeling algorithms are similar to those described in Pecin et al. (2017b). The experiments were run on a 2 Deca-core Ivy-Bridge Intel Xeon E5-2670 v2 server running at 2.50 GHz. The 128 GB of available RAM was shared between 8 copies of the algorithm running in parallel on the server. Each instance is solved by one copy of the algorithm using a single thread.

We considered the following existing instances from the literature.

HFVRP instances All Golden-Taillard instances with 50–100 customers considered by Baldacci and Mingozzi (2009) were used. Those instances have from 3 to 6 vehicle types and are are divided into classes, we kept the nomenclature used in that paper:

- HVRP. Referred as FIX-VAR class in Choi and Tcha (2007) and Pessoa et al. (2009). For each $u \in M$, there are positive fixed costs f_u, type-dependent travel costs r_u and potentially constraining limits K_u.
- FSMF. Referred as FIX class in those previous works. Obtained from HVRP by setting $K_u = \infty$ and $r_u = 1.0$, for each $u \in M$.
- FSMD. Referred as VAR class in those previous works. Obtained from HVRP by setting $K_u = \infty$ and $f_u = 0$, for each $u \in M$.
- FSMFD. Obtained from HVRP by setting $K_u = \infty$, for each $u \in M$.
- HD. Obtained from HVRP by setting $f_u = 0$ for each $u \in M$.

There are 8 Golden-Taillard instances per class, numbered from 13 to 20, making in total 40 instances. In addition, we used larger instances with 100–199 customers proposed by Brandão (2011) named

N1–N4. Each name involves two instances, one belonging to the class FSMD presented above, and another one to the class HD. Thus, in total there are 8 instances. Note that the instances by Brandão (2011) named N5 were not considered because of large vehicle capacities. The best known solutions for all the instances above are taken from Subramanian et al. (2012) and from Subramanian (2016).

SDVRP instances We consider 23 standard instances by Nag et al. (1988) and by Chao et al. (1999) with 27–324 customers, named p01–p23. The best known solutions are taken from Cordeau and Maischberger (2012).

MDVRP instances We consider 11 standard instances by Cordeau et al. (1997) with 50–360 customers, named p01–p07, p12, p15, p18, and p21. In addition we use 8 instances with 151–200 customers proposed by Baldacci and Mingozzi (2009). The best known solutions are taken from Cordeau and Maischberger (2012) and from Baldacci and Mingozzi (2009).

5.1. Overall results on existing instances

In total 90 existing instances are considered. Among all these instances, only two were not solved to optimality by our algorithm within the time limit of 36 hours. These are the largest SDVRP instances p17 and p18 with 270 and 324 customers. So, the largest solved HFVRP instances have 199 customers, the largest solved SDVRP instance has 216 customers, and the largest solved MDVRP instance has 360 customers (although only 2 instances with more than 200 customers were considered). The detailed results for all those instances are presented in Appendix A.

Using our algorithm, we managed to improve the best known solutions for 10 instances. In Table 1, for each such instance, we give the best known in the literature solution value, the reference to the paper and the new solution value. Here [SPUS12] stands for Subramanian et al. (2012), [S16] stands for Subramanian (2016), [BM09] stands for Baldacci and Mingozzi (2009), and [CM12] stands for Cordeau and Maischberger (2012).

Problem	Instance	Previous BKS	Reference	Improved value
HVRP	BrandaoN1fsmd	2212.77	[SPUS12]	2211.63
HVRP	BrandaoN1hd	2234.13	[S16]	2233.90
HVRP	BrandaoN2fsmd	2823.75	[SPUS12]	2810.20
HVRP	BrandaoN2hd	2859.82	[S16]	2851.94
HVRP	c100_20fsmf	4032.81	[SPUS12]	4029.61
HVRP	c100_20hvrp	4761.26	[SPUS12]	4760.68
MDVRP	n200-k16-3-80	1757.86	[BM09]	1756.48
SDVRP	p16	3393.55	[CM12]	3393.31
SDVRP	p18	4751.27	[CM12]	4747.75[1]
SDVRP	p21	1263.71	[CM12]	1260.01

[1] optimality is not proved, other values are optimal

Table 1: Instances with improved best known solution values

5.2. Comparison of algorithm variants

Here we compare the reference variant of our algorithm with its variants in which we remove or change one of its ingredients. Namely we test

- the variant without Extended Capacity Cuts,
- the variant in which Rank-1 Cuts limited arc memory is changed to node memory,
- the variant in which Rank-1 Cuts limited arc memory is subproblem independent.

For proper comparison all algorithm variants were run with the initial upper bound equal to the optimum solution value (or with the best known upper bound if optimal solution is not known). Aggregated results are presented in Table 2: root gap, root time (in seconds), number of active ECCs and R1Cs in the end of the root node, number of nodes in the search tree, number of solved instances and total running time (in seconds) over the instances solved by the variant. The columns corresponding to times are geometric means, the other numbers are averages.

As can be seen in the table, the impact of subproblem dependent R1C memory is noticeable but quite limited. On the contrary, ECCs and limited

Cuts	Rank-1 cuts memory	Root gap	Root time	ECC num.	R1C num.	Nodes num.	Solved	Total time
R1C only	arc, sp.dep	0.323%	93	0	124	67.8	87/90	181
R1C+ECC	node, sp.dep	0.133%	106	12	98	38.7	86/90	178
R1C+ECC	arc	0.108%	115	11	110	31.0	88/90	174
R1C+ECC	arc, sp.dep	0.105%	113	11	112	29.6	88/90	170

Table 2: Comparison of different algorithm variants on the whole set of instances

arc memory for R1C cuts are critical for solving two difficult instances. These ingredients are very useful to decrease the root gap and the number of nodes in the branch-and-price tree. ECCs decrease root gaps directly, by cutting fractional solutions that possibly would not be cut by any R1C. On the other hand, node and arc memory R1Cs are in principle equivalent. However, arc memory R1Cs decrease the root gaps actually obtained, by allowing more cuts to be added before separation is stopped because the maximum pricing time is exceeded. Although the geometric mean of total running time does not differ much among the variants, the solution time difference for some individual instances may change a lot. To illustrate this, in Figure 1, we show the performance profile for three variants of the algorithm.

It can be seen from the picture that Extended Capacity Cuts slow a bit the solution of many instances but may speed up a lot solution of some (difficult) instances. To illustrate this further, in Table 3, we compare the variants with and without ECC on a selection of instances.

5.3. Comparison with the literature

Here we compare our algorithm with the best algorithms available for the literature. In Tables 4–7, we indicate the algorithm, the number of solved instances, and the geometric mean of the total running time over all instances solved to optimality by all algorithms being compared. Note that here we launched our algorithm with the same initial upper bound as for the algorithms of Baldacci and Mingozzi (2009) and Contardo and Martinelli (2014).

Our algorithm is able to solve a significantly larger number of instances.

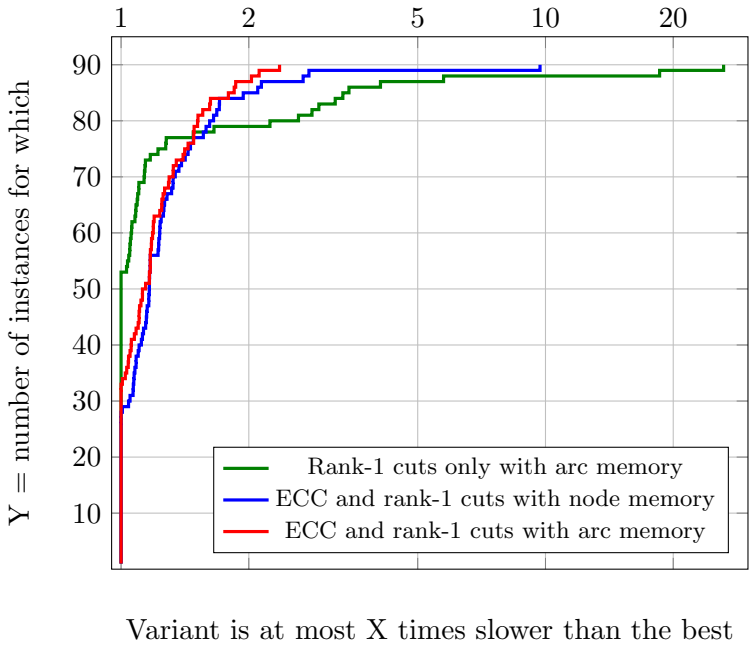

Figure 1: Performance profile comparing different variants of the algorithm

For HVRP and SDVRP instances, we double the size of instances which can be solved to optimality. Total solution time comparison is also favourable to our algorithm. Although, in these tables we do not take into account the difference in the speed of the computers used. For information, according to PassMark® benchmark, a single thread of our processor obtains the score of 1576, the processor Intel Xeon E5462 at 2.8 GHz used by Contardo and Martinelli (2014) obtains the score of 1187, the processor AMD Athlon 64 X2 Dual Core 4200+ used by Baldacci and Mingozzi (2009) obtains the score of 667, and the processor Core 2 Duo at 2.13 GHz used by Pessoa et al. (2009) obtains the score of 832. Even taking into account those differences and considering only the instances solved by all algorithms, our algorithm is still the fastest.

5.4. New HFVRP Instances

We also performed tests on newly created HFVRP instances. Our objectives are twofold. First, we would like to offer to the community a well-

Instance	Opt. solution	Cuts	Root bound	Root time	ECC num.	R1C num.	Nodes num.	Total time
c50_14fsmf	9119.03	R1C only	8767.26	28s	–	158	11	2m45s
		R1C+ECC	9119.03	40s	16	0	1	40s
c50_16fsmfd	3168.92	R1C only	3137.82	16s	–	43	23	6m05s
		R1C+ECC	3168.92	14s	7	0	1	14s
c50_15fsmf	2586.37	R1C only	2563.94	30s	–	68	53	12m02s
		R1C+ECC	2586.37	39s	16	0	1	39s
c100_19fsmf	8661.81	R1C only	8454.19	1m45s	–	240	17	11m22s
		R1C+ECC	8661.81	3m53s	9	187	1	3m53s
c100_19hvrp	10420.3	R1C only	10069.3	5m00s	–	176	21	49m15s
		R1C+ECC	10316.7	13m57s	3	153	3	21m34s
SDVRP_p22	1008.71	R1C only	1004.64	10m21s	–	54	41	2h02m
		R1C+ECC	1007.88	17m19s	28	131	3	21m
BrandaoN3hd	2378.99	R1C only	2316.50	8m56s	–	83	365	>36h[2]
		R1C+ECC	2353.96	19m31s	37	182	91	10h48m

[2] cannot be solved without using ECC

Table 3: Impact of ECC on the solution time of selected instances

Algorithm	Solved	Total time (17)	Total time (35)
Pessoa et al. (2009)	17/24	7m00s	-
Baldacci and Mingozzi (2009)	35/40	3m32s	5m17s
Our Branch-Cut-and-Price	40/40	38s	53s

Table 4: Comparison with the best published exact algorithm on the HFVRP Golden-Taillard instances. Total time is the geometric mean for the 17 instances solved by all algorithms and for 35 instances solved by the last two.

designed benchmark with 100 instances (the XH set) for both exact and heuristic methods. The set includes some large instances, with up to 1000 customers, that probably will remain very challenging for many years to come. Second, we want to better assess the strengths and limitations of our BCP algorithm, understanding how the characteristics of an instance impact in its performance.

5.4.1. Instance Generation Mechanism

Each new HFVRP instance is always created from an original CVRP instance, keeping the number of clients n, demand vector q, and customer

Algorithm	Solved	Total time
Baldacci and Mingozzi (2009)	9/13	2m01s
Our Branch-Cut-and-Price	21/23	18s

Table 5: Comparison with the best published exact algorithm on the SDVRP instances. Total time is the geometric for the 9 instances solved by both algorithms.

Algorithm	Solved	Total time (7)	Total time (10)
Baldacci and Mingozzi (2009)	7/9	2m49s	-
Contardo and Martinelli (2014)	10/10	2m26s	6m55s
Our Branch-Cut-and-Price	11/11	46s	1m27s

Table 6: Comparison with the best published exact algorithms on the MDVRP instances of Cordeau et al. (1997). Total time is the geometric mean for the 7 instances solved by all algorithms and the 10 instances solved by the last two ones.

Algorithm	Solved	Total time
Baldacci and Mingozzi (2009)	7/8	14m20s
Our Branch-Cut-and-Price	8/8	4m05s

Table 7: Comparison the best published exact algorithm on the MDVRP instances of Baldacci and Mingozzi (2009). Total time is the geometric mean for the 7 instances solved by both algorithms.

and depot coordinates. The Euclidian distance matrix d is obtained from the coordinates; as usual in the HFVRP literature, distances are not rounded. The following pieces of data should be added: the number of vehicle types m and, for each $u \in M = \{1, \ldots, m\}$, the values of Q_u, f_u, r_u, and K_u. All those new values are integer, except by the type-dependent travel cost factors r_u, that are given with 2 decimal places.

The concept of instance attributes used in Uchoa et al. (2017) for generating a diversified set of CVRP instances was adapted for HFVRP. We define the following HFVRP instance attributes and their possible values:

- Problem Type (PT): HVRP, FSMF, FSMD, FSMFD, or HD.
- Number of vehicle types (m): 3, 5, or 9.
- Capacity Variability (QV): 2, 4 or 6. Attribute QV defines the ratio between the largest and the smallest capacity (Q_m/Q_1).
- Economy of Scale (ES) on travel costs: Linear (no economy of scale, travel cost factor r_u proportional to Q_u), Moderate (r_u increases less than proportionally to Q_u) , or Heavy (r_u increases much less than Q_u). This attribute does not apply for instances of FSMF type.
- Fleet Distribution (FD): Uniform (almost the same values of K_u, for each $u \in M$), Inversely Proportional (K_u inversely proportional to Q_u, so, many small vehicles and few large vehicles), In-between (values of K_u between Uniform and Inversely Proportional). This attribute only applies for instances of HVRP and HD types.
- Fixed Cost Magnitude (FCM): Small (a small part of the total cost is due to f_u values), Medium, Large. The attribute does not apply on FSMD and HD types.

Given an original CVRP instance, an upper bound on the optimal solution value of that instance (Z_{CVRP}) and the values for the 6 above defined attributes, the new data for the HFVRP instance is obtained by the following steps:

1. The capacity vector Q is (uniquely) defined in such a way that: (i) $Q_1, \ldots, Q_m$ define a geometric progression, (ii) $Q_m/Q_1 = QV$, and (iii) the average capacity $\sum_{u=1}^{m} Q_u/m$ is equal to the capacity of the original CVRP instance (Q_{CVRP}). Actually, the capacities are rounded to the nearest integer. For example, if $m = 5$, $QV = 4$ and $Q_{CVRP} = 1000$, then $Q = (445, 629, 889, 1258, 1779)$.

2. If PT=FSMF, all travel cost factors are unitary by definition. Otherwise, the r vector depends on Q and ES. Recalling that m is odd, define $t = (m+1)/2$ as the median vehicle type. Define $h_u = Q_u/Q_t$. If ES=Linear, then r_u is set to h_u (rounded to 2 decimal places) for all $u \in M$. If ES=Moderate, then $r_u = 1.4621 h_u((\exp(-h_u) + 1)/2)$ for all $u \in M$. If ES=Heavy, then $r_u = 1.9015 h_u((3\exp(-h_u) + 1)/4)$ for all $u \in M$. The constants in those formulas make $r_t = 1.00$ in any case. For example, for the previous vector Q, ES=Linear gives $r = (0.50, 0.71, 1.00, 1.41, 2.00)$, ES=Moderate yields $r = (0.59, 0.77, 1.00, 1.29, 1.66)$, ES=Heavy yields $r = (0.67, 0.83, 1.00, 1.16, 1.34)$.

3. If PT=HVRP or PT=HD , the K vector depends on Q, on $q(V') = \sum_{i \in V'} q_i$ and on FD. In any case, K is chosen in such a way that the total fleet capacity $\sum_{u \in M} Q_u K_u$ is close to $1.1\, q(V')$. In the previous example, if FD=Uniform then $K = (6, 6, 6, 6, 5)$, if FD=Inversely Proportional then $K = (13, 9, 7, 5, 3)$, if FD=In-between then $K = (9, 7, 6, 5, 5)$.

4. If PT=FSMD or PT=HD, all fixed costs are zero by definition. Otherwise, the f vector depends on Q_{CVRP}, $q(V')$, Z_{CVRP} and FCM. Regardless of the attribute FCM, the values of f_u are always proportional to $h_u(2 - \exp(-h_u))/2$. This means that there are diseconomies of scale in the fixed costs. Similar diseconomies can observed in most existing instances in the literature. In fact, we observed in preliminary experiments that if fixed costs only grow linearly with the capacity, then FSMF instances have optimal solutions that almost only use the vehicle with the largest capacity. If FCM=Small, then fixed costs

are set in such a way that $f_t = (0.342\, Z_{CVRP})/(q(V')/Q_{CVRP})$. If FCM=Medium, all fixed costs are two times larger; if FCM=Large they are three times larger than that with FCM=Small. The constant 0.342 in those formulas was obtained by linear regression over the results of preliminary experiments, in order to make typical instances generated with PT=HVRP, $m = 5$, QV=4, ES=Moderate, FD=In-between and FCM=Medium to have optimal solutions where (roughly) half of the total cost consists of fixed costs. In the previous example, if FCM=Medium, $Z_{CVRP} = 50000$ and $q(V') = 27018$, then $f = (541, 827, 1266, 1928, 2894)$.

5.4.2. The New XH Set

The X set is composed by 100 CVRP instances, ranging from 100 to 1000 customers, proposed in Uchoa et al. (2017) and available in CVRPLIB web page (*http://vrp.atd-lab.inf.puc-rio.br/*). For each CVRP instance in the X set, a new HFVRP instance was created using the previously presented generation mechanism. The values for the 6 attributes were randomly selected, Z_{CVRP} was set to the best know solution value. The resulting XH set of HFVRP instances is available as an electronic companion to this paper and also in CVRPLIB.

For each new instance we tried to perform 20 runs of the HILS algorithm proposed in Penna et al. (2017). However, as we set a time limit of 24 hours per instance, on most instances the number of runs was smaller than 20. Using the HILS best solution as initial upper bounds, we run the BCP algorithm on the 68 instances of the XH set with less than 500 customers (the remaining 32 instances are clearly out of the reach of current exact HFVRP algorithms). It was possible to solve 23 instances within the time limit of 60 hours. The largest solved instance (X376-HD) has 375 customers and 3 vehicle types. On the other hand, an instance with 125 customers and 9 vehicle types (X126-HVRP) could not be solved. The detailed results can be seen in Tables B.17 and B.18.

5.4.3. Assessing the impact of each attribute on instance difficulty

The experiments reported in Tables B.17 and B.18 are statistically insufficient to assess the impact of each of the attributes on the performance of the proposed BCP. In order to do that, we created 300 additional medium-sized HFVRP instances, following a one-factor-at-a-time (OFAT) experiment design approach (Frey and Wang, 2006). We thus defined the *standard attribute configuration* as (PT=HVRP, $m = 5$, QV=4, ES=Moderate, FD=In-between, FCM=Medium). By changing one attribute value at a time (and keeping the other attributes with their standard values), we obtain 14 other configurations. For example, the effect of m is assessed by comparing the results of the standard configuration with the results of configurations (PT=HVRP, $m = 3$, QV=4, ES=Moderate, FD=In-between, FCM=Medium) and (PT=HVRP, $m = 9$, QV=4, ES=Moderate, FD=In-between, FCM=Medium). The complete experiment is described next:

- We first created a set of 20 original CVRP instances, all of them with 120 customers, with the same generator used in Uchoa et al. (2017). The CVRP attributes were randomly chosen, so the original instances are well diversified.
- Each CVRP instance is used for generating 15 HFVRP instances, one for each attribute configuration.
- We computed the median BCP times for solving the 20 instances created for each configuration. In order to eliminate the effect of the quality of the initial upper provided by HILS, a preliminary BCP run determines the optimal solutions values. Those values are used as initial upper bounds in the runs that are actually used for computing the medians reported in Table 8.

The results for attribute PT indicate that instances with limits on the number of vehicles of each type (HVRP and HD instances) are significantly harder than the same instances without those limits (FSM instances). This is a bit unexpected, since FSM instances require optimizing both fleet composition and their routes. On the other hand, the results for attribute m

are quite expected; the larger the number of vehicle types, the harder the instance. The effect of the economy of scale on travel costs is also strong: instances generated with ES=Moderate are already harder than those with ES=Linear, ES=Heavy makes them even harder. A small Capacity Variability ($QV = 2$) makes instances easier. However, no differences were observed between $QV = 4$ and $QV = 6$. Instances with FD=In-between were found to be a little harder than FD=Uniform or FD=Inversely Proportional. Finally, the Fixed Cost Magnitude does not seem to have a big effect, FCM=Large makes instances just a bit harder.

Table 8: BCP median times for each configuration

	secs	hours	ratio		secs	hours	ratio
PT				*m*			
HVRP	9903	2.75	1	3	2116	0.59	0.21
FSMF	5445	1.51	0.55	5	9903	2.75	1
FSMD	1611	0.45	0.16	9	25162	6.99	2.54
FSMFD	1827	0.51	0.18				
HD	7440	2.07	0.75	*QV*			
				2	5013	1.39	0.51
ES				4	9903	2.75	1
Linear	2890	0.80	0.29	6	9940	2.76	1.00
Moderate	9903	2.75	1				
Heavy	20396	5.67	2.06	*FCM*			
				Small	9855	2.74	1.00
FD				Medium	9903	2.75	1
Uniform	6953	1.93	0.70	Large	11383	3.16	1.15
InBet.	9903	2.75	1				
Inv.Prop.	5893	1.64	0.60				

6. Conclusions

This work presented a Branch-Cut-and-Price algorithm for the Heterogeneous Fleet VRP, including the related Multi-Depot VRP and Site Dependent VRP. The algorithm includes elements found in previous HFVRP algorithms (like route enumeration and Extended Capacity Cuts) and also elements (like limited memory R1Cs) only found in the most recent algorithms for the two most classical homogeneous VRP variants, Capacitated VRP and VRP with Time Windows. However, many of those elements were adapted in order to take advantage of the existence of several distinct

subproblems, corresponding to each vehicle type. The computational results obtained were good. It seems that typical instances with up to 200 customers can now be expected to be solved to optimality (often in long runs). In contrast, the best previous algorithms had difficulties on solving instances with 100 customers. The new algorithm can also find several optimal solutions that can not be found by existing heuristic methods.

Acknowledgments

Experiments presented in this paper were carried out using the PlaFRIM (Federative Platform for Research in Computer Science and Mathematics), created under the Inria PlaFRIM development action with support from Bordeaux INP, LABRI and IMB and other entities: Conseil Régional d'Aquitaine, Université de Bordeaux, CNRS and ANR in accordance to the "Programme d'Investissements d'Avenir". AP and EU visited Bordeaux with funding from project FAPERJ E26/110.075/2014 (International Cooperation FAPERJ/INRIA). We thank Puca Penna and Anand Subramanian for providing the code of their HILS heuristic.

References

Achterberg, T., 2007. Constraint integer programming. Ph.D. thesis, Technische Universitat Berlin.

Baldacci, R., Battarra, M., Vigo, D., 2008a. Routing a heterogeneous fleet of vehicles. In: The vehicle routing problem: latest advances and new challenges. Springer, pp. 3–27.

Baldacci, R., Christofides, N., Mingozzi, A., 2008b. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Math. Program. 115 (2), 351–385.

Baldacci, R., Mingozzi, A., 2009. A unified exact method for solving different classes of vehicle routing problems. Math. Program. 120, 347–380.

Baldacci, R., Mingozzi, A., Roberti, R., 2011. New route relaxation and pricing strategies for the vehicle routing problem. Operations research 59 (5), 1269–1283.

Brandão, J., 2011. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. Computers and Operations Research 38 (1), 140–151.

Chao, I.-M., Golden, B., Wasil, E., 1999. A computational study of a new heuristic for the site-dependent vehicle routing problem. INFOR: Information Systems and Operational Research 37 (3), 319–336.

Choi, E., Tcha, D.-W., 2007. A column generation approach to the heterogeneous fleet vehicle routing problem. Comput. & Oper. Res. 34, 2080–2095.

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

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.

Cordeau, J.-F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30 (2), 105–119.

Cordeau, J.-F. C., Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. Computers and Operations Research 39 (9), 2033 – 2050.

Frey, D., Wang, H., 2006. Adaptive One-Factor-at-a-Time Experimentation and Expected Value of Improvement. Technometrics 48 (3), 418–431.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., Werneck, R. F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical programming 106 (3), 491–511.

Golden, B., Assad, A., Levy, L., Gheysens, F., 1984. The fleet size and mix vehicle routing problem. Computers and Operations Research 11, 49–66.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., Løkketangen, A., 2010. Industrial aspects and literature survey: Fleet composition and routing. Comput. & Oper. Res. 37, 2041–2061.

Jepsen, M., Petersen, B., Spoorendonk, S., Pisinger, D., 2008. Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research 56 (2), 497–511.

Kullmann, O., 2009. Fundaments of branching heuristics. Handbook of Satisfiability, A. Biere, M. Heule, H. van Maaren and T. Walsh (Eds), IOS Press, 205–244.

Laporte, G., Norbert, Y., 1983. A branch and bound algorithm for the capacitated vehicle routing problem. Operations Research Spektrum 5, 77–85.

Lysgaard, J., 2003. CVRPSEP: a package of separation routines for the capacitated vehicle routing problem.
URL `http://www.hha.dk/lys/CVRPSEP.htm`

Nag, B., Golden, B. L., Assad, A., 1988. Vehicle routing with site dependencies. Vehicle routing: Methods and studies, 149–159.

Pecin, D., Contardo, C., Desaulniers, G., Uchoa, E., 2017a. New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing, To appear.

Pecin, D., Pessoa, A., Poggi, M., Uchoa, E., 2017b. 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., 2017c. Limited memory rank-1 cuts for vehicle routing problems. Operations Research Letters 45 (3), 206–209.

Penna, P. H. V., Subramanian, A., Ochi, L. S., Vidal, T., Prins, C., 2017. A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Annals of Operations Research (On-line first), 1–70.

Pessoa, A., de Aragão, M. P., Uchoa, E., 2007. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. In: International Workshop on Experimental and Efficient Algorithms. Springer, pp. 150–160.

Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2013. In-out separation and column generation stabilization by dual price smoothing. In: International Symposium on Experimental Algorithms. Springer, pp. 354–365.

Pessoa, A., Uchoa, E., de Aragão, M. P., 2009. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks 54 (4), 167–177.

Petersen, B., Pisinger, D., Spoorendonk, S., 2008. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 397–419.

Poggi de Aragão, M., Uchoa, E., 2003. Integer program reformulation for robust branch-and-cut-and-price. In: Wolsey, L. (Ed.), Annals of Mathematical Programming in Rio. Búzios, Brazil, pp. 56–61.

Røpke, S., 2012. Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems. Presentation in Column Generation 2012.

Subramanian, A., 2016. Personal communication.

Subramanian, A., Penna, P. H. V., Uchoa, E., Ochi, L. S., 2012. A hybrid algorithm for the heterogeneous fleet vehicle routing problem. European Journal of Operational Research 221 (2), 285–295.

Taillard, Éric., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO-Oper. Res. 33 (1), 1–14.

Uchoa, E., Fukasawa, R., Lysgaard, J., Pessoa, A., Poggi de Aragão, M., Andrade, D., 2008. Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation. Mathematical Programming 112, 443–472.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Subramanian, A., Vidal, T., 2017. New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research 257 (3), 845–858.

Vanderbeck, F., Sadykov, R., Tahiri, I., 2017. BaPCod — a generic Branch-And-Price Code.
URL `https://realopt.bordeaux.inria.fr/?page_id=2`

Westerlund, A., Göthe-Lundgren, M., Larsson, T., 2003. Mathematical formulations of the heterogeneous fleet vehicle routing problem. In: ROUTE 2003, International Workshop on Vehicle Routing. pp. 1–2.

Yaman, H., 2006. Formulations and valid inequalities for the heterogeneous vehicle routing problem. Mathematical Programming 106, 365–390.

Appendix A. Detailed results

In the Appendix, we present the detailed results for the reference variant of our algorithm for all tested instances. As the initial upper bound, we used the best solution available (including solutions obtained by us) with a small ϵ added for a double check. For some instances, the algorithm was launched the second time with another upper bound either 1) for a fair comparison with Baldacci and Mingozzi (2009), or 2) when the best known solution in the literature is greater than the solution obtained by us.

In the following tables, we show the instance name, n — the number of customers, m — the number of different vehicle types (or the number of depots), the initial upper bound, the dual bound obtained in the root, the root node time, the number of ECC and R1C at the end of the root node, the nodes number, the total time, and the best solution value obtained by the algorithm. In the tables, * near the best solution value indicates that its optimality was proven for the first time. If the best solution value is underlined, it improves on the best known solution value. All obtained solutions were proven to be optimal except when it is specially indicated.

Instance	n	m	Initial	Root				Nodes	Total	Final
			UB	bound	time	#ECC	#R1C	number	time	UB
c50_13hvrp	50	6	3185.2	3185.09	18s	0	0	1	18s	3185.09
c50_14hvrp	50	3	10107.7	10107.50	34s	26	0	1	34s	10107.50
c50_15hvrp	50	3	3065.4	3065.29	23s	8	38	1	23s	3065.29
c50_16hvrp	50	3	3265.6	3265.41	19s	27	0	1	19s	3265.41
c75_17hvrp	75	4	2077.1	2076.96	3m13s	13	109	1	3m13s	2076.96
c75_18hvrp	75	6	3743.7	3743.58	2m08s	28	0	1	2m08s	3743.58
c100_19hvrp	100	3	10420.5	10316.70	13m57s	3	153	3	21m34s	10420.30

c100_19hvrp	100	3	10423.4	10316.10	11m24s	4	118	3	19m18s	10420.30*
c100_20hvrp	100	3	4760.8	4745.35	8m26s	4	366	35	1h23m	4760.68
c100_20hvrp	100	3	4806.7	4740.46	5m02s	2	294	63	3h27m	4760.68*

Table A.9: Detailed results for the class HVRP of heterogeneous fleet instances

Instance	n	m	Initial	Root				Nodes	Total	Final
			UB	bound	time	#ECC	#R1C	number	time	UB
c50_13fsmf	50	6	2406.5	2406.36	18s	0	0	1	18s	2406.36
c50_14fsmf	50	3	9119.2	9119.03	40s	16	0	1	40s	9119.03
c50_15fsmf	50	3	2586.5	2586.37	38s	25	0	1	38s	2586.37
c50_16fsmf	50	3	2720.6	2720.43	19s	22	0	1	19s	2720.43
c75_17fsmf	75	4	1734.7	1734.53	3m42s	3	191	1	3m42s	1734.53
c75_17fsmf	75	4	1744.9	1732.44	7m41s	4	209	5	24m14s	1734.53
c75_18fsmf	75	6	2369.8	2369.65	2m21s	8	96	1	2m21s	2369.65
c75_18fsmf	75	6	2371.5	2369.65	2m56s	5	145	1	2m56s	2369.65
c100_19fsmf	100	3	8662.0	8661.81	3m53s	9	187	1	3m53s	8661.81
c100_20fsmf	100	3	4029.8	4018.26	4m13s	9	197	111	56m29s	4029.61
c100_20fsmf	100	3	4039.5	4016.90	3m32s	4	150	2279	28h41m	4029.61*

Table A.10: Detailed results for the class FSMF of heterogeneous fleet instances

Instance	n	m	Initial UB	Root				Nodes	Total	Final UB
				bound	time	#ECC	#R1C	number	time	
c50_13fsmfd	50	6	2964.8	2964.65	16s	0	0	1	16s	2964.65
c50_14fsmfd	50	3	9127.0	9126.90	39s	6	0	1	39s	9126.90
c50_15fsmfd	50	3	2635.1	2634.96	25s	9	0	1	25s	2634.96
c50_16fsmfd	50	3	3169.1	3168.92	13s	7	0	1	13s	3168.92
c75_17fsmfd	75	4	2004.6	2004.48	1m40s	20	87	1	1m40s	2004.48
c75_17fsmfd	75	4	2023.7	2001.55	2m57s	0	139	5	6m27s	2004.48
c75_18fsmfd	75	6	3148.1	3147.99	53s	0	0	1	53s	3147.99
c100_19fsmfd	100	3	8662.0	8661.81	3m36s	8	183	1	3m36s	8661.81
c100_19fsmfd	100	3	8664.3	8659.93	4m33s	10	200	7	7m09s	8661.81
c100_20fsmfd	100	3	4153.2	4137.08	2m55s	10	89	3	3m26s	4153.02
c100_20fsmfd	100	3	4154.5	4137.45	3m46s	8	89	3	4m43s	4153.02*

Table A.11: Detailed results for the class FSMFD of heterogeneous fleet instances

Instance	n	m	Initial UB	Root				Nodes	Total	Final UB
				bound	time	#ECC	#R1C	number	time	
c50_13hd	50	6	1518.0	1517.84	18s	0	0	1	18s	1517.84
c50_14hd	50	3	607.7	607.53	39s	29	0	1	39s	607.53
c50_15hd	50	3	1015.4	1015.29	25s	18	0	1	25s	1015.29

c50_16hd	50	3	1145.1	1144.94	14s	15	0	1	14s	1144.94
c75_17hd	75	4	1062.1	1061.96	3m28s	12	92	1	3m28s	1061.96
c75_18hd	75	6	1823.7	1823.58	1m31s	14	0	1	1m31s	1823.58
c100_19hd	100	3	1120.5	1120.34	5m58s	14	99	1	5m58s	1120.34*,3
c100_20hd	100	3	1534.3	1534.17	1m30s	28	39	1	1m30s	1534.17

[3] We could not obtain solution 1117.51 claimed by Taillard (1999)

Table A.12: Detailed results for the class HD of heterogeneous fleet instances

Instance	n	m	Initial	Root				Nodes	Total	Final
			UB	bound	time	#ECC	#R1C	number	time	UB
c50_13fsmd	50	6	1492.0	1491.86	16s	0	0	1	16s	1491.86
c50_14fsmd	50	3	603.4	603.21	51s	36	0	1	51s	603.21
c50_15fsmd	50	3	1000.0	999.82	14s	0	0	1	14s	999.82
c50_16fsmd	50	3	1131.1	1131.00	12s	8	0	1	12s	1131.00
c75_17fsmd	75	4	1038.7	1038.60	1m51s	13	26	1	1m51s	1038.60
c75_18fsmd	75	6	1800.9	1800.80	1m12s	27	0	1	1m12s	1800.80
c75_18fsmd	75	6	1801.4	1800.80	1m16s	26	0	1	1m16s	1800.80
c100_19fsmd	100	3	1105.6	1105.44	2m43s	20	58	1	2m43s	1105.44
c100_20fsmd	100	3	1530.6	1530.43	1m38s	13	64	1	1m38s	1530.43

Table A.13: Detailed results for the class FSMD of heterogeneous fleet instances

Instance	n	m	Initial UB	Root bound	Root time	Root #ECC	Root #R1C	Nodes number	Total time	Final UB
BrandaoN1fsmd	150	6	2211.8	2211.63	14m06s	1	275	1	14m06s	2211.63
BrandaoN1fsmd	150	6	2212.8	2211.63	17m37s	1	312	1	17m37s	2211.63*
BrandaoN1hd	150	6	2234.0	2228.53	19m18s	4	276	7	27m23s	2233.90
BrandaoN1hd	150	6	2234.2	2228.30	20m18s	2	237	9	33m32s	2233.90*
BrandaoN2fsmd	199	5	2810.3	2802.28	1h12m	1	553	89	7h18m	2810.20
BrandaoN2fsmd	199	5	2823.8	2796.84	56m47s	4	441	247	26h29m	2810.20*
BrandaoN2hd	199	5	2852.1	2842.60	56m20s	0	443	9	2h08m	2851.94
BrandaoN2hd	199	5	2859.9	2839.09	44m16s	6	283	65	7h47m	2851.94*
BrandaoN3fsmd	120	4	2234.7	2234.57	1m53s	0	0	1	1m53s	2234.57*
BrandaoN3hd	120	4	2379.1	2353.96	19m31s	37	182	91	10h48m	2378.99*
BrandaoN4fsmd	100	4	1822.9	1822.78	1m37s	31	0	1	1m37s	1822.78*
BrandaoN4hd	100	4	1839.4	1839.22	1m43s	51	0	1	1m43s	1839.22*

Table A.14: Detailed results for heterogeneous fleet instances by Brandão (2011)

Instance	n	m	Initial UB	Root bound	Root time	Root #ECC	Root #R1C	Nodes number	Total time	Final UB
sdvrp_p01	55	3	640.5	640.32	12s	0	0	1	12s	640.32
sdvrp_p02	55	2	598.2	598.10	13s	0	0	1	13s	598.10

sdvrp_p03	80	3	954.5	954.32	1m14s	7	101	1	1m14s	954.32
sdvrp_p03	80	3	957.1	948.78	1m20s	2	108	3	1m53s	954.32
sdvrp_p04	76	2	854.6	854.43	51s	8	86	1	51s	854.43
sdvrp_p05	103	3	1003.7	1003.57	2m02s	8	95	1	2m02s	1003.57
sdvrp_p06	104	2	1028.7	1025.08	7m24s	0	288	3	8m54s	1028.52*
sdvrp_p07	27	3	391.4	391.30	0s	0	0	1	0s	391.30
sdvrp_p08	54	3	664.6	664.46	2s	0	0	1	2s	664.46
sdvrp_p09	81	3	948.4	948.23	20s	0	0	1	20s	948.23
sdvrp_p10	108	3	1218.9	1218.75	1m38s	0	74	1	1m38s	1218.75*
sdvrp_p11	135	3	1448.3	1448.17	5m38s	0	354	1	5m38s	1448.17*
sdvrp_p12	162	3	1665.7	1650.98	9m29s	1	722	247	24h42m	1665.55*
sdvrp_p13	54	3	1194.3	1194.18	46s	27	0	1	46s	1194.18
sdvrp_p14	108	3	1960.1	1959.96	2m19s	13	90	1	2m19s	1959.96
sdvrp_p14	108	3	1960.7	1959.96	3m04s	13	95	1	3m04s	1959.96*
sdvrp_p15	162	3	2685.2	2680.24	5m46s	8	230	3	6m34s	2685.09*
sdvrp_p16	216	3	3393.5	3368.41	6m16s	3	165	197	5h13m	3393.31
sdvrp_p16	216	3	3393.7	3366.81	5m21s	2	242	183	5h37m	3393.31*
sdvrp_p17	270	3	4066.3	4038.14	9m52s	3	306	1065	>36h	4066.15[a]
sdvrp_p18	324	3	4751.4	4708.99	13m19s	1	468	677	>36h	4748.41[a,6]
sdvrp_p19	104	3	843.3	840.19	22m28s	0	356	3	34m55s	843.15*
sdvrp_p20	156	3	1030.9	1023.32	38m43s	1	506	21	4h21m	1030.78*
sdvrp_p21	209	3	1260.2	1255.66	53m18s	0	562	17	3h42m	1260.01

sdvrp_p21	209	3	1263.8	1254.25	47m42s	0	497	185	26h43m	1260.01*
sdvrp_p22	122	3	1008.9	1007.88	17m19s	28	131	3	21m15s	1008.71*
sdvrp_p23	102	3	803.4	803.29	1m27s	30	0	1	1m27s	803.29*

[a] optimality not proved

[6] better solution with value 4747.75 was obtained by another variant of the algorithm

Table A.15: Detailed results for site-dependent instances

Instance	n	m	Initial	Root				Nodes	Total	Final
			UB	bound	time	#ECC	#R1C	number	time	UB
p01	50	4	577.0	576.87	11s	0	0	1	11s	576.87
p02	50	4	473.7	473.53	31s	0	0	1	31s	473.53
p03	75	2	641.3	641.19	48s	0	0	1	48s	641.19[4]
p04	100	2	1001.2	1001.04	1m40s	8	121	1	1m40s	1001.04
p05	100	2	750.2	750.03	7m47s	4	194	1	7m47s	750.03
p05	100	2	751.3	749.02	10m37s	4	310	3	18m34s	750.03
p06	100	3	876.6	876.50	49s	6	29	1	49s	876.50
p07	100	4	882.1	881.97	1m29s	11	48	1	1m29s	881.97
p12	80	2	1319.1	1318.95	59s	7	10	1	59s	1318.95
p15	160	4	2505.6	2505.42	3m13s	24	10	1	3m13s	2505.42
p18	240	6	3703.0	3702.85	10m15s	16	25	1	10m15s	3702.85
p21	360	9	5475.0	5474.84	37m01s	9	121	1	37m01s	5474.84*

mdvrp-n151-k12-3-100	150	3	1197.7	1197.51	6m50s	10	205	1	6m50s	1197.51
mdvrp-n151-k12-3-80	150	3	1375.5	1375.37	2m13s	17	100	1	2m13s	1375.37[5]
mdvrp-n151-k12-4-100	150	4	1058.5	1058.38	2m51s	34	46	1	2m51s	1058.38
mdvrp-n151-k12-4-80	150	4	1200.7	1200.54	1m30s	29	0	1	1m30s	1200.54
mdvrp-n200-k16-3-100	199	3	1511.5	1508.19	8m59s	12	328	3	11m22s	1511.35
mdvrp-n200-k16-3-80	199	3	1756.6	1752.27	5m35s	8	261	9	10m34s	1756.48
mdvrp-n200-k16-3-80	199	3	1757.9	1752.23	5m23s	3	240	15	16m23s	1756.48*
mdvrp-n200-k16-4-100	199	4	1347.3	1347.19	7m03s	4	193	1	7m03s	1347.19
mdvrp-n200-k16-4-80	199	4	1535.2	1534.56	3m31s	28	58	1	3m31s	1534.56

[4] We and Contardo and Martinelli (2014) could not obtain solution 640.65 claimed by Baldacci and Mingozzi (2009)

[5] We could not obtain solution 1374.03 claimed by Baldacci and Mingozzi (2009)

Table A.16: Detailed results for multi-depot instances

Appendix B. Results on the new XH set

Tables B.17 and B.18 present results on the 68 XH instances with up to 500 customers. The instance name indicate the number of points (n customers plus the depot) and the attribute Problem Type (PT). Next, there is more information about the instance creation: the original CVRP instance, the Z_{CVRP} value, and attributes m, QV, ES, FD, and FCM. Then, there is the upper bound obtained with the HILS heuristic Penna et al. (2017), the value that was used as initial upper bound in the BCP run. Subsequent columns are bound the root node: the lower bound, time and number of active ECCs and R1Cs in the end of the node. Next, there is the number of nodes explored and the total time (with a limit of 60 hours). Finally, the final upper bound after the BCP run, underlined values indicate improvement upon HILS. There run of instance X459-HD went out-of-memory still in the root node. In the end of Table B.18 there are some lines with statistics for some columns: minimum, maximum, average and median. The values under column root bound are actually percent gaps between the root lower bound and final upper bound.

Table B.17: New set of benchmark instances: characteristics and BCP results (Part I)

		CVRP Instance		HFVRP attributes					HILS	Root				Nodes	Total	Final
#	Name	Name	UB	m	QV	ES	FD	FCM	UB	bound	time	#ECC	#R1C	number	time	UB
1	X101-FSMFD	X-n101-k25	27591	5	2	H	–	S	35170.2	35110.1	3m45s	16	119	3	4m08s	35170.2
2	X106-FSMD	X-n106-k14	26362	3	6	L	–	–	31566.3	31566.3	14m16s	11	72	1	14m16s	31566.3
3	X110-HD	X-n110-k13	14971	9	4	M	IP	–	15859.3	15729.8	8m51s	8	101	23	21m20s	15859.3
4	X115-HVRP	X-n115-k10	12747	3	6	H	Inb	S	19412.6	19381.2	7m27s	4	228	3	9m11s	19412.6
5	X120-FSMF	X-n120-k6	13332	5	4	–	–	L	26778.8	26729.9	15m51s	4	264	27	2h16m	26778.8
6	X125-HVRP	X-n125-k30	55539	9	2	M	U	M	95401.0	94639.1	14m02s	18	53	1353	>60h	95288.4
7	X129-FSMFD	X-n129-k18	28940	9	4	M	–	L	59184.5	59150.2	2m59s	28	107	3	3m12s	59184.5
8	X134-FSMD	X-n134-k13	10916	3	2	H	–	–	10271.9	10039.7	23m44s	24	44	521	>60h	10271.9
9	X139-HD	X-n139-k10	13590	5	6	L	IP	–	16810.9	16255.3	14m53s	4	159	1151	>60h	16810.9
10	X143-FSMF	X-n143-k7	15700	3	6	–	–	L	11706.7	10572.8	2h52m	0	0	26	>60h	11706.7
11	X148-HVRP	X-n148-k46	43448	5	4	L	IP	M	80285.3	80285.3	28m49s	26	18	1	28m49s	80285.3
12	X153-FSMFD	X-n153-k22	21220	3	2	M	–	S	27151.8	27002.3	13m13s	19	161	2305	>60h	27151.8
13	X157-HD	X-n157-k13	16876	9	4	M	IP	–	17246.5	17138.5	5m55s	10	190	51	3h55m	17246.5
14	X162-FSMD	X-n162-k11	14138	9	6	L	–	–	11853.4	11853.4	1h59m	23	158	1	1h59m	11853.4
15	X167-FSMF	X-n167-k10	20557	5	2	–	–	M	32035.2	31284.9	34m05s	10	80	379	>60h	31811.6
16	X172-HVRP	X-n172-k51	45607	5	4	H	IP	L	97395.9	97224.6	16m28s	30	209	135	10h51m	97395.9
17	X176-FSMFD	X-n176-k26	47812	3	6	L	–	M	101177.1	100665.4	12m15s	16	198	79	3h16m	101095.0
18	X181-HD	X-n181-k23	25569	9	2	M	Inb	–	26017.3	25930.1	3m29s	17	106	39	31m02s	26017.3
19	X186-FSMD	X-n186-k15	24145	5	2	M	–	–	23959.9	23769.0	1h13m	36	42	13	18h08m	23955.7
20	X190-FSMF	X-n190-k8	16980	3	4	–	–	S	18899.1	18877.0	1h05m	12	150	149	18h40m	18896.9
21	X195-FSMF	X-n195-k51	44225	9	6	–	–	M	68102.1	66187.0	35m00s	15	56	287	>60h	68102.1
22	X200-HD	X-n200-k36	58578	9	2	M	IP	–	60110.1	59496.2	1h06m	21	99	97	>60h	60110.1
23	X204-FSMD	X-n204-k19	19565	5	6	H	–	–	19612.7	19214.9	42m30s	0	0	195	>60h	19612.7
24	X209-FSMFD	X-n209-k16	30656	3	4	L	–	S	41409.7	41298.0	10m43s	42	476	2751	>60h	41409.7
25	X214-HVRP	X-n214-k11	10856	9	6	L	Inb	S	16120.1	15761.9	2h08m	0	6	45	>60h	16120.1
26	X219-HD	X-n219-k73	117595	5	2	M	U	–	120737.0	120737.0	12m36s	0	0	1	7s	120737.0
27	X223-HVRP	X-n223-k34	40437	3	4	H	IP	M	72251.1	72058.8	10m22s	41	329	763	>60h	72251.1
28	X228-FSMFD	X-n228-k23	25742	9	4	L	–	M	42312.9	42196.3	32m52s	24	272	385	>60h	42312.9
29	X233-FSMD	X-n233-k16	19230	3	6	H	–	–	19898.9	19476.0	36m59s	0	0	297	>60h	19898.9
30	X237-FSMF	X-n237-k14	27042	5	2	–	–	S	32210.2	31013.8	25m10s	12	210	185	>60h	32210.2
31	X242-FSMFD	X-n242-k48	82751	9	6	M	–	L	187390.6	187333.0	6m04s	14	169	5	7m59s	187368.9
32	X247-HVRP	X-n247-k50	37274	5	2	H	IP	S	49971.1	49761.3	37m27s	4	185	759	>60h	49904.9
33	X251-FSMD	X-n251-k28	38684	3	4	L	–	–	39993.8	39917.7	5m27s	43	305	541	7h26m	39993.1
34	X256-FSMF	X-n256-k16	18839	9	6	–	–	M	31414.9	30302.9	7h33m	0	0	9	>60h	31414.9
35	X261-HD	X-n261-k13	26558	5	4	M	U	–	28867.3	28053.2	2h51m	0	0	36	>60h	28867.3
36	X266-HD	X-n266-k58	75478	3	2	H	IP	–	70368.6	70242.2	10m49s	15	305	501	6h10m	70363.6

Table B.18: New set of benchmark instances: characteristics and BCP results (Part II)

		CVRP Instance		HFVRP attributes					HILS	Root				Nodes	Total	Final
#	Name	Name	UB	m	QV	ES	FD	FCM	UB	bound	time	#ECC	#R1C	number	time	UB
37	X270-FSMD	X-n270-k35	35291	5	6	M	–	–	38652.9	38216.2	35m59s	0	35	223	>60h	38652.9
38	X275-HVRP	X-n275-k28	21245	9	4	L	U	S	30687.3	30504.3	10m18s	9	168	51	2h58m	30673.4
39	X280-FSMF	X-n280-k17	33503	3	2	–	–	L	65796.4	64177.8	30m42s	0	0	177	>60h	65796.4
40	X284-FSMFD	X-n284-k15	20226	9	6	L	–	M	31901.8	31826.1	48m13s	37	512	641	>60h	31901.8
41	X289-HVRP	X-n289-k60	95151	5	2	M	IP	S	127795.8	127357.0	1h43m	6	206	355	>60h	127795.8
42	X294-HD	X-n294-k50	47161	3	4	H	U	–	44089.0	43887.1	1h14m	0	624	856	>60h	44089.0
43	X298-FSMD	X-n298-k31	34231	5	6	L	–	–	35025.4	34944.4	13m53s	62	243	1613	20h21m	35022.1
44	X303-FSMFD	X-n303-k21	21744	3	2	M	–	M	35993.5	35277.2	1h50m	0	0	93	>60h	35993.5
45	X308-FSMF	X-n308-k13	25859	9	4	–	–	L	51965.1	50136.9	3h43m	0	0	13	>60h	51965.1
46	X313-FSMD	X-n313-k71	94044	9	2	L	–	–	93377.8	93158.7	2h23m	40	180	435	>60h	93361.6
47	X317-HVRP	X-n317-k53	78355	5	4	M	IP	L	165763.4	165568.0	6m22s	29	118	11	22m55s	165763.0
48	X322-HD	X-n322-k28	29834	3	6	H	Inb	–	33507.8	32974.7	59m09s	0	0	245	>60h	33507.8
49	X327-FSMFD	X-n327-k20	27532	5	4	M	–	S	38672.8	38021.6	44m35s	63	63	75	>60h	38672.8
50	X331-FSMF	X-n331-k15	31102	9	6	–	–	L	63082.7	61165.8	5h09m	21	56	37	>60h	63082.7
51	X336-FSMF	X-n336-k84	139135	3	2	–	–	M	212635.9	209457.0	15m17s	0	0	277	>60h	212635.9
52	X344-FSMD	X-n344-k43	42056	5	4	L	–	–	42370.5	42343.1	13m54s	34	224	7	18m45s	42369.7
53	X351-HVRP	X-n351-k40	25928	9	2	H	IP	L	54124.9	53310.3	2h50m	0	0	43	>60h	54124.9
54	X359-HD	X-n359-k29	51505	3	6	M	U	–	60737.7	59115.7	11m55s	13	0	155	>60h	60737.7
55	X367-FSMFD	X-n367-k17	22814	9	6	H	–	L	51605.0	50338.2	2h41m	0	20	49	>60h	51605.0
56	X376-HD	X-n376-k94	147713	3	4	L	Inb	–	161394.2	161394.2	1m30s	43	48	1	1m30s	161394.2
57	X384-FSMF	X-n384-k52	65943	5	2	–	–	M	105143.6	100030.0	1h54m	0	75	108	>60h	105143.6
58	X393-HVRP	X-n393-k38	38260	9	6	M	IP	M	72748.1	71790.4	33m03s	0	0	49	>60h	72748.1
59	X401-FSMFD	X-n401-k29	66187	3	2	L	–	S	89755.7	87914.5	3h43m	0	0	111	>60h	89755.7
60	X411-FSMD	X-n411-k19	19718	5	4	H	–	–	18430.9	17379.9	2h15m	0	0	43	>60h	18430.9
61	X420-FSMD	X-n420-k130	107798	5	6	M	–	–	112984.8	112331.0	22m17s	33	519	289	>60h	112984.8
62	X429-HVRP	X-n429-k61	65483	3	2	L	U	S	91732.2	90553.3	50m29s	0	0	109	>60h	91732.2
63	X439-FSMF	X-n439-k37	36391	9	4	–	–	L	71877.0	70010.0	2h18m	31	196	72	>60h	71877.0
64	X449-FSMFD	X-n449-k29	55269	5	4	L	–	L	113204.3	110256.1	9h53m	0	0	24	>60h	113204.3
65	X459-HD	X-n459-k26	24145	9	6	M	IP	–	25359.1	o.m.						
66	X469-HD	X-n469-k138	221909	3	2	H	Inb	–	217177.8	215938.0	11m12s	0	0	193	>60h	217177.8
67	X480-FSMD	X-n480-k70	89458	3	4	L	–	–	100583.5	100399.0	24m55s	56	311	2067	>60h	100583.5
68	X491-FSMF	X-n491-k59	66510	5	2	–	–	L	131315.5	125200.7	5h35m	0	0	37	>60h	131315.5
	Min									0.00%	1m30s	0	0	1	7s	
	Max									9.70%	9h53m	63	624	2751	>60h	
	Avg.									1.43%	1h15m	15	126	322	>40h	
	Median									0.68%	32m52s	11	80	97	>60h	