CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

MASTER'S THESIS

Modeling and solving vehicle routing problems with many available vehicle types

SANDRA ERIKSSON BARMAN

Department of Mathematical Sciences
Division of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2014

Thesis for the Degree of Master of Science

Modeling and solving vehicle routing problems with many available vehicle types

Sandra Eriksson Barman

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology and University of Gothenburg
SE – 412 96 Gothenburg, Sweden
Gothenburg, May 2014

Matematiska vetenskaper
Göteborg 2014

Abstract

In this thesis, models have been formulated and mathematical optimization methods developed for the heterogeneous vehicle routing problem with a very large set of available vehicle types, called *many*-hVRP. This is an extension of the standard heterogeneous vehicle routing problem (hVRP), in which typically fairly small sets of vehicle types are considered.

Two mathematical models based on standard models for the hVRP have been formulated for the *many*-hVRP. Column generation and dynamic programming have been applied to both these models, following a successful algorithm for the hVRP. Benders' decomposition algorithm has also been applied to one of the models. In addition to the standard cost structure, where the cost of a pair of a vehicle and a route is determined by the length of the route and the vehicle type, we have studied costs that depend also on the load of the vehicle along the route. These load dependent costs were easily incorporated into the models, and other extensions could be similarly incorporated.

By using a standard set of test instances (with between three and six vehicle types in each instance) we have been able to compare our implementation with published results for hVRP. For *many*-hVRP, we have extended these instances to include larger sets of vehicle types (with between 91 and 381 vehicle types in each instance). The results show that the algorithms implemented for the two models find optimal solutions in a similar amount of time, but Benders' algorithm at times takes much longer to verify optimality. However, some other properties of Benders' algorithm suggests that it may constitute a good basis for a heuristic, when instances with even larger sets of vehicle types are used.

Acknowledgements

First I would like to thank my supervisor Peter Lindroth at Volvo Group Trucks Technology, for suggesting this interesting project and guiding the work throughout the process. I also want to thank my supervisor and examinor Ann-Brith Strömberg at the University of Gothenburg for her help and guidance.

I would like to thank Oskar for his fantastic support. I also want to thank Oskar and my friends for making the years studying here so wonderful. Finally I want to thank my mother for always supporting me and encouraging my interest in mathematics.

Contents

1 Introduction

In this thesis we have developed mathematical optimization models and algorithms for vehicle transport missions, called *vehicle routing problems*, in the special case where vehicle types can be chosen from a very large set. The thesis work has been performed in a collaboration between University of Gothenburg and Volvo Group Trucks Technology.

Vehicle routing problems have been studied for many years. The first model and algorithm were proposed by Dantzig and Ramser ([1]) in 1959, and since then hundreds of models and algorithms have been studied (see [2]). The general vehicle routing problem consists of determining an optimal set of vehicles, using an optimal set of routes, for distributing goods over a customer network. Figure 1[1] shows an example of what a network could look like—in Rennes, France.

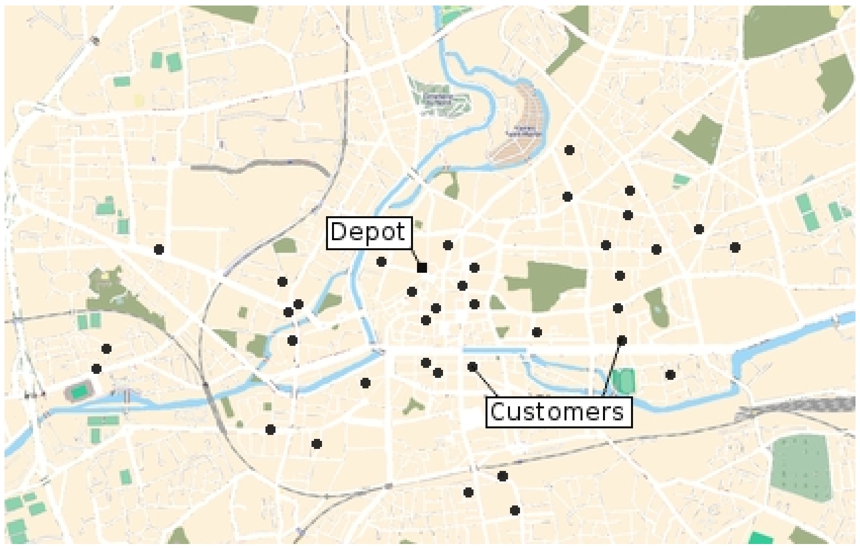

Figure 1: A network of roads and customers in a transport mission, where goods are to be distributed to 37 customers, marked by circles—via the white roads—from a central depot, marked by a square.

When modeling this type of problem, decisions have to be made about what aspects should be included in the model. From Volvo's perspective, including a very large set of vehicle types is of great interest. Since there is an immense set of possible vehicle configurations, the ability to find optimal vehicle configurations for specific transport missions would be useful. This may help customers to make more qualified decisions when purchasing vehicles, which in turn can lead

[1]The image shown in Figure 1 has been adapted from an image created by ©openstreetmap.org contributors, where data is available under the Open Database License, and cartography is licensed as CC BY-SA; see opendatacommons.org and creativecommons.org. The altered image in Figure 1 is thus available under the same license.

to more satisfied customers. This type of optimization tool can also help Volvo to better understand their customers' needs, which may then influence strategic vehicle development decisions and make Volvo even more competitive on the tough global vehicle market.

Research on vehicle routing problems has been successful, and has proved to be relevant in industrial applications. There is a growing industry of software for transportation planning based on methods developed by the scientific community for vehicle routing problems, and increasingly complex models and larger sized problems are solved ([3]). In this thesis, the focus is on finding models and algorithms appropriate for vehicle routing problems with a very large set of vehicle types. We have taken as a starting point so called *heterogeneous vehicle routing problems*, hVRP, in which it is assumed that more than one vehicle type is available. In previous hVRP models, to the best of our knowledge only a few vehicle types have been included. We have adapted these models and algorithms to accomodate a much larger set of vehicle types, here denoted *many*-hVRP. In addition, some new models and algorithms are proposed. Load dependent costs, which are not included in the standard hVRP, have also been implemented to illustrate how the solution framework developed can be expanded to include more aspects of real transportation problems.

To solve the *many*-hVRP, exact algorithms and heuristics based on column generation, dynamic programming and Benders' decomposition will be used. These three mathematical optimization techniques are presented in Section 2. A literature review of modeling and solution approaches for relevant vehicle routing problems is found in Section 3, problem formulations for *many*-hVRP are presented in Section 4 together with the implemented algorithms, after which tests and results are presented in Section 5. Finally, a discussion of the results and some suggestions for further research are presented in Section 6.

2 Mathematical optimization techniques applicable to vehicle routing problems

In optimization, problems are classified into different types that share certain characteristics and that can be solved using similar algorithms. Problems that can be stated as to

$$\underset{\mathbf{x}}{\text{minimize}}\ \mathbf{c}^\top \mathbf{x}, \tag{1a}$$
$$\text{subject to}\ \mathbf{D}\mathbf{x} = \mathbf{d}, \tag{1b}$$
$$\mathbf{x} \geq \mathbf{0}^n, \tag{1c}$$

for $\mathbf{c}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{d} \in \mathbb{R}^m$ and $\mathbf{D} \in \mathbb{R}^{m \times n}$—see Appendix A for an explanation of the notation is used in this thesis—are called *linear optimization problems* or *linear programs*[2]. A vector $\mathbf{x} \in \mathbb{R}^n$ that satisfies the constraints (1b)–(1c) is called a *feasible solution*. The aim is thus to minimize the linear objective function $\mathbf{c}^\top \mathbf{x}$ over the set of feasible solutions. The problem is said to be *feasible* if there exists at least one feasible solution. The matrix $\mathbf{D}$ is called the *constraint matrix*.

If the variables in the model (1) are required to be integer or binary, i.e. if $\mathbf{x} \in \mathbb{Z}^n$ or $\mathbf{x} \in \{0, 1\}^n$, then we have an *integer optimization problem*/*integer program* [4].

Vehicle routing problems are often formulated as integer programs; see formulations (12) and (15) in Section 3.1.1. Integer programs can be computationally very hard to solve—much harder than linear programs. We present three algorithms that can be used to solve integer programs, all of which have been applied to the *many*-hVRP in this thesis, as reported in Section 4. Column generation, which is presented in Section 2.1, is a method that solves linear programs, and which is not guaranteed to find an optimal solution to an integer program unless it is combined with, for instance, a branch-and-bound algorithm. Column generation has previously and with great success been applied to vehicle routing problems ([5]). In Section 2.2 dynamic programming is presented, which is a technique that is especially suited to solve the subproblems that occur in column generation for vehicle routing problems. Benders' decomposition algorithm, which can solve integer programs to optimality, is presented in Section 2.3. To the best of our knowledge, Benders' decomposition has not previously been applied to the hVRP; it has, however, been applied to other related problems, such as the simultaneous aircraft routing and crew scheduling problem ([6]) and network optimization problems (see [7]).

2.1 Column generation

The following outline of column generation is based on [8] and [9, Chapter 3.3], in addition to the textbook [4]. A more detailed account is found in Appendix

[2]The problem (1) can be formulated in different ways. For instance, the constraints (1c) can be excluded since they can be incorporated into the constraints (1b), and the equality in (1b) can be replaced by two inequalities.

B.

A property of linear programs such as (1), that is used in the construction of solution algorithms, is that if the problem is feasible and has a finite optimal solution, then there exists an extreme point to the set of feasible solutions that is optimal [4, p. 219, Theorem 8.10]. This is illustrated by the following example. Let

$$\begin{pmatrix} -1 & -1 \\ -1 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} -2 \\ -3 \\ 0.5 \end{pmatrix}, \tag{2a}$$

$$x_1, x_2 \geq 0, \tag{2b}$$

be the constraints of a linear program (here defined by inequality constraints). The feasible set defined by the constraints of a linear program is always a polyhedron. The polyhedron that comprises the feasible set defined by the constraints (2) is shown in Figure 2. This polyhedron has five extreme points—the five corners of the polyhedron. The feasible set defined by the constraints (2) would be very simple to optimize over, since the optimum will be attained in at least one of those extreme points for any linear objective function. All that needs to be done is to check the objective function value in each of the five extreme points.

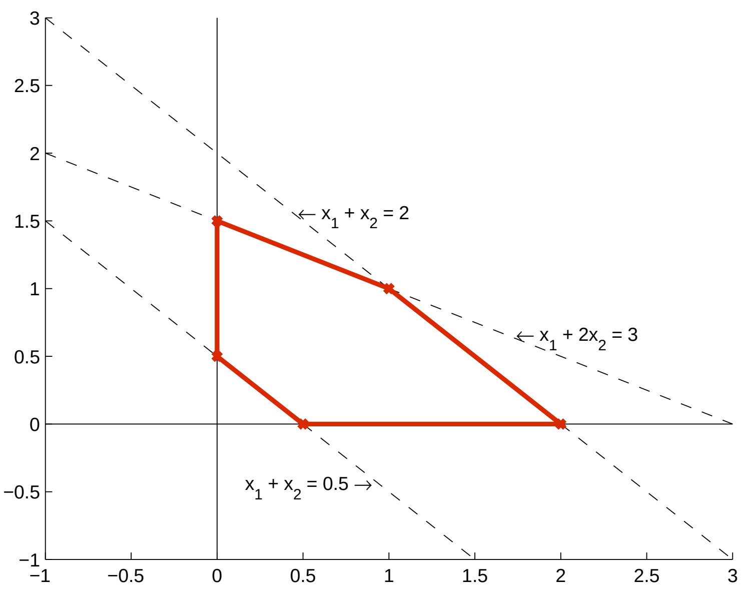

Figure 2: The feasible set defined by the constraints (2). Each of the dotted lines indicate where the corresponding constraint is satisfied with equality.

For bigger linear programs, however, the feasible set can have a huge number of extreme points. One solution technique that is used to solve such problems, is called the simplex method. To use the simplex method, the linear program must be expressed on the form (1), where $n \geq m$. The idea is to start in one extreme point and in each step of the algorithm go to the neighbouring extreme point that is most promising with respect to the change of the objective function value. Determining the most promising extreme point can be done without having to explicitly enumerate all of the extreme points, by utilizing the so called *reduced costs*[3], denoted $\hat{c}_i$, $i = 1, \ldots, n$. While standing at one extreme point, the reduced cost $\hat{c}_i$ measure how much the value of the objective function changes if the value of the variable x_i is increased by one—for variables x_i that has the value zero[4]—while moving in the direction of one particular neighbouring extreme point. If $\min_{i \in \{1,\ldots,n\}} \hat{c}_i \geq 0$, then the current extreme point is an optimal solution.

If the constraint matrix $\mathbf{D}$ in problem (1) has many columns, thousands or perhaps even millions, by using column generation it is possible to solve the problem to optimality while only considering a small subset of the columns. Let $I \subset \{1, \ldots, n\}$ be such that $|I| \geq m$ represent a subset of the columns in the problem (1). The constraint matrix restricted to the columns in I is denoted $\mathbf{D}_I := (\mathbf{d}_i)_{i \in I}$, where $\mathbf{d}_i$, $i = 1, \ldots, n$ denote the columns of $\mathbf{D}$. Defining analogously, $\mathbf{c}_I := (c_i)_{i \in I}$ and $\mathbf{x}_I := (x_i)_{i \in I}$, we have the following model, having much fewer columns/variables than the model (1):

$$\min_{\mathbf{x}_I} \ \mathbf{c}_I^\top \mathbf{x}_I, \tag{3a}$$

$$\text{s.t.} \quad \mathbf{D}_I \mathbf{x}_I = \mathbf{d}, \tag{3b}$$

$$\mathbf{x}_I \geq \mathbf{0}^{|I|}. \tag{3c}$$

The model (3) is called the *restricted master problem*, in contrast to the complete problem (1) which is called the *master problem*. Here, the number of columns/variables can be much smaller than in the master problem, i.e., $|I| \ll n$. The restricted master problem is feasible if the original problem (1) is feasible and $\text{rank}(\mathbf{D}_I) = m$.

The column generation algorithm iteratively solves the restricted master problem, and uses information about its optimal solution to extend the set I, after which the restricted master problem is solved again. This is repeated until the optimal solution to the restricted master problem is verified to be optimal also in the master problem. The column generation subproblem uses the reduced costs to extend I and verify optimality, similarly to the simplex method, as described above. Given an optimal extreme point to the restricted master problem, using the corresponding optimal solution $\boldsymbol{\pi}^*$ to its linear programming

[3]The reduced costs are defined in Appendix B.1.

[4]For a linear program on the form (1) with $n \geq m$, it holds that every extreme point corresponds to at most m non-zero variable values x_i, $i = 1, \ldots, n$.

dual[5], which is given by

$$\begin{aligned} \max_{\boldsymbol{\pi}} \ & \mathbf{d}^\top \boldsymbol{\pi}, \\ \text{s.t.} \ & \mathbf{D}_I^\top \boldsymbol{\pi} \le \mathbf{c}_I, \end{aligned}$$

where $\boldsymbol{\pi} \in \mathbb{R}^m$, the reduced costs of the master problem variables x_i, $i = 1, \dots, n$, are given by

$$\hat{c}_i := c_i - \mathbf{d}_i^\top \boldsymbol{\pi}^*, \qquad i = 1, \dots, n.$$

Since the reduced costs measure how much the objective function value decreases if the corresponding variable is increased by one, by adding a variable $i \in \{1, \dots, n\} \setminus I$ with strictly negative reduced cost $\hat{c}_i$ to the restricted master problem (3), the optimal solution to (3) would be improved (unless the current optimal extreme point is degenerate, in which case it may not be possible to take a non-zero step in the direction defined by the new variable). Thus, to determine which column to add to the restricted master problem, the problem,

$$\hat{c}^* := \min_{i \in \{1,\dots,n\}} \{\hat{c}_i\} = \min_{i \in \{1,\dots,n\}} \left\{ c_i - \mathbf{d}_i^\top \boldsymbol{\pi}^* \right\}, \tag{4}$$

called the ***subproblem*** is solved, and the set I is extended by adding an index $i \in \arg\min_{i \in \{1,\dots,n\}} \{\hat{c}_i\}$. If all the reduced costs are non-negative, i.e., if $\hat{c}^* \ge 0$, then the current solution to the restricted master problem is also optimal in the complete master problem. This holds even if not all columns $\mathbf{d}_i$, $i = 1, \dots, n$, have been generated, i.e., even if $I \neq \{1, \dots, n\}$. This means that the linear program can be solved while only generating a fraction of all its columns.

For integer programs, it is possible to apply column generation by relaxing the integrality requirement. Solving the relaxed problem using column generation not only gives a lower bound on the optimal value of the original integer program, but can also give a feasible (but not necessarily optimal) solution to the integer program. The latter is achieved by, as a last step, solving the integer program using only the columns generated during the column generation process—this restricted master problem with integer requirements on the variables can be much easier to solve than the original integer program if the number $|I|$ of columns is large. See Appendix B.2 for details.

For problems that can be formulated as a set partitioning problem with binary variables, column generation has been shown to be a good solution strategy—for vehicle routing and crew pairing assignment problems, among others. It is often implemented in combination with a branch-and-bound algorithm, called branch-and-price, in which an optimal integer solution is found using the lower bounds from column generation in each node ([5]). The most successful exact algorithms for vehicle routing problems are based on branch-and-price with additional cut generation, so called branch-and-cut-and-price (see [10]).

[5]The dual of a linear program is the Lagrangian dual problem of the linear program (called the primal), see the textbook [4], and many useful relationships between the primal problem and the dual problem exists.

In this master thesis branch-and-price has not been implemented, but the column generation algorithm can be improved by adding such a branch-and-bound framework.

2.2 Dynamic programming for shortest path problems with resource constraints

The column generation subproblem (4) for vehicle routing problems are *elementary shortest path problems with resource constraints* (denoted ESPPRC in [11]). These problems are NP-hard in the strong sense ([11, 12]), which makes it important to find efficient solution strategies. Dynamic programming is the most commonly used method for solving subproblems when column generation is used in connection with vehicle routing problems ([12, p. 155]), although due to the complexity of the problem often only a relaxed non-elementary shortest path problem with resource constraints (denoted SPPRC in [11]) is solved, in which case it has a pseudo-polynomial complexity ([13]).

In the following subsections the dynamic programming algorithm for ESPPRC is presented, along with some methods that can be implemented to make the solution process more efficient. Some of these methods for improving the algorithm have been tested in this project (as detailed in Section 4.1.1) while other methods are left as suggestions for further research (see Section 6.1).

2.2.1 A dynamic programming algorithm for ESPPRC

The following presentation is based on [11, 13].

A general shortest path problem with resource constraints is defined on a network of nodes $\mathcal{N} := \{1, \ldots, N\}$, arcs $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$, and a set of resources $\{1, \ldots, R\}$. A *path* is a sequence of nodes, denoted $P_{i_0 i_H} := (i_0, \ldots, i_H)$, where $i_h \in \mathcal{N}$, $h = 0, \ldots, H$, and $(i_{h-1}, i_h) \in \mathcal{A}$, $h = 1, \ldots, H$. For $i \in \mathcal{N}$, the notation (i) will be used to denote a single-node path. The set $\mathcal{V}_{P_{i_0 i_H}} := \{i_0, \ldots, i_H\}$ is called the set of *visited nodes*. A path $P_{i_0 i_H}$ can be *extended* by adding a node $i_{H+1} \in \mathcal{N}$. The extended path is denoted $(P_{i_0 i_H}, i_{H+1})$. A similar notation is used for two paths that are concatenated.

Each path $P_{i_0 i_H}$ has an associated cost, denoted $C_{P_{i_0 i_H}}$. The *resource consumption vector* $\left(T^1_{i_h}, \ldots, T^R_{i_h}\right)_{P_{i_0 i_H}} \in \mathbb{R}^R$, for a given path $P_{i_0 i_H}$ at node i_h, represents the amount of each resource $r \in \{1, \ldots, R\}$ that has been consumed at node i_h. The individual elements of the resource consumption vectors associated with path $P_{i_0 i_H}$, are written as $T^r_{i_h}(P_{i_0 i_H})$, $h = 1, \ldots, H$, $r = 1, \ldots, R$. The resource consumption vector $\left(T^1_{i_0}, \ldots, T^R_{i_0}\right)_{P_{i_0 i_H}}$ is given for the inital node i_0. *Resource functions* $f^r : \mathbb{R}^R \times \mathcal{A} \mapsto \mathbb{R}$ are given, so that for $h \in \{1, \ldots, H\}$, $r \in \{1, \ldots, R\}$,

$$T^r_{i_h}(P_{i_0 i_H}) := f^r \left((T^1_{i_{h-1}}, \ldots, T^R_{i_{h-1}})_{P_{i_0 i_H}}, (i_{h-1}, i_h) \right).$$

For each node $i \in \mathcal{N}$, a set of feasible resource consumption vectors

$$\mathcal{T}_i := \left\{ (T_i^1, \ldots, T_i^R) \;\middle|\; T_i^r \in [a_i^r, b_i^r], r \in \{1, \ldots, R\} \right\},$$

is given. A path $P_{i_0 i_H}$ is said to be *feasible with respect to the resource constraints*, or *resource feasible*, if for each node i_h, $h = 1, \ldots, H$, it holds that $\left(T_{i_h}^1, \ldots, T_{i_h}^R\right)_{P_{i_0 i_H}} \in \mathcal{T}_{i_h}$.

The objective of the shortest path problem with resource constraints is to minimize the cost $C_{P_{st}}$, over the set P_{st} of paths from node $s \in \mathcal{N}$ to node $t \in \mathcal{N}$, that are resource feasible.

Dynamic programming is based either on the Ford-Bellman algorithm, or on Djikstra's algorithm, depending on how paths are extended (see e.g. [13]). For a general shortest path problem with resource constraints, dynamic programming can work as follows ([11, 13]):

Paths starting at node s and ending in node $i \neq t$ are maintained in a family of non-processed paths, denoted $\mathcal{P}_{\text{NPP}}$, which at the beginning of the algorithm contains only the single-node path (s). A path $P_{si} \in \mathcal{P}_{\text{NPP}}$ is processed by extending it to all nodes $j \in \{j \in \mathcal{N}$, such that (P_{si}, j) is resource feasible. Each resulting path that does not end in node t is added to $\mathcal{P}_{\text{NPP}}$, while paths ending in t are added to a set of processed paths, denoted $\mathcal{P}_{\text{PP}}$. After path P_{si} has been processed, it is added to $\mathcal{P}_{\text{PP}}$. Extending paths in this way ensures that all paths, in $\mathcal{P}_{\text{NPP}}$ as well as in $\mathcal{P}_{\text{PP}}$, are resource feasible.

From here on, $U_{i_0 i_H}$, $Q_{i_0 i_H}$ and $Q^*_{i_0 i_H}$ are defined analogously to $P_{i_0 i_H}$. Let $\mathcal{E}_{P_{si}} := \{(P_{si}, U_{jt}) \mid (P_{si}, U_{jt}) \text{ is resource feasible}\}$, so that $\mathcal{E}_{P_{si}}$ is the set of all feasible extensions of the path P_{si}, ending in node t. For the paths Q_{si} and Q^*_{si}, both ending at the node $i \neq t$, Q^*_{si} is said to be dominated by Q_{si} if

$$\min_{P_{st} \in \mathcal{E}_{Q_{si}}} C_{P_{st}} \leq \min_{P_{st} \in \mathcal{E}_{Q^*_{si}}} C_{P_{st}}.$$

This means that no path in $\mathcal{E}_{Q^*_{si}}$ can have a lower cost than the best extension (Q_{si}, U_{jt}) of Q_{si}. A path in $\mathcal{P}_{\text{NPP}} \cup \mathcal{P}_{\text{PP}}$ that is dominated by another path in the same set can be removed.

When the set of non-processed paths $\mathcal{P}_{\text{NPP}}$ is empty, an optimal path from s to t is found among the paths in $\mathcal{P}_{\text{PP}}$.

A simple example of a shortest path problem with resource constraints is shown in Figure 3. It contains five nodes and one resource. In the context of vehicle routing problems, this resource may correspond to the available time for traveling to the nodes of the network. Next to each arc (i, j), the arc cost c_{ij} and the resource consumption t_{ij} along that arc are shown in red. The resource consumption vectors are given by $(T_1^1)_{(1)} := 0$, $\left(T_j^1\right)_{(P_{1i}, j)} := \left(T_i^1\right)_{P_{1i}} + t_{ij}$, $i, j \in \{2, \ldots, 5\}$, $(i, j) \in \mathcal{A}$. The set of feasible resource consumption vectors are defined by $[a_i^1, b_i^1] := [0, 20]$, $i = 1, \ldots, 5$. The costs are given by $C_{(1)} := 0$, $C_{(P_{1i}, j)} := C_{P_{1i}} + c_{ij}$, $i, j \in \{2, \ldots, 5\}$, $(i, j) \in \mathcal{A}$.

The cheapest path from node 1 to node 5, disregarding resource consumption, is $(1, 2, 3, 5)$, with the associated cost $C_{(1,2,3,5)} = 0.43$. However, the path $(1, 2, 3, 5)$ is not resource feasible, since $\left(T_5^1\right)_{(1,2,3,5)} = 22 > 20$. The cheapest (shortest) path from node 1 to node 5, that is also resource feasible, is $(1, 3, 5)$, with the cost $C_{(1,3,5)} = 0.88$. For this example, dynamic programming is not very useful since the optimal path can be found easily by inspection. For more complicated networks, the efficiency of the dynamic programming algorithm depends strongly on the ability to "identify and discard paths which are not useful" [11, p. 46]. In the extreme case where no paths can be discarded (i.e. removed due to domination), then dynamic programming is simply an enumeration of feasible paths, which is not a very efficient algorithm.

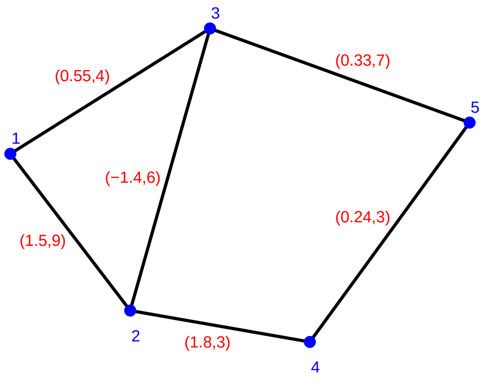

Figure 3: A network consisting of five nodes and six arcs, with one resource T^1. Each arc (i, j) has an associacted consumption t_{ij} of resource T^1, and a cost c_{ij}, shown next to the arcs as (t_{ij}, c_{ij}). The resource T^1 is accumulated along each path, and the resource constraints state that $T_i^1 \in [0, 20]$, $i \in \{1, ..., 5\}$.

For an ESPPRC, paths must be elementary, meaning that no node $i \in \mathcal{N}$ may be visited more than once. For (non-elementary) SPPRC with non-decreasing resource functions and with cost given by $C_{(s)} := c_0$, $C_{(P_{si},j)} := C_{P_{si}} + c_{ij}$, $i, j \in \mathcal{N}$, $(i, j) \in \mathcal{A}$, for paths Q_{si} and Q_{si}^* which end in the same node i, if the inequality[6]

$$\left(T_i^1, \ldots, T_i^R\right)_{Q_{si}} \leq \left(T_i^1, \ldots, T_i^R\right)_{Q_{si}^*}$$

holds, then Q_{si}^* is dominated by Q_{si} and can be discarded. For ESPPRC with non-decreasing resource fucntions, the set of visited nodes $\mathcal{V}_{Q_{si}}$ of Q_{si} must also be a subset of the visited nodes $\mathcal{V}_{Q_{si}^*}$ of Q_{si}^*, for Q_{si}^* to be dominated by Q_{si} (see [11, p. 50]). This is formalized in Claim 1, the proof of which is essentially the same proof as that of [13, Claim 1]. Claim 1 is the basis for the criteria for

[6]The notation $\left(T_i^1, \ldots, T_i^R\right)_{Q_{si}} \leq \left(T_i^1, \ldots, T_i^R\right)_{Q_{si}^*}$ means $T_i^r(Q_{si}) \leq T_i^r(Q_{si}^*), \forall r \in \{1, \ldots, R\}$.

dominating paths in the dynamic programming algorithm used in this master thesis. The specific details of this implementation are given in Section 4.1.1.

Claim 1. *For an elementary shortest path problem with resource constraints, starting in node $s \in \mathcal{N}$ and ending in node $t \in \mathcal{N}$, assume that the resource consumption is given by*

$$T_j^r\left((P_{si}, j)\right) := \max\left\{a_i^r, T_i^r(P_{si}) + h_{ij}^r\right\},$$

where $h_{ij}^r \geq 0, i, j \in \mathcal{N}$, $(i,j) \in \mathcal{A}$, $r \in \{1, \ldots, R\}$, and that the cost is given by

$$C_{(P_{si},j)} := C_{P_{si}} + c_{ij}, i, j \in \mathcal{N}, (i,j) \in \mathcal{A}$$

where $C_{(s)} := c_0$. For two elementary and resource feasible paths Q_{si} and Q_{si}^* ending in node $i \neq t$, Q_{si}^* is dominated by Q_{si} if the relations

$$\left(T_i^1, \ldots, T_i^R\right)_{Q_{si}} \leq \left(T_i^1, \ldots, T_i^R\right)_{Q_{si}^*},$$

$$C_{Q_{si}} \leq C_{Q_{si}^*},$$

and

$$\mathcal{V}_{Q_{si}} \subseteq \mathcal{V}_{Q_{si}^*}$$

hold.

Proof. Since $i \neq t$, it holds that $\mathcal{V}_{Q_{si}^*} \subsetneq \mathcal{N}$. If $\left(T_j^1, \ldots, T_j^R\right)_{(Q_{si}^*,j)} \notin \mathcal{T}_j$ for all $j \in \mathcal{N} \setminus \mathcal{V}_{Q_{si}^*}$, then Q_{si}^* can not be extended further. Otherwise, pick any $j \in \mathcal{N} \setminus \mathcal{V}_{Q_{si}^*}$ for which $\left(T_j^1, \ldots, T_j^R\right)_{(Q_{si}^*,j)} \in \mathcal{T}_j$. The inclusion $\mathcal{V}_{Q_{si}} \subseteq \mathcal{V}_{Q_{si}^*}$ implies that the extension (Q_{si}, j) is an elementary path. It holds that $\left(T_j^1, \ldots, T_j^R\right)_{(Q_{si},j)} \in \mathcal{T}_j$, since for $r \in \{1, \ldots, R\}$,

$$\begin{aligned} T_j^r\left((Q_{si}, j)\right) &:= \max\left\{a_i^r, T_i^r(Q_{si}) + h_{ij}^r\right\} \leq \max\left\{a_i^r, T_i^r(Q_{si}^*) + h_{ij}^r\right\} \\ &= T_j^r\left((Q_{si}^*, j)\right) \leq b_i^r. \end{aligned}$$

It also holds that

$$C_{(Q_{si},j)} = C_{Q_{si}} + c_{ij} \leq C_{Q_{si}^*} + c_{ij} = C_{(Q_{si}^*,j)}.$$

Hence, both (Q_{si}, j) and (Q_{si}^*, j) are resource feasible elementary paths, and the relations

$$\left(T_i^1, \ldots, T_i^R\right)_{(Q_{si},j)} \leq \left(T_i^1, \ldots, T_i^R\right)_{(Q_{si}^*,j)},$$

$$C_{(Q_{si},j)} \leq C_{(Q_{si}^*,j)},$$

and

$$\mathcal{V}_{(Q_{si},j)} \subseteq \mathcal{V}_{(Q_{si}^*,j)}$$

hold. By induction with respect to extension of paths, any feasible extension (Q_{si}^*, U_{ij}^*) of Q_{si}^* also provides a feasible extension (Q_{si}, U_{ij}^*) of Q_{si}, for which $C_{(Q_{si},U_{ij}^*)} \leq C_{(Q_{si}^*,U_{ij}^*)}$. This shows that Q_{si}^* is dominated by Q_{si}. $\square$

It is clear that the dominance rule of SPPRC is more efficient than that of ESPPRC, since more paths can be dominated if the restriction on visited nodes is left out. To increase the number of paths that can be eliminated for ESPPRC—if it is possible to determine nodes that are unreachable from a given path using time-windows, for instance—unreachable nodes can be added to the set of visited nodes, which makes the dominance rule in Claim 1 a bit more efficient ([11, pp. 50–51]).

2.2.2 Simplifying the problem

Relaxing the ESPPRC to the SPPRC by allowing paths containing cycles, i.e., non-elementary paths, will result in problems having a larger state space but also more efficient dominance rules. When column generation is used as part of the solution procedure for vehicle routing problems, by using branch-and-price for instance, the lower bounds of the original problem provided by column generation are weaker when the ESPPRC subproblems are relaxed to the SPPRC. This may be compensated by the reduction of the complexity of the subproblem, which can be great. A compromise between SPPRC and ESPPRC that is often implemented for vehicle routing problems is to allow non-elementary paths, but forbid cycles of length two. This only duplicates the number of labels compared to SPPRC and is quite easy to implement.

A compromise between the ESPPRC and the SPPRC has been proposed, in which, after solving the SPPRC, a restriction is imposed on the nodes that are visited more than once. This new problem is solved, and analogous restrictions are added to every node that is visited several times in the new solution. This procedure is repeated until an elementary path is returned. In the worst case scenario, restrictions have to be added to all nodes.

Yet another way to simplify the ESPPRC is to use a so-called state-space relaxation. Instead of keeping track of the visited nodes, the number of visited nodes are used in the dominance criterion. This is different relaxation of the ESPPRC, and also in this case an elimination of cycles of length two can be implemented (see [14, pp. 417–418] and [12, pp. 160–161,165]).

2.2.3 Heuristic methods for the ESPPRC

A heuristic for improving the speed of column generation, when using dynamic programming to solve the ESPPRC subproblems, can be implemented in several ways.

The dynamic programming algorithm can be terminated before an optimal path has been verified, and return a path with a negative reduced cost which is not necessarily optimal. Since adding any path with a negative reduced cost may improve the solution in the next iteration of the column generation, it is only necessary to solve the ESPPRC subproblem to optimality when trying to prove that the current solution to the restricted master problem is optimal in the complete problem. This does not need to be done at all when using a heuristic column generation procedure.

Another heuristic approach is to temporarily relax the requirement that paths be elementary, and allow non-elementary paths in the beginning of the dynamic programming process. When paths get longer than a certain threshold value, the elementarity requirement is added in the hope that an elementary path with negative reduced cost is returned. If no such paths are found, the process is restarted and the elementarity requirement is added earlier than before. This is repeated until an elementary path with negative reduced cost is found (see [11, pp. 58–59]).

Betinelli et al. ([15]) take a different approach, solving the ESPPRC column generation subproblems in three steps. The first step is a simple heuristic. In the second step, the domination requirement (see Claim 1) that the inclusion $\mathcal{V}_{Q_{si}} \subseteq \mathcal{V}_{Q^*_{si}}$ holds is relaxed, and replaced by another requirement. This results in a problem that is easier to solve but in which a path that is optimal in the original, non-relaxed, problem may be dominated by a non-optimal path. Hence, optimality in the original problem, is not guaranteed in the solution to the relaxed problem. In a final step, the relaxed criterion $\mathcal{V}_{Q_{si}} \subseteq \mathcal{V}_{Q^*_{si}}$ is reinstated, and the column generation subproblems are solved to optimality.

2.3 Benders' decomposition

To handle the large set of vehicle types in *many*-hVRP, decomposing the problem in several levels can be a good idea. Using Benders' decomposition, problems can be decomposed in such a way that in each iteration one set of variables—which we call *complicating variables*—are fixed, and the problem with respect to the remaining variables is solved to optimality. This procedure is iterated, so that information about solutions from previous iterations is used to fix the complicating variables in each iteration, until the optimal solution to the restricted problem (with fixed variables) is verified to be an optimal solution to the original problem.

In the context of *many*-hVRP, the complicating variables can be chosen to be the vehicle types, so that in each iteration a subset of the vehicle types—of a size that can be efficiently handled—is chosen among the whole set of vehicle types. The details of our implementation of Benders' algorithm for *many*-hVRP is presented in Section 4.2.

For routing and scheduling problems arising in areas such as airline planning, Benders' decomposition together with column generation has successfully been applied by Cordeau et al. in [6] to simultaneously consider both the aircraft routing problem and the crew pairing problem—something that traditionally have been done sequentially due to the large complexity of the problem. Their implementation first uses a linear programming (LP) relaxation of both Benders master problem and Benders subproblem, and later reintroduces the integer requirements and performs a heuristic branching on the integer variables. Branching on variables has not been performed in this thesis, however it would be an interesting extention of the implemented algorithm. The review [16] of airline scheduling planning, by Dunbar et al., describes how the method by Cordeau et al. ([6]) has been expanded and improved by several authors. The expansions include time windows, reversing the order of Benders master- and subproblems which improved the convergence rate, and applying a heuristic algorithm to parts of the problem to avoid the tailing off-effect of column generation.

Here, a version of Benders' decomposition that in this thesis is used on *many*-hVRP is presented. The theory is based on [9, Chapter 7.3]. The goal is to solve an optimization problem of the form

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \ & \mathbf{c}^\top \mathbf{x}, && \text{(5a)}\\
\text{s.t. } & \mathbf{D}\mathbf{x} + \mathbf{F}\mathbf{y} = \mathbf{b}, && \text{(5b)}\\
& \mathbf{x} \geq \mathbf{0}^{n_1}, && \text{(5c)}\\
& \mathbf{y} \in \{0,1\}^{n_2}, && \text{(5d)}
\end{aligned}
$$

for $\mathbf{c}$, $\mathbf{x} \in \mathbb{R}^{n_1}$, $\mathbf{y} \in \mathbb{R}^{n_2}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{D} \in \mathbb{R}^{m \times n_1}$, $\mathbf{F} \in \mathbb{R}^{m \times n_2}$. The set of feasible solutions to (5) is assumed to be non-empty and bounded. Here, the complicating variables are the binary variables $\mathbf{y}$. If variables $\mathbf{y}$ are fixed, the remaining problem is a linear program, which is assumed to be much easier to solve than the integer program (5). Defining the set $R := \{\mathbf{y} \in \{0,1\}^{n_2} \mid \exists \mathbf{x} \geq \mathbf{0}^{n_1}, \mathbf{D}\mathbf{x} = \mathbf{b} - \mathbf{F}\mathbf{y}\}$, a formulation that is equivalent to the model (5) is given by

$$
\min_{\mathbf{y} \in R} \left\{ \min_{\mathbf{x}} \ \mathbf{c}^\top \mathbf{x} \ \middle| \ \mathbf{D}\mathbf{x} = \mathbf{b} - \mathbf{F}\mathbf{y}, \mathbf{x} \geq \mathbf{0}^{n_1} \right\}. \tag{6}
$$

The inner problem, given by

$$
\begin{aligned}
\min_{\mathbf{x}} \ & \mathbf{c}^\top \mathbf{x}, && \text{(7a)}\\
\text{s.t. } & \mathbf{D}\mathbf{x} = \mathbf{b} - \mathbf{F}\mathbf{y}, && \text{(7b)}\\
& \mathbf{x} \geq \mathbf{0}^{n_1}, && \text{(7c)}
\end{aligned}
$$

is called *Benders subproblem.* The feasible set of (7) is non-empty and bounded for $\mathbf{y} \in R$. Due to strong duality for linear programs, which says that the optimal objective value of (7) for fixed $\mathbf{y} \in R$ is equal to the optimal objective value of its dual [4, p. 248], (7) can be replaced by its dual

$$
\begin{aligned}
\max_{\mathbf{u}} \ & (\mathbf{b} - \mathbf{F}\mathbf{y})^\top \mathbf{u}, && \text{(8a)}\\
\text{s.t. } & \mathbf{D}^\top \mathbf{u} \leq \mathbf{c}, && \text{(8b)}\\
& \mathbf{u} \in \mathbb{R}^m. && \text{(8c)}
\end{aligned}
$$

This problem is also feasible and bounded for $\mathbf{y} \in R$, so there exists an optimal solution in at least one of the extreme points of its feasible set. It can thus be reformulated as

$$
\max_{i \in \mathcal{P}} \left\{ (\mathbf{b} - \mathbf{F}\mathbf{y})^\top \mathbf{u}_i \right\}, \tag{9}
$$

where the set $\{\mathbf{u}_i, i \in \mathcal{P}\}$ denotes the extreme points of the set $\{\mathbf{u} \mid \mathbf{D}^\top \mathbf{u} \leq \mathbf{c}\}$. The model (6) can now be written as

$$\min_{\mathbf{y} \in R} \left\{ \max_{i \in \mathcal{P}} \left\{ (\mathbf{b} - \mathbf{F}\mathbf{y})^\top \mathbf{u}_i \right\} \right\},$$

which in turn can be reformulated as

$$v^*_{\mathrm{BMP}} := \min_{\mathbf{y}, v} \; v, \tag{10a}$$

$$\text{s.t.} \;\; v \geq (\mathbf{b} - \mathbf{F}\mathbf{y})^\top \mathbf{u}_i, \qquad i \in \mathcal{P}, \tag{10b}$$

$$\mathbf{y} \in R. \tag{10c}$$

The problem (10) is called *Benders master problem.* As with column generation[7], a restricted version of this problem is introduced where in this case only a subset of the constraints are included, resulting in *Benders restricted master problem*

$$v^*_{\mathrm{BRMP}} := \min_{\mathbf{y}, v} \; v, \tag{11a}$$

$$\text{s.t.} \;\; v \geq (\mathbf{b} - \mathbf{F}\mathbf{y})^\top \mathbf{u}_i, \qquad i \in \widetilde{\mathcal{P}}, \tag{11b}$$

$$\mathbf{y} \in R, \tag{11c}$$

where $\widetilde{\mathcal{P}} \subseteq \mathcal{P}$. For $(v^*_{\mathrm{BMP}}, \mathbf{y}^*)$ optimal in Benders master problem it holds that v^*_{BMP} equals the optimal objective value of the original problem (5), and solving Benders subproblem (7) with $\mathbf{y} := \mathbf{y}^*$ will yield $\mathbf{x}^*$, for which $(\mathbf{y}^*, \mathbf{x}^*)$ is optimal in the original problem.

Since not all constraints are included in Benders master problem, the inequality $v^*_{\mathrm{BRMP}} \leq v^*_{\mathrm{BMP}}$ holds if $(v^*_{\mathrm{BRMP}}, \widetilde{\mathbf{y}}^*)$ is optimal in (11). Such an optimal solution $(v^*_{\mathrm{BRMP}}, \widetilde{\mathbf{y}}^*)$ exists and is bounded if $\widetilde{\mathcal{P}} \neq \emptyset$, since R is closed, bounded, and non-empty. Solving Benders subproblem (7) with $\mathbf{y} := \widetilde{\mathbf{y}}^*$ provides a feasible solution $(\widetilde{\mathbf{y}}^*, \widetilde{\mathbf{x}}^*)$ to the original problem, and $\mathbf{c}^\top \widetilde{\mathbf{x}}^*$ is consequently an upper bound on its optimal value v^*_{BMP}. Solving the dual (9) of Benders subproblem, again with $\mathbf{y} := \widetilde{\mathbf{y}}^*$, provides an extreme point $\mathbf{u}_{i^*}$, $i^* \in \mathcal{P}$, for which the equivalence $(\mathbf{b} - \mathbf{F}\widetilde{\mathbf{y}}^*)^\top \mathbf{u}_{i^*} = \mathbf{c}^\top \widetilde{\mathbf{x}}^*$ holds. This is an upper bound on the optimal value v^*_{BMP}, and for which v^*_{BRMP} is a lower bound. Thus, if it holds that

$$(\mathbf{b} - \mathbf{F}\widetilde{\mathbf{y}}^*)^\top \mathbf{u}_{i^*} = v^*_{\mathrm{BRMP}},$$

then $(\widetilde{\mathbf{y}}^*, \widetilde{\mathbf{x}}^*)$ is an optimal solution to the original problem (5). Otherwise, the inqualities

$$(\mathbf{b} - \mathbf{F}\widetilde{\mathbf{y}}^*)^\top \mathbf{u}_{i^*} \geq v^*_{\mathrm{BMP}} \geq v^*_{\mathrm{BRMP}},$$

hold and i^* is added to $\widetilde{\mathcal{P}}$ defining a new constraint to Benders restricted master problem. Since u_{i^*} is optimal in (9), the new constraint is the constraint in Benders master problem which is the most unsatisfied by the solution $(v^*_{\mathrm{BRMP}}, \widetilde{\mathbf{y}}^*)$.

[7]If the problem (5) were linear, i.e., if the constraint (5d) was replaced by $\mathbf{y} \geq \mathbf{0}^{n_2}$, then Benders' algorithm would be equivalent to applying Dantzig-Wolfe decomposition and column generation to the dual of (5). See [9, Chapter 7.3.2] for details.

Solving Benders restricted master problem with the updated set of constraints will yield a solution that is different from $(v^*_{\text{BRMP}}, \widetilde{\mathbf{y}}^*)$, which is no longer feasible, and the new optimal objective value of Benders restricted master problem will be greater than or equal to v^*_{BRMP}. Since the number of constraints in Benders master problem is finite, Benders' algorithm is guaranteed to converge in a finite number of iterations.

3 Vehicle routing problems

Vehicle routing problems are difficult optimization problems, which have applications in many fields, including transportation, logistics, communication and manufacturing, and they belong to the most studied combinatorial optimization problems. The classic problem called the *capacitated vehicle routing problem*, cVRP, consists of finding a solution to a simplified transport mission in which customers are serviced by a set of indentical vehicles delivering goods from a central depot, where the number of customers that each vehicle can service is limited by a vehicle capacity restriction. The cVRP is an extension of the traveling salesman problem, and is NP-hard in the strong sense (see e.g. [3, 17, 18]).

A wide range of extensions of the classic cVRP are available and several thousands of articles have been dedicated to the subject. These extensions can be classified into the three main groups

1. Assignment of customers and routes to resources, including *multiple depots*, a *heterogeneous fleet of vehicles*, *multiple periods*, in which customers are serviced more than once, and *split deliveries*, in which customers can be serviced by more than one vehicle.

2. Sequence choices, including *backhauls* and *pickup-and-delivery* where deliveries are made to one set of customers and goods are picked up from a set of vendors/customers, and orders can be recieved dynamically, *precedence* constraints such that some customers should be serviced before others, and *multiple trips*, in which vehicles can depart from and return to the depot more than once.

3. Evaluation of fixed sequences, including *time windows* where each customer needs to be serviced within a given time-frame, networks with *time-dependent* features such as time-dependent travel times or time-dependent service costs, and *2D and 3D loading constraints.* [3]

Many different solution methods, both exact and heuristic, have been developed for different types of vehicle routing problems, and even though the methods need to be tailored to the specific problem, several methods can be applied to different extensions of cVRP. Consistently, exact methods for vehicle routing problems can only solve problems with up to 200 customers (see [3, 10, 18]).

Successful heuristic methods for vehicle routing problems almost always use a combination of different classical heuristics. Recently, combinations of exact methods—based on mathematical programming techniques—and heuristic methods, have been proposed. These are sometimes called matheuristics (see [10]). According to [10], p. 61 “an exact solution of real-world problems with many additional side constraints will remain impossible in the short and medium term. However, close-to-optimal solutions of more and more complex and integrated problems, increasingly based on incomplete optimization approaches and mathematical-programming-based heuristics, are possible, and this is sufficient to provide useful decision support in practice”.

In this thesis we are interested in vehicle routing problems with a very large set of vehicle types, which we call *many*-hVRP, in constrast to the standard vehicle routing problem with a heterogeneous fleet, hVRP, in which usually only a few vehicle types are considered. The sets of vehicle types for *many*-hVRP could be so large that it would not be practical to consider the whole set, making the algorithms that have previously been used for hVRP impractical. Parts of the models and algorithms for *many*-hVRP have been based on mathematical programming algorithms developed for hVRP, which are presented in Section 3.1. In Section 3.2 a model including load dependent costs which has been incorporated into the *many*-hVRP model, is presented. Other extensions such as time-windows can easily be implemented. Algorithms for *many*-hVRP are presented in Section 4.

3.1 Heterogeneous vehicle routing problems

The vehicle routing problem with a set of non-identical vehicle types was first formulated by Golden et al. ([19]) in 1984. The problem is known as the heterogeneous VRP, or the fleet size and mix VRP (see [17]).

When modeling real-life problems, there is often a trade-off between the number of practical considerations that can be included and the maximum problem size that can be solved within a reasonable amount of time. A heterogenous fleet of vehicles is one such aspect of real-life problems that may be important to consider. Hoff et al. state in [7, p. 2043] that "there is generally a strong dependency between fleet composition and routing aspects", and decisions about fleet composition and routing need to be integrated. Given a transport mission, the optimal routing depends strongly on the characteristics of the available fleet of vehicles. Hoff et al. directs critisism against the operation research community, saying that it has been too focused on idealized models and that there has not been enough effort directed towards the fleet composition problem [7].

Nevertheless, there has been a number of contributions relating to hVRP from the operation research community. In the following sections standard models, solution methods and test instances for hVRP are presented.

3.1.1 Standard models

In hVRP a heterogeneous set of vehicles is considered, in constrast to cVRP where all vehicles are of the same type. Vehicle types differ by their capacity and cost structure. A limited or unlimited number of vehicles of each type is available. One part of the cost of each route can be fixed (denoted fixed cost), and one part of the cost can depend the route length (denoted variable cost)—sometimes only a fixed cost or only a variable cost is used. Sometimes, the variable cost does not depend on the vehicle type used for the route ([17]). In this thesis, no restriction has been put on the numbers of vehicles of each type that are available, and a combination of fixed and variable costs is used—both of which depend on vehicle type.

There exist three main types of mathematical models for vehicle routing problems in general. In the first type, integer vehicle flow variables are used to model both the vehicle routes and the commodity flow. In the second type, additional continuous variables are used to model the commodity flow. The third type of model uses a set-partitioning formulation. The first type of models has mostly been used for basic versions of vehicle routing problems, and is according to Toth et al. [18], not suited for heterogeneous vehicle problems. Therefore, the second and third types of models of the hVRP is presented below. The formulation using commodity flow variables is from [7], and the set-partitioning formulation can be found in [20].

Both models are defined on a directed graph $(\mathcal{N}, \mathcal{A})$, where $\mathcal{N} := \{0, 1, \dots, N\}$ denotes the set of nodes representing the customers and the depot, and $\mathcal{A}$ denotes the set of directed arcs between the nodes in $\mathcal{N}$ representing the network. The depot is denoted by 0, making $\mathcal{N} \setminus \{0\} =: \mathcal{N}_0$ the set of nodes representing the customers. Each customer $i \in \mathcal{N}_0$ has a demand d_i. There are K different vehicle types, represented by the set $\mathcal{K} := \{1, \dots, K\}$. Each vehicle k has a limited capacity, denoted D_k. It is assumed that every customer may be served by at least one vehicle type, i.e., the constraint $\max_{k \in \mathcal{K}} D_k \geq \max_{i \in \mathcal{N}} d_i$ must hold.

Associated with each vehicle type $k \in \mathcal{K}$ are routing (or variable) costs c_{ij}^k, $(i, j) \in \mathcal{A}$, and a fixed cost f_k. In the model (12) feasible routes are defined by the constraints in the model, while in the model (15) feasible routes are implicitly defined by the sets $\mathcal{R}_k$, $k \in \mathcal{K}$, containing all feasible routes r with respect to vehicle type k. A route is a sequence of nodes $(i_0, i_1, \dots, i_{H-1}, i_H)$, where $i_h \in \mathcal{N}$, $h = 0, \dots, H$, and $(i_{h-1}, i_h) \in \mathcal{A}$, $h = 1, \dots, H$. A route $r := (i_0, i_1, \dots, i_{H-1}, i_H)$ is feasible if it starts and ends at the depot, i.e. if $i_0 = i_H = 0$. A route-vehicle pair (r, k), such that $r := (i_0, i_1, \dots, i_{H-1}, i_H)$, and $k \in \mathcal{K}$, is feasible if the route is feasible and if the capacity constraints of the vehicle type are satisfied by the route, i.e., if the total demand—given by $\sum_1^{H-1} d_h$—of the customers along the route is less than or equal to the vehicle's capacity D_k. The cost for a feasible route-vehicle pair (r, k) equals $c_r^k := f_k + \sum_{h=1}^H c_{i_{h-1} i_h}^k$.

Formulation using commodity flow variables:

Define the variables

$$x_{ij}^k := \begin{cases} 1, & \text{if arc } (i,j) \text{ is used by a vehicle of type } k, \\ 0, & \text{otherwise,} \end{cases} \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K},$$

and e_{ij} := flow of goods from node i to j, $(i, j) \in \mathcal{A}$. The commodity flow formulation is given by:

$$\min_{\mathbf{x},\mathbf{e}} \left(\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{N}:(0,j)\in\mathcal{A}} f_k x_{0j}^k + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \right), \tag{12a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij}^k = 1, \qquad j \in \mathcal{N}_0, \tag{12b}$$

$$\sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji}^k = 0, \qquad j \in \mathcal{N}_0, k \in \mathcal{K}, \tag{12c}$$

$$\sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} e_{ij} - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} e_{ji} = d_j, \qquad j \in \mathcal{N}_0, \tag{12d}$$

$$e_{0j} \le \sum_{k\in\mathcal{K}} D_k x_{0j}^k, \qquad (0,j) \in \mathcal{A}, \tag{12e}$$

$$e_{ij} \le \sum_{k\in\mathcal{K}} M_{ijk} x_{ij}^k, \qquad (i,j) \in \mathcal{A}, \tag{12f}$$

$$e_{ij} \ge 0, \qquad (i,j) \in \mathcal{A}, \tag{12g}$$

$$x_{ij}^k \in \{0,1\}, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}. \tag{12h}$$

The objective (12a) is to minimize the sum of the fixed costs and the route costs of each vehicle used. The constraints (12b) ensure that each customer is visited exactly once, and the constraints (12c) that any vehicle that visits a customer departs from that customer. The constraints (12d) ensure that a vehicle that visits a customer delivers exactly the amount of goods that the customer demands. The constraints (12e)–(12f) impose the vehicle capacity restrictions, where the parameter M_{ijk} is chosen to be sufficiently large.

A slightly different formulation is found in [17], in which constraints limit the number of vehicles of each type, and the constraints (12e)–(12f) have been replaced by

$$d_j x_{ij}^k \le e_{ij} \le (D_k - d_i) x_{ij}^k, \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}. \tag{13}$$

This formulation is not consistent, since x_{ij}^k is always zero for some $k \in \mathcal{K}$ if $|\mathcal{K}| > 1$. We suggest that the constraints (12e)–(12f) be replaced by

$$d_j x_{ij}^k \le e_{ij} \le (D_k - d_i) x_{ij}^k + M_{ijk}(1 - x_{ij}^k), \qquad (i,j) \in \mathcal{A}, k \in \mathcal{K}. \tag{14}$$

Set-partitioning formulation:

Define the parameters

$$\delta_{ir}^k := \begin{cases} 1, & \text{if route } (r,k) \text{ visits customer } i, \\ 0, & \text{otherwise,} \end{cases} \qquad i \in \mathcal{N}_0, r \in \mathcal{R}_k, k \in \mathcal{K},$$

and the variables

$$x_r^k := \begin{cases} 1, & \text{if route } (r,k) \text{ is used,} \\ 0, & \text{otherwise,} \end{cases} \qquad r \in \mathcal{R}_k, k \in \mathcal{K}.$$

The set-partitioning formulation is then given by:

$$\min_{\mathbf{x}} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} c_r^k x_r^k, \tag{15a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} \delta_{ir}^k x_r^k = 1, \qquad i \in \mathcal{N}_0, \tag{15b}$$

$$x_r^k \in \{0,1\}, \qquad r \in \mathcal{R}_k, k \in \mathcal{K}. \tag{15c}$$

The objective (15a) is to minimize the sum of the costs over the routes that are used. The constraints (15b) ensure that each customer is visited by exactly one vehicle.

This type of set-partitioning formulation was originally proposed for the vehicle routing problem by Balinski and Quandt in 1964 ([21]). The model is general, and has the advantage that many restrictions can easily be incorporated, since the feasibility of routes is implicitly defined by the sets $\mathcal{R}_k$, $k \in \mathcal{K}$. The LP relaxation of this type of model for VRP is often very tight. Also, the number of feasible routes, $\sum_{k \in \mathcal{K}} |\mathcal{R}_k|$, will for most problems be extremely big, which makes column generation an appropriate solution method—for problems with tens of customers, there may be billions of feasible routes ([18]). The set-partitioning formulation is then seen as a column generation master problem, where the column generation subproblems generate routes x_r^k to be added to the the restricted master problem (in which not all routes in the sets $\mathcal{R}_k$, $k \in \mathcal{K}$ are included).

3.1.2 Previously developed solution strategies

Up until recently, no exact algorithm had been implemented for the hVRP "due to the intrinsic difficulty of this family of routing problems" ([17, p. 13]). Exact solution methods have now been developed based on branch-and-cut-and-price by Pessoa et al. in 2007 ([22]) and on a set-partitioning formulation using additional constraints by Baldacci et al. in 2009 ([23]). The exact method in [23] can solve instances with up to 75 customers and some instances with 100 customers; it works well for other types of vehicle routing problems as well ([3]).

Baldacci et al. review in [17] the many different heuristic solution methods that have been applied to hVRP, as do Hoff et al. in [7]. Of the methods reviewed, the set-partitioning based heuristic proposed by Taillard in [20] and the column generation based heuristic of Choi and Tcha in [24] are of particular relevance for this project; both methods are mentioned in both review articles [7, 17].

Taillard [20] implemented a set-partitioning based heuristic algorithm, that solves one homogeneous VRP (where only one vehicle type is allowed) for each vehicle type using an adaptive memory procedure. From each homogeneous VRP a set of routes are stored. These routes are added as columns to a set-partitioning formulation of hVRP, but here each route r defines one column for each vehicle type k for which $r \in \mathcal{R}_k$—not just for the vehicle type(s) that the route was associated with in the homogenous problem. Unfortunately, according to [20, p. 6], the method does not perform well on very small instances when the number of vehicles allowed of each vehicle type was restricted, in which case "there was a higher probability that a run would not produce a good or even feasible solution" compared to the larger instances.

Inspired by the approach in [20], in [24] Choi and Tcha presented a column generation based heuristic using the following set-partitioning formulation:

$$\min_{\mathbf{x}} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} c_r^k x_r^k, \tag{16a}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} \delta_{ir}^k x_r^k \geq 1, \qquad i \in \mathcal{N}_0, \tag{16b}$$

$$\sum_{r \in \mathcal{R}_k} x_r^k = \alpha^k, \qquad r \in \mathcal{R}_k, k \in \mathcal{K}, \tag{16c}$$

$$x_r^k \in \{0, 1\}, \qquad r \in \mathcal{R}_k, k \in \mathcal{K}, \tag{16d}$$

$$\alpha^k \geq 0, \text{ integer}, \qquad r \in \mathcal{R}_k, k \in \mathcal{K}. \tag{16e}$$

Compared to the formulation (15), the constraints (16c) and (16e) have been added to the set-partitioning problem, relating to the number α_k of vehicles of type k that is used. The new constraints do not impose any restriction on the number of vehicles that are allowed but are used to accelerate a branch-and-bound implementation. In addition, the problem is formulated as a set-covering rather than a set-partitioning problem. Since arc costs correspond to Euclidean distances the two models are equivalent in the sense that the optimal solution to (15) is optimal also in (16). The subproblems that generate new columns for the restricted master problem corresponding to (16) are defined as

$$\min_{r \in \mathcal{R}_k, k \in \mathcal{K}} \left(c_r^k - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir}^k - \pi_k^* \right), \tag{17}$$

where π_i^* and π_k^* are the optimal dual variable values corresponding to the constraints (16b) and (16c), respectively. In [24] this subproblem is divided into one subproblem for each vehicle type which are relaxed, both by a state-space relaxation and by relaxing the elementary constraint, while introducing a 2-cycle elimination procedure. The relaxed subproblems are solved using dynamic programming. It was found in [24] that only very small problem instances with up to 20 customers could be solved to optimality using this approach.

Therefore the column generation procedure is employed in a heuristic branch-and-bound algorithm. In a preprocessing step, some vehicle types are discarded by calculating a lower bound on the optimal value of (16) by requiring that at least one vehicle of a particular type is used, taking only the fixed costs into account. This lower bound is compared with an upper bound provided by a heuristic, enabling the rejection of the vehicle types for which the corresponding lower bound is higher than the upper bound of the original formulation (16) (see [24]).

This column generation algorithm by Choi and Tcha has been reported to provide good results compared with other heuristic solution methods in the two review articles [17, 7]. In [7, p. 2048], it is stated that “results confirm the dominance of this algorithm, both in terms of quality and computation time”. According to Pessoa et al. ([22]), who developed an exact branch-and-cut-and-price algorithm, the lower bounds computed by Choi and Tcha [24] were not effective enough for an exact branching algorithm, but the columns generated in the root node yielded a heuristic solution that outperformed other heuristics on several test instances.

The approach to solve hVRP heuristically implemented by Choi and Tcha in [24] has been used as a starting point for the algorithms developed in this thesis—both the splitting of the column generation subproblem into one problem per vehicle type and the solution of subproblems using dynamic programming—which will be explained in Section 4.1. The results obtained by the column generation algorithm implemented in this thesis are compared with the results in [24], and presented in Section 5.1.1.

3.1.3 Standard benchmarks

In 1984 Golden et al. ([19]) introduced test instances for the hVRP based on customer node data from instances found in [25, 26], which include twenty sets of test instances, twelve of which possessing Euclidean distances (denoted G12 in [17]). The test instances in G12 are commonly used for hVRP with only fixed costs. Taillard ([20]), whose set-partitioning based heuristic is described in Section 3.1.2, adapted eight of the twelve sets in G12 by defining variable costs (denoted T8 in [17]). These are typically used as benchmarks to test algorithm performance for hVRP with variable costs. None of the test instances in G12 and T8 possesses a combination of fixed and variable costs of which both depend on the vehicle type ([17, 3]). Another set of test instances that should be mentioned are those proposed by Li et al. in [27], containing 200–360 customers located on concentric circles around the depot, with four to six vehicle types in each instance. These test instances are also frequently used as benchmarks for hVRP ([3]).

The algorithm presented in [24], also described in Section 3.1.2, was tested on extensions of the test instances T8 and G12. The eight instances that are common to the sets T8 and G12 were combined, so that both the fixed and variable costs that depend on the vehicle type—from the respective sets T8 and G12—were applied simultaneously. The set T8 was also extended to include all of the instances in G12, by adding variable costs which depend on the vehicle types to the four instances in G12 that were not included in T8. The four new instances are the smallest in the set, with only 20 customers each (a reason why they were not included in T8 may be that the algorithm implemented in [20] did not perform well on small instances). The twelve new instances are here denoted CT12. In [24], the algorithm was also tested on T8 and G12, to be able to compare with other published results. In [17], the best solution values obtained using different heuristics are compared with the best known solutions at the time (2008). The best known solutions was found by Choi and Tcha for all instances in T8 (see [17, Table 6]), and the average relative gap with respect to the best known solution values for instances in G12 was 0.004% (see [17, Table 4]). The best solutions known today (2014) for CT12 are presented in Section 5.1.1—better solutions than those obtained in [24] have only been found for two of the twelve instances in CT12.

The test instances CT12 have been used to test the performance of our algorithms, both in their original form and their extensions to include more vehicle types (presented below). Problem specifications for CT12 are listed in Table 1[8]. The four test instances to which Choi and Tcha added vehicle specific variable costs to are named Choi3–Choi6, and the instances from T8 are named T13–T20, following the numbering of the original instances in [19]. The ranges of the capacities, the fixed costs, and the variable costs, for the vehicles in each instance are given in Table 1. For $r \in \mathcal{R}_k$, $k \in \mathcal{K}$, the cost c_r^k of the route-vehicle pair (r, k) is given by $c_r^k := f_k + v_k \alpha_k$, where f_k denotes the fixed cost, v_k denotes the variable cost, and α_k denotes the length of the route r. A smaller vehicle is always less expensive than a larger vehicle, i.e., for $k_1, k_2 \in \mathcal{K}$, the inequality $D_{k_1} < D_{k_2}$ implies the inequalities $f_{k_1} < f_{k_2}$ and $v_{k_1} < v_{k_2}$. For each test instance in CT12, vehicles are denoted by letters, so that, e.g., the five vehicles of test instance Choi3 are denoted A–E, where A (E) corresponds to the vehicle with the smallest (largest) capacity.

Each of the twelve test instances in CT12 contains three to six different vehicle types and 20–100 customers. The instances Choi3–Choi6 share customer nodes $\mathcal{N}_0$ and customer demands. The difference between the instances Choi3–Choi4 and Choi5–Choi6 is that the depot has been moved. The instances T13–T14 also have the same node specifications, as do the instances T15–T16, T17–T18, and T19–T20, respectively.

To test the *many*-hVRP problem formulation, the instance set CT12 has been extended to include vehicles with capacities in the ranges of the corresponding original instances Choi3–Choi6 and T13–T20. All integer values in this capacity interval have been included, since the demands of the customers are all integer. The fixed and variable costs have been extended accordingly using the spline function in Matlab. The new set of test instances is denoted CT12EXT, and the individual test instances are denoted Choi3EXT–Choi6EXT, T13EXT–T20EXT. Consequently, the test instances in CT12EXT comprise 91–381 different vehicle capacities. Accordingly, e.g., does Choi3EXT includes 101 vehicle types with capacities ranging from 20 to 120, fixed cost ranging from 20 to 225,

[8]The instances CT12 can be constructed as follows; the instances in G12, are constructed using the instances which were introduced by Christofides et al. ([25])—these are downloaded from [28]. The networks of customer nodes of CT12 are constructed according to the instructions given by Golden et al. [19] for G12, while the vehicle specifications for CT12 are set according to [24, Table 1] (which are also found in [23, Table 2]).

and variable cost ranging from 1.0 to 2.5. For each instance in CT12EXT, vehicles are denoted by numbers, with the lowest (highest) number corresponding to the vehicle with the smallest (largest) capacity.

For each test instance in CT12EXT, all unnecessary vehicle capacities were excluded from the set of vehicles according to the following procedure: A vehicle capacity value is excluded from the test instance if no route in the set $\cup_{k \in \mathcal{K}} \mathcal{R}_k$ has a total customer demand equal to this value (except for the smallest value). This since any route driven by the "excluded vehicle" (capacity) can be driven by a smaller vehicle—at a lower cost. No vehicle capacity could, however, be excluded from any of the twelve instances.

Instance	Customers	Vehicles			
	# customers	# vehicles	Capacity	Fixed cost	Variable cost
Choi3	20	5	20 – 120	20 – 225	1.0 – 2.5
Choi4	20	3	60 – 150	1000 – 3000	1.0 – 4.0
Choi5	20	5	20 – 120	20 – 225	1.0 – 2.5
Choi6	20	3	60 – 150	1000 – 3000	1.0 – 4.0
T13	50	6	20 – 200	20 – 400	1.0 – 3.2
T14	50	3	120 – 300	1000 – 3500	1.0 – 1.4
T15	50	3	50 – 160	50 – 160	1.0 – 2.0
T16	50	3	40 – 140	100 – 400	1.0 – 2.1
T17	75	4	50 – 120	25 – 320	1.0 – 1.8
T18	75	6	20 – 400	10 – 800	1.0 – 3.2
T19	100	3	100 – 300	500 – 1200	1.0 – 1.7
T20	100	3	60 – 200	100 – 500	1.0 – 2.0

Table 1: The individual test instances, Choi3–Choi6 and T13–T20, of the set CT12. "# customers" refers to the number or customer nodes, and "# vehicles" refers to the number of vehicle types in each instance. The capacity, fixed cost, and variable cost ranges are given for each set of vehicle types.

3.2 Load dependent vehicle routing problems

Most vehicle routing problem settings restrict variable traveling costs to depend only on the distance traveled. Since traveling costs depend on many factors, this may be too limiting. According to Xiao et al. ([29]), factors determining real life traveling costs can be divided into two groups, of which the first includes distance, but also speed, load, fuel consumtion, and road conditions. The factors in the second group are less related to the route traveled and include vehicle depreciation and maintenance costs, wages, and taxes. Most factors in the first group are related to fuel consumption, which is highly dependent on the distance traveled and the vehicle load. They argue, using statistical data, that fuel consumption for a homogeneous vehicle routing problem can be modeled by the following objective function

$$\min_{\mathbf{x},\mathbf{e}} \left(\sum_{j \in \mathcal{N}} f x_{0j} + \sum_{(i,j) \in \mathcal{A}} (c_{ij} x_{ij} + b_{ij} e_{ij}) \right) \tag{18}$$

and constraints corresponding to (12b)–(12h), but considering only one vehicle type. Analogously to the flow formulation (12) for hVRP, the variables are defined as

$$x_{ij} := \begin{cases} 1, & \text{if arc } (i,j) \text{ is used} \\ 0, & \text{otherwise} \end{cases} \quad (i,j) \in \mathcal{A},$$

and $e_{ij} :=$ flow of goods from node i to j, $(i,j) \in \mathcal{A}$. Also, the parameters c_{ij}, $(i,j) \in \mathcal{A}$, are routing (variable) costs and the parameter f is a fixed cost. In addition, there are parameters b_{ij}, $(i,j) \in \mathcal{A}$, which determines the contribution of the load across each arc to the total cost. Here c_{ij} and b_{ij} are parameters not only depending on distance but also on fuel consumption rates and fuel prices, and the load variable e_{ij} is measured in weight units. The authors of [29] use a simulated annealing based heuristic algorithm to solve this homogeneous vehicle routing problem.

The objective function (18) can easily be adjusted for the column generation subproblems of hVRP or *many*-hVRP by letting f, c_{ij}, and b_{ij} depend on the vehicle type, since each vehicle type can be treated separately in the subproblems. This has been done in this thesis; see Section 4.3. It would be more difficult to use this type of load dependent cost function for a formulation such as the flow formulation (12) of hVRP. Making the parameters b_{ij} depend on the vehicle type would require the variables e_{ij}^k, representing the load on the vehicle of type k on link (i,j), $(i,j) \in \mathcal{A}$, $k \in \mathcal{K}$.

4 Problem formulation and solution methods for the *many*-hVRP

The *many*-hVRP problem, for which we wish to find a suitable model and solution method, is the version of the standard hVRP where the number of different vehicle types is very large. If some of the parameters that determine the configurations of the vehicles can be chosen from a continuous interval, there may even be an infinite number of possible vehicle types.

The standard test instances for hVRP, some of which were presented in Section 3.1.3, contain relatively few vehicle types. To the best of our knowledge, hVRP with a very large set of vehicle types—here denoted *many*-hVRP—have not been studied before. To efficiently handle *many*-hVRP, solution techniques tailored for the large set of vehicles are necessary. Starting from the solution technique for hVRP from [24], as presented in Section 3.1.2, a column generation algorithm has been implemented for *many*-hVRP; it is presented in Section 4.1. This implementation uses a straightforward adaptation of the hVRP model (15), and is therefore denoted the *straightforward model*. A slightly different model, in which the number of vehicle types that can be used in a feasible solution is restricted, is presented in Section 4.2. There are two main reasons for formulating *many*-hVRP using this model, denoted the *restricted model*. The first reason is simply that a restriction on the number of vehicle types in a solution may be a desirable feature in a solution, when the set of vehicle types is very large. A fleet with fewer vehicle types may for instance be more flexible. The second reason is due to algorithmic considerations. The restriction of the number of vehicle types used in a solution enables a Benders' decomposition of an LP relaxation of the model, which then results in a subproblem having the same form as the master problem in a column generation approach to the straightforwad model. Benders subproblem can thus be solved using the column generation algorithm presented in Section 4.1 for the straightforward model, but with a smaller set of vehicle types. The number of vehicle types allowed in a feasible solution to the restricted model determines the size of the set of vehicle types considered in Benders subproblem, influencing its computational complexity.

To illustrate how these models adapt to more complex problem settings, the straightforward and the restricted model, as well as the corresponding solution algorithms have been adapted to consider load dependent costs, following the model presented in Section 3.2. The details are presented in Section 4.3.

To the best of our knowledge, the restriction on the number of vehicle types used in a solution to the restricted model has not been used in previous models for hVRP, probably because the number of vehicle types considered is usually small. Neither have we encountered the application of Benders' algorithm to hVRP in the literature. We envision that the algorithms developed in this thesis—with some modifications of the implementation of the sets of vehicle types which showed to fit well within our current solution framework; see Section 6—can be successfully applied to real life instances of larger size than the benchmarks CT12 and CT12EXT here used to test the algorithms.

4.1 The straightforward model with column generation

The heterogeneous vehicle routing problem can, as shown in Section 3.1.1, be formulated as the set-partitioning model (15). Hence, it can also be used to model the *many*-hVRP. Using the model (15) has the advantage that standard solution methods that have been proven successful for hVRP can be applied also to *many*-hVRP. We denote the model (15), when used for *many*-hVRP, as the *straightforward model*. A version of the column generation approach implemented in [24]—which has proved successful for hVRP; see Sections 3.1.2 and 3.1.3—has been implemented for *many*-hVRP, and is presented here, along with some adaptations to the case of many different vehicle types.

To be able to use column generation, as detailed in Appendix B.2, the binary restrictions on the variables x_r^k in (15) are relaxed, resulting in the following column generation master problem:

$$z^*_{\mathrm{LP}} := \min_{\mathbf{x}} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} c_r^k x_r^k, \tag{19a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} \delta_{ir}^k x_r^k = 1, \qquad i \in \mathcal{N}_0, \tag{19b}$$

$$x_r^k \geq 0, \qquad r \in \mathcal{R}_k, k \in \mathcal{K}. \tag{19c}$$

The corresponding restricted master problem, with $\widetilde{\mathcal{R}}_k \subseteq \mathcal{R}_k$, $k \in \mathcal{K}$, is formulated as

$$\text{(RMP)} \qquad z^*_{\mathrm{RMP}} := \min_{\mathbf{x}} \sum_{k\in\mathcal{K}} \sum_{r\in\widetilde{\mathcal{R}}_k} c_r^k x_r^k, \tag{20a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{r\in\widetilde{\mathcal{R}}_k} \delta_{ir}^k x_r^k = 1, \qquad i \in \mathcal{N}_0, \tag{20b}$$

$$x_r^k \geq 0, \qquad r \in \widetilde{\mathcal{R}}_k, k \in \mathcal{K}, \tag{20c}$$

and its linear programming dual is formulated as

$$\begin{aligned} \text{(RMPDual)} \qquad & \max_{\boldsymbol{\pi}} \sum_{i\in\mathcal{N}_0} \pi_i, \\ & \text{s.t.} \sum_{i\in\mathcal{N}_0} \pi_i \delta_{ir}^k \leq c_r^k, \qquad r \in \widetilde{\mathcal{R}}_k, k \in \mathcal{K}. \end{aligned}$$

In each iteration of the column generation algorithm, the sets $\widetilde{\mathcal{R}}_k$ are expanded by routes $r \in \mathcal{R}_k \setminus \widetilde{\mathcal{R}}_k$, having low reduced costs. For $r \in \mathcal{R}_k$, $k \in \mathcal{K}$, the reduced cost for the variable x_r^k, denoted $\hat{c}_r^k$, is given by

$$\hat{c}_r^k = c_r^k - \sum_{i\in\mathcal{N}_0} \pi_i^* \delta_{ir}^k, \tag{21}$$

where $\boldsymbol{\pi}^*$ denotes an optimal solution to (RMPDual). As defined in Section 3.1.1, $c_r^k := f_k + \sum_{h=1}^{H} c_{i_{h-1}i_h}^k$, where $r = (i_0, i_1, \dots, i_{H-1}, i_H)$. Let

$$\begin{aligned} \hat{c}_{ij}^k &:= c_{ij}^k - \pi_j^*, \qquad && (i,j) \in \mathcal{A}, j \in \mathcal{N}_0, \\ \hat{c}_{i0}^k &:= c_{i0}^k, && (i,0) \in \mathcal{A}. \end{aligned}$$

The reduced cost $\hat{c}_r^k$ can then be expressed as

$$\hat{c}_r^k := f_k + \sum_{h=1}^{H} c_{i_{h-1}i_h}^k - \sum_{h=1}^{H-1} \pi_{i_h}^* = f_k + \sum_{h=1}^{H} \hat{c}_{i_{h-1}i_h}^k.$$

The subproblem that must be solved in order to find the route-vehicle pair (r,k) corresponding to the variable x_r^k, $r \in \mathcal{R}_k$, $k \in \mathcal{K}$, having the lowest reduced cost $\hat{c}_r^k$, can be divided into one subproblem for each vehicle type, as done in [24]. In the last column generation iteration binary restrictions on the variables x_r^k are added to (RMP), to obtain a feasible solution to (15) as explained in Appendix B.2. The column generation algorithm is detailed in Algorithm 1.

1. Initialization: Choose the subsets $\widetilde{\mathcal{R}}_k \subseteq \mathcal{R}_k$, $k \in \mathcal{K}$, such that the restricted master problem (RMP) is feasible, and such that $\sum_{k\in\mathcal{K}} |\widetilde{\mathcal{R}}_k| \geq |\mathcal{N}_0|$.
2. Solve (RMP).
3. Solve, for each $k \in \mathcal{K}$, the subproblem (where $r = (i_0, i_1, \dots, i_H)$)

   $$(\hat{c}^k)^* := \min_{r\in\mathcal{R}_k} \{\hat{c}_r^k\} = f_k + \min_{r\in\mathcal{R}_k} \left\{ \sum_{h=1}^{H} c_{i_{h-1}i_h}^k - \sum_{h=1}^{H-1} \pi_{i_h}^* \right\},$$

   where $\boldsymbol{\pi}^*$ is an optimal solution to (RMPDual).
4. If $\mathcal{K}^- := \{k \in \mathcal{K} \mid (\hat{c}^k)^* < 0\} = \emptyset$, go to Step 5. Otherwise, for each $k \in \mathcal{K}^-$, include the index

   $$\tilde{r} \in \arg\min_{r\in\mathcal{R}_k} \{\hat{c}_r^k\},$$

   in the set $\widetilde{\mathcal{R}}_k$. Then go to Step 2.
5. No variables in (19) have negative reduced costs, so the current solution to (RMP) is optimal in (19), i.e., $z_{\text{LP}}^* = z_{\text{RMP}}^*$. Solve the current (RMP) with binary restrictions on x_r^k, $r \in \widetilde{\mathcal{R}}_k$, $k \in \mathcal{K}$. The solution is a feasible in the original problem (15), and $z_{\text{RMP}}^* \geq z^*$.

Algorithm 1: Column generation algorithm.

We have implemented some modifications of Algorithm 1 in order to speed up the convergence to the optimal value of (19). It is not necessary to solve the column generation subproblem(s) to optimality, except for in the last iteration, when verifying that the solution to (RMP) is optimal in the complete master problem (19); (see [8, 30]). Therefore, a route $r \in \mathcal{R}_k \backslash \widetilde{\mathcal{R}}_k$, such that the variable x_r^k has a negative but not necessarily minimal reduced cost $\hat{c}_r^k = c_r^k - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir}^k$ can be added to $\widetilde{\mathcal{R}}_k$ in Step 4. In addition, since the original subproblem to find a variable x_r^k with minimal reduced cost has been divided into one subproblem for each vehicle type k, it is not necessary to solve the subproblem for every vehicle type every iteration. Instead a so called *partial column generation* can be employed, such that only a subset of the subproblems are considered each iteration. However, at least one variable x_r^k with a negative reduced cost must be added to (RMP), provided that such a variable exists ([8]). Since there are as many subproblems as there are vehicle types, and the set of vehicle types is assumed to be very large, using partial column generation can be beneficial. Details on the solution of the subproblems are presented in Section 4.1.1.

In Section 4.1.2, upper and lower bounds, $\bar{z}$ and $\underline{z}$, on z_{LP}^* are derived. Since z_{LP}^* is a lower bound on z^*, $\underline{z}$ is also a lower bound on z^*. The upper and lower bounds can be used to terminate the column generation prior to has convergence, i.e., when $\bar{z} - \underline{z} \leq \varepsilon$, for some pre-determined value $\varepsilon > 0$. When the column generation algorithm has converged, i.e., when the optimal solution to the restricted master problem (RMP) has been verified as an optimal solution to the complete master problem (19), then it holds that $\bar{z} = z_{\text{LP}}^* = \underline{z}$.

4.1.1 Subproblem

For a fixed value of $k \in \mathcal{K}$, the column generation subproblem to find

$$(\hat{c}^k)^* := \min_{r \in \mathcal{R}_k} \left\{ \hat{c}_r^k \right\} = \min_{r \in \mathcal{R}_k} \left\{ c_r^k - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir}^k \right\} = f_k + \min_{r \in \mathcal{R}_k} \left\{ \sum_{h=1}^{H} \hat{c}_{i_{h-1} i_h}^k \right\} \quad (22)$$

is an elementary shortest path problem with resource constraints, since for $k \in \mathcal{K}$ the set $\mathcal{R}_k$ consist of elementary routes $r = (i_0, i_1, \ldots, i_{H-1}, i_H)$, starting and ending at the depot, and that must satisfy $\sum_{h=1}^{H-1} d_{i_h} \leq D_k$ (i.e., the total demand of the customer nodes visited by route r can not be greater than the capacity of vehicle k).

Two different approaches have been used to solve (22) in this thesis, one using AMPL and CPLEX, and one using dynamic programming in Matlab. Both approaches have been implemented in such a way that the solution algorithm can be interrupted before an optimal solution to the subproblem has been found, either when a predefined time limit has been exceeded, or after a certain number of routes with negative reduced cost has been found. It was noted that often when no route with negative reduced cost had been found for a given subproblem in the later iterations, then no route with negative reduced cost was found in the next couple of iterations. Therefore, a kind of *tabu-strategy of partial column* *generation* was also implemented for the subproblems (see [31] for an introduction to tabu search), according to the following. If, for one specific subproblem, no routes with negative reduced cost have been found during a pre-determined number of consecutive column generation iterations, that specific subproblem is not solved for a pre-determined number of iterations. When the extended test instances CT12EXT were used, it was also noted that subproblems corresponding to vehicles with similar capacities sometimes yielded the same routes—both when solved to optimality and when not. The tabu-strategy was then updated, so that only the route-vehicle pair with the lowest reduced cost was added to the restricted master problem, and the other subproblems which yielded the same route were recorded as not providing a route with negative reduced cost—thus potentially leading to them not being solved for a number of iterations.

Mathematical formulations implemented in AMPL and CPLEX:

Below, two mathematical formulations of the problem (22) are presented; these can be directly implemented in AMPL and CPLEX. The model (23) has been adapted to the column generation subproblem (22), using the flow formulation (12) of hVRP. The constraints (23c)–(23e) have been added, and the constraints (12d)–(12g) have been replaced with the constraints (23f)–(23h). The variables e_{ij}, $(i,j) \in \mathcal{A}$, now represent an artificial flow, where each node that is visited by a route (in terms of the variables x_{ij}) has a demand of one unit of goods. The variables e_{ij} are needed to make sure that the feasible set in (23) consists of connected paths, and can assume values in the range $[0, N]$, where N is the number of customers. The resulting formulation is given by

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{e}} \quad & \sum_{(i,j)\in\mathcal{A}} \hat{c}^k_{ij} x_{ij}, && && \text{(23a)}\\
\text{s.t.} \quad & \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = 0, && j \in \mathcal{N}_0, && \text{(23b)}\\
& \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} \le 1, && j \in \mathcal{N}_0, && \text{(23c)}\\
& \sum_{i\in\mathcal{N}:(0,i)\in\mathcal{A}} x_{0i} = 1, && && \text{(23d)}\\
& \sum_{i\in\mathcal{N}:(i,0)\in\mathcal{A}} x_{i0} = 1, && && \text{(23e)}\\
& \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} e_{ij} - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} e_{ji} = \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij}, && j \in \mathcal{N}_0, && \text{(23f)}\\
& x_{ij} \le e_{ij} \le N x_{ij}, && (i,j) \in \mathcal{A}, && \text{(23g)}\\
& \sum_{i\in\mathcal{N}_0} \sum_{j\in\mathcal{N}} d_i x_{ij} \le D_k, && && \text{(23h)}\\
& x_{ij} \in \{0,1\}, && (i,j) \in \mathcal{A}. && \text{(23i)}
\end{aligned}
$$

The objective (23a) is to minimize the reduced route costs $\hat{c}_{ij}$ only, since the fixed cost f_k will be present in any route. The constraints (23b) ensure that the vehicle departs from each customer that it arrives at. The constraints (23c) states that a customer may be visited at most once, while the constraints (23d)–(23e) ensure that at least one customer is visited. The constraints (23f)–(23g) are artificial constraints that are needed to make sure that the route is connected. Since the constraints (23f) should hold for all the customer nodes (i.e. $\mathcal{N}_0$), only cycles (with respect to the variables x_{ij}) that include the depot are feasible, and the highest value taken by any variable e_{ij}, $(i,j) \in \mathcal{A}$, will be e_{0j}, for the unique arc $(0,j) \in \mathcal{A}$ for which $x_{0j} = 1$. The constraints (23h) impose the vehicle capacity restrictions. For $\mathbf{x}^*$ being an optimal solution to the model (23), we have that $(\hat{c}^k)^* = f_k + \sum_{(i,j)\in\mathcal{A}} \hat{c}^k_{ij} x^*_{ij}$.

We have adapted the formulation of the VRP with wime windows [32, p. 58]—which employs variables representing time instead of flow variables—for the column generation subproblem (22). Here the variables e_{ij} and the constraints (23f)–(23g) are replaced by the variables s_i, $i \in \mathcal{N}$, (representing order with respect to time) and

$$s_i + 1 \le s_j + N(1 - x_{ij}), \qquad i \in \mathcal{N}, j \in \mathcal{N}_0 : (i,j) \in \mathcal{A}, \tag{24a}$$

$$0 \le s_i \le N, \qquad i \in \mathcal{N}, \tag{24b}$$

resulting in the formulation (23a)–(23e), (23h)–(23i), (24). These constraints are analogous to modelling very large time-windows, which are not restrictive in the sense that they do not cut away any solution that is feasible (the time-windows are $[0, N]$ for each customer node, where N is the number of customers, and it takes 1 time-unit to cross each arc in the network $\mathcal{A}$). The constraints (24) ensure that the resulting feasible set consists of connected elementary routes. The time variables s_i are fewer than the flow variables e_{ij}; their use may therefore lead to a better formulation. Further, the alternative constraints make the constraints (23c) redundant; they are not included in the formulation in [32, p. 58]. However, they strengthen the formulation (in terms of a tighter representation of the convex hull of the set of feasible solutions) and—according to the following paragraph— yield great improvements in solution time when applied to the subproblem (22).

The formulations (23) and (23a)–(23e), (23h)–(23i), (24) have been solved by AMPL and CPLEX. Our preliminary tests indicated that the two formulations are rather similar with respect to solution time, although the latter formulation with constraints (23c) removed is a lot more time-consuming—see Section 5.1.3.

Dynamic programming:

Instead of formulating integer linear programming models that can be solved by AMPL and CPLEX, more efficient solution methods, specifically tailored for shortest path problems with resource constraints, can be applied. Dynamic programming algorithms are especially suited for column generation subproblems arising from vehicle routing problems, as discussed in Section 2.2. Solving the subproblem (22), for a specific vehicle k using dynamic programming methods can be done as in Algorithm 2.

1. Initialization: Let $\mathcal{P}_{\text{NPP}} := \{(1)\}$, $\mathcal{P}_{\text{PP}} := \emptyset$, $\mathcal{P}_{\text{FP}} := \emptyset$.
2. Pick one path P_{1i} from the set $\mathcal{P}_{\text{NPP}}$ of non-processed paths and extend it to all nodes $j \in \mathcal{N}_0$ for which (P_{1i}, j) is a feasible path (a path is feasible if the total demand of all nodes visited is not greater than the capacity of the vehicle (i.e., resource feasible) and no customer node has been visited more than once (i.e., the path is elementary)).
3. Find paths in $\mathcal{P}_{\text{NPP}} \cup \mathcal{P}_{\text{PP}}$ that are dominated by, or that dominates the extended paths from Step 2. Remove the dominated paths from their respective sets. Include the extended paths that were not dominated by any path in $\mathcal{P}_{\text{NPP}} \cup \mathcal{P}_{\text{PP}}$ in $\mathcal{P}_{\text{NPP}}$, and include the path P_{1i} in the set $\mathcal{P}_{\text{PP}}$ of processed paths.
4. Extend the path P_{1i} to include the depot, and include it in the set $\mathcal{P}_{\text{FP}}$ of full paths, if it has a lower objective value than all paths in $\mathcal{P}_{\text{FP}}$.
5. If $\mathcal{P}_{\text{NPP}} = \emptyset$, an optimal route is found in $\mathcal{P}_{\text{FP}}$, and the algorithm terminates. Otherwise, go to Step 2. Alternatively: Terminate the algorithm when a time limit has been exceeded, or when a pre-determined number of paths have been processed.

Algorithm 2: Dynamic programming algorithm for the column generation subproblem.

Since the subproblems (22) do not have to be solved to optimality in every iteration of the column generation algorithm, Algorithm 2 may be terminated before $\mathcal{P}_{\text{NPP}}$ is empty. The dynamic programming algorithm is allowed to terminate when a full path with a negative reduced cost that is lower than some given threshold is found, and/or when a time limit has been exceeded. It is often the case that an optimal route is found, but not verified, early in the solution course. Terminating the algorithm before optimality is verified may then substantially lower the solution time for the column generation.

The path $P_{1i} \in \mathcal{P}_{\text{NPP}}$ to be extended in Step 2 can be chosen according to different criteria. Our implementation chooses the path having the lowest objective value. Also, paths that can not be extended to any customer nodes, i.e., paths for which the sum of the demand of its visited customers exceeds the value $(D_k - \min_{i \in \mathcal{N}_0} d_i)$, are processed in Step 4 instead of being included in $\mathcal{P}_{\text{NPP}}$.

In Step 3, the domination criterion in Claim 1 (Section 2.2.1) is used. Hence, path P_{1i} dominates path P^*_{1i} if (i) the sum of the customer demands of the nodes visited by path P_{1i} is less than or equal to that of P^*_{1i}, (ii) the cost of P_{1i} is less than or equal to that of P_{1i}^*, and (iii) the nodes visited by the path P_{1i} form a subset of the nodes visited by the path P_{1i}^*. With this domination criterion, the dynamic programming algorithm is guaranteed to find the optimal route. The criterion has been strengthened by including in the set of nodes visited by a path also the nodes that are not reachable due to the capacity limitation of the vehicle. A relaxation of the domination criterion that has been tested is to disregard the criterion (iii), concerning the sets of visited nodes. In this case the algorithm is not guaranteed to provide an optimal route, but it may find good routes faster. Preliminary tests, comparing the domination criterion of Claim 1 with the strengthened domination criterion and the relaxed domination criterion did not show any great difference in solution time. The exact domination criterion of Claim 1 has been used in the standard implementation.

There are many other ways to make the dynamic programming algorithm more efficient; see Sections 2.2.2 and 2.2.3. In addition to the improvements listed in those sections, information from the solution course of one subproblem should be used to warm-start the dynamic programming for other subproblems with similar vehicle types—using the fact that there are many subproblems with similar parameter values. In our current implementation, solving the subproblems by AMPL and CPLEX has the advantage that the solutions of subproblems is warm-started. The dynamic programming implementation is made in Matlab. Great savings in solution time would probably be achieved if it were instead implemented in, e.g., C. Even so, our Matlab implementation does not perform much worse than the AMPL implementation of (23) and (23a)–(23e), (23h)–(23i), (24); see Section 5.1.1 for details.

Among these implementations, however, AMPL is sometimes much more efficient in solving subproblems to optimality. Even though the dynamic programming algorithm manages to find an optimal route early in the solution course, preliminary tests indicated that it may take much more time to verify optimality compared to AMPL. Therefore, dynamic programming is always substituted by AMPL in the latter part of the column generation solution course, when optimality has to be verified.

4.1.2 Upper and lower bounds

We next derive upper and lower bounds on the optimal value z_{LP}^* of the master problem (19). These can be used to terminate the column generation algorithm prior to convergence, in which case a measure of the quality of the best feasible solution to the problem (19) found so far is given by the lower bound.

An upper bound $\bar{z}$ on the optimal value z_{LP}^* of (19) is given by the solution to the (RMP) in each iteration of the column generation, since the optimal solution to (RMP) is feasible in (19). For such an optimal solution $\big((\bar{x}_r^k)^*\big)_{r\in\widetilde{\mathcal{R}}_k, k\in\mathcal{K}}$, it holds that

$$z_{\text{LP}}^* \leq \bar{z} := z_{\text{RMP}}^* = \sum_{k\in\mathcal{K}} \sum_{r\in\widetilde{\mathcal{R}}_k} c_r^k (\bar{x}_r^k)^*.$$

By strong duality (e.g., [4, p. 248]), it holds that $\bar{z} = \sum_{i\in\mathcal{N}_0} \pi_i^*$ where $\boldsymbol{\pi}^*$ is optimal in (RMPDual) ([8]).

The following reasoning provides a lower bound on z^*_{LP}. Given an optimal solution $\left((x^k_r)^*\right)_{r\in\mathcal{R}_k,k\in\mathcal{K}}$ to (19), it holds that

$$z^*_{\mathrm{LP}} - \bar{z} = \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k} c^k_r (x^k_r)^* - \sum_{i\in\mathcal{N}_0} \pi^*_i.$$

Since it holds that $\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k} \delta^k_{ir}(x^k_r)^* = 1$, $i \in \mathcal{N}_0$, we have that

$$\begin{aligned} z^*_{\mathrm{LP}} - \bar{z} &= \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k} c^k_r (x^k_r)^* - \sum_{i\in\mathcal{N}_0} \left(\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k} \delta^k_{ir}(x^k_r)^*\right)\pi^*_i \\ &= \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k} \left(c^k_r - \sum_{i\in\mathcal{N}_0}\delta^k_{ir}\pi^*_i\right)(x^k_r)^* \\ &\geq \left(\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k}(x^k_r)^*\right)\hat{c}, \end{aligned}$$

where $\hat{c} := \min_{r\in\mathcal{R}_k,k\in\mathcal{K}} \left\{c^k_r - \sum_{i\in\mathcal{N}_0}\delta^k_{ir}\pi^*_i\right\}$ is the lowest reduced cost over all indices $r \in \mathcal{R}_k$ and $k \in \mathcal{K}$, which is found by solving subproblem (22) for all $k \in \mathcal{K}$. Since it holds that $\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k}(x^k_r)^* \leq |\mathcal{N}_0|$, and $\hat{c} \leq 0$ it follows that

$$z^*_{\mathrm{LP}} - \bar{z} \geq \hat{c}\,|\mathcal{N}_0|.$$

Hence a lower bound on z^*_{LP} is given by $\underline{z} := \hat{c}\,|\mathcal{N}_0| + \bar{z}$. This implies that the column generation can be terminated when $\hat{c}$ is small enough, since $\hat{c}\,|\mathcal{N}_0| \leq \varepsilon$ and $\varepsilon > 0$ implies that $\bar{z} - \underline{z} \leq \varepsilon$.

The result that $z^*_{\mathrm{LP}} \geq \kappa\,\hat{c} + \bar{z}$, where $\kappa \geq \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_k}(x^k_r)^*$ is given without proof in [8]. A derivation of the lower bound $\underline{z}$, similar to the one above, is found in [9, Chapter 3.7] for $\kappa = 1$.

4.2 The restricted model with Benders' decomposition and column generation

We next present a model for the *many*-hVRP, which is an alternative to the straightforward model (cf. (15)) presented in Section 4.1. We have formulated this model for the *many*-hVRP; to the best of our knowledge it has not previously been used for hVRP. In this new formulation, a constraint is added to (15) limiting the number of vehicle types that can be used. This is intended to make the model more flexible, since in some applications a fleet with a limited number of vehicle types is to prefer. The constraint is also intended to allow for a Benders' decomposition of the problem. The new model is called the *restricted model*.

Define the parameters M = the allowed number of vehicles of each type, and C = the allowed number of different vehicle types, denoted the *vehicle type* *limit*, and the variables

$$y^k := \begin{cases} 1, & \text{if vehicle type } k \text{ is allowed,} \\ 0, & \text{otherwise,} \end{cases} \qquad k \in \mathcal{K}$$

The restricted model is then given by

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \ & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} c_r^k x_r^k, && && \text{(25a)}\\
\text{s.t.} \ & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} \delta_{ir}^k x_r^k = 1, && i \in \mathcal{N}_0, && \text{(25b)}\\
& \sum_{r\in\mathcal{R}_k} x_r^k \leq M y^k, && k \in \mathcal{K}, && \text{(25c)}\\
& \sum_{k\in\mathcal{K}} y^k \leq C, && && \text{(25d)}\\
& x_r^k \in \{0,1\}, && r \in \mathcal{R}_k, k \in \mathcal{K}, && \text{(25e)}\\
& y^k \in \{0,1\}, && k \in \mathcal{K}. && \text{(25f)}
\end{aligned}$$

The difference between the formulation (25) and (15), is that the constraints (25c)–(25d) and the binary variables y^k, $k \in \mathcal{K}$, have been added; the constraints (25c) limit the number of vehicles of each type, and the constraint (25d) sets a limit on the number of different vehicle types that may be used. If the formulation (25) and the formulation (15) share an optimal solution, for a specific problem instance, then the vehicle type limit C is said to be non-restrictive.

4.2.1 Outline of Benders' decomposition applied to the restricted model

To use Benders' decomposition, the binary requirements on the variables x_r^k are relaxed, resulting in the following formulation[9]

$$\begin{aligned}
v^* := \min_{\mathbf{x},\mathbf{y}} \ & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} c_r^k x_r^k, && && \text{(26a)}\\
\text{s.t.} \ & \sum_{k\in\mathcal{K}} \sum_{r\in\mathcal{R}_k} \delta_{ir}^k x_r^k = 1, && i \in \mathcal{N}_0, && \text{(26b)}\\
& \sum_{r\in\mathcal{R}_k} x_r^k \leq M y^k, && k \in \mathcal{K}, && \text{(26c)}\\
& \sum_{k\in\mathcal{K}} y^k \leq C, && && \text{(26d)}\\
& x_r^k \geq 0, && r \in \mathcal{R}_k, k \in \mathcal{K}, && \text{(26e)}\\
& y^k \in \{0,1\}, && k \in \mathcal{K}. && \text{(26f)}
\end{aligned}$$

[9]If the formulation (26) and the formulation (19) share an optimal solution for a specific problem instance, then the vehicle type limit C is also said to be non-restrictive.

In model (26), the variables y^k, $k \in \mathcal{K}$ are considered complicating. Define the set

$$R := \left\{ (y^k)_{k=1}^K \in \{0,1\}^K \;\middle|\; \sum_{k \in \mathcal{K}} y^k \leq C, \sum_{k \in \mathcal{K}_{\text{capacity}}} y^k \geq 1 \right\},$$

for $\mathcal{K}_{\text{capacity}} := \{k \in \mathcal{K} \mid D_k \geq \max_{i \in \mathcal{N}_0}\{d_i\}\}$. The set $\mathcal{K}_{\text{capacity}}$ consists of those vehicle types k that can service all customers, thus the set R consists of those variable values of the complicating variables $\mathbf{y}$ for which the remaining problem (26), in the variables $\mathbf{x}$, has at least one feasible solution.

When applying Benders' decomposition to (26) (see Section 2.3) the Benders subproblem for fixed values of the variables $\mathbf{y} := (\tilde{y}^k)_{k \in \mathcal{K}}$ appears as

$$\begin{aligned}
(\text{BendersSP}(\tilde{\mathbf{y}})) \qquad w^*(\tilde{\mathbf{y}}) := \min_{\mathbf{x},\mathbf{y}} & \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} c_r^k x_r^k, \\
\text{s.t.} & \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} \delta_{ir}^k x_r^k = 1, && i \in \mathcal{N}_0, \\
& \sum_{r \in \mathcal{R}_k} x_r^k \leq M \tilde{y}^k, && k \in \mathcal{K}, \\
& x_r^k \geq 0, && r \in \mathcal{R}_k, k \in \mathcal{K},
\end{aligned}$$

and the corresponding dual problem is given by

$$\begin{aligned}
(\text{BendersSPDual}(\tilde{\mathbf{y}})) \quad \max_{\boldsymbol{\pi},\boldsymbol{\gamma}} & \left(\sum_{i \in \mathcal{N}_0} \pi_i + \sum_{k \in \mathcal{K}} M \tilde{y}^k \gamma_k \right), && && \text{(27a)} \\
\text{s.t.} & \sum_{i \in \mathcal{N}_0} \pi_i \delta_{ir}^k + \gamma_k \leq c_r^k, && r \in \mathcal{R}_k, k \in \mathcal{K}, && \text{(27b)} \\
& \gamma_k \leq 0, && k \in \mathcal{K}. && \text{(27c)}
\end{aligned}$$

We denote the feasible set (corresponding to the feasible set defined by the constraints (8b)–(8c) in Section 2.3)

$$F_{\text{BendersSPDual}} := \left\{ (\boldsymbol{\pi}, \boldsymbol{\gamma}) \in \mathbb{R}^{|\mathcal{N}_0| + |\mathcal{K}|} \;\middle|\; (\boldsymbol{\pi}, \boldsymbol{\gamma}) \text{ satisfies (27b)–(27c)} \right\}.$$

Defining the constrained set of vehicles as $\widetilde{\mathcal{K}}(\tilde{\mathbf{y}}) := \{k \in \mathcal{K} \mid \tilde{y}^k = 1\}$, an equivalent formulation to (BendersSP($\tilde{\mathbf{y}}$)) is

$$\begin{aligned}
\min_{\mathbf{x}} & \sum_{k \in \widetilde{\mathcal{K}}(\tilde{\mathbf{y}})} \sum_{r \in \mathcal{R}_k} c_r^k x_r^k, && && \text{(28a)} \\
\text{s.t.} & \sum_{k \in \mathcal{K}(\tilde{\mathbf{y}})} \sum_{r \in \mathcal{R}_k} \delta_{ir}^k x_r^k = 1, && i \in \mathcal{N}_0, && \text{(28b)} \\
& \sum_{r \in \mathcal{R}_k} x_r^k \leq M, && k \in \widetilde{\mathcal{K}}(\tilde{\mathbf{y}}), && \text{(28c)} \\
& x_r^k \geq 0, && r \in \mathcal{R}_k, k \in \widetilde{\mathcal{K}}(\tilde{\mathbf{y}}). && \text{(28d)}
\end{aligned}$$

If M is chosen large enough (e.g., $M = |\mathcal{N}_0|$), then the constraints (28c) are not restrictive and can be removed from this formulation. We assume from now on that M has been chosen so that it is not restrictive. The models (28) and (19) are thus equivalent, except that the set of allowed vehicle types is smaller in the former, and Algorithm 1 can be applied to find an optimal solution to (28a)–(28b), (28d) (which will give an optimal solution to (BendersSP($\widetilde{\mathbf{y}}$)) by setting $x_r^k := 0$, $r \in \mathcal{R}_k$, $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$).

Each time the subproblem (BendersSP($\widetilde{\mathbf{y}}$)) is solved, we seek an optimal extreme point $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ to (BendersSPDual($\widetilde{\mathbf{y}}$)). How to find such an optimal extreme point is detailed in Section 4.2.2. Let L denote the current Benders iteration, and let $\mathcal{L} := \{1, \ldots, L-1\}$ be the set of previous iterations. The dual solution $(\boldsymbol{\pi}^L, \boldsymbol{\gamma}^L) := (\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ defines a new constraint

$$v \geq \sum_{i \in \mathcal{N}_0} \pi_i^L + \sum_{k \in \mathcal{K}} M y^k \gamma_k^L, \tag{29}$$

to be added to Benders restricted master problem, which is given by

$$\begin{aligned} \text{(BendersRMP)} \qquad \tilde{v}^* := \min_{v, \mathbf{y}} \;& v, \\ \text{s.t. } \; & v \geq \sum_{i \in \mathcal{N}_0} \pi_i^l + \sum_{k \in \mathcal{K}} M y^k \gamma_k^l, \qquad l \in \mathcal{L}, \\ & \mathbf{y} \in R. \end{aligned}$$

After adding the new constraint (29) to (BendersRMP) it is solved for optimal values $(\tilde{v}^L, \widetilde{\mathbf{y}}^L)$. This defines the new problems (BendersSP($\widetilde{\mathbf{y}}^L$)) and (BendersSPDual($\widetilde{\mathbf{y}}^L$)), which are solved to generate a new constraint (29) to (BendersRMP). This process is iterated—adding one constraint (29) in each Benders iteration—until an optimal solution to (26) is found and verified. How to determine that such an optimal solution has been found, and the resulting Benders' algorithm, is described in Section 4.2.3. Note that (BendersRMP), (BendersSP($\widetilde{\mathbf{y}}$)), and (BendersSPDual($\widetilde{\mathbf{y}}$)), correspond to the general problems, (10), (7), and (8), respectively, in Section 2.3.

4.2.2 Optimal extreme point to Benders subproblem

Since it is the compact formulation (28a)–(28b), (28d) that is solved each Benders iteration, and not the equivalent problem (BendersSP($\widetilde{\mathbf{y}}$)), some extra effort has to be put into finding an extreme point of the feasible set $F_{\text{BendersSPDual}}$ that is optimal in the dual (27) of (BendersSP($\widetilde{\mathbf{y}}$)). To this end, Claims 2 and 3 have been formulated and proven.

The final column generation iteration in the solution course for the model (28a)–(28b), (28d), results in optimal dual variable values $\bar{\boldsymbol{\pi}}$ which are optimal for the dual of (28a)–(28b), (28d), which is expressed as

$$\max_{\boldsymbol{\pi}} \sum_{i \in \mathcal{N}_0} \pi_i, \tag{30a}$$

$$\text{s.t.} \sum_{i \in \mathcal{N}_0} \pi_i \delta_{ir}^k \le c_r^k, \qquad r \in \mathcal{R}_k, k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}). \tag{30b}$$

The problems (28) and (28a)–(28b), (28d) possesses the same optimal objective value, and by strong duality the optimal objective values of (30) and (28a)-(28b), (28d) are equal ([4, p. 248]). Since the formulations (28) and (BendersSPDual($\widetilde{\mathbf{y}}$)) are equivalent, the optimal objective value of (30) is therefore equal to the optimal value $w^*(\widetilde{\mathbf{y}})$ of (BendersSP($\widetilde{\mathbf{y}}$)) ((BendersSPDual($\widetilde{\mathbf{y}}$)) and (BendersSPDual($\widetilde{\mathbf{y}}$)) has the same optimal objective value, by strong duality). This relationship will be used in the proof of Claim 2.

For $k \in \mathcal{K}$ and $\bar{\boldsymbol{\pi}} \in \mathbb{R}^{|\mathcal{N}_0|}$, we define the problem

$$(\text{Gamma}(k, \bar{\boldsymbol{\pi}})) \qquad \gamma_k^* := \max_{\gamma_k} \gamma_k, \tag{31a}$$

$$\text{s.t. } \gamma_k \le c_r^k - \sum_{i \in \mathcal{N}_0} \bar{\pi}_i \delta_{ir}^k, \qquad r \in \mathcal{R}_k, \tag{31b}$$

$$\gamma_k \le 0. \tag{31c}$$

Problem (Gamma($k, \bar{\boldsymbol{\pi}}$)) is closely related to (BendersSPDual($\widetilde{\mathbf{y}}$)). The optimal value γ_k^* of (Gamma($k, \bar{\boldsymbol{\pi}}$)) equals the maximum value that can be taken by γ_k in (BendersSPDual($\widetilde{\mathbf{y}}$)), when the variables $\boldsymbol{\pi}$ are fixed to $\bar{\boldsymbol{\pi}}$.

We have the following results, the proofs of which are found in Appendix C.1.

Claim 2. *Any vector* $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ *that satisfies the conditions*

1. $\bar{\boldsymbol{\pi}}$ *is optimal in* (30),

2. $\bar{\gamma}_k = 0$, $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, *and*

3. $\bar{\gamma}^k = \gamma_k^*$ *where* γ_k^* *is optimal in* (Gamma($k, \bar{\boldsymbol{\pi}}$)), *for* $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$,

is optimal in (BendersSPDual($\widetilde{\mathbf{y}}$)).

Claim 3. *Any vector* $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ *that satisfies the conditions*

1. $\bar{\boldsymbol{\pi}}$ *is an extreme point to the set* $\left\{ \boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{N}_0|} \;\middle|\; \boldsymbol{\pi} \text{ satisfies (30b)} \right\}$,

2. $\bar{\gamma}_k = 0$, $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, *and*

3. $\bar{\gamma}_k = \gamma_k^*$ *where* γ_k^* *is optimal in* (Gamma($k, \bar{\boldsymbol{\pi}}$)), *for* $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$,

is an extreme point to the set $F_{BendersSPDual}$.

After solving (28a)–(28b), (28d)—the solution of which is optimal in Benders subproblem (BendersSP($\widetilde{\mathbf{y}}$)), where the excluded variables are set to zero ($x_r^k := 0, r \in \mathcal{R}_k, k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$)—using column generation, a value of $\bar{\boldsymbol{\pi}}$ satisfying condition 1 of Claim 2 results from the last iteration. The important consequences of Claims 2 and 3 are the following: if $\bar{\boldsymbol{\gamma}}$ satisfies conditions 2 and 3 of Claim 2, then $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ is optimal in (BendersSPDual($\widetilde{\mathbf{y}}$)). If $\bar{\boldsymbol{\pi}}$ also satisfies condition 1 of Claim 3—making $\bar{\boldsymbol{\pi}}$ an optimal extreme point to (30)—and $\bar{\boldsymbol{\gamma}}$ satisfies conditions 2 and 3 of Claim 2 (and consequently also of Claim 3) for the chosen value of $\bar{\boldsymbol{\pi}}$, then $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ is an extreme point to $F_{\text{BendersSPDual}}$ that is optimal in (BendersSPDual($\widetilde{\mathbf{y}}$)).

Notice that the variable values $\bar{\boldsymbol{\gamma}} = (\bar{\gamma}_k)_{k \in \mathcal{K}}$, that satisfy conditions 2 and 3 of Claim 2, are given by $\bar{\gamma}_k = \min\left\{0, \min_{r \in \mathcal{R}_k}\left\{\hat{c}_r^k\right\}\right\}$ (where $\bar{\gamma}_k$ is the optimal solution value of (Gamma($k, \bar{\boldsymbol{\pi}}$))). In the context of the linear relaxation (19) of the straightforward model, $\hat{c}_r^k$ is the reduced cost of variable x_r^k (compare (31) with the reduced cost (21)). Hence, $\bar{\gamma}_k$ has the interpretation of the change in objective value relative to the optimal solution to Benders subproblem (BendersSP($\widetilde{\mathbf{y}}$)) that would be achieved, if the route-vehicle pair corresponding to the variable x_r^k—with minimal reduced cost among the variables x_r^k, $r \in \mathcal{R}_k$—was allowed in a solution to Benders subproblem (and the value of x_r^k is increased by one, moving in the direction of a particular neighbouring extreme point)[10]. Interestingly, the right-hand side of each constraint (29) that is added to (BendersRMP) according to Claim 2, is the sum of $\sum_{i \in \mathcal{N}_0} \pi_i^L$ (which equals the optimal objective value of Benders subproblem in Benders iteration L) and $\sum_{k \in \mathcal{K}} M y^k \gamma_k^L$ (being the sum of scaled "minimal reduced costs", γ_k^L, from Benders iteration L, for all vehicle types in the set $\{k \in \mathcal{K} \mid y^k = 1\} =: \widetilde{\mathcal{K}}(\mathbf{y})$).

4.2.3 Benders' algorithm

With $(\tilde{v}^*, \widetilde{\mathbf{y}}^*)$ optimal in (BendersRMP), $\widetilde{\mathbf{x}}^*$ optimal in (BendersSP($\widetilde{\mathbf{y}}^*$)), and $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$ optimal in (BendersSPDual($\widetilde{\mathbf{y}}^*$)), we have that $(\widetilde{\mathbf{x}}^*, \widetilde{\mathbf{y}}^*)$ must be an optimal solution to (26) if it holds that

$$\tilde{v}^* = \sum_{i \in \mathcal{N}_0} \pi_i^* + \sum_{k \in \mathcal{K}} M(\tilde{y}^k)^* \gamma_k^*. \tag{32}$$

This, since $\tilde{v}^*$ is a lower bound on the optimal value v^* of (26), and $\sum_{i \in \mathcal{N}_0} \pi_i^* + \sum_{k \in \mathcal{K}} M(\tilde{y}^k)^* \gamma_k^* = \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}_k} c_r^k (\tilde{x}_r^k)^* = w^*(\widetilde{\mathbf{y}}^*)$ is an upper bound on v^*. If the equation (32) does not hold,the following inequality

$$\tilde{v}^* < \sum_{i \in \mathcal{N}_0} \pi_i^* + \sum_{k \in \mathcal{K}} M(\tilde{y}^k)^* \gamma_k^*, \tag{33}$$

[10]Similarly to how the reduced cost in the context of the column generation algorithm measures the change in objective value relative to the current optimal solution value of the column generation RMP, if the corresonding variable is allowed; see Section 2.1 and Appendix B.1.

must hold. Thus, adding to (BendersRMP) the constraint $v \geq \sum_{i\in\mathcal{N}_0} \pi_i^* + \sum_{k\in\mathcal{K}} My^k\gamma_k^*$, would make the current solution $(\tilde{v}^*, \widetilde{\mathbf{y}}^*)$ infeasible.

Following the procedure described in Section 2.3, Benders' algorithm iteratively solves (BendersRMP) for $(\tilde{v}^*, \widetilde{\mathbf{y}}^*)$ and (BendersSPDual($\widetilde{\mathbf{y}}^*$)) for $(\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*)$—each iteration adding a constraint to (BendersRMP)—until an optimal solution to (26) is found and verified, i.e., until the equation (32) holds. Benders' algorithm for the restricted model (25) is outlined in Algorithm 3.

1. Initialization: Choose $\widetilde{\mathbf{y}}^1 \in R$ and solve (BendersSP($\widetilde{\mathbf{y}}^1$)) for $(\boldsymbol{\pi}^1, \boldsymbol{\gamma}^1)$, defining the first constraint,
$$v \geq \sum_{i\in\mathcal{N}_0} \pi_i^1 + \sum_{k\in\mathcal{K}} My^k\gamma_k^1,$$
of (BendersRMP).
2. Solve (BendersRMP) for $(\tilde{v}^L, \widetilde{\mathbf{y}}^L)$.
3. For $\widetilde{\mathbf{x}}^L$, solve (BendersSP($\widetilde{\mathbf{y}}^L$)) by solving (28a)–(28b), (28d) using column generation. In the last column generation, add binary restrictions on the variables x_k^r, $r \in \widetilde{\mathcal{R}}_k, k \in \mathcal{K}$, and find a solution $\widetilde{\mathbf{x}}^L_{\text{binary}}$, such that $(\widetilde{\mathbf{y}}^L, \widetilde{\mathbf{x}}^L_{\text{binary}})$ is feasible in (25).
4. If $w^*(\widetilde{\mathbf{y}}^L) > \tilde{v}^L$, find an extreme point $(\boldsymbol{\pi}^L, \boldsymbol{\gamma}^L)$ that is optimal in (BendersSPDual($\widetilde{\mathbf{y}}^L$)), by using an optimal extreme point $\bar{\boldsymbol{\pi}}$ to (30)—the dual of (28a)–(28b), (28d)—as prescribed in Claims 2 and 3. Add the new constraint
$$v \geq \sum_{i\in\mathcal{N}_0} \pi_i^L + \sum_{k\in\mathcal{K}} My^k\gamma_k^L,$$
to (BendersRMP) and go to Step 2. Otherwise $w^*(\widetilde{\mathbf{y}}^L) = \tilde{v}^L = v^*$; hence $(\widetilde{\mathbf{y}}^L, \widetilde{\mathbf{x}}^L)$ is optimal in (26). Terminate the algorithm.

Algorithm 3: Benders' decomposition algorithm.

A lower bound on the optimal value v^* of the problem (26) is given in each Benders iteration by $\tilde{v}^L$. This lower bound is non-decreasing. An upper bound on v^* is given in each Benders iteration by the optimal objective value $w^*(\widetilde{\mathbf{y}}^L)$ of Benders subproblem (BendersSP($\widetilde{\mathbf{y}}^L$)). This upper bound is, however, not non-increasing. A better upper bound, which is non-increasing, is given by $\min_{l\in\{1,\dots,L\}} \left\{w^*(\widetilde{\mathbf{y}}^l)\right\}$ ([33]). Since the optimality condition (32) (which is the termination criterion that has been implemented) is equivalent to $\tilde{v}^L = w^*(\widetilde{\mathbf{y}}^L)$, this implies that a stronger optimality condition would be given by
$$\tilde{v}^L = \min_{l\in\{1,\dots,L\}} \left\{w^*(\widetilde{\mathbf{y}}^l)\right\}. \tag{34}$$

Benders' algorithm could be set to terminate when $\min_{l\in\{1,\ldots,L\}} \{w^*(\widetilde{\mathbf{y}}^l)\} - \tilde{v}^L \le \varepsilon$, for some $\varepsilon > 0$, since then the difference between the best feasible solution that has been found so far—with the objective value $\min_{l\in\{1,\ldots,L\}} \{w^*(\widetilde{\mathbf{y}}^l)\}$—and an optimal solution to (26)—with the objective value v^*—is not greater than ε.

Since Benders' algorithm is applied to the relaxation (26) of the restricted model (25), v^* (and thus also $\tilde{v}^L$) is only a lower bound on the optimal value of (25). An upper bound is given in Benders iteration L by the smallest objective value in (25) of the feasible solutions $(\widetilde{\mathbf{y}}^l, \widetilde{\mathbf{x}}^l_{\text{binary}})$, $l = 1, \ldots, L$.

It may not necessarily be the case that Benders' algorithm terminates when $(\widetilde{\mathbf{x}}^L, \widetilde{\mathbf{y}}^L)$ optimal in (26) are found. This has not been proven, but an intuitive argument for why optimality would not be verified is the following: if all constraints (29) that are binding in the optimal solution have not been generated in Benders iteration L, it may happen that $\tilde{v}^L < v^*$, even if $\widetilde{\mathbf{y}}^L$ is an optimal choice of variable values in (26), i.e., even if $w^*(\widetilde{\mathbf{y}}^L) = v^*$.

However, Algorithm 3 will converge after a finite number of iterations, if, in each Benders iteration L, the optimal solution $(\boldsymbol{\pi}^L, \boldsymbol{\gamma}^L)$ is chosen as an extreme point to the feasible set $F_{\text{BendersSPDual}}$ of (BendersSPDual($\widetilde{\mathbf{y}}^L$)). The set R defining the set of feasible values for $\mathbf{y}$ in (BendersRMP) is finite, thus there is only a finite number of possible problems (BendersSPDual($\mathbf{y}$)), for each of which an optimal extreme point can be found for each (BendersSPDual($\mathbf{y}$)). If for two distinct Benders iterations, l_1 and l_2, it holds that $\mathbf{y}^{l_1} = \mathbf{y}^{l_2}$, then the optimality criterion (32) is fulfilled, and since the set R is finite, finite convergence follows. The same result should hold, also if a solution $(\boldsymbol{\pi}^L, \boldsymbol{\gamma}^L)$ that is optimal but not necessarily an extreme point in (BendersSPDual($\widetilde{\mathbf{y}}^L$)) is chosen. This is just an intuitive argument, however.

An argument for the finite convergence of Algorithm 3 can be based on the proof of finite convergence for a generalized Benders' decomposition found in [34, Theorem 6.3.4, p 125]. The argument for the general Benders' algorithm, as presented in Section 2.3, is that the algorithm is guaranteed to converge finitely since there is only a finite number of extreme points to the feasible set of the dual of Benders subproblem [9].

With the proposed procedure for finding $(\boldsymbol{\pi}^L, \boldsymbol{\gamma}^L)$ in Algorithm 3, it may be the case that only one of possibly several optimal extreme points to the dual (BendersSPDual($\widetilde{\mathbf{y}}^L$)) can be found. Thus, we can not be sure that all extreme points to $F_{\text{BendersSPDual}}$ can be found in the solution course of the proposed Benders' algorithm.

4.2.4 Additions to Benders' algorithm

To improve the speed of convergence of Algorithm 3, another step has been implemented, in which optimal solutions $\widetilde{\mathbf{x}}^l$ to (BendersSP($\widetilde{\mathbf{y}}^l$)), from previous iterations $l = 1, \ldots, L-1$ of Benders' algorithm, are tested to see if the corresponding routes would yield a better objective value when used in combination with other vehicles. This procedure is called *projection of routes*.

Define the set of all routes "used" in iteration l by

$$\mathcal{R}^l := \cup_{k\in\mathcal{K}}\{r \in \mathcal{R}_k \mid \left(\tilde{x}_r^k\right)^l = 1\}.$$

For $r \in \cup_{k\in\mathcal{K}}\mathcal{R}_k$, denote by $\mathcal{K}_r := \arg\min_{k\in\mathcal{K}:r\in\mathcal{R}_k}\left\{c_r^k\right\}$ the set of optimal vehicle types for route r. Only vehicles in $\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}^l)$ are allowed to be used in a solution to (BendersSP($\widetilde{\mathbf{y}}^l$)) in iteration l, in Benders' algorithm. Hence, routes $r \in \mathcal{R}^l$ are not necessarily matched with vehicle types in $\mathcal{K}_r$. For $\widetilde{\mathbf{y}} \in R$ such that $\mathcal{R}^l \subseteq \cup_{k\in\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})}\mathcal{R}_k$, we denote by $\widetilde{\mathbf{x}}^l(\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}))$ the solution to (BendersSP($\widetilde{\mathbf{y}}$)) that uses all the routes in $\mathcal{R}^l$, matched with their respective optimal vehicle types in $\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$ (i.e., $r \in \mathcal{R}^l$ is matched with $k \in \arg\min_{k\in\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})}\left\{c_r^k\right\}$). There may exist some set of variable values $\widetilde{\mathbf{y}} \in R$, for which the objective value in (26) of $(\widetilde{\mathbf{y}}, \widetilde{\mathbf{x}}^l(\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})))$ is lower than the value $w^*(\widetilde{\mathbf{y}}^l)$ of $(\widetilde{\mathbf{y}}^l, \widetilde{\mathbf{x}}^l)$.

In the projection of routes procedure, variable values $\widetilde{\mathbf{y}}$, for which the following holds, are sought; the objective value of $(\widetilde{\mathbf{y}}, \widetilde{\mathbf{x}}^l(\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})))$ in (26) is lower than the value $\min_{l\in\{1,\dots,L\}}\left\{w^*(\widetilde{\mathbf{y}}^l)\right\}$ of the best feasible solution found so far. If such variable values $\widetilde{\mathbf{y}}$ are found, then $\mathbf{y}^{L+1} := \widetilde{\mathbf{y}}$. Thus, the projection of routes is used in place of Benders RMP in iteration $L+1$ to define the next subproblem (BendersSP($\widetilde{\mathbf{y}}^{L+1}$)). This subproblem has an optimal objective value $w^*(\widetilde{\mathbf{y}}^{L+1})$ that is lower than or equal to the objective value of $\widetilde{\mathbf{x}}^l(\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}))$; hence, $w^*(\widetilde{\mathbf{y}}^{L+1}) < \min_{l\in\{1,\dots,L\}}\left\{w^*(\widetilde{\mathbf{y}}^l)\right\}$.

The projection of routes procedure has been implemented in a rather rudimentary fashion. For each solution $\widetilde{\mathbf{x}}^l$, $l = 1,\dots,L$, routes $r \in \mathcal{R}^l$ are matched with vehicles types from $\mathcal{K}_r$. Denote the resulting solution $\mathbf{x}(\mathcal{R}^l)$, and let $\mathbf{y}(\mathcal{R}^l) = (y^k(\mathcal{R}^l))_{k\in\mathcal{K}}$, where

$$y^k(\mathcal{R}^l) := \begin{cases} 1, & \text{if vehicle } k \text{ is used in the solution } \mathbf{x}(\mathcal{R}^l), \\ 0, & \text{otherwise.} \end{cases}$$

If the best solution $(\mathbf{x}(\mathcal{R}^{\hat{l}})), \mathbf{y}(\mathcal{R}^{\hat{l}}))$, $\hat{l} \in \{1,\dots,L\}$ has an objective value in (26) that is lower than $\min_{l\in\{1,\dots,L\}}\left\{w^*(\widetilde{\mathbf{y}}^l)\right\}$, and if $|\widetilde{\mathcal{K}}(\mathbf{y}(\mathcal{R}^{\hat{l}}))| \leq C$, then $\mathbf{y}^{L+1} := \mathbf{y}(\mathcal{R}^{\hat{l}})$. However, if $|\widetilde{\mathcal{K}}(\mathbf{y}(\mathcal{R}^{\hat{l}}))| > C$, so that $\mathbf{y}(\mathcal{R}^{\hat{l}}) \notin R$, then a restricted set $\widetilde{\mathcal{K}}_{\text{restricted}} \subset \widetilde{\mathcal{K}}(\mathbf{y}(\mathcal{R}^{\hat{l}}))$ for which $\widetilde{\mathcal{K}}_{\text{restricted}} \leq C$, is defined using the constraints of (BendersRMP). This restricted set is used analogously to $\mathcal{K}_r$ to find $(\hat{\mathbf{x}}(\mathcal{R}^{\hat{l}})), \hat{\mathbf{y}}(\mathcal{R}^{\hat{l}}))$. If $(\hat{\mathbf{x}}(\mathcal{R}^{\hat{l}})), \hat{\mathbf{y}}(\mathcal{R}^{\hat{l}}))$ still has a lower objective value than $\min_{l\in\{1,\dots,L\}}\left\{w^*(\widetilde{\mathbf{y}}^l)\right\}$, then $\widetilde{\mathbf{y}}^{L+1} := \hat{\mathbf{y}}(\mathcal{R}^{\hat{l}})$. Otherwise (BendersRMP) is solved as usual. If $\widetilde{\mathbf{y}}^{L+1}$ is defined without solving (BendersRMP), then Step 3 of Algorithm 3 is performed to find a new constraint to add to (BendersRMP), but the optimality criterion is not checked since $\widetilde{\mathbf{y}}^L$ was found without solving (BendersRMP), thus no value $\tilde{v}^L$ is provided.

The projection of routes can be set to be performed every i_{repeat}:th Benders iteration, starting from iteration i_{start}, and the number of solutions from previous Benders subproblems that are tested may be restricted to solutions from the last i_{restrict} iterations, for i_{repeat}, i_{start}, $i_{\text{restrict}} \in \mathbb{N}$. This procedure has been shown to greatly reduce the time before an optimal solution to (26) is found; see Section 5.2.3. Even greater improvements in solution time would probably be achieved by improving this procedure of projecting routes, especially by improving the handling of restrictive values of the vehicle type limit C.

To improve the solution speed of Benders subproblems, a so called *warm start* can be utilized, according to the following. The routes $\cup_{l=1}^{L}\mathcal{R}^l$ that are part of the previous optimal solutions to Benders subproblems, i.e., $\widetilde{\mathbf{x}}^l$, $l = 1, \ldots, L$, and which can be taken by some vehicle in the current Benders iteration—i.e., the routes in the set $\left(\cup_{l=1}^{L}\mathcal{R}^l\right) \cap \left(\cup_{k \in \cup \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}^{L+1})}\mathcal{R}_k\right)$—are included in the set of routes used to start the column generation algorithm for solving Benders subproblem in iteration $L+1$. This has been shown to reduce the time spent in the column generation algorithm. It is even the case that in most iterations of Benders' algorithm, an optimal solution to Benders subproblem can be found among the routes provided by the warm start. See Section 5.2.2 for details.

The projection of routes procedure and the warm start, described in this section, use some ideas from the set-partitioning based heuristic for hVRP by Taillard ([20]), described in Section 3.1.2. In [20] each route r generated in a homogeneous VRP determines route-vehicle pairs (r, k) in a set-partitioning formulation for hVRP, for each vehicle type in the set $\{k \in \mathcal{K} \mid r \in \mathcal{R}_k\}$.

4.2.5 Suggestions for further improvements of Benders' algorithm

To get an optimal extreme point to $F_{\text{BendersSPDual}}$, by Claims 2 and 3, Benders subproblem should be solved to optimality using column generation to find an optimal extreme point $\bar{\boldsymbol{\pi}}$ to the problem (30). However, at the later iterations of the column generation often very small improvements are made, known as the tailing off effect (see [8]). It is possible that good quality constraints of (BendersRMP) can be provided by using points in the set $F_{\text{BendersSPDual}}$ that are not necessarily extreme points, or even non-optimal in the current Benders subproblem. This, since for any $(\boldsymbol{\pi}, \boldsymbol{\gamma}) \in F_{\text{BendersSPDual}}$, the inequality

$$v \geq \sum_{i \in \mathcal{N}_0} \pi_i + \sum_{k \in \mathcal{K}} M y^k \gamma_k \tag{35}$$

is a valid constraint in (BendersRMP) (see [35, p. 308]). Thus, the column generation algorithm for Benders subproblem can be terminated before an optimal extreme point to (30) is found, and the variable values $\boldsymbol{\pi}$ that are optimal in the dual of the current column generation RMP, can be used to define a new constraint—for any $\boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{N}_0|}$ it holds that $(\boldsymbol{\pi}, \boldsymbol{\gamma}) \in F_{\text{BendersSPDual}}$ whenever it holds that $\boldsymbol{\gamma} := (\gamma_k)_{k \in \mathcal{K}}$, $\gamma_k \leq \gamma_k^*$, and γ_k^* is optimal in (Gamma$(k, \boldsymbol{\pi})$).

It follows that, instead of solving the problem (Gamma$(k, \boldsymbol{\pi})$), $k \in \mathcal{K}$, in each Benders iteration[11] (as prescribed in Claim 2), lower bounds on the optimal values of problems the (Gamma$(k, \boldsymbol{\pi})$), $k \in \mathcal{K}$, can be used to define a

[11]We investigated if it would be possible to find a set of conditions under which $\tilde{\gamma} \in \arg\min\,(\text{Gamma}(k_1, \boldsymbol{\pi}))$ implies the inclusion $\tilde{\gamma} \in \arg\min\,(\text{Gamma}(k_2, \boldsymbol{\pi}))$, for $k_1 \neq k_2$. We did not, however, find any such conditions; see Appendix C.2.

new constraint to (BendersRMP). This could greatly reduce the computational effort, since the problem (Gamma($k, \boldsymbol{\pi}$)) is essentially the column generation subproblems of the straightforward model, and these are computationally hard problems—see Section 2.2. Dynamic programming for SPPRC with 2-cycle elimination, described in Section 2.2.2, is an efficient algorithm for calculating a lower bound of good quality to (Gamma($k, \boldsymbol{\pi}$)), and is a commonly used method for solving relaxed column generation subproblems in connection with VRP (see [12]). This method is used by, among others, Choi and Tcha ([24]) in their column generation solution approach for hVRP, as mentioned in Section 3.1.2. This usage of lower bounds has not been implemented in this thesis, but it is suggested as a way of improving the algorithm.

4.3 Load dependent costs

Load dependent costs can be used for both the straightforward model and the restricted model, developed in Sections 4.1 and 4.2 respectively, by redefining the parameters c_r^k, and thus altering the column generation subproblems. For the restricted model, this has implications for Benders subproblems. The property of the set-partitioning model (15)—on which both models are based—that only the subproblems need to be altered is quite useful. Other extensions made to the models, e.g., imposing time-windows, would also require changes of the column generation subproblems only, and not of other parts of the algorithms (although the set R of Benders RMP of the restricted model would also need to be altered).

We use the load dependent objective function introduced by Xiao et al. in [29], as presented in Section 3.2. Define the parameters $Q_{\text{dist}} > 0$ and $Q_{\text{load}} > 0$. Parameter Q_{dist} is used to weight the part of the objective function that does not depend on the vehicle load on the arcs, and parameter Q_{load} is used to weight the part of the objective function that depends on the vehicle load. Thus, the load dependent cost of the route-vehicle pair (r, k), where $r = (i_0, i_1, \ldots, i_{H-1}, i_H)$ and $i_0 = i_H = 0$—and the sum $\sum_{\hat{h}=h}^{H-1} d_{\hat{h}}$ represents the load of the vehicle after the route has visited customer $h - 1$—is defined as

$$(c_{\text{load}})_r^k := \left(f_k + \sum_{h=1}^{H-1} c_{i_{h-1}i_h}^k \left(Q_{\text{dist}} + Q_{\text{load}} \left(\sum_{\hat{h}=h}^{H-1} d_{\hat{h}} \right) \right) \right) + c_{H-1,H}^k Q_{\text{dist}}.$$

Hence, the cost of the route-vehicle pair (r, k) is given by the sum of the fixed cost f_k and a weighted sum of the arc costs $c_{i_{h-1}i_h}^k$, $h = 1, \ldots, H$, where the weights increase with the vehicle load on the arcs. Xiao et al. ([29]) model the parameter values Q_{dist} and Q_{load} based on fuel consumption rates and fuel prices, as described in Section 3.2. Since we have used artificial test instances and not real data to test our algorithms in this thesis, the values of Q_{dist} and Q_{load} are chosen by trial and error.

Algorithm 1 is adapted accordingly, resulting in the following subproblem for $k \in \mathcal{K}$:

$$\min_{r \in \mathcal{R}_k} (\hat{c}_{\text{load}})_r^k = \min_{r \in \mathcal{R}_k} \left\{ (c_{\text{load}})_r^k - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir}^k \right\} \tag{36}$$

The same dynamic programming algorithm, i.e., Algorithm 2, that is used for the standard column generation subproblems (22) can also be used for the subproblems (36) by altering the caluclation of the cost for each path. The condition in Claim 1 pertaining to the cost of each path can be modified, with corresponding modifications of the proof, so that the claim holds also for the subproblems (36).

Of the two formulations (23) and (23a)–(23e), (23h)–(23i), (24) used for the standard column generation subproblems (22), only the former can be used in the case of load dependent costs (36) (with some alterations). Now the variables e_{ij}, $(i,j) \in \mathcal{A}$, instead of representing an artificial flow—as in formulation (23)—represents the actual flow of goods—as in formulation (12). The resulting subproblem formulation used in this thesis, is given by

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{e}} \quad & \sum_{(i,j)\in\mathcal{A}} \left(c_{ij}^k (Q_{\text{dist}} x_{ij} + Q_{\text{load}}\, e_{ij}) - \pi_i^* x_{ij} \right), && && \text{(37a)}\\
\text{s.t.} \quad & \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} x_{ji} = 0, && j \in \mathcal{N}_0, && \text{(37b)}\\
& \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} \le 1, && j \in \mathcal{N}_0, && \text{(37c)}\\
& \sum_{i\in\mathcal{N}:(0,i)\in\mathcal{A}} x_{0i} = 1, && && \text{(37d)}\\
& \sum_{i\in\mathcal{N}:(i,0)\in\mathcal{A}} x_{i0} = 1, && && \text{(37e)}\\
& \sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} e_{ij} - \sum_{i\in\mathcal{N}:(j,i)\in\mathcal{A}} e_{ji} = d_j \left(\sum_{i\in\mathcal{N}:(i,j)\in\mathcal{A}} x_{ij} \right), && j \in \mathcal{N}_0, && \text{(37f)}\\
& d_j x_{ij} \le e_{ij} \le (D_k - d_i) x_{ij}, && (i,j) \in \mathcal{A}, && \text{(37g)}\\
& x_{ij} \in \{0,1\}, && (i,j) \in \mathcal{A}. && \text{(37h)}
\end{aligned}$$

The objective (37a) is to minimize the reduced cost. The constraints (37b)–(37e) are equivalent to the constraints (23b)–(23e). The constraints (37f) have been adapted from the constraints (12d), and the constraints (37g) are equivalent to the constraints (13), reformulated for a homogeneous problem.

5 Tests and results

Both the straightforward and the restricted model has been tested on the original (CT12) and the extended (CT12EXT) test instances; the results are presented in Sections 5.1 and 5.2, respectively. The latter of the instance sets—including more vehicle types than the former—are described in Section 3.1.3.

The test were performed on Linux computers with a Pentium Dual-Core CPU 2.5 GHz with 2048 KB cache. The mathematical models and algorithms have been implemented using the modelling software AMPL 12.1.0 ([36]), and the optimization solver CPLEX 12 ([37]). Dynamic programming for the column generation subproblems and some other calculations (e.g., finding initial solutions to the column generation RMP, and for warm start and projection of routes in Benders' algorithm) have been implemented in Matlab 2012b [38]. The communication between AMPL and Matlab is done via text files. The bash command `screen` is used to enable a single Matlab session being opened and called several times from AMPL. Matlab has also been used for the visualization of solutions and algorithm performance data.

5.1 The original test instances, CT12

The test instances CT12 have been used to test the column generation algorithm, implemented as described in Section 4.1. Since these instances contain only three to six vehicle types, the corresponding tests of the column generation algorithm concern the hVRP, not the *many*-hVRP.

In Section 5.1.1, the results from these tests are compared with the best published results for hVRP on CT12, as well as with an implementation of the flow formulation (12) solved directly by AMPL and CPLEX. In Sections 5.1.2 and 5.1.3, tests of different ways initializations of the column generation algorithm and different implementations of the algorithms (described in Section 4.1.1) for the column generation subproblems, are presented.

5.1.1 Comparison of solution methods

In this section, the column generation applied to the model (15) (the straightforward model of Section 4.1) is compared with a straightforward implementation of the flow formulation (12), as well as with the results of Choi and Tcha in [24] and with the best known results for CT12. The column generation applied to (15), described in Section 4.1, has been implemented in AMPL and CPLEX, using Matlab with dynamic programming and/or AMPL and CPLEX for the column generation subproblems (22). The flow formulation (12) has been implemented in AMPL and CPLEX.

In Table 2 results for the flow formulation (12) are presented. Table 5 shows the results from the column generation algorithm by Choi and Tcha ([24]) (these results were obtained using a Pentium IV 2.6 GHz processor). Table 3 and 4 lists the results from our standard implementation of column generation on CT12. The results in Table 3 were obtained using dynamic programming for the column generation subproblems in the former part and AMPL and CPLEX in the latter part of the solution course, whereas the results in Table 4 were obtained using only AMPL and CPLEX for the column generation subproblems. Both the solve time—i.e. the CPU time spent in the column generation algorithm in AMPL and CPLEX, and the clock time spent in the dynamic programming in Matlab for the column generation subproblems—and the total time—i.e., the clock time from the algorithm is starts until it finishes—are presented in the tables. In addition to time spent in the column generation algorithm, the total time also includes the time spent calculating the heuristic solution used to start the column generation, and the time spent in communications between AMPL/CPLEX and Matlab. The calculations were interrupted after 10000 CPU seconds[12]. The lower bound $\underline{z}$, was calculated after the column generation algorithm was terminated (for some of the larger instances, these calculations were aborted with an out-of-memory error). More details about the column generation implementation are presented in Sections 5.1.2 and 5.1.3.

In Tables 2–5, "Obj. value" refers to the objective value of the best feasible solution to (15) found, and "Opt. gap" = ("Obj. value" - $z_{\text{bestknown}})/z_{\text{bestknown}}$, where $z_{\text{bestknown}}$ is the objective value of the best feasible solutions known for the instances in CT12, (see [23, Table 7]).

Instance	Obj. value	LB	Opt. gap [%]
Choi3	1154.8	986.1	0.89
Choi4	6437.3	6281.7	0
Choi5	1357.2	1027.8	2.64
Choi6	7021.3	6310.8	7.75
T13	3238.0	2401.6	9.22
T14	9672.1	1034.8	5.97
T15	2760.6	424.8	4.77
T16	3524.1	843.5	11.21
T17	2846.9	1190.8	42.03
T18	4744.8	2449.5	50.72
T19	10274.1	2110.6	18.61
T20	4778.8	116.1	15.03

Table 2: Results from a straightforward implementation of the model (12) in AMPL. The computations were interrupted after 10000 CPU seconds. "LB" refers to the current lower bound provided by AMPL and CPLEX when the computations were interrupted.

[12]The column generation algorithm is set to terminate when the solve time exceeds 10000 CPU seconds. The solve time is, however measured only at the start of each column generation iteration. This explains why the total solve time for T16 in Table 4 is 22810 CPU seconds—the subproblems are solved to optimality after approximately 5000 CPU seconds, which leads to the last column generation iteration (before the algorithm is terminated) taking approximately 17000 CPU seconds.

Comparing the results from the column generation algorithm implementations presented in Tables 3 and 4, using dynamic programming for the subproblems in the first part of the column generation algorithm (Table 3) yields better objective values than when not using dynamic programming (Table 4). Comparing the best feasible solutions to (25) found, the column generation with dynamic programming found solutions with lower or equal objective values for all instances except Choi3. Comparing the upper bounds $\bar{z}$ on the optimal objective value of (26) for the instances T14–T20 (for which the column generation did not converge in 10000 CPU seconds), the column generation with dynamic programming provided upper bounds that were strictly lower for all instances except T14. For these reasons, the column generation with dynamic programming for the subproblems is set as the standard, even though the column generation where dynamic programming is not used converges faster.

Instance	Obj. value	$\bar{z}$	$\underline{z}$	T_{OPT} [CPU s]	T_{TOT} [s]	Opt. gap [%]
Choi3	1150.9	1139.8	1139.8	299	348	0.55
Choi4	6484.9	6370.8	6370.8	691	747	0.74
Choi5	1322.3	1308.4	1308.4	556	516	0
Choi6	6524.4	6451.6	6451.6	765	779	0.12
T13	2969.7	2959.8	2959.8	8107	5657	0.17
T14*	12771.5	9311.0	< 0	10519	10743	39.93
T15	2692.5	2608.8	2153.4	10139	8243	2.18
T16	3210.1	3139.0	2286.3	10089	7989	1.30
T17*	2556.8	2159.3	–	10134	10347	27.55
T18	3249.1	3188.0	–	10282	10507	3.21
T19*	12957.5	9743.4	–	10029	10348	49.59
T20*	5659.2	4517.6	–	10145	10484	36.22

Table 3: Results from the column generation, implemented in AMPL and Matlab and using dynamic programming. * indicates that no improvement of the initial, heuristic solution was made, in the objective value "Obj. value" of the best feasible solution to (15) found. $\bar{z}$ and $\underline{z}$ denote the upper and lower bound, respectively, defined in Section 4.1.2, of the optimal value, z^*_{LP}, of the LP relaxation (19). "T_{OPT}" refers to the total solve time in AMPL/CPLEX and Matlab. "T_{TOT}" refers to the total time of the whole algorithm.

Solving the flow formulation directly in AMPL and CPLEX requires a very long computing time. The calculations were interrupted after a pre-defined time limit of 10000 CPU seconds, and for none of the test instances was an optimal solution verified. Comparing the results presented in Tables 2 and 3, column generation with dynamic programming performs better than the implementation of the flow formulation on the instances Choi3–Choi6, T13, T15–T16 and T18 (for which an improvement of the initial heuristic solution was made during the column generation), except for Choi4. For the instances Choi3–Choi6 and

T13 (for which the column generation converged before 10000 CPU seconds), the lower bound $\underline{z}$ provided by the column generation was higher than that resulting from the flow formulation.

Instance	Obj. value	$\bar{z}$	$\underline{z}$	T_{OPT} [CPU s]	T_{TOT} [s]	Opt. gap [%]
Choi3	1145.6	1139.8	1139.8	66	66	0.09
Choi4	6961.1	6370.8	6370.8	128	104	8.14
Choi5	1322.3	1308.4	1308.4	547	335	0
Choi6	6563.2	6451.6	6451.6	194	144	0.72
T13	2975.6	2959.8	2959.8	6675	4129	0.37
T14*	12771.5	8873.2	6558.4	10035	6341	39.93
T15	2787.6	2624.6	2094.1	11240	7047	5.79
T16	3219.2	3152.4	2165.4	22810	13855	1.59
T17*	2556.8	2469.1	–	10042	6424	27.55
T18	3677.9	3471.4	–	10119	6600	16.83
T19*	12957.5	9883.5	–	10050	6703	49.59
T20*	5659.2	5292.0	–	10079	6738	36.22

Table 4: Results from the column generation, implemented in AMPL and Matlab, without using dynamic programming. * indicates that no improvement of the initial, heuristic solution was made, in the objective value "Obj. value" of the best feasible solution to (15) found. $\bar{z}$ and $\underline{z}$ denote the upper and lower bound, respectively, defined in Section 4.1.2, of the optimal value, z^*_{LP}, of the LP relaxation (19). "T_{OPT}" refers to the total solve time in AMPL/CPLEX and Matlab. "T_{TOT}" refers to the total time of the whole algorithm.

Comparing the column generation results presented in Tables 3 and 4 with those resulting from the column generation heuristic of Choi and Tcha ([24]) (these are presented in Table 5), the implementation of Choi and Tcha—described in Section 3.1.2—manages to find better solutions in a fraction of the solution time T_{OPT}. Using the relaxed column generation subproblems, a set-covering formulation instead of the set-partitioning formulation (15), and, in addition, an implementation in C, the method of Choi and Tcha definitely out-performs our column generation implementation. This suggests that great improvements can be made to our implementation. The reason why we did not improve our column generation implementation, is that the focus of our work was to extend the hVRP to the *many*-hVRP.

The best solutions for CT12 obtained by the method of Baldacci et al. ([23]), which is the best exact method for hVRP according to Vidal et al. ([3]), differed from those reported in Table 5 only for the instances T17 and T19. The computations of Baldacci et al. were interrupted after 7200 CPU s. Optimality was proven for all instances, except T20. The optimal objective values for T17 and T19 was found to be 2004.5 and 8661.8, respectively ([23, Table 7]).

Instance	Obj. value	LB	Time [CPU s]	Opt. gap [%]
Choi3	1144.6	1138.6	0.25	0
Choi4	6437.3	6369.2	0.45	0
Choi5	1322.3	1307.7	0.19	0
Choi6	6516.5	6451.6	0.41	0
T13	2964.7	2959.6	3.95	0
T14	9126.9	8748.6	51.70	0
T15	2635.0	2597.2	4.36	0
T16	3168.9	3114.0	5.98	0
T17	2023.6	1979.9	68.11	0.95
T18	3148.0	2959.8	18.78	0
T19	8664.3	8431.9	905.20	0.03
T20	4154.5	4082.3	53.02	0

Table 5: Results from the column generation algorithm of Choi and Tcha [24]. "LB" refers to the optimal objective value of the LP relaxation of the problem (16), with some additional relaxations.

5.1.2 Column generation: heuristic to find initial columns

A few different heuristics to initialize the sets $\widetilde{R}_k, k \in \mathcal{K}$, in the restricted master problem (20) has been tried.

The first heuristic starts with finding a route for the vehicle type with smallest capacity, by adding the unvisited node $i \in \mathcal{N}_0$ that is closest to the depot, and subsequently adding the unvisited node that is closest to the previously added node, until the capacity limit is reached. This is repeated for another vehicle type, in order of capacity (after a route has been added for the largest vehicle type, the smallest vehicle type is considered again), until all nodes have been visited.

The second heuristic treats each vehicle type separately, and constructs routes that visits all customer nodes in $\mathcal{N}_0$ that the vehicle type k can service, i.e., all nodes $i \in \mathcal{N}_0$ such that $d_i \leq D_k$. The routes are constructed in the same fashion as in the first heuristic, i.e., they are extended to the closest unvisited node until the capacity limit is reached.

The routes provided by the first heuristic generally defines a restricted master problem (20) with a higher optimal objective value z^*_{RMP} than the second heuristic does, since often many more routes are constructed with the second heuristic. Using the second heuristic, the column generation also tends to converge faster to the optimal value z^*_{LP} of the master problem (19) as compared with the first heuristic. However, in the final step of the column generation when binary restrictions on the variables are added to find a feasible solution to the model (15), the first heuristic results in feasible solutions that are clearly better on the four test instances Choi3–Choi6 that were compared; see Table 6.

For this reason, the first heuristic has been chosen for our final tests. To initialize the sets $\widetilde{R}_k$, $k \in \mathcal{K}$, so that a basis matrix exists, the total number of columns in the initial restricted master problem (20) should be greater than or equal to the number of constraints, i.e., $\sum_{k\in\mathcal{K}} |\widetilde{R}_k| \geq |\mathcal{N}_0|$ (see Appendix B.1). This has been achieved by adding, for each customer node, one route that visits only that customer, using the vehicle with the largest capacity. However, by adding these routes to the routes generated by the heuristic, the best feasible solution to the formulation (15) found in the last column generation iteration, appears to be of lower quality (compare the objective values obtained when the algorithm is initiated using the "First heuristic" with those obtained using the "First heuristic, extra routes" in Table 6). For this reason extra routes have not been added in our standard version of the column generation algorithm. Instead, the problem of determining a basis matrix (adding artificial columns) is left to AMPL.

Instance	Objective value		
	First heuristic	Second heuristic	First heuristic, extra routes
Choi3	1150.9	1157.8	1155.2
Choi4	6484.8	6512.9	6987.1
Choi5	1322.3	1333.6	1323.2
Choi6	6524.4	6537.6	6537.6

Table 6: Results from column generation started using two different heuristics, on the four smallest test instances in CT12. "Objective value" refers to the objective value of the best feasible solution to the formulation (15) found, i.e., the value obtained in the last column generation iteration when binary requirements are added.

5.1.3 Column generation: algorithms applied to the subproblem

The subproblems have been implemented such that they can either be solved using the mathematical formulations (23) and (23a)–(23e), (23h)–(23i), (24) in Section 4.1.1 (implemented in AMPL and CPLEX), or by using the dynamic programming algorithm described in Section 4.1.1 (implemented in Matlab).

Preliminary tests on the smallest test instances, i.e., Choi3–Choi6, solving all subproblems to optimality, indicated that none of the two mathematical formulations (23) and (23a)–(23e), (23h)–(23i), (24) is significantly better than the other; each formulation outperforms the other on some test instance. However, the latter formulation with the constraints (23c) removed (which is a valid formualtion) does not perform well on the small test instances; after only a few column generation iterations for the instance Choi3, the computations were terminated by AMPL with termination code 9, which is "an error code from the operating system level which has generally been associated with out-of-memory errors" ([39]). Contrast this to the formulation (23), for which the column generation algorithm for the instance Choi3 converges after 110 CPU seconds, and the formulation (23a)–(23e), (23h)–(23i), (24), for which the column generation algorithm for the instance Choi3 converges after 75 CPU seconds.

In our standard implementation, subproblems are first solved using the dynamic programming algorithm, which is terminated after a time limit is exceeded or after a certain number of partial paths in $\mathcal{P}_{\mathrm{NPP}}$ has been processed (it has been set so that the limit on the number of paths is not restrictive). The time limit is increased if, in a certain iteration, no routes with negative reduced cost are found for any of the subproblems. When the time limit has been increased a predefined number of times and no more routes with negative reduced cost are found, the subproblem is solved by AMPL and CPLEX, using the formulation (23a)–(23e), (23h)–(23i), (24). Here, each subproblem is also terminated either when a time limit is reached or when a certain number of routes with negative reduced cost have been found. This time limit can also be increased when no more routes with negative reduced cost are found for any subproblem. In a final step—if the column generation has not converged in the previous steps—all subproblems are solved to optimality. Subproblems that are solved by AMPL and CPLEX use the CPLEX-option `uppercutoff=0`—meaning that nodes in the branch-and-bound tree with an optimal value higher than zero are fathomed ([40, p. 71])—since only routes with negative reduced costs are interesting.

Many configurations for solving the subproblems are possible. The configurations for the time limit used in the standard implementation are shown in Table 7. The great difference in size between the test instances makes it necessary to use different time limit configurations for different instances. Configurations CONF1 and CONF4 are used for the small instances Choi3–Choi6, configurations CONF2 and CONF5 are used for the medium-sized instances T13, T15–T16, and CONF3 and CONF6 are used for instances T14, T17–T20 in the standard implementation. Since T14 is much more computationally demanding than the other medium sized instances T13, T15–T16 (see the results in Section 5.1.1) it is therefore included in the latter group.

In addition to the time limit, a kind of tabu-strategy of partial column generation—described in Section 4.1.1—has also been implemented for the subproblems. When the tabu-strategy is used, subproblems that do not yield any route with negative reduced cost for a pre-determined number of consecutive column generation iterations, are not considered for a pre-determined number of iterations. This strategy reduces the number of subproblems that need to be solved each iteration. It is often the case that a subproblem that has not provided a route with negative reduced cost for a number of iterations will not provide one for many more iterations, so the negative impact of this procedure on the number of iterations needed for the column generation to converge should not be great. In the standard implementation, a subproblem is not considered again for seven or four consecutive iterations (the number depending on the current phase of the column generation solutiouree), if that subproblem has not provided a route with negative reduced cost for two consequtive iterations (when dynamic programming is used) or for four consecutive iterations (when AMPL and CPLEX are used). Note that when a time limit is set for the subproblems, there may exist a route with negative reduced cost for a specific subproblem, even if no route is found during the solution course for that subproblem.

Config	Dynamic programming			AMPL	
	T_{START} [CPU s]	S_{MULT}	S_{NUMINC}	T_1 [CPU s]	T_2 [CPU s]
CONF1	2	2	1	15	500
CONF2	5	3	1	50	500
CONF3	5	5	3	250	500
CONF4	-	-	-	2	10
CONF5	-	-	-	5	30
CONF6	-	-	-	25	300

Table 7: Different configurations of the time limit for the subproblems. T_{START} denotes the time limit for the dynamic programming in the first part of the column generation algorithm. If no route with negative reduced cost is found for any of the subproblems in a specific column generation iteration, and optimality has not been verified, then the time limit is multiplied by S_{MULT}. This is done S_{NUMINC} times, after which the subproblem is solved by AMPL and CPLEX. The first time limit in AMPL and CPLEX is T_1. If no route with negative reduced cost is found for any of the subproblems in a specific iteration, and optimality has not been verified, then the time limit is increased to T_2. If no route with negative reduced cost is found and optimality is still not been verified, then all subproblems are solved to optimality until the column generation algorithm has converged. No values given for T_{START}, S_{MULT}, and S_{NUMINC} means that dynamic programming is not used. The computations can be interrupted after a pre-defined time limit for the complete column generation algorithm has been exceeded, for any of the configurations.

The gain from the CPLEX option `uppercutoff=0` is in itself quite substantial, and together with the time limit and tabu-strategy for the subproblems, these extensions of the column generation algorithm led to greatly reduced solution times. So for example did the solution time for test instance T13 decrease from $2.3 \cdot 10^6$ to $7.3 \cdot 10^3$ CPU seconds, when `uppercutoff=0`, time limit, and the tabu-strategy were used instead of solving all subproblems to optimality in AMPL and CPLEX.

The improvements in solution time when using `uppercutoff=0` and time limits are illustrated for the instance T13 in Figures 4 and 5. The tabu-strategy for subproblems was not used here, to illustrate why it is useful (it is often the case that when a subproblem has not provided a route with negative reduced cost for a few iterations, it often will not do so for many more iterations, as can be seen in both Figures 4 and 5).

In Figures 4 and 5, solution times for column generation subproblems for all vehicles A–F in the instance T13 are shown; for each specific vehicle, the subproblems for which no route with negative reduced cost were found are indicated by circles. Figure 4 shows the subproblem solution times when all the subproblems are solved to optimality in AMPL—for the larger vehicles the subproblem solution times are very long (up to $50,000$ CPU s) in the second half of the column generation algorithm. This amount of time is more time than what is used by the whole column generation algorithm when using dynamic programming with time limits and `uppercutoff=0`, as shown in Figure 5.

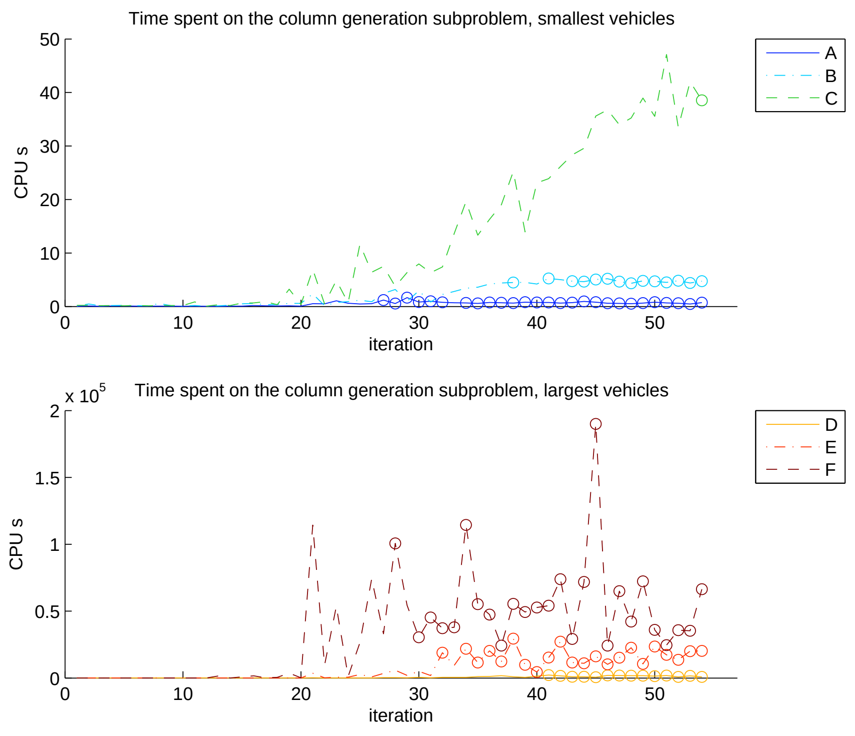

Figure 4: Time spent on solving the column generation subproblems for each of the vehicles A–F in the instance T13. In each iteration the subproblems are solved to optimality in AMPL. The vehicles D–F—having larger capacities—give rise to longer computing times than the vehicles A–C—with a smaller capacities. A circle indicates that the optimal value (the minimum reduced cost) of the corresponding subproblem is non-negative.

A few more iterations are needed when the subproblems are not solved to optimality in each iteration in this way (see Figure 5) but the gain in total solution time is great. After 58 column generation iterations, dynamic programming with the given time limit can no longer find any route with negative reduced cost for any of the vehicles. The subproblems are then solved to optimality in AMPL. This takes a lot more time than the time limit that was set for the dynamic programming, but nothing is actually gained in terms of the optimal objective value of the restricted master problem. Even though AMPL manages to find a route with negative reduced cost in each of the iterations 59 and 60, no improvement is made in the objective value of the restricted master problem after adding those routes. However, adding these routes makes it possible to verify that the solution to the restricted master problem in iteration 58 (the last iteration employing dynamic programming) is optimal in the complete master problem.

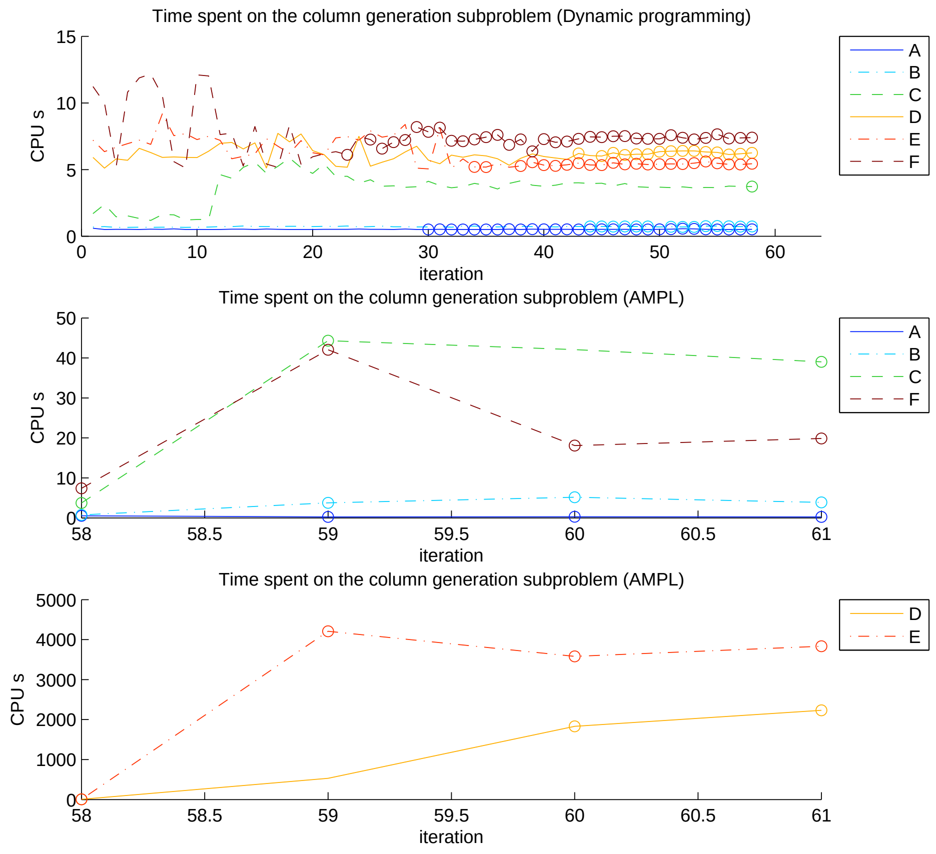

Figure 5: Time spent on the column generation subproblems for each vehicle A–F in the instance T13. Subproblems are solved by dynamic programming in Matlab with a time limit the first 58 iterations, whereas in iterations 59–61 subproblems are solved to optimality using CPLEX. The rings indicate that the minimum objective value (the minimum reduced cost) found for a subproblem is non-negative.

The impact of using `uppercutoff=0` is demonstrated by the fact that the solution time of the the subproblem for the largest vehicle (F) actually is less than that of C, D, and E (Figure 5). Compare with Figure 4, in which the solution time of the subproblem for vehicle F is much greater than for the other vehicles. AMPL sends previously found solutions to CPLEX as a warm-start when solving the next subproblem [40, pp. 60–61]. Since only the parameters $\hat{c}_{ij}^{k}$, $(i, j) \in \mathcal{A}$, and D_k differ between the subproblems ((23a)–(23e), (23h)–(23i), (24)), when using `uppercutoff=0` it is probably the case that the solution to the previously solved subproblem for vehicle E is utilized to start the subproblem solution for vehicle F. Hence AMPL can quickly conclude that the optimal value is greater than zero, without having to find this value.

5.2 The extended test instances, CT12EXT

To test the performance of the proposed algorithms in Section 4 for *many*-hVRP, the extended test instances, CT12EXT, were used. In CT12EXT, the vehicle sets are extended to include vehicles with all (integer) capacities between the lowest and highest capacity of the original instances; see Section 3.1.3. The resulting instances thus contain between 91 and 381 different vehicle types.

In Section 5.2.1, the straightforward model (15) of *many*-hVRP is assessed. The column generation applied to CT12 in Section 5.1.1 is here tested on the extended test instances of CT12EXT and the results are compared.

The extended instances CT12EXT have also been used to test the algorithms proposed for the restricted model (25). This model, when the vehicle type limit C is not restrictive, is equivalent to the straightforward model in that share optimal solutions and thus also optimal values—although different solution methods are applied to the two models. Benders' algorithm applied to the restricted model, with a non-restrictive vehicle type limit C, is assessed in Section 5.2.2, in which the results are also compared with the results obtained for the straightforward model presented in Section 5.2.1. A modified Benders' algorithm, using projection of routes, which yielded great improvements in the performance of the algorithm, is presented in Section 5.2.3. The convergence of Benders' algorithm is discussed in Section 5.2.4, and Benders' algorithm with a restrictive limit C is evaluated in Section 5.2.5.

Results of tests of the straightforward restricted models using load dependent costs are presented in Section 5.2.6.

The notation in Sections 5.2.2—5.2.4 follows that in Section 4.2.

5.2.1 Straightforward model with column generation

When the column generation algorithm for the straightforward model (15) was tested on the extended test instances CT12EXT, it was found that subproblems for vehicles with similar capacities often provide the same routes in the current column generation iteration—although with different objective values/reduced costs, because of the slightly different cost structures. For the instances CT12, having more dissimilar vehicle cost structures, the subproblems mostly provide different routes. The column generation algorithm was therefore adjusted so that routes were added only from the subproblem that provided the lowest objective value/reduced cost for the same route. Subproblems that did not provide any route—either because no route with a negative reduced cost was found, or because the route found with a negative reduced cost was also found in another subproblem, and with a lower reduced cost—for a pre-determined number of consecutive column generation iterations were as before not solved again for a number of iterations (the tabu-strategy of partial column generation). A 25 % (10 %) decrease in total solve time was found for the instance T13EXT (T15EXT), when the tabu strategy was included in the algorithm.

The extended instances CT12EXT represent relaxations of the original instances, since all solutions that are feasible in an original instance are also feasible in the corresponding extended instance. Comparing the results of the straightforward model on the smaller instances of CT12EXT (8) to those of the instances CT12 (Table 3) the lower bounds for the instances in CT12EXT are lower than those for CT12 as can be expected. The objective value of the best feasible solution to the binary program (15) is also lower for the instances Choi3EXT, Choi5EXT, and T13EXT. However, the for the instances Choi4EXT and Choi6EXT are higher than those for the instances Choi4 and Choi6, respectively. This is somewhat counterintuitive. Somehow the routes generated during the column generation solution course for problem (19), with test instances Choi4EXT and Choi6EXT, are not as suited for the binary program (15) as those routes generated for Choi4 and Choi6. Solving the problem again, allowing only the vehicle types that appear in the optimal solution to the problem (19) might yield better objective values, as indicated by the fact that the objective values of the best feasible solutions found using Benders' algorithm for the restricted model, with a non-restrictive vehicle type limit C (Table 9) for the instances Choi4EXT and Choi6EXT are lower than the corresponding values reported in Table 8.

Instance	Obj. value	LB	T_{OPT} [CPU s]	T_{TOT} [s]	Opt. gap [%]
Choi3EXT	1010.0	1010.0	2335	2364	0.00
Choi4EXT	6544.2	6366.0	3913	3902	2.80
Choi5EXT	1187.0	1180.6	2717	2601	0.54
Choi6EXT	6691.7	6436.6	3885	3817	3.96
T13EXT	2756.5	2748.9	55533	38522	0.28

Table 8: Results for the straightforward model with column generation. "Obj. value" refers to the objective value of the best feasible solution to (15) found, i.e. the value obtained in the last column generation iteration when binary requirements are added. "LB" refers to the optimal value of the LP relaxation (19) of the problem (15). "T_{OPT}" and "T_{TOT}" are defined as in Table 3. "Opt. gap" is given by ("Obj. value" - "LB")/"LB".

Comparing the solution times listed in Tables 8 and 3, it seems that the solution times scale quite well, considering that there are between 20 and 30 times more vehicles in the extended problems than in the original ones. The larger instances T14EXT-T20EXT have not been tested, because the original test instances T14-T20 were more difficult to solve than the instances Choi3-Choi6 and T13, and the test instance T13EXT took a substantial amount of time to solve as can be seen in Table 8.

5.2.2 The restricted model with Benders' algorithm—non-restrictive vehicle type limit C

Knowing an optimal solution to the LP relaxation (19) of the straightforward model, for an instance in CT12EXT, the vehicle type limit C in the restricted model can be set higher than the number of vehicle types used in the known optimal solution. This makes the optimal solution to the relaxation (26) of the straightforward model, an optimal solution to (19). This enables a comparison of the different solutions strategies for the straightforward model (column generation of Algorithm 1 and the restricted model (Benders' algorithm of Algorithm 3). As described in Section 4.2, when Benders' algorithm is implemented for the restricted model, Benders subproblems equal (19), but with a smaller set of vehicle types (including at most C types). Also, in each Benders iteration a feasible solution to the restricted model (26) is given—in contrast to the column generation applied to the straightforward model and which guarantees a feasible solution only in the last column generation iteration (when binary requirements are added in the restricted master problem).

The results for the restricted model with a non-restrictive limit C, for the smallest instances of CT12EXT, are given in Table 9. These results are compared with those obtained by using column generation for the straightforward model, listed in Table 8. Results for T13EXT are not included here, because calculating the values of the Benders dual variables $\boldsymbol{\gamma}$ is too time-consuming for this instance; especially for γ_k for the larger vehicle types k. Only vehicles among the 32 smallest vehicle types are used in an optimal solution, so using the lower bound suggested in Appendix C.3 to reduce the set of vehicle types would probably result in big improvements—as would using the lower bound on γ_k suggested in Section 4.2.5.

Benders' algorithm performs better than the the column generation for the straightforward model for some of the small instances, and quite a lot worse for some. For the instances Choi4EXT and Choi6EXT, Benders' algorithm converges to the optimal solution to (26) (being an optimal solution also to (19)) in just two and three iterations, respectively, taking less computing time than the column generation approach for the straightforward model. The best objective values of (25) found for these instances, when applying Benders' algorithm to the restricted model, are also better than those found applying column generation to the straightforward model. On the contrary, for the instances Choi3EXT and Choi5EXT, Benders' algorithm does not pick the vehicles that are used in the optimal solution to (26) even after 100 Benders iterations, even though this takes a lot more time than solving the straightforward model; see Table 8. The biggest difference between the original and extended sets of instances is that for Choi4EXT and Choi6EXT, the cost structure is such that only vehicle types among the five smallest are used in the optimal solutions to (26); this might explain why Benders' algorithm converges before 100 Benders iterations for these instances but not for Choi3EXT and Choi5EXT.

Instance	C	Obj_1	Obj_2	LB	T_{OPT} [CPU s]	T_{TOT} [s]	It	Opt. gap [%]
Choi3EXT	10	1010.5	1010.5	893.3	10124	31573	100	0.05
Choi4EXT	4	6366.0	6484.7	6366.0	1101	1119	3	1.86
Choi5EXT	13	1181.4	1188.7	1067.6	59302	52205	100	0.69
Choi6EXT	6	6436.6	6580.8	6436.6	1304	1298	4	2.24

Table 9: Results for Benders' algorithm. "C" is the parameter value of the vehicle type limit that restricts the number of vehicle types used in any solution to the problems (26) and (25). It is set to two units larger than the number of vehicle types used in the optimal solution to (19), which is thus not restrictive. "Obj_1" refers to the objective value of the best optimal solution to Benders subproblem, which is an upper bound on the optimal objective value of (26). "Obj_2" refers to the objective value of the best solution found when adding binary restrictions in the last iteration of the column generation algorithm performed to solve the Benders subproblem in each Benders iteration; it is an upper bound on the optimal objective value of (25). "LB" refers to the optimal objective value of Benders restricted master problem in the last Benders iteration, which is a lower bound on the optimal objective value of (26), and therefore also a lower bound on the optimal objective value of (25). "T_{OPT}" and "T_{TOT}" are defined as in Table 3. "It" refers to the number of Benders iterations performed. Maximally 100 Benders iterations have been performed. "Opt. gap" is given by ("Obj. value" - v^*)/v^*, where v^* is the optimal value of (26). Since the limit C is not restrictive, v^* equals the lower bound presented in Table 8.

Next, more detailed results are provided for the instance Choi3EXT. The optimal objective values of Benders subproblem and Benders restricted master problem with vehicle type limit $C = 10$, i.e., the restricted model with at most ten different vehicle types used in any solution, are shown for each Benders iteration in Figures 6 and 7. The best objective value of Benders subproblem found, 1010.5, is not found until iteration 81. It is close to the optimal value, 1010.0; see Table 8. The optimal objective value of Benders restricted master problem increases in each iteration, but after 20 iterations the increase is very small, although it is still far from the optimal objective value of Benders master problem.

Even though the optimal solution to (26) is not found during the 100 iterations for Choi3EXT, the vehicle types that are used in the optimal solution[13] are chosen quite frequently, which is illustrated in Figure 8. A similar pattern was found for the other instances tested as well. Figure 8 also shows that of the 101 vehicle types, only vehicles among the 39 smallest types are chosen in any Benders iteration.

[13]The optimal solution to the relaxed problem (26) is integer for Choi3EXT, and is therefore also an optimal solution to (25). This is, however, not a general property.

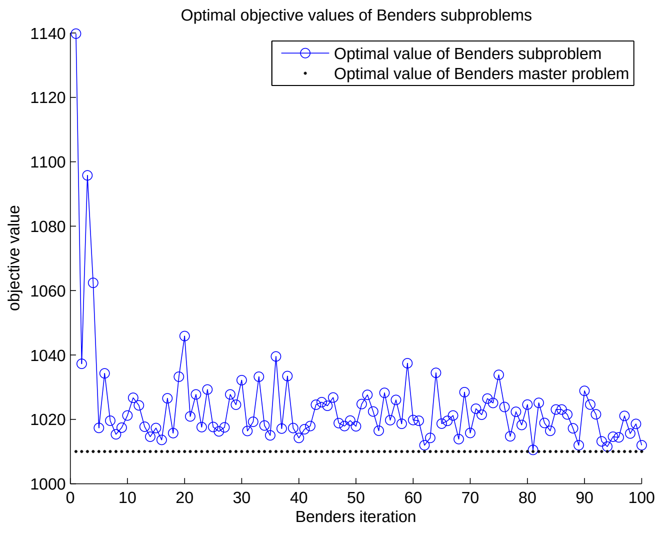

Figure 6: Optimal objective values of Benders subproblems over 100 iterations, for Choi3EXT with vehicle type limit $C = 10$; the best value, 1010.5, is attained at iteration 81. The optimal objective value of Benders master problem is 1010.

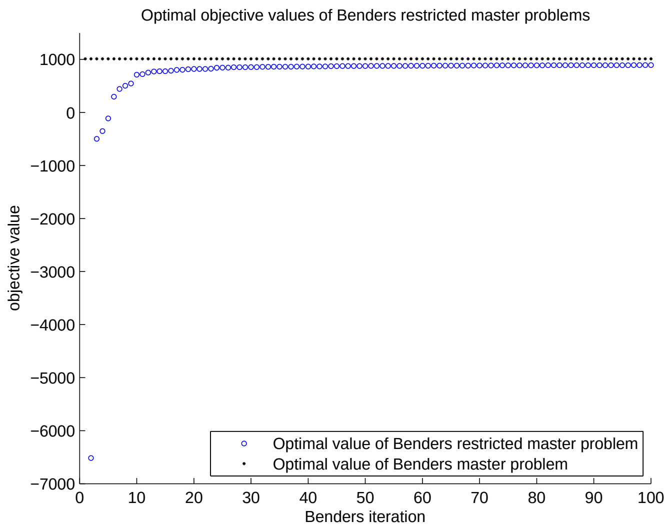

Figure 7: Optimal objective values of Benders restricted master problems over 100 iterations, for Choi3EXT with vehicle type limit $C = 10$. The mean increase in objective value between two successive Benders iterations is 4.1 (0.3) for iterations 10–50 (50–100); the best value attained is 893 (at iteration 100) as compared to the optimal value, 1010.0, of Benders master problem.

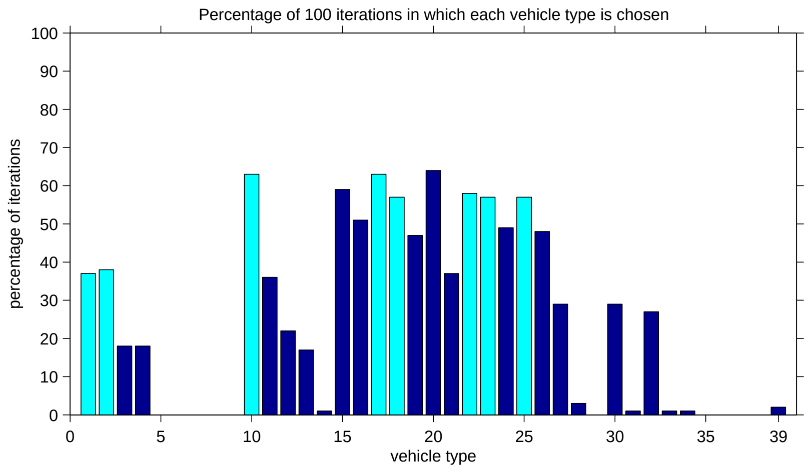

Figure 8: The percentage of 100 Benders iterations in which each of the vehicles (of type/size 1–101) is chosen, for Choi3EXT with vehicle type limit $C = 10$. The known optimal solution uses the vehicle set $\{1, 2, 10, 17, 18, 22, 23, 25\}$, which is marked by lighter bars.

An interesting fact is that after a few Benders iterations, the difference between the objective values of the first and last iterations (before adding binary restrictions on the variables) of the column generation applied to Benders subproblem is often very small when the column generation is warm-started with routes from solutions $\widetilde{\mathbf{x}}^l$ to previous Benders iterations; see Section 4.2.4. The usefulness of re-using routes in this way is demonstrated in Figure 9 for the instance Choi3EXT with vehicle type limit $C = 10$. In only thirteen of the 100 Benders iterations was the difference non-zero, and the greatest difference was found during the first iterations. It is also in those first Benders iterations that the objective value of Benders restricted master problem increases the most, see Figure 7.

This effect is also apparent in the computation time required for the column generation of Benders subproblems, as shown in Figure 10. The column generation for Benders subproblem in the first two Benders iterations requires more than twice the time that is required for the following iterations; this is consistent with the pattern apparent in Figure 9, in which the column generation in the three first Benders iterations results in a much larger difference than do the later iterations. The time spent on calculating the dual variable value $\boldsymbol{\gamma}^L$ from the dual variable value $\boldsymbol{\pi}^L$—found in the last iteration of the column generation—is less than the time spent on the column generation in all the Benders iterations. It takes less than one second to solve Benders restricted master problem, which is only a fraction of the time spent on Benders subproblems, hence the time spent on solving Benders restricted master problem is not included in Figure 10.

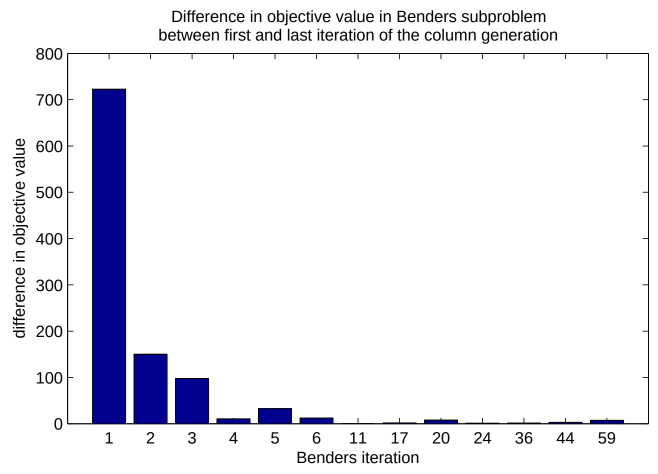

Figure 9: Difference in objective value between the first and last iterations of the column generation in Benders subproblem over 100 Benders iterations, for Choi3EXT with vehicle type limit $C = 10$. Only the thirteen iterations with a non-zero difference is shown. In the iterations $11, 17, 24$, and 36 the difference was in the interval $[0.2, 2]$, which is not clearly visible in the plot.

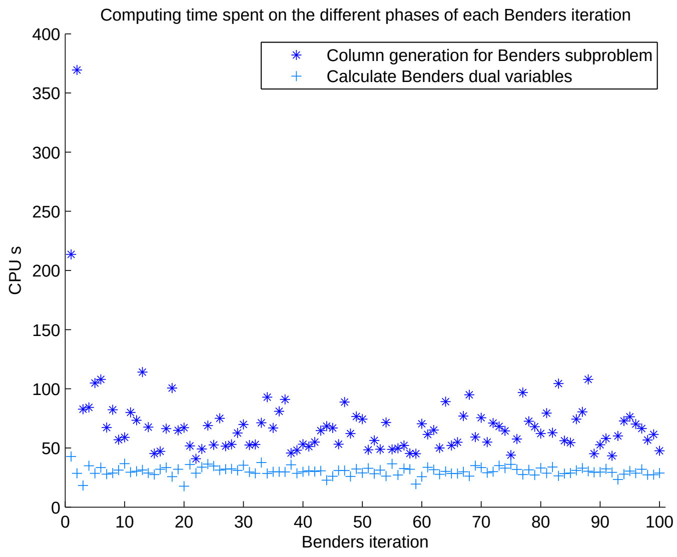

Figure 10: Time spent on the column generation algorithm applied to Benders subproblem and on calculating the dual variable values γ^L that define the new constraint in Benders restricted master problem, for Choi3EXT, with $C = 10$, over 100 Benders iterations.

5.2.3 Improved Benders' algorithm using projection of routes

Studying the optimal routes that are part of the solutions $\widetilde{\mathbf{x}}^l$ to (BendersSP($\widetilde{\mathbf{y}}^l$)), for Benders iterations $l = 1, \ldots, L$, i.e., studying the sets $\mathcal{R}^l$, defined in Section 4.2.4, it becomes clear that in some iterations the routes used in the optimal solution to (BendersSP($\widetilde{\mathbf{y}}^l$)) actually form an optimal solution to (26), however, they are not paired with the correct vehicle types.

In Section 4.2 we describe the procedure "projection of routes", in which routes being solutions to Benders subproblems in earlier iterations are combined with other vehicle types (that were not allowed in those subproblems but which provide lower objective values). This procedure has been implemented in order to improve the performance of Benders' algorithm. If such vehicle types are found and define a feasible Benders subproblem, then they are used in the next Benders iteration. This procedure seems to yield great improvements to Benders' algorithm for the instances Choi3EXT and Choi5EXT (for which Benders' algorithm was found too perform poorly in Section 5.2.2).

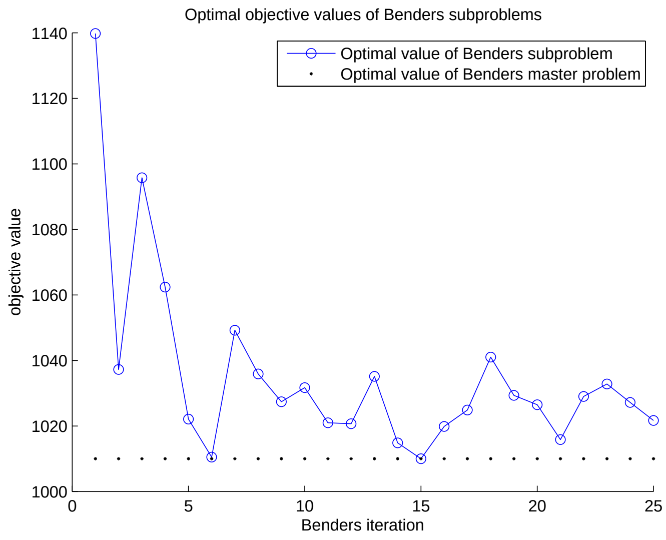

Figure 11: Optimal objective values of Benders subproblems over 24 iterations when the projection of routes is used, for Choi3EXT with the vehicle type limit $C = 10$. The best objective value 1010.0, which equals the optimal value of Benders master problem, was found in iteration 15 by projecting routes from iteration 14.

These improvements are illustrated in Figure 11 for the instance Choi3EXT, with the vehicle type limit $C = 10$. Projection of routes is tried out in every iteration from iteration 4; it is successful in iterations 5, 6, and 15. We compare this with the results shown in Figure 6 (Section 5.2.2), for the same instance solved without the projection of routes procedure. Without the projection of routes, the best objective value (1010.5) of Benders subproblem over 100 Benders iterations is found at iteration 81. When using projection of routes, this solution is found already at iteration six. Also, the best objective value (1010.0), found at iteration 15 when using the projection of routes, is optimal in (26).

Similar results were obtained for Choi5EXT, when projection of routes was used. The optimal objective value of (26) for Choi5EXT, 1180.6, was found in Benders subproblem in Benders iteration 16, while the best objective value of Benders subproblem over 100 iterations withouth using the projection of routes was 1181.4.

5.2.4 Convergence of Benders' algorithm

Unfortunately, as discussed in Section 4.2.3, Benders' algorithm does not always terminate when the optimal solution to (26) has been found, if not all necessary constraints to Benders master problem have been generated. This happens for the instance Choi3EXT with the vehicle type limit $C = 10$ (which is not restrictive), both when the projection of routes is used and when it is not (see Sections 5.2.2 and 5.2.3). Using the projection of routes, the optimal value of Benders master problem is found at Benders iteration 15, but the optimality is not verified until iteration 2342 (in order to speed up the convergence only the 30 smallest vehicle types of Choi3EXT were considered). A similar late verification of the optimality was observed for the instance Choi5EXT.

The optimality for Choi3EXT, with $C = 10$, could actually have been verified in iteration 16 if other values of Benders dual variables $\boldsymbol{\gamma}^{15}$ would have been found in Benders iteration 15. The values $\boldsymbol{\pi}^{15}$ found in Benders iteration 15 yields that $\boldsymbol{\gamma}^{15} \leq 0$ and $\boldsymbol{\gamma}^{15} \neq 0$ (here, $\boldsymbol{\pi}^{15}$ is the dual variable value to the problem (30) found in the last iteration of the column generation for Benders subproblem at Benders iteration 15, and $\boldsymbol{\gamma}^{15}$ is calculated according to Claim 2). The optimality criterion (32) of Benders' algorithm is not satisfied in Benders iteration 15. However, if the column generation for Benders subproblem is continued a couple of iterations, allowing for all vehicle types in $\mathcal{K}$, no improvement is made to the objective value of the column generation restricted master problem, and optimality of $\widetilde{\mathbf{x}}^{15}$ in the model (19) is verified by the optimality criterion for column generation (implying that $(\widetilde{\mathbf{x}}^{15}, \widetilde{\mathbf{y}}^{15})$ is optimal in (26)). A new dual solution $\hat{\boldsymbol{\pi}}$ to the column generation restricted master problem is found, for which $\hat{\boldsymbol{\gamma}} = 0$ (where $\hat{\boldsymbol{\gamma}}$ is calculated using $\hat{\boldsymbol{\pi}}$ according to Claim 2). It holds that $\hat{\boldsymbol{\pi}}$ is feasible and optimal in the dual of Benders subproblem in Benders iteration 15. If $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\gamma}})$ were used to define the new constraint to Benders restricted master problem in Benders iteration 15, instead of $(\boldsymbol{\pi}^{15}, \boldsymbol{\gamma}^{15})$, optimality of $(\widetilde{\mathbf{x}}^{15}, \widetilde{\mathbf{y}}^{15})$ in (26) could have been verified in Benders iteration 16 using the criterion (34), since then the inequality $\tilde{v}^{16} \geq \sum_{i \in \mathcal{N}_0} \pi_i^0$ holds, and we have that the inequalities $\tilde{v}^{16} \leq v^*$ and $\sum_{i \in \mathcal{N}_0} \pi_i^0 \geq v^*$ hold. This is a general result, i.e., if the values of the Benders dual variables $\boldsymbol{\gamma}$ equal zero in some Benders iteration, then optimality would be verified in the next Benders iteration if the criterion (34) was used.

Assume that the vehicle type limit C is not restrictive. According to (31), the inequality in $\boldsymbol{\gamma}^l = (\gamma_k^l)_{k\in\mathcal{K}} \leq \mathbf{0}^{|\mathcal{K}|}$ holds, where γ_k^l equals the minimum reduced cost over the variables x_r^k, $r \in \mathcal{R}_k$, if it is negative[14]. Therefore, there should exist a vector $(\boldsymbol{\pi}^{l_0}, \boldsymbol{\gamma}^{l_0})$ such that $\boldsymbol{\gamma}^{l_0} = \mathbf{0}^{|\mathcal{K}|}$, which is optimal in the dual of Benders subproblem in Benders iteration l_0, if $(\widetilde{\mathbf{x}}^{l_0}, \widetilde{\mathbf{y}}^{l_0})$ is an optimal solution to (26). This happens for the instances Choi4EXT and Choi6EXT with a non-restrictive limit C and for which Benders' algorithm converges quickly. As can be seen in Table 9, optimality was verified in iterations three and four, respectively. The optimal solutions were found in iterations two and three, respectively, and the values of Benders dual variables $\boldsymbol{\gamma}$ were in both cases found to be $\mathbf{0}^{|\mathcal{K}|}$.

5.2.5 The restricted model with Benders' algorithm—restrictive vehicle type limit C

If the goal is to choose a limited number of vehicle types from a larger set of vehicle types, then the straightforward model can not be used. That such a lim-

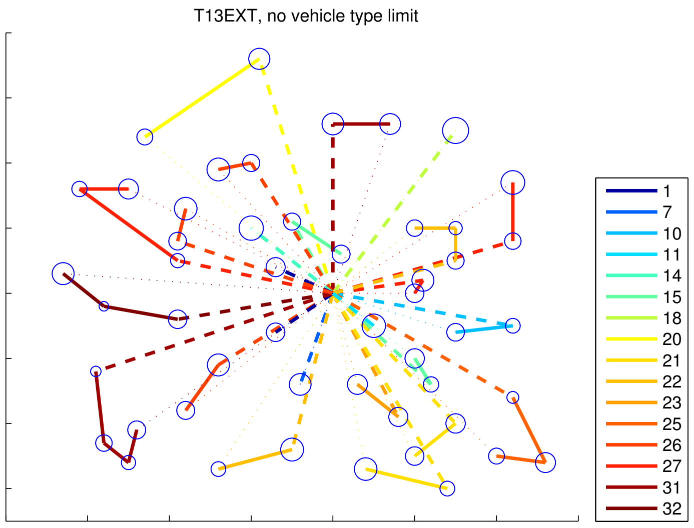

Figure 12: An illustration of the best solution to (15) found for the instance T13EXT with objective value 2753.6, when no vehicle type limit C is used ($C = \infty$). The depot is centrally located among the customer nodes, whose areas are proportional to their respective demand. The routes are marked with a dashed line for the first arc (leaving the depot), solid lines for following arcs, and a dotted line for the last arc (returning to the depot). Routes taken by different vehicle types have different colors. In this solution, only vehicles among the 32 smallest (out of 181) types are used.

[14]The reduced cost is defined with respect to the optimal solution $\widetilde{\mathbf{x}}^l$ to Benders subproblem in Benders iteration l, where Benders subproblem is seen as a column generation restricted master problem to (19) (since only a subset of all route-vehicle pairs is allowed); see Section 4.2.2.

itation can be imposed constitutes a valuable feature of the restrictive model, which could result in a greater flexibility of the resulting fleet.

Figure 12 illustrates a feasible solution for the instance T13EXT in which the allowed number of vehicle types used is unlimited, and Figure 13 shows a feasible solution in which the number of vehicle types is limited to four. The solution illustrated in Figure 13 was obtained using Benders' algorithm, by restricting the vehicle set to the 50 smallest vehicle types (only vehicles from the 32 smallest vehicle types were used in the optimal solution to T13EXT, when no vehicle type limit was set).

For the instance T13EXT with $C = 4$ (Figure 13), four vehicle types in the range $[22, 31]$ were used, whereas for the case with a non-restrictive vehicle type limit (Figure 12), 16 vehicle types in the range $[1, 32]$ were used. Hence, for this instance, out of the totally 181 vehicle types only vehicles among the 32 smallest types were used. Even though the objective value (2838.2) of the solution for the case with $C = 4$ is greater than that (2753.6) for the case with non-restrictive vehicle type limit, it is still lower than the optimal objective value (2964.7) of the original test instance T13 (see Section 5.2.1), which includes six vehicle types.

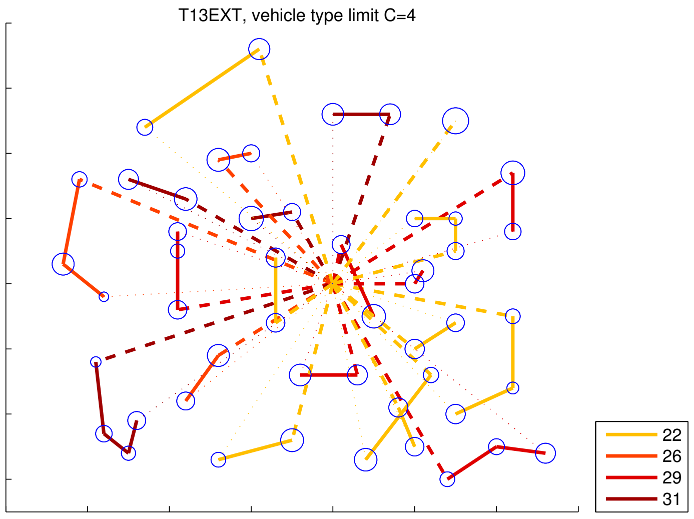

Figure 13: An illustration of the best solution to (25) found for the instance T13EXT with objective value 2838.2, with vehicle type limit $C = 4$. In this solution, only vehicle types in the range $[22, 32]$ are used, out of 181 vehicle types. Information about how to interpret the plot is found in Figure 12.

Choosing only a limited number of vehicle types, by using Benders' algorithm with projection of routes for the restricted model (25), with a restrictive vehicle type limit C, seems to work quite well. The biggest disadvantage is that it may take a long time to verify optimality using Benders' algorithm. When terminating after a large number of Benders iterations it is often revealed that the best solution was found early in the solution course.

For the instance Choi3EXT with $C = 6$ a feasible solution with objective value 1012.4 is found after eight Benders iterations when applying the projection of routes, and this value is not improved during the following 142 Benders iterations (since with $C = \infty$ the optimal solution, with objective value 1010.0, contains eight different vehicle types, the limit $C = 6$ is restrictive). This suggests that Benders' algorithm can be successfully used as a heuristic, thus terminated after a pre-specified time limit or no improvement of the objective value of Benders subproblem has been observed during a pre-specified number of iterations.

5.2.6 Load dependent costs

Both the straightforward model (15) and the restricted model (25) has been tested using load dependent costs, as described in Section 4.3. The parameters Q_{dist} and Q_{load} were set to 1.4 and 0.05, respectively, by testing different combinations for T13EXT using the straightforward model, and choosing those values that resulted in reasonable solutions.

Load dependent cost	Obj. value	LB	T_{OPT} [CPU s]	T_{TOT} [s]
Yes	2775.6	2801.9	34113	27243
No	2753.6	2748.9	35032	17658

Table 10: Results for the straightforward model using the column generation for the instance T13EXT, with and without load dependent costs. "Obj. value" refers to the objective value of the best feasible solution to (15) found, i.e. the value obtained in the last column generation iteration when binary requirements are added—given in the original cost of T13EXT. "LB" refers to the optimal value of the LP relaxation (19) of the problem (15). "T_{OPT}" refers to the total solve time in AMPL/CPLEX and Matlab. "T_{TOT}" refers to the total time of the whole algorithm.

Since some changes were made to the mathematical formulation of the column generation subproblem to enable the inclusion of load dependent costs (compare formulations (23a)–(23e), (23h)–(23i), (24) and (37)), the results using load dependent costs are compared to results obtained by setting $Q_{\text{dist}} = 1$ and $Q_{\text{load}} = 0$, which is equivalent to not having load dependent costs. This makes the comparison of solution times more relevant. In Table 10, results for the straightforward model using column generation, with and without load dependent costs, are presented for the instance T13EXT. The load dependent objective values have been converted to the original cost (by calculating the cost of the solution using (15a)), for ease of comparison. The total solve time T_{OPT} when using load dependent costs is less than when load dependent costs are not used, while the opposite is true for the total time T_{TOT}.

The difference between the objective value of the best feasible solution to (15) obtained using the load dependent cost model but with parameters $Q_{\text{dist}} = 1$ and $Q_{\text{load}} = 0$—presented in Table 10—and the objective value of the best feasible solution to (15) obtained using the straightforward model—presented in Table 8—can probably be explained by the fact that different formulations of the subproblem have been used in AMPL and CPLEX ((23a)–(23e), (23h)–(23i), (24) for the straightforward model and (37) for the load dependent cost model).

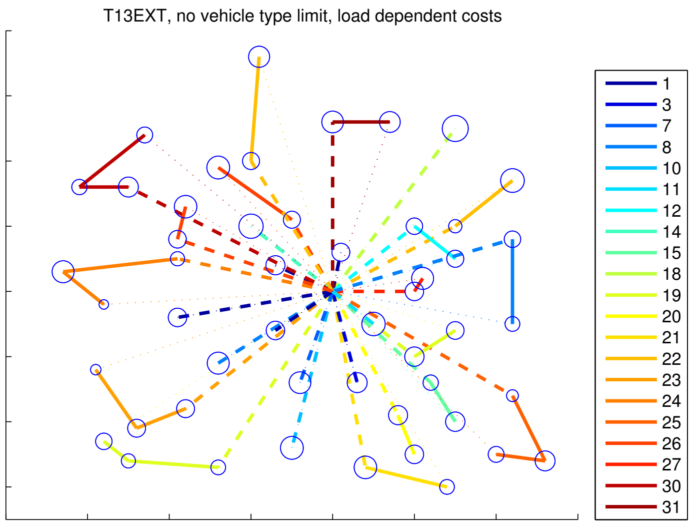

Figure 14: An illustration of the best solution to (15) (no vehicle type limit C) found for the instance T13EXT with load dependent costs. Objective value 2775.6 (given in the original cost of T13EXT). In this solution, only vehicles among the 32 smallest (out of 181) types are used. Information about how to interpret the plot is found in Figure 12.

The best feasible solution (with objective value 2775.6) obtained for T13EXT with load dependent costs is shown in Figure 14; compare this solution with the best feasible solution obtained for the case of load independent costs (Figure 12). A feasible solution to T13EXT with vehicle type limit $C = 4$ and load dependent costs is shown in Figure 15; this solution is quite different from the solution to T13EXT with load dependent cost, but no vehicle type limit (shown in Figure 14). Also, compare the solution in Figure 15 with the corresponding solution shown in Figure 13, which was obtained using the original, load independent, costs.

Smaller vehicle types are used to a larger extent when the costs are load dependent (Figures 14 and 15) than when they are load independent (Figures 12 and 13).

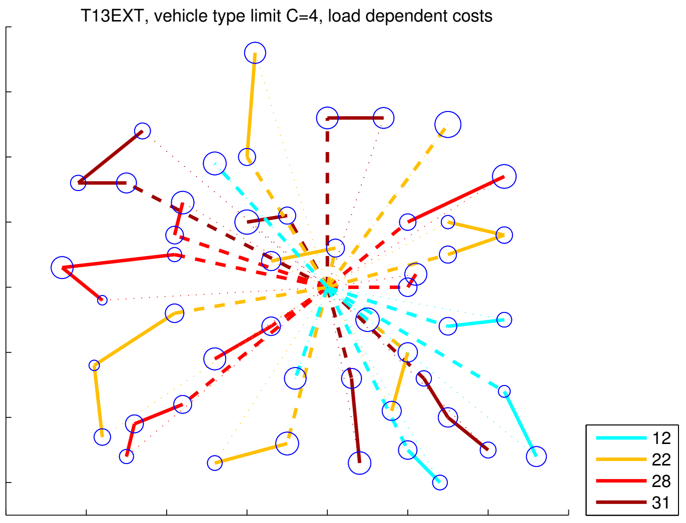

Figure 15: An illustration of the best solution to (25) found for the instance T13EXT with load dependent costs and with vehicle type limit $C = 4$, for test instance T13EXT with load dependent costs. Objective value 2882.6 (given in the original costs of T13EXT). In this solution, only vehicle types in the range $[12, 31]$ are used, out of 181 vehicle types. Information about how to interpret the plot is found in Figure 12.

6 Discussion

We have extended the standard heterogeneous vehicle routing problem (hVRP) to include a large set of vehicle types, then called *many*-hVRP. Further, we have developed and tested models and algorithms for *many*-hVRP.

The results of the tests performed on the original test instances CT12 (see Section 3.1.3) show that great improvements in the solution times were achieved—as demonstrated for the instance T13—using the time limit, the CPLEX-option `uppercutoff=0`, and the tabu-strategy of partial column generation (see Section 5.1.3). The tabu-strategy takes advantage of the fact that many subproblems are similar for extended test instances CT12EXT. The time limit also affects the tabu-strategy, since subproblems which do not yield a route with negative reduced cost within the given time limit for a few consecutive column generation iterations—so that those subproblems are skipped the following iterations—could have yielded routes with negative reduced cost given a more generous time limit. For the larger test instances, this has a great positive effect on the solution time of the column generation algorithm.

The results of the test performed on the extended test instances, CT12EXT, indicate that the straightforward model with column generation performs better than the restricted model with Benders' algorithm, when a non-restrictive vehicle type limit is set (which then has the same optimal solutions). Benders' algorithm converges very slowly for some test instances. With the implemented projection of routes procedure, the performance of Benders' algorithm can be improved—as demonstrated for the instance Choi3EXT—so that an optimal solution is found much sooner; however, the optimality of the solution still takes a very long time to verify.

Advantages of using Benders' algorithm are indicated by the following: a clear pattern emerges in which some vehicles types, which are part of an optimal set of vehicle types, are chosen more often than other vehicle types. Each constraint that is added to Benders restricted master problem has a nice interpretation as the objective value of the optimal solution to Benders subproblem plus a weighted sum of "reduced costs" (see Section 4.2.2), thus the collected information gained from the "reduced costs" of previous solutions to Benders subproblem is used when new vehicle types are chosen in each Benders iteration—this seems to provide a good guide for choosing new vehicle types. But, even though Benders' algorithm succeeds in choosing optimal vehicle types more often than non-optimal vehicle types, the specific configuration of the optimal set of vehicle types is sometimes hard to find. After a few iterations, the vehicles that are chosen do not seem to yield any improvements to previous solutions, in the sense that there is no difference between the first and last iteration of the column generation algorithm used to solve Benders subproblem (when the column generation is warm-started with routes from optimal solutions to Benders subproblems from previous iterations).

These properties of Benders' algorithm suggest that it may constitue a good basis for a heuristic, when the set of vehicle types is even larger than in the test instances CT12EXT. Since an optimal configuration of vehicle types can often be found among the vehicle types that are frequently chosen by Benders' algorithm, and those vehicle types that are frequently chosen typically form a much smaller set than the whole set of vehicle types, the set of vehicle types can after a few Benders iterations be reduced to the types that have been frequently chosen—resulting in a problem instance that is easier to solve.

The set of optimal routes for a given problem depends to a great extent on the vehicle types that are available. The improvements gained by using the projection of routes procedure suggest that Benders' algorithm succeeds in choosing vehicle type configurations which yield Benders subproblems whose optimal solutions are composed by high quality routes. Using only those routes that were part of previous optimal solutions to Benders subproblems, projection of routes has been found to determine the optimal configuration of vehicles and routes. Thus, for problems instances where the set of vehicle types are much larger than those in the set of extended test instances CT12EXT, a good approach may be to temporarily restrict the set of vehicle types, and use some procedure similar to the projection of routes—where vehicle types that are not part of the restricted set are allowed to be matched with routes from solutions to Benders subproblems—to find better solutions after a number of Benders iterations have been performed, and extend the restricted set accordingly.

The restricted model, with a restrictive vehicle type limit C, was shown to yield solutions with different characteristics than with an non-restrictive limit. Extending the straightforward and the restricted model, to include load dependent cost, also worked well; the solution times were not greatly impacted, and solutions with slightly different characteristics were found when load dependent costs were considered—vehicle types with smaller capacities were used more often than when load independent costs were considered. Other extensions, mentioned in Section 3, can be similarly incorporated into the models.

6.1 Future research and development

To improve the performance of the proposed algorithms, several important adjustments can be made. For the straightforward model, using column generation, the dynamic programming algorithm should be implemented in C rather than in Matlab. Also, the column generation subproblems should be relaxed, as is done in many implementations of column generation for vehicle routing problems (see Section 2.2) by Choi and Tcha [24], among others. The set-partitioning problem should be turned into a set-covering problem, solved using (possibly heuristic) branch-and-price. The lower bound, described in Appendix C.3, should be used to reduce the set of vehicle types. The results in Section 5.1.1 suggest these changes would result in shorter computation times and that solutions with better objective values would be found.

For the restricted model, using Benders' algorithm, the relaxation suggested for the column generation subproblems should be applied to the problems (31) that are solved in each Benders iteration, as is suggested in Section 4.2.5. This is an important change, since in each Benders iteration, the model (31) needs to be solved for each vehicle type.

The models should also be changed to include more features of real life problems. Even bigger sets of vehicles could be handled as suggested in Section 6. The load dependent costs could be made more realistic by letting the parameters Q_{dist} and Q_{load} be arc-dependent; only minor adjustments to the code would be necessary to allow for this. Other extensions, such as time windows can easily be incorporated into the models. With tight time-windows, the problem may actually be easier to solve ([15]). An extended model could be based on the techniques used by Ceselli et al. in [41], who consider a hVRP with multiple depots, as well as

> time windows associated with depots and customers; incompatibility constraints between goods, depots, vehicles, and customers; maximum route length and duration; upper limits on the number of consecutive driving hours and compulsory drivers' rest periods; the possibility of skipping some customers and using express courier services instead of the given fleet to fulfill some orders; the option of splitting up the orders; and the possibility of open routes that do not terminate at depots [41, abstract],

using a column generation algorithm with dynamic programming for the subproblems. In Betinelli et al. ([15]), a similar hVRP with multiple depots and time windows is solved, using a branch-and-cut-and-price heuristic. The authors state that it is probably the heterogeneous fleet that "really complicates the problem" [15, p. 735], but that their implementation often performs better than other heuristics from the literature. Heuristics such as these—based on mathematical programming techniques—are becoming more popular, as was mentioned in Section 3. The findings in [15] indicate that the models developed in this thesis could be competitive as heuristics for the *many*-hVRP.

We think that the solution methods developed in this thesis, in which the difficult problem of handling a large set of vehicle types is dealt with, form a promising foundation on which to build more complex models. As indicated in the beginning of Section 3, there are many possible extensions of the standard vehicle routing problem. To further develop the models and algorithms developed in this thesis in order to construct a framework that is relevant to real-life problems, extensions of the *many*-hVRP models should be considered, more complex data sets should be used, and the solution algorithms should be further developed according to the suggestions above.

Appendices

A Notation

A vector $\mathbf{x} \in \mathbb{R}^n$ is understood to be a column-vector, and the scalar product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is written $\mathbf{x}^\top \mathbf{y}$. The two vectors $\mathbf{x} \in \mathbb{R}^{n_1}, \mathbf{y} \in \mathbb{R}^{n_2}$ are concatenated as $\mathbf{z} := (\mathbf{x}, \mathbf{y})$, where $\mathbf{z} \in \mathbb{R}^{n_1+n_2}$ is assumed to be a column vector. Furthermore, when given variables $(x_i)_{i\in\mathcal{I}}$, where $\mathcal{I}$ denotes a set of indices, the vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{I}|}$ is implicitly assumed to equal $(x_i)_{i\in\mathcal{I}}$. If no order is imposed on the index sets $\mathcal{I}$ and $\mathcal{J}$, given a matrix $\mathbf{A} := (\mathbf{a}_i^\top)_{i\in\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{J}|}$, where $\mathbf{a}_i$, $i \in \mathcal{I}$ are vectors in $\mathbb{R}^{|\mathcal{J}|}$ that define the rows of $\mathbf{A}$, and given a vector $\mathbf{x} := (x_j)_{j\in\mathcal{J}}$, the matrix-vector product $\mathbf{Ax}$ is defined by the assumed ordering, so that for each $i_0 \in \mathcal{I}$, $(\mathbf{Ax})_{i_0} := \mathbf{a}_{i_0}^\top \mathbf{x} := \sum_{j\in\mathcal{J}} (\mathbf{a}_{i_0})_j x_j$. Similar notation is used when variables are indexed using several sets of indices $\mathcal{I}_1, \ldots, \mathcal{I}_n$, as $(x_{i_1,\ldots,i_n})_{i_1\in\mathcal{I}_1,\ldots,i_n\in\mathcal{I}_n}$.

B Column generation

Here a thorough description of column generation is given. In Appendix B.1, a Dantzig-Wolfe decomposition of a linear program is presented together with the mathematical theory that column generation of the decomposed problem is based. In Appendix B.2 a Dantzig-Wolfe decomposition and column generation algorithm for an integer program is presented. It is shown to provide a better lower bound than the linear programming relaxation of the original problem (i.e., when the integrality requirements are relaxed).

B.1 Linear program

This presentation, unless otherwise indicated, is based on [9, Chapters 2.6, 3.3–3.4], and on linear programming theory from the textbook [4].

Here, column generation in connection with a Dantzig-Wolfe decomposition is presented in more detail, but the same theory can be applied to column generation as it is presented in Section 2.1. Using Dantzig-Wolfe decomposition has the advantage that special structures in the problem can be utilized [8]. Consider the linear program

$$\text{(LP)} \qquad \min_{\mathbf{x}} \ \mathbf{c}^\top \mathbf{x}, \tag{38a}$$

$$\text{s.t.} \ \mathbf{Dx} = \mathbf{d}, \tag{38b}$$

$$\mathbf{Ax} = \mathbf{b}, \tag{38c}$$

$$\mathbf{x} \geq \mathbf{0}^n, \tag{38d}$$

where $\mathbf{c}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{d} \in \mathbb{R}^{m_1}$, $\mathbf{b} \in \mathbb{R}^{m_2}$, $\mathbf{D} \in \mathbb{R}^{m_1\times n}$ and $\mathbf{A} \in \mathbb{R}^{m_2\times n}$. The set of feasible solutions to (LP) is assumed to be non-empty and bounded.

When decomposing the problem, the constraints (38b) are regarded as *complicating*. This could be desirable if the problem has a specific structure, so that the constraints (38b) and (38c) are easier to deal with separately, or if (38b) represents linking constraints [42]. Then, defining the polyhedron $X_{\text{LP}} := \{\mathbf{x} \geq \mathbf{0}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$, any $\mathbf{x} \in X_{\text{LP}}$ can be expressed as a convex combination of its extreme points $\mathbf{x}^p$, $p \in \mathcal{P}$, i.e., $\mathbf{x} := \sum_{p\in\mathcal{P}} \alpha_p \mathbf{x}^p$ such that $\sum_{p\in\mathcal{P}} \alpha_p = 1, \alpha_p \geq 0, p \in \mathcal{P}$ hold. The model (LP) can thus be rewritten as the *complete master problem*

$$
\begin{aligned}
\text{(DW)} \qquad \min_{\boldsymbol{\alpha}} \ & \sum_{p\in\mathcal{P}} \alpha_p \mathbf{c}^\top \mathbf{x}^p, \\
\text{s.t.} \ & \sum_{p\in\mathcal{P}} \alpha_p \mathbf{D}\mathbf{x}^p = \mathbf{d}, \\
& \sum_{p\in\mathcal{P}} \alpha_p = 1, \\
& \alpha_p \geq 0, \qquad p \in \mathcal{P}.
\end{aligned}
$$

(DW) is called the Dantzig-Wolfe formulation of (LP). Whereas (LP) has n columns, the decomposed problem (DW) contains one column for each extreme point of the polyhedron X_{LP}; hence, there are $|\mathcal{P}|$ columns in (DW). If $|\mathcal{P}|$ is a very large number, column generation can be a good approach to solving (DW). Defining a subset $\widetilde{\mathcal{P}} \subset \mathcal{P}$, the column generation *restricted master problem* is given by

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}} \ & \sum_{p\in\widetilde{\mathcal{P}}} \alpha_p \mathbf{c}^\top \mathbf{x}^p, && \text{(39a)} \\
\text{s.t.} \ & \sum_{p\in\widetilde{\mathcal{P}}} \alpha_p \mathbf{D}\mathbf{x}^p = \mathbf{d}, && \text{(39b)} \\
& \sum_{p\in\widetilde{\mathcal{P}}} \alpha_p = 1, && \text{(39c)} \\
& \alpha_p \geq 0, \qquad p \in \widetilde{\mathcal{P}}. && \text{(39d)}
\end{aligned}
$$

The linear programming dual problem of the restricted master problem is given by

$$
\begin{aligned}
\max_{\boldsymbol{\pi},q} \ & \left(\mathbf{d}^\top \boldsymbol{\pi} + q\right), && \text{(40a)} \\
\text{s.t.} \ & (\mathbf{D}\mathbf{x}^p)^\top \boldsymbol{\pi} + q \leq \mathbf{c}^\top \mathbf{x}^p, \qquad p \in \widetilde{\mathcal{P}}, && \text{(40b)}
\end{aligned}
$$

where $\boldsymbol{\pi} \in \mathbb{R}^n$, $q \in \mathbb{R}$. The column generation subproblem is given by

$$
z^*_{\text{SubP}} := \min_{p\in\mathcal{P}} \left(\mathbf{c}^\top \mathbf{x}^p - (\mathbf{D}\mathbf{x}^p)^\top \boldsymbol{\pi}^* - q^*\right), \tag{41}
$$

where $(\boldsymbol{\pi}^*, q^*)$ is an optimal solution to (40). As is described in Section 2.1, in each column generation iteration the restricted master problem—in this case

(39)—is solved, after which the solution to the subproblem—in this case (41)—either yields a column/extreme point to be added to the restricted master problem (i.e., if $z^*_{\mathrm{SubP}} < 0$) or establishes that an optimal solution to the complete master problem has been found (i.e., if $z^*_{\mathrm{SubP}} = 0$). The problem (LP) has been decomposed, so that constraints (38b) and (38c) are dealt with separately, with (38b) taken care of in the restricted master problem and (38c) in the subproblem. In the remainder of this section, the theory behind this algorithm is presented.

For ease of notation, let $\omega_p := \mathbf{c}^\top \mathbf{x}^p$, $\bar{\mathbf{d}}_p := (\mathbf{D}\mathbf{x}^p, 1)$, and $\bar{\mathbf{d}} := (\mathbf{d}, 1)$, so that (39a) can be written as $\min_{\boldsymbol{\alpha}} \sum_{p\in\mathcal{P}} \alpha_p \omega_p$ and (39b)–(39c) as

$$\sum_{p\in\mathcal{P}} \alpha_p \bar{\mathbf{d}}_p = \bar{\mathbf{d}}.$$

As mentioned in Section 2.1, an optimal solution to the restricted master problem (39) can be found in an extreme point to the feasible set of solutions defined by the constraints (39b)–(39d). It holds that any such extreme point can have at most $m_1 + 1$ non-zero coefficients, since $\bar{\mathbf{d}} \in \mathbb{R}^{m_1+1}$. Let $\boldsymbol{\alpha}^*_{\mathrm{RMP}}(\mathbf{B}^*) := (\alpha^*_p)_{p\in\widetilde{\mathcal{P}}}$ be such an optimal solution to the restricted master problem, where $\mathbf{B}^* := \big(\bar{\mathbf{d}}_p\big)_{p\in\mathcal{B}^*}$ is a corresponding optimal basis matrix, i.e. $\{p \in \widetilde{\mathcal{P}} \mid \alpha^*_p > 0\} \subseteq \mathcal{B}^* \subseteq \widetilde{\mathcal{P}}$, such that $|\mathcal{B}^*| = m_1 + 1$, $\mathbf{B}^*$ is non-singular and $(\alpha^*_p)_{p\in\mathcal{B}^*} := (\mathbf{B}^*)^{-1}\bar{\mathbf{d}}$. Such a basis matrix $\mathbf{B}^*$ always exists, assuming the rank of $\big(\bar{\mathbf{d}}_p\big)_{p\in\mathcal{P}} = m_1 + 1$ and $|\widetilde{\mathcal{P}}| \geq m_1 + 1$.

Let $\boldsymbol{\alpha}_{\mathbf{B}} := (\alpha_p)_{p\in\mathcal{P}}$ be a basic feasible solution to the complete master problem (DW), which is also an extreme point to its feasible set, given by

$$(\alpha_p)_{p\in\mathcal{B}} := \mathbf{B}^{-1}\bar{\mathbf{d}}, \qquad (\alpha_p)_{p\in\mathcal{P}\setminus\mathcal{B}} := \mathbf{0},$$

where $\mathbf{B} := \big(\bar{\mathbf{d}}_p\big)_{p\in\mathcal{B}}$, $\mathcal{B} \subseteq \mathcal{P}$, is a (non-singular) basis matrix, and define $\boldsymbol{\omega}_{\mathbf{B}} := (\omega_p)_{p\in\mathcal{B}}$. The reduced cost $\hat{\omega}_p$ of the variable α_p, $p \in \mathcal{P}$, can be written as

$$\hat{\omega}_p = \omega_p - \boldsymbol{\omega}_{\mathbf{B}}^\top \mathbf{B}^{-1}\bar{\mathbf{d}}_p;$$

see [4, p. 227]. If $\boldsymbol{\alpha}_{\mathbf{B}}$ is not optimal in (DW), at least one reduced cost $\hat{\omega}_p$, $p \in \mathcal{P} \setminus \mathcal{B}$, must be negative. The basis matrix $\mathbf{B}$ can then be changed by removing one column $\bar{\mathbf{d}}_{p_{\mathrm{out}}}$, $p_{\mathrm{out}} \in \mathcal{B}$, and adding a column $\bar{\mathbf{d}}_{p_{\mathrm{in}}}$, $p_{\mathrm{in}} \in \{p \in \mathcal{P}\setminus\mathcal{B} : \hat{\omega}_p < 0\}$ (so that the new basis matrix is defined by the set $\mathcal{B} \setminus p_{\mathrm{out}} \cup p_{\mathrm{in}}$) such that the the objective value of (DW) corresponding to $\boldsymbol{\alpha}_{\mathbf{B}}$ is improved (assuming that $\boldsymbol{\alpha}_{\mathbf{B}}$ is non-degenerate, i.e., that $\mathbf{B}^{-1}\bar{\mathbf{d}} > \mathbf{0}^{m_1+1}$). This holds since the reduced cost $\hat{\omega}_p$ represents the change in objective value of (DW) when the variable α_p is increased by one, moving in the direction of a specific neighbouring extreme point. If all the reduced costs $\hat{\omega}_p$, $p \in \mathcal{P} \setminus \mathcal{B}$, are non-negative, then $\boldsymbol{\alpha}_{\mathbf{B}}$ is optimal in (DW).

With $\mathbf{B}^*$ being an optimal basis matrix in the restricted master problem (39), $\big(\boldsymbol{\alpha}^*_{\mathrm{RMP}}(\mathbf{B}^*)\big)_p = \big(\boldsymbol{\alpha}_{\mathbf{B}^*}\big)_p$, for $p \in \widetilde{\mathcal{P}}$, and $(\boldsymbol{\alpha}_{\mathbf{B}^*})_p = 0$, $p \in \mathcal{P} \setminus \widetilde{\mathcal{P}}$ since $\mathcal{B}^* \subseteq \widetilde{\mathcal{P}}$. Thus, if $\boldsymbol{\alpha}_{\mathbf{B}^*}$ is not optimal in the complete master problem (DW), the inclusion of any of the indices $\left\{ p \in \mathcal{P} \setminus \widetilde{\mathcal{P}} \,\middle|\, \hat{\omega}_p < 0 \right\}$ (where we have that $\hat{\omega}_p = \omega_p - \boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1}\bar{\mathbf{d}}_p$), to $\widetilde{\mathcal{P}}$ would improve the value of the optimal solution to (39) (again, assuming non-degeneracy of $\boldsymbol{\alpha}_{\mathbf{B}^*}$). On the other hand, if all reduced costs $\hat{\omega}_p$, $p \in \mathcal{P}$, are non-negative, then $\boldsymbol{\alpha}_{\mathbf{B}^*}$ is optimal in the complete problem (DW). As shown next, the reduced costs $\hat{\omega}_p$, $p \in \mathcal{P}$, given an extreme point $\boldsymbol{\alpha}_{\mathbf{B}^*}$ of the feasible set of (DW) with basis representation $\mathbf{B}^*$, can be rewritten using a set of optimal dual variables to the restricted master problem—this will determine the column generation subproblem (41).

Since $\alpha_p^* = 0$ for all $p \in \widetilde{\mathcal{P}} \setminus \mathcal{B}^*$, it holds that

$$\sum_{p \in \widetilde{\mathcal{P}}} \alpha_p^* \mathbf{c}^\top \mathbf{x}^p = \boldsymbol{\omega}_{\mathbf{B}^*}^\top (\alpha_p^*)_{p \in \mathcal{B}^*} = \boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1} \bar{\mathbf{d}} = \boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1} (\mathbf{d}, 1). \tag{42}$$

Also, for any optimal solution to a linear program there exists a basis such that all reduced costs are non-negative, although if the solution is degenerate some bases may yield negative reduced costs [4, p. 228]. Assuming that the basis matrix $\mathbf{B}^*$ above has been chosen properly, the reduced costs $\hat{\omega}_p$, $p \in \widetilde{\mathcal{P}}$, must therefore be non-negative since $\boldsymbol{\alpha}_{\mathrm{RMP}}^*(\mathbf{B}^*) := (\alpha_p^*)_{p \in \widetilde{\mathcal{P}}}$ is an optimal solution to the restricted master problem (39). Consequently, $\boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1}$ satisfies (40b), and is thus feasible in the restricted master problem dual (40). For any $(\boldsymbol{\pi}, q)$ feasible in (40), by (39b)–(39c) and (40b), it holds that

$$\sum_{p \in \widetilde{\mathcal{P}}} \alpha_p^* \mathbf{c}^\top \mathbf{x}^p \geq \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p^* \left((\mathbf{D}\mathbf{x}^p)^\top \boldsymbol{\pi} + q \right) = (\boldsymbol{\pi}, q)^\top \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p^* (\mathbf{D}\mathbf{x}^p, 1) = (\boldsymbol{\pi}, q)^\top (\mathbf{d}, 1).$$

Together with (42), this implies that

$$\boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1} (\mathbf{d}, 1) = \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p^* \mathbf{c}^\top \mathbf{x}^p \geq (\boldsymbol{\pi}, q)^\top (\mathbf{d}, 1),$$

which shows that $\boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1}$ is an optimal solution to (40). Define

$$(\boldsymbol{\pi}^*, q^*) := \boldsymbol{\omega}_{\mathbf{B}^*}^\top (\mathbf{B}^*)^{-1}. \tag{43}$$

The reduced costs, corresponding to the extreme point $\boldsymbol{\alpha}_{\mathbf{B}^*}$ with basis representation $\mathbf{B}^*$, can now be written as

$$\hat{\omega}_p = \mathbf{c}^\top \mathbf{x}^p - (\mathbf{D}\mathbf{x}^p)^\top \boldsymbol{\pi}^* - q^*, \qquad p \in \mathcal{P},$$

with the particular choice of optimal solution $(\boldsymbol{\pi}^*, q^*)$ to (40) given by (43). Thus, finding the variable α_p, $p \in \mathcal{P} \setminus \widetilde{\mathcal{P}}$, with minimal reduced cost amounts to solving the column generation subproblem (41). If $z_{\mathrm{SubP}}^* < 0$, then there exists some variables with negative reduced cost, and adding such a variable to the restricted master problem (39) improves the value of its optimal solution, provided that it is non-degenerate. If $z_{\mathrm{SubP}}^* = 0$, then there are no variables with negative reduced cost, and the current solution $\boldsymbol{\alpha}^*_{\mathrm{RMP}}(\mathbf{B}^*)$ to the restricted master problem (39) defines an optimal solution $\boldsymbol{\alpha}_{\mathbf{B}^*}$ the complete master problem (DW).

If (DW) has an optimal solution which is degenerate, then it may be the case that $z^*_{\mathrm{SubP}} < 0$ even when the optimal solution to the restricted master problem defines an optimal solution to (DW). This holds, since some basis representations of an optimal solution to (DW) may cause some reduced costs to be strictly negative [4, p. 228]. Thus, $\mathbf{B}^*$ in (43) may need to be replaced by some other basis, using some variables that are not included in the restricted master problem, for optimality to be verified.

B.2 Binary program

A binary linear programming problem of the type

$$\text{(BLP)} \quad \begin{aligned} \min_{\mathbf{x}} \ & \mathbf{c}^\top \mathbf{x}, \\ \text{s.t. } & \mathbf{D}\mathbf{x} = \mathbf{d}, \\ & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \in \{0,1\}^n, \end{aligned}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{d} \in \mathbb{R}^{m_1}$, $\mathbf{b} \in \mathbb{R}^{m_2}$, $\mathbf{D} \in \mathbb{R}^{m_1 \times n}$ and $\mathbf{A} \in \mathbb{R}^{m_2 \times n}$, might be very difficult to solve due to the binary requirements on the variables. We assume that the feasible set of (BLP) is non-empty. Solving a convexified linear program

$$\text{(BLPConv)} \quad \begin{aligned} \min_{\mathbf{x}} \ & \mathbf{c}^\top \mathbf{x}, \\ \text{s.t. } & \mathbf{D}\mathbf{x} = \mathbf{d}, \\ & \mathbf{x} \in \mathrm{conv}(X_{\mathrm{BLP}}), \end{aligned}$$

where $X_{\mathrm{BLP}} := \{\mathbf{x} \in \{0,1\}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$, using column generation may be a good approach to finding a near-optimal solution to (BLP) (see [42]). The solution of (BLPConv) can also yield a good quality lower bound that can be utilized in a branch-and-bound algorithm to find an optimal solution to (BLP). This combination of column generation and branch-and-bound is called branch-and-price [8].

Even though the optimal solution to (BLPConv) only provides a lower bound of the optimal value on the original problem (BLP), it is at least as good as that of the LP relaxation of (BLP) given by

$$\min_{\mathbf{x}} \ \mathbf{c}^\top \mathbf{x}, \tag{44a}$$
$$\text{s.t. } \mathbf{D}\mathbf{x} = \mathbf{d}, \tag{44b}$$
$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{44c}$$
$$\mathbf{x} \in [0,\ 1]^n, \tag{44d}$$

since the set

$$\operatorname{conv}(X_{\mathrm{BLP}}) := \operatorname{conv}\left(\{\, \mathbf{x} \in \{0,1\}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b} \,\}\right) \subseteq \{\, \mathbf{x} \in [0,\ 1]^n : \mathbf{A}\mathbf{x} = \mathbf{b} \,\}.$$

The set $\operatorname{conv}(X_{\mathrm{BLP}})$ is a bounded polyhedron since X_{BLP} is a finite set, so (BLPConv) can be reformulated using the Dantzig-Wolfe decomposition as

$$\begin{aligned} \text{(DWConv)} \qquad \min_{\boldsymbol{\alpha}} \ & \sum_{p \in \mathcal{P}} \alpha_p \mathbf{c}^\top \mathbf{x}^p, \\ \text{s.t.} \ & \sum_{p \in \mathcal{P}} \alpha_p \mathbf{D}\mathbf{x}^p = \mathbf{d}, \\ & \sum_{p \in \mathcal{P}} \alpha_p = 1, \\ & \alpha_p \geq 0, p \in \mathcal{P}, \end{aligned}$$

where $\mathbf{x}^p, p \in \mathcal{P}$, denote the extreme points of $\operatorname{conv}(X_{\mathrm{BLP}})$. Since the variables of the original problem (BLP) are binary, these extreme points are simply the elements in the set X_{BLP}. An optimal solution to this problem is found using column generation by iteratively solving the restricted master problem of (DWConv), as described in Appendix (see also B.1 [8, 30]) [15].

There are many different ways to obtain a feasible solution to the original problem (BLP), given a solution to (BLPConv)/(DWConv). In this thesis, using the set $\widetilde{\mathcal{P}} \subseteq \mathcal{P}$ generated during the column generation algorithm, α_p, $p \in \widetilde{\mathcal{P}}$, in (DWConv) are restricted to take binary values and the following problem is solved

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p \mathbf{c}^\top \mathbf{x}^p, && \text{(45a)} \\ \text{s.t.} \ & \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p \mathbf{D}\mathbf{x}^p = \mathbf{d}, && \text{(45b)} \\ & \sum_{p \in \widetilde{\mathcal{P}}} \alpha_p = 1, && \text{(45c)} \\ & \alpha_p \in \{0,1\}, \qquad p \in \widetilde{\mathcal{P}}. && \text{(45d)} \end{aligned}$$

For the case when $\widetilde{\mathcal{P}} = \mathcal{P}$, (45) is equivalent to the original problem (BLP), but when $\widetilde{\mathcal{P}} \subsetneq \mathcal{P}$ the value of the optimal solution to (45) is only an upper bound on that of the optimal solution to the original problem. The set $\operatorname{conv}\{\mathbf{x}^p \mid p \in \widetilde{\mathcal{P}}\}$ contains an optimal solution to (BLPConv), but the set $\{\mathbf{x}^p \mid p \in \widetilde{\mathcal{P}}\}$, which is used in (45), does not necessarily contain an optimal solution to (BLP) [42].

[15] In the standard decomposition of integer programs in the literature, integrality conditions are kept in the formulation, i.e., $\alpha_p \in \{0,1\}$, $p \in \mathcal{P}$, in (DWConv). This formulation is equivalent to (BLP) ([30]). However, the LP relaxation of that decomposition, which is solved by column generation in the literature, and (DWConv) are equivalent.

In order to obtain an optimal solution to (BLP) after solving (BLPConv) ((DWConv)) using column generation, the branch-and-bound algorithm can be employed, or the branch-and-bound tree can be heuristically explored, which is another way to obtain a feasible, possibly non-optimal, solution to (BLP) (see [30]).

If the binary program (BLP) is instead formulated as

$$\begin{aligned} \min_{\mathbf{x}} \ & \mathbf{c}^\top \mathbf{x}, \\ \text{s.t.} \ & \mathbf{D}\mathbf{x} = \mathbf{d}, \\ & \mathbf{x} \in S, \\ & \mathbf{x} \in \{0,1\}^n, \end{aligned}$$

for some bounded set S, the results gained from the Dantzig-Wolfe reformulation still hold. I.e., the problem can be reformulated as (DWConv) with $\mathbf{x}^p$, $p \in \mathcal{P}$, being the extreme points of the set $S \cap \{\{0,1\}^n$ ([30]), which is the type of problem present in the flow formulation (12) of hVRP, where the constraints relating to the variables y_{ij}, $(i,j) \in \mathcal{A}$, can be incorporated in S.

Sometimes, column generation is applied directly on the LP relaxation (44). This is the case with the set partitioning formulation (15) for hVRP, which is already a tight formulation; see Sections 4.1 and 3.1.1. Set partitioning formulations can be related to more compact formulations, such as the flow formulation (12), using Dantzig-Wolfe decompositions. A decomposition for a more complex problem—including time windows, multiple depots, split deliveries, and pickups and deliveries—of which hVRP is a special case, is found in [43]. Another decomposition is found in [44]—for the vehicle routing problem with time windows and a fixed number of vehicles, resulting in one subproblem for each vehicle. This could be adapted to hVRP, but a different flow formulation than (12) would then have to be used. In [45] the authors state that being aware of the connection between the flow formulation and the set partitioning formulation is important to help construct "efficient branching and cutting strategies compatible with the column generation approach in order to obtain integer solutions" [45, p. 169].

C Algorithmic issues

In Appendix C.1, proofs of two claims that have been formulated specifically for Benders' algorithm as it is used in this thesis is presented. Since it is not the original Benders subproblem that is solved, but the equivalent formulation (28a)–(28b), (28d), the standard results on how to obtain an optimal extreme point to the feasible set of Benders subproblem can not be directly applied. Therefore, Claims 2 and 3 have been formulated and proven, showing that such an optimal extreme point can be obtained using the (dual) solution to (28a)–(28b), (28d). In Appendix C.2, results are shown demonstrating that there is no simple way of reducing the number of subproblems that need to be solved in the course of Benders' algorithm and the column generation algorithm. In

Appendix C.3, suggestions for how to reduce the vehicle set are presented, based on similar procedures presented in [24].

C.1 Proofs of claims about sufficient conditions for an optimal extreme point

Proof of Claim 2. Assume that $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ satisfies conditions 1–3. Then, it holds that

$$0 \leq c_r^k - \sum_{i \in \mathcal{N}_0} \bar{\pi}_i \delta_{ir}^k, \qquad r \in \mathcal{R}_k, k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}),$$

since $\bar{\boldsymbol{\pi}}$ satisfies (30b), which implies that $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ satisifies (27b)–(27c). Also, since $\bar{\gamma}^k = 0$, $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, and $\tilde{y}^k = 0$, $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, the equivalences

$$\sum_{i \in \mathcal{N}_0} \bar{\pi}_i + \sum_{k \in \mathcal{K}} M \tilde{y}^k \bar{\gamma}^k = \sum_{i \in \mathcal{N}_0} \bar{\pi}_i = w^*(\widetilde{\mathbf{y}}),$$

hold by the optimality of $\bar{\boldsymbol{\pi}}$ in (30). The result follows. $\square$

Proof of Claim 3. Assume that $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ satisfies conditions 1–3. Then, the inclusion $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}}) \in F_{\text{BendersSPDual}}$ follows analogously to Claim 2, since $\bar{\boldsymbol{\pi}}$ satisfies (30b). Define the vector $\Delta_{(r,k)}$ by

$$(\Delta_{(r,k)})_i := \delta_{ir}^k, \qquad i \in \mathcal{N}_0,$$

and let $\{\mathbf{e}_k\}_{k \in \mathcal{K}}$ denote the standard basis, i.e., $\mathbf{e}_k \in \mathbb{R}^{|\mathcal{K}|}$ and $e_{kk} = 1$, and $e_{kj} = 0$, $j \neq k$. The feasible set $F_{\text{BendersSPDual}}$ of (27) can then be expressed as

$$F_{\text{BendersSPDual}} = \left\{ (\boldsymbol{\pi}, \boldsymbol{\gamma}) \in \mathbb{R}^{|\mathcal{N}_0| + |\mathcal{K}|} \;\middle|\; \begin{array}{l} (\Delta_{(r,k)}, \mathbf{e}_k)^\top (\boldsymbol{\pi}, \boldsymbol{\gamma}) \leq c_r^k, r \in \mathcal{R}_k, k \in \mathcal{K} \\ (\mathbf{0}^{|\mathcal{N}_0|}, \mathbf{e}_k)^\top (\boldsymbol{\pi}, \boldsymbol{\gamma}) \leq 0, \quad k \in \mathcal{K} \end{array} \right\}.$$

Let $m_{\mathcal{R}} := \sum_{k \in \mathcal{K}} |\mathcal{R}_k|$ and

$$\mathbf{c} = (c_r^k)_{r \in \mathcal{R}_k, k \in \mathcal{K}}, \Delta = (\Delta_{(r,k)}^\top)_{r \in \mathcal{R}_k, k \in \mathcal{K}}, \mathbf{E} = (\mathbf{E}_{(r,k)})_{r \in \mathcal{R}_k, k \in \mathcal{K}},$$

where $\mathbf{E}_{(r,k)} := \mathbf{e}_k^\top$ defines one row of $\mathbf{E}$. Thus, $\mathbf{c} \in \mathbb{R}^{m_{\mathcal{R}}}$, $\Delta \in \mathbb{R}^{m_{\mathcal{R}} \times |\mathcal{N}_0|}$, and $\mathbf{E} \in \mathbb{R}^{m_{\mathcal{R}} \times |\mathcal{K}|}$. Defining

$$\mathbf{A} := \begin{pmatrix} \Delta & \mathbf{E} \\ \mathbf{0}^{|\mathcal{K}| \times |\mathcal{N}_0|} & \mathrm{I}^{|\mathcal{K}|} \end{pmatrix},$$

we also have that

$$F_{\text{BendersSPDual}} = \left\{ (\boldsymbol{\pi}, \boldsymbol{\gamma}) \in \mathbb{R}^{|\mathcal{N}_0| + |\mathcal{K}|} \;\middle|\; \mathbf{A}(\boldsymbol{\pi}, \boldsymbol{\gamma}) \leq (\mathbf{c}, \mathbf{0}^{|\mathcal{K}|}) \right\}.$$

Since there is at least one vehicle that can service each customer $i \in \mathcal{N}_0$, the identity matrix $\mathrm{I}^{|\mathcal{N}_0|}$ is a submatrix of Δ. Hence, rank(Δ)= $|\mathcal{N}_0|$ and rank$(\mathbf{A})$= $|\mathcal{N}_0| + |\mathcal{K}|$. Define

$$I_1\left((\boldsymbol{\pi}, \boldsymbol{\gamma})\right) := \left\{ (r,k) \;\middle|\; r \in \mathcal{R}_k, k \in \mathcal{K}, (\Delta_{(r,k)}, \mathbf{e}_k)^\top (\boldsymbol{\pi}, \boldsymbol{\gamma}) = c_r^k \right\},$$

and

$$I_2\left((\boldsymbol{\pi},\boldsymbol{\gamma})\right) := \left\{ k \in \mathcal{K} \;\middle|\; \left(\mathbf{0}^{|\mathcal{N}_0|}, \mathbf{e}_k\right)^\top (\boldsymbol{\pi},\boldsymbol{\gamma}) = 0 \right\}.$$

Then $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})$ is an extreme point to the set $F_{\text{BendersSPDual}}$ if and only if there are $|\mathcal{N}_0| + |\mathcal{K}|$ linearly independent constraint rows

$$(\Delta_{(r,k)}, \mathbf{e}_k)^\top, \qquad (r,k) \in I_1\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right),$$

and

$$(\mathbf{0}^{|\mathcal{N}_0|}, \mathbf{e}_k)^\top, \qquad k \in I_2\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right),$$

by the algebraic characterization of extreme points in [4, Theorem 3.17].

Define $\Gamma_0 := \left\{ k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}) \;\middle|\; \bar{\gamma}_k = 0 \right\}$, and $\Gamma_- := \left\{ k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}) \;\middle|\; \bar{\gamma}_k < 0 \right\}$. For $k \in \Gamma_-$ it holds that the set $\mathcal{R}_k(I_1) := \{ r \in \mathcal{R}_k \mid (r,k) \in I_1\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right) \} \neq \emptyset$, since otherwise $\bar{\gamma}_k$ would not be optimal in the problem (Gamma$(k, \bar{\boldsymbol{\pi}})$). For $k \in \Gamma_-$, let $r(k)$ be any element in $\mathcal{R}_k(I_1)$. Then, the inclusions

$$\Gamma_0 \subseteq I_2\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right),$$

and

$$\{ (r(k), k) \mid k \in \Gamma_- \} \subseteq I_1\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right),$$

hold. Also, the inequalities $\bar{\gamma}_k = 0$, $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, imply that the inclusion

$$\widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}) \subseteq I_2\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right)$$

holds. Thus, each $k \in \mathcal{K}$ defines a row among

$$(\Delta_{(r,k)}, \mathbf{e}_k)^\top, \qquad (r,k) \in \{ (r(k), k) \mid k \in \Gamma_- \}, \tag{46a}$$

$$(\mathbf{0}^{|\mathcal{N}_0|}, \mathbf{e}_k)^\top, \qquad k \in \Gamma_0 \cup \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}). \tag{46b}$$

The system (46) constists of $|\mathcal{K}|$ linearly independent rows, since the vectors $\mathbf{e}_k$, $k \in \mathcal{K}$, form the identity matrix $\mathrm{I}^{|\mathcal{K}|}$, $\Gamma_0 \cup \Gamma_- = \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, and $\Gamma_0 \cap \Gamma_- = \emptyset$.

Let $I_3(\boldsymbol{\pi}) := \left\{ (r,k) \;\middle|\; r \in \mathcal{R}_k, k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}}), \Delta_{(r,k)}^\top \boldsymbol{\pi} = c_r^k \right\}$. Since $\bar{\boldsymbol{\pi}}$ is an extreme point to $\left\{ \boldsymbol{\pi} \in \mathbb{R}^{|\mathcal{N}_0|} \;\middle|\; \boldsymbol{\pi} \text{ satisfies (30b)} \right\}$, by the algebraic characterization of extreme points in [4, Theorem 3.17], there exists $|\mathcal{N}_0|$ linearly independent constraint rows in the problem (30) whose corresponding indices (r,k) are included in the set $I_3(\bar{\boldsymbol{\pi}})$. Let $\Gamma := \{(r,k)\} \subseteq I_3(\bar{\boldsymbol{\pi}})$ denote the set of those indices, so that the $|\mathcal{N}_0|$ linearly independent constraint rows are given by

$$\Delta_{(r,k)}^\top, \qquad (r,k) \in \Gamma.$$

The $|\mathcal{N}_0|$ constraint rows of BendersSPDual$(\widetilde{\mathbf{y}})$,

$$(\Delta_{(r,k)}, \mathbf{e}_k)^\top, \qquad (r,k) \in \Gamma, \tag{47}$$

must then also be linearly independent. The inclusion $\Gamma \subseteq I_1\left((\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{\gamma}})\right)$ follows from the inclusion $\Gamma \subseteq I_3(\bar{\boldsymbol{\pi}})$.

Define

$$V_k := \left\{ \mathbf{v} \in \mathbb{R}^{|\mathcal{N}_0|} \;\middle|\; (\mathbf{v}, \mathbf{e}_k)^\top \text{ is a constraint row in (46)–(47)} \right\}.$$

For $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, then $|V_k| = 1$ holds. Therefore, the corresponding constraint row can not be written as a linear combination of other constraint rows in (46)–(47). For $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, because of the linear independence of (47), the row vector $(\mathbf{0}^{|\mathcal{N}_0|}, \mathbf{e}_k)^\top$ occurs as a constraint row only once, and $V_k \setminus \{\mathbf{0}^{|\mathcal{N}_0|}\}$ is a linearly independent set. Therefore, no constraint row $(\mathbf{v}, \mathbf{e}_k)^\top$, where $\mathbf{v} \in V_k$, can be written as a linear combination of other constraint rows. Hence, the system (46)–(47) forms $|\mathcal{N}_0| + |\mathcal{K}|$ linearly independent constraint rows and the result follows. $\square$

C.2 Reducing the number of subproblems that need to be solved in each Benders iteration

Assume that column generation subproblems (22), $k \in \mathcal{K}$, given by

$$(\text{CGSubproblem}(k)) \quad (\hat{c}^k)^* := \min_{r \in \mathcal{R}_k} \left\{ \hat{c}_r^k \right\} = \min_{r \in \mathcal{R}_k} \left\{ c_r^k - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir}^k \right\},$$

are solved to optimality. The question is whether or not it can be determined that an optimal route r^* in the subproblem CGSubproblem(k_1) is optimal also in CGSubproblem(k_2), where $k_1 \neq k_2$, by only solving CGSubproblem(k_1). This would be very helpful both since CGSubproblem(k), $k \in \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$, are the subproblems of the column generation algorithm in Benders subproblem, and since the problem

$$\min \left\{ 0, \min_{r \in \mathcal{R}_k} \; \hat{c}_r^k \right\}$$

needs to be solved once for each $k \in \mathcal{K} \setminus \widetilde{\mathcal{K}}(\widetilde{\mathbf{y}})$ at the end of each Benders iteration.

For the sets of the instances CT12 and CT12EXT, the cost of route (r, k), where $r \in \mathcal{R}_k$ and $k \in \mathcal{K}$, is defined as $c_r^k := f_k + \sum_{h=1}^{H} c_{i_{h-1} i_h}^k$, if the route r visits the nodes $(i_0, i_1, \ldots, i_{H-1}, i_H)$ in that order. The cost structure of the instances is such that $c_{ij}^k := \text{dist}(i,j) \cdot c^k$, where $\text{dist}(i,j)$ denotes the distance between nodes i and j, and the constant $c^k > 0$ depends on the vehicle type k. I.e., $c_r^k := f_k + \alpha_r \cdot c^k$ where $\alpha_r := \sum_{h=1}^{H} \text{dist}(i_{h-1}, i_h)$. Also for $k_1, k_2 \in \mathcal{K}$ such that $k_1 \neq k_2$, either $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$ or $\mathcal{R}_{k_2} \subset \mathcal{R}_{k_1}$ hold; further the inclusion $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$ implies the inequalities $f_{k_1} < f_{k_2}$ and $c_{k_1} < c_{k_2}$.

We define $\mathcal{R}_k^* := \arg\min_{r \in \mathcal{R}_k} \hat{c}_r^k$. Claim 4 shows that, for $k_1, k_2 \in \mathcal{K}$ such that $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$, a route which is optimal in CGSubproblem(k_2) as well as feasible in CGSubproblem(k_1) does need not be optimal in CGSubproblem(k_1), i.e., $r_2^* \in \mathcal{R}_{k_2}^* \cap \mathcal{R}_{k_1} \not\Rightarrow r_2^* \in \mathcal{R}_{k_1}^*$.

Claim 4. *Given the cost structure of the instances* CT12 *and* CT12EXT*, for* $k_1, k_2 \in \mathcal{K}$ *such that* $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$ *and* $\mathcal{R}^*_{k_2} \cap \mathcal{R}_{k_1} \neq \emptyset$*, pick*

$$r^*_2 \in \mathcal{R}^*_{k_2} \cap \mathcal{R}_{k_1}.$$

If there exists an $r^0 \in \mathcal{R}_{k_1}$ *such that* $\alpha_{r^0} > \alpha_{r^*_2}$ *and such that the inequalities*

$$(\alpha_{r^0} - \alpha_{r^*_2})c^{k_1} < \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^0} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^*_2} \leq (\alpha_{r^0} - \alpha_{r^*_2})c^{k_2}, \tag{48}$$

hold, then

$$r^*_2 \notin \mathcal{R}^*_{k_1}.$$

Proof. If possible pick $r^*_2 \in \mathcal{R}^*_{k_2} \cap \mathcal{R}_{k_1}$ and $r^0 \in \mathcal{R}_{k_1}$ such that $\alpha_{r^0} > \alpha_{r^*_2}$. Since $c^{k_1} < c^{k_2}$ holds, we have that $(\alpha_{r^0} - \alpha_{r^*_2})c^{k_1} < (\alpha_{r^0} - \alpha_{r^*_2})c^{k_2}$ holds, so the system (48) of inequalities is consistent. The former of these inequalities implies that

$$\begin{aligned}
c^{k_1}_{r^0} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^0} &= f_{k_1} + \alpha_{r^0} \cdot c^{k_1} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^0} \\
&< f_{k_1} + \alpha_{r^*_2} \cdot c^{k_1} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^*_2} \\
&= c^{k_1}_{r^*_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^*_2}.
\end{aligned}$$

The latter inequality implies that

$$\begin{aligned}
c^{k_2}_{r^0} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_2}_{ir^0} &= f_{k_2} + \alpha_{r^0} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_2}_{ir^0} \\
&= f_{k_2} + \alpha_{r^0} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^0} \\
&\geq f_{k_1} + \alpha_{r^*_2} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_1}_{ir^*_2} \\
&= f_{k_1} + \alpha_{r^*_2} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_2}_{ir^*_2} \\
&= c^{k_2}_{r^*_2} - \sum_{i \in \mathcal{N}_0} \pi^*_i \delta^{k_2}_{ir^*_2}.
\end{aligned}$$

Thus, the relations $\hat{c}^{k_2}_{r^0} \geq \hat{c}^{k_2}_{r^*_2} = (\hat{c}^{k_2})^*$ hold. Since it holds that $(\hat{c}^{k_1})^* \leq \hat{c}^{k_1}_{r^0} < \hat{c}^{k_1}_{r^*_2}$, it follows that $r^*_2 \notin \mathcal{R}^*_{k_1}$. The result follows. $\square$

It is, however, not apparent after solving CGSubproblem(k_2) whether there exists an r^0 as in Claim 4 or not.

Conversely, if $r^*_1 \in \mathcal{R}^*_{k_1}$, where $k_1, k_2 \in \mathcal{K}$ are such that $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$, even though this implies that $r^*_1 \in \mathcal{R}_{k_2}$, it can not be established that $r^*_1 \in \mathcal{R}^*_{k_2}$, since CGSubproblem($k_2$) has a larger feasible set than CGSubproblem(k_1).

Thus there is no way, with the cost structure of the test instances CT12 and CT12EXT, to determine that an optimal route to one subproblem is also an optimal route to another subproblem by solving only one of the problems CGSubproblem(k_1), CGSubproblem(k_2), $k_1, k_2 \in \mathcal{K}$, $k_1 \neq k_2$.

The relationship that is indicated in Claim 4 can, however, be used to strengthen CGSubproblem(k_1), given an optimal solution to CGSubproblem(k_2) which is also feasible in CGSubproblem(k_1). This, since it follows from Claim 5 that if k_1 and k_2 satisfies the given conditions, then the inequalities (49), below, define valid inequalities in CGSubproblem(k_1).

Claim 5. *Given the cost structure of the instances* CT12 *and* CT12EXT*, for $k_1, k_2 \in \mathcal{K}$ such that $\mathcal{R}_{k_1} \subset \mathcal{R}_{k_2}$ and $\mathcal{R}^*_{k_2} \cap \mathcal{R}_{k_1} \neq \emptyset$, the inclusions*

$$r_1^* \in \mathcal{R}^*_{k_1}, r_2^* \in \mathcal{R}^*_{k_2} \cap \mathcal{R}_{k_1}$$

imply the inequalities

$$\alpha_{r_1^*} \geq \alpha_{r_2^*} \tag{49a}$$

and

$$(\alpha_{r_1^*} - \alpha_{r_2^*})c^{k_1} \leq \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_1^*}^{k_1} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_2^*}^{k_1} \leq (\alpha_{r_1^*} - \alpha_{r_2^*})c^{k_2}. \tag{49b}$$

Proof. Since the inequalities $\hat{c}_{r_1^*}^{k_1} \leq \hat{c}_{r_2^*}^{k_1}$ and $\hat{c}_{r_1^*}^{k_2} \geq \hat{c}_{r_2^*}^{k_2}$ hold, it follows that

$$\alpha_{r_1^*} \cdot c^{k_1} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_1^*}^{k_1} \leq \alpha_{r_2^*} \cdot c^{k_1} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_2^*}^{k_1},$$
$$\alpha_{r_1^*} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_1^*}^{k_1} \geq \alpha_{r_2^*} \cdot c^{k_2} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_2^*}^{k_1}.$$

This implies that the inequalities

$$(\alpha_{r_1^*} - \alpha_{r_2^*})c^{k_1} \leq \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_1^*}^{k_1} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_2^*}^{k_1} \tag{50a}$$

and

$$(\alpha_{r_1^*} - \alpha_{r_2^*})c^{k_2} \geq \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_1^*}^{k_1} - \sum_{i \in \mathcal{N}_0} \pi_i^* \delta_{ir_2^*}^{k_1} \tag{50b}$$

hold. For the system (50) to be consistent, the inequality $\alpha_{r_1^*} \geq \alpha_{r_2^*}$ must hold, since the cost structure assumed implies that $c^{k_1} < c^{k_2}$. The result follows. □

Thus, for k_1 and k_2 satisfying the conditions in Claim 5, we have the following. From (49a) it follows that the length of a route r_1^* which is optimal in CGSubproblem(k_1), must be greater than or equal to the length of a route r_2^*, which is optimal in CGSubproblem(k_2) and feasible in CGSubproblem(k_1). From (49b) it follows that the the sum of the dual variables along route r_1^* must be greater than or equal to the sum of the dual variables along route r_2^*, and the difference must lie in the interval $[c_{r_1^*}^{k_1} - c_{r_2^*}^{k_1}, c_{r_1^*}^{k_2} - c_{r_2^*}^{k_2}]$.

C.3 Reducing the set of vehicles

For all of the test instances in the original set CT12 as well as in the extended set CT12EXT (that have been tested), as the column generation algorithm progresses a pattern emerges, in which, for large vehicles, fewer and fewer subproblems provide routes with negative reduced costs. This since the optimal solution to the column generation problem seldom contains any vehicle from the larger half of the capacity range.

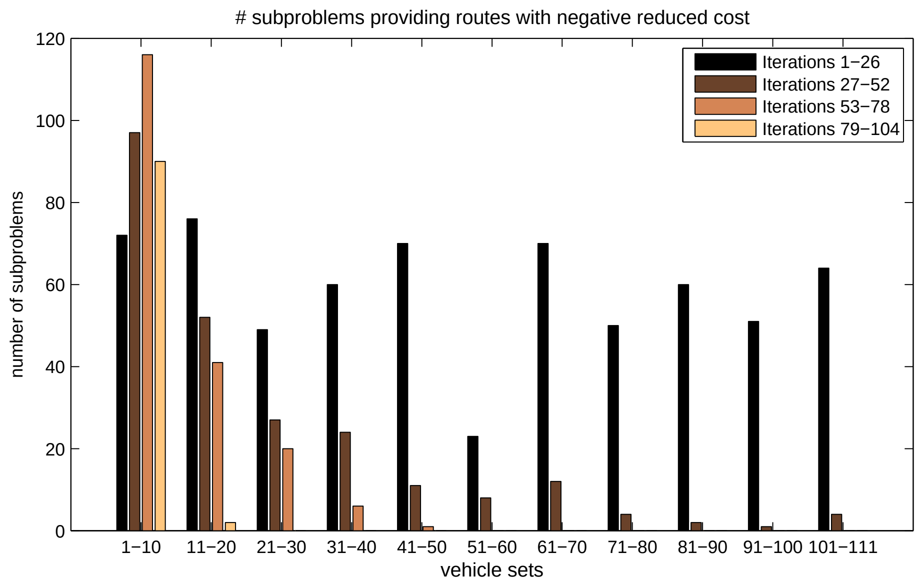

Figure 16: The number of subproblems—for each group of column generation iterations and each set of vehicle types—in which a route with a negative reduced cost is found, for the straightforward model of the instance T15EXT. For each vehicle set of 10 (11) vehicle types and each group of 26 iterations at most 260 (286) subproblems can yield routes with negative reduced costs.

The 111 vehicle types in T15EXT were divided—by increasing capacity—into the eleven vehicle types sets: 1–10, 11–20, . . . , 91–100, 101–111. The column generation iterations were divided into four groups, each of 26 iterations. Figure 16 shows the number of subproblems—for each group of 26 column generation iterations and each set of 10 (11) vehicle types—in which a route with a negative reduced cost was found, for the straightforward model (15) of the instance T15EXT. The reason why a specific subproblem provides no route with a negative reduced cost is either that no such route exists, or that no such route was found within the time limit, or that the subproblem was skipped due to the tabu-strategy of partial column generation (see Section 4.1.1). For each vehicle set and group of iterations the number of subproblems that provided a route with negative reduced cost was at most 116. No subproblems corresponding to vehicles in the sets 51-60, ..., 101–111, provide any route with a negative reduced cost in the latter half of the column generation iterations. This fits well with the fact that the optimal solution to the problem (19) for the instance T15EXT includes vehicles only from the vehicle set 1-10, and the solution to (15) given in the last column generation iteration includes vehicles only from the sets 1-10 and 11-20.

In addition to not providing any route with a negative reduced cost, subproblems for vehicles with larger capacities often require much longer solution times. Therefore, a lot may be gained by reducing the number of vehicle types. Choi and Tcha [24] reduce the number of vehicle types by using an upper bound on the optimal solution of the original problem, and a lower bound on the optimal solution value of the original problem with the extra requirement that at least one vehicle of a particular type is used. Requiring that at least one vehicle of type $\hat{k} \in \mathcal{K}$ is used; a lower bound on the optimal value of the restricted model model (25) (see Section 4.2) is given by

$$\begin{aligned} \underline{z}(\hat{k}) := \min_{\mathbf{s}} & \sum_{k \in \mathcal{K}} f_k s_k, && \text{(51a)} \\ \text{s.t.} & \sum_{k \in \mathcal{K}} D_k s_k \geq \sum_{i \in \mathcal{N}_0} d_i, && \text{(51b)} \\ & s_{\hat{k}} \geq 1, && \text{(51c)} \\ & s_k \in \mathbb{N}, \qquad k \in \mathcal{K}, && \text{(51d)} \end{aligned}$$

where the variable s_k denotes the number of vehicles of type k that is used. The objective (51a) is to minimize the sum of the fixed costs f_k over all vehicles used. The constraint (51b) makes sure that the total capacity of the used vehicles is not less than the total demand of the customers, and the constraint (51c) ensures that at least one vehicle of type $\hat{k}$ is used.

Provided an upper bound $\bar{z}$ on the optimal objective value of (25), vehicles $\hat{k}$ for which $\underline{z}(\hat{k}) \geq \bar{z}$ can not be part of an optimal solution to (25) and can thus be eliminated from the set of vehicles. When Benders' algorithm is used to find a solution to (26), an upper bound $\bar{z}$ to (25) is provided in each iteration. So hopefully, even though the lower bound is probably quite weak since it does not include variable costs, this procedure might eliminate some of the vehicle types with large capacities: this may result in great reductions of the solution time.

The same procedure can be applied to the straightforward model (15) with column generation from Section 4.1.

References

[1] Dantzig GB, Ramser JH. The truck dispatching problem. Management Science. 1959;6(1):80–91.

[2] Toth P, Vigo D. Preface. In: Toth P, Vigo D, editors. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM Publishing; 2002. pp. xvii–xviii.

[3] Vidal T, Crainic TG, Gendreau M, Prins C. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. European Journal of Operational Research. 2013;231(1):1–21.

[4] Andréasson N, Evgrafov A, Patriksson M. An Introduction to Continuous Optimization. Lund: Studentlitteratur; 2005.

[5] Vanderbeck F. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research. 2000;48(1):111–128.

[6] Cordeau J, Stojković G, Soumis F, Desrosiers J. Benders decomposition for simultaneous aircraft routing and crew scheduling. Transportation Science. 2001;35(4):375–388.

[7] Hoff A, Andersson H, Christiansen M, Hasle G, Løkketangen A. Industrial aspects and literature survey: fleet composition and routing. Computers & Operations Research. 2010;37(12):2041–2061.

[8] Lübbecke ME, Desrosiers J. Selected topics in column generation. Operations Research. 2005;53(6):1007–1023.

[9] Lasdon LS. Optimization Theory for Large Systems. London, UK: Macmillan; 1970.

[10] Drexl M. Rich vehicle routing in theory and practice. Logistics Research. 2012;5(1–2):47–63.

[11] Irnich S, Desaulniers G. Shortest path problems with resource constraints. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column Generation. New York: Springer; 2005. pp. 33–65.

[12] Righini G, Salani M. New dynamic programming algorithms for the resource constrained shortest path problem. Networks. 2008;51(3):155–170.

[13] Feillet D, Dejax P, Gendreau M, Gueguen C. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks. 2004;44(3):216–229.

[14] Feillet D. A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR. 2010;8(4):407–424.

[15] Bettinelli A, Ceselli A, Righini G. A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. Transportation Research Part C. 2011;19(5):723–740.

[16] Dunbar M, Froyland G, Wu C-L. Robust airline schedule planning: Minimizing propagated delay in an integrated routing and crewing framework. Transportation Science. 2012;46(2):204–216.

[17] Baldacci R, Battarra M, Vigo D. Routing a heterogeneous fleet of vehicles. In: Golden BL, Raghavan S, Wasil EA, editors. The Vehicle Routing Problem: Latest Advances and New Challenges. vol. 43 of Operations Research/Computer Science Interfaces. New York: Springer; 2008. pp. 3–27.

[18] Toth P, Vigo D. An overview of vehicle routing problems. In: Toth P, Vigo D, editors. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM Publishing; 2002. pp. 1–26.

[19] Golden BL, Assad A, Levy L, Gheysens F. The fleet size and mix vehicle routing problem. Computers & Operations Research. 1984;11(1):49–66.

[20] Taillard ED. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO: Operations Research. 1999;33(1):1–14.

[21] Balinski LM, Quandt RE. On an integer program for a delivery problem. Operations Research. 1964;12(2):300–304.

[22] Pessoa A, de Aragão MP, Uchoa E. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. In: Demetrescu Camil, editor. Experimental Algorithms. vol. 4525 of Lecture Notes in Computer Science. Germany: Springer; 2007. pp. 150–160.

[23] Baldacci R, Mingozzi A. A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming. 2009;120(2):347–380.

[24] Choi E, Tcha D-W. A column generation approach to the heterogeneous fleet vehicle routing problem. Operations Research. 2007;34(7):2080–2095.

[25] Christofides N, Eilon S. An algorithm for the vehicle-dispatching problem. OR. 1969;20(3):309–318.

[26] Clarke GU, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research. 1964;12(4):568–581.

[27] Li F, Golden B, Wasil E. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. Computers & Operations Research. 2007;34(9):2734–2742.

[28] Díaz BD. VRP web: VRP Instances; 2006 [cited March 2014]. Available from: `http://www.bernabe.dorronsoro.es/vrp/Problem_Instances/instances.html`.

[29] Xiao Y, Zhao Q, Kaku I, Xu Y. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. Computers & Operations Research. 2012;39(7):1419–1431.

[30] Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH. Branch-and-price: Column generation for solving huge integer programs. Operations Research. 1998;46(3):316–329.

[31] Glover F, Laguna M. Tabu search. In: Pardalos PM, Du D, Graham RL, editors. Handbook of Cobinatorial Optimization. Springer. New York: SIAM Publishing; 2013. pp. 3261–3362.

[32] Larsen J. Parallelization of the vehicle routing problem with time windows [PhD thesis]. Department of Mathematical Modelling, Technical University of Denmark; 1999. IMM-PHD-1999-62.

[33] McDaniel D, Devine M. A modified Benders' partitioning algorithm for mixed integer programming. Management Science. 1977;24(3):312–319.

[34] Floudas CA. Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Cary, NC, USA: Oxford University Press; 1995.

[35] Boschetti M, Maniezzo V. Benders decomposition, Lagrangean relaxation and metaheuristic design. Journal of Heuristics. 2009;15(3):283–312.

[36] AMPL, a modeling language for mathematical programming: AMPL Downloads; 2013 [cited March 2014]. Available from: `http://www.ampl.com/DOWNLOADS/index.html`.

[37] IBM ILOG CPLEX optimization studio; [cited March 2014]. Available from: `http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/`.

[38] The MathWorks Inc. Matlab: the language of technical computing; 2014. Available from: `http://www.mathworks.se/products/matlab/`.

[39] Fourer B. Google groups, AMPL modeling language, Error running cplexamp: termination code 9; 2011 [cited March 2014]. Available from: `https://groups.google.com/d/msg/ampl/6W4qIOssLio/_iQX6IkxcWcJ`.

[40] ILOG AMPL CPLEX System Version 10.0 User's Guide; 2006. Available from: `http://www.ampl.com/BOOKLETS/amplcplex100userguide.pdf`.

[41] Ceselli A, Righini G, Salani M. A column generation algorithm for a rich vehicle routing problem. Transportation Science. 2009;43(1):56–69.

[42] Vanderbeck F. Implementing mixed integer column generation. In: Desaulniers G, Desrosiers J, Solomon MM, editors. Column Generation. New York: Springer; 2005. pp. 331–358.

[43] Desaulniers G, Desrosiers J, Ioachim I, Solomon MM, Soumis F, Villeneuve D. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In: Crainic TG, Laporte G, editors. Fleet Management and Logistics. New York: Springer; 1998. pp. 57–93.

[44] Chabrier A. Vehicle routing problem with elementary shortest path based column generation. Computers & Operations Research. 2006;33(10):2972–2990.

[45] Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. VRP with time windows. In: Toth P, Vigo D, editors. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM Publishing; 2002. pp. 157–193.