Contributions to Exact Algorithms for Packing and Routing Problems

Dissertation
zur Erlangung des Grades einer Doktorin der
wirtschaftlichen Staatswissenschaften
(Dr. rer. pol.)
des Fachbereichs Rechts- und Wirtschaftswissenschaften
der Johannes Gutenberg-Universität Mainz

vorgelegt von
M. Sc. Katrin Heßler
in Mainz

im Jahre 2020

Tag der mündlichen Prüfung: 1.10.2020

Contents

List of Papers

- Katrin Heßler[1], Timo Gschwind[1], Stefan Irnich[1] (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research* **271**(2), 401–419.
- Katrin Heßler[1], Stefan Irnich[1], Tobias Kreiter[2], Ulrich Pferschy[3] (2020). Lexicographic Bin-Packing Optimization for Loading Trucks in a Direct-Shipping System. Technical Report LM-2020-05, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany. Submitted to *OR Spectrum*.
- Katrin Heßler[1], Stefan Irnich[1] (2020). A Branch-and-Cut Algorithm for the Soft-Clustered Vehicle-Routing Problem. *Discrete Applied Mathematics* **288**, 218–234.
- Katrin Heßler[1] (2020). Exact Algorithms for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes. Technical Report LM-2020-04, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany. Submitted to *European Journal of Operational Research*.

[1]Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany
[2]scc EDV-Beratung AG, Wambachergasse 10, A-1130 Wien, Austria
[3]Department of Statistics and Operations Research, University of Graz, Universitätsstraße 15/E3, A-8010 Graz, Austria

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Combinatorial optimization problems aim to find an optimal solution within a finite set of possible solutions (Schrijver, 2003, p.1). Important real-world applications of combinatorial optimization problems in logistics are packing and routing problems – besides cutting, scheduling, network design, location, and assignment problems, among others. When modeling these problems as optimization problems, the objective is to minimize or maximize a function that represents costs, delay, profit, etc., subject to side constraints that define the feasible set of candidate solutions. Typically, this set of possible solutions is too large for exhaustive search and often even grows exponentially in the problem size (Schrijver, 2003, p.1). Moreover, the set is not listed explicitly but implicitly defined by restrictions (Korte and Vygen, 2006, p.5). From a theoretical point of view, many combinatorial optimization problems are $\mathcal{NP}$-hard (Schrijver, 2003, p.1) which means that no algorithm exists that runs in polynomial time unless $\mathcal{P} = \mathcal{NP}$. Therefore, efficient mathematical methods are necessary for solving such problems exactly.

In the past decades not only computer performance has been improved but also several exact algorithms have been developed that are capable of solving even large instances of some problems in acceptable time. Prominent examples are the traveling salesman problem (Applegate *et al.*, 2011), knapsack problems (Kellerer *et al.*, 2004), as well as the bin packing problem and the cutting stock problem (Delorme and Iori, 2020). Moreover, developing exact algorithms is reasonable because heuristics often apply concepts and ideas derived from exact methods.

In this thesis, we focus on exact algorithms for different packing and routing problems. Section 1.1 provides an overview of exact solution methods that are used in this thesis. The packing and routing problems studied in this thesis are introduced in Section 1.2. Finally, Section 1.3 summarizes contributions and depicts the outline.

1.1 Exact Solution Methods

Combinatorial optimization problems can often be formulated as integer linear programs (IPs). A prominent method to solve such IPs is the branch-and-bound (BB) algorithm that systematically enumerates candidate solutions to ensure feasible integer solutions (Dakin, 1965). Instead of a brute-force enumeration, the essential concept of the algorithm is to use bounding and pruning strategies to reduce the search space.

However, for combinatorial optimization problems, there exist not only compact formulations having a polynomial number of variables and constraints but also extensive formulations having an exponential number of variables or constraints. For such extensive formulations, the set of variables or constraints is often too large for explicitly representing the model. Nevertheless, this large-scale formulations can be solved by considering only a subset of columns (=variables) or rows (=constraints) at the beginning and then adding required columns or rows dynamically during the solution process. Note that adding a row (column) to the primal is equivalent to adding a column (row) to the respective dual. Branch-and-cut (BC) algorithms employ this technique of dynamically adding constraints to solve extensive formulations with a huge number of constraints. Also extensive formulations with a huge number of variables can be solved by means of a sophisticated branch-cut-and-price (BCP) algorithm. The different components of these algorithms are explained and discussed in more detail in the following.

Branch-and-cut. The idea of BC algorithms is to start with a relaxation of the problem, usually by removing the integrality constraints and constraint families of exponential size. Combining two techniques then solves the problem: cutting planes and BB. Cutting planes are valid inequalities, so-called cuts, that either cut off fractional points to strengthen the linear relaxation or were omitted in the initial relaxed formulation but are necessary to ensure feasibility. For the former type, generic cuts exist that are applicable to all IPs, for example Chvátal-Gomory cuts (Chvátal, 1973; Gomory, 1963). Moreover, there are specialized cuts for certain problems, for example capacity cuts (Lysgaard *et al.*, 2004). For the latter type, constraints initially omitted are added dynamically during the solution procedure to ensure feasibility. Note that typically not all such constraints are added but it must be guaranteed that the solution is feasible. Violated cuts are detected by a so-called separation procedure that may be costly, depending on the cut. When embedded in a BB algorithm, the main goal is to decide how and when to separate which type of cut.

The performance of a BC algorithm depends on the one hand on the specific model formulation that should avoid symmetry between solutions to keep the size of the BB tree limited. On the other hand, it is crucial to find a fast separation

procedure that detects strong cuts. However, the advantage of a BC approach is that it is much simpler to implement compared to sophisticated BCP approaches.

Branch-cut-and-price. Extensive formulations with a huge number of variables can often be solved efficiently by a BCP approach. Note that such large-scale models result in many applications from a Dantzig-Wolfe decomposition of a compact formulation (Dantzig and Wolfe, 1960). BCP algorithms combine three techniques: cutting planes, BB, and column generation. The basic idea of column generation (CG) is to start with a restricted master problem (RMP) only consisting of a small subset of variables (=columns) and then add dynamically negative-reduced cost variables by solving a so-called pricing problem (=subproblem). The algorithm alternates between re-optimizing the RMP and solving the pricing problem. If there are no more variables with negative reduced cost, the linear relaxation of the master problem is solved. Note that not all columns are generated because most columns are nonbasic and their corresponding variable values are equal to zero in the optimal solution. If the master problem contains integrality constraints, the problem can be solved by a so-called branch-and-price (BP) algorithm that is a hybrid of BB and column generation (Barnhart *et al.*, 1998). Additionally adding cutting planes to strengthen the linear relaxation yields a BCP algorithm.

The main advantages of CG solution approaches are as follows: (i) the linear relaxation of the reformulated model is typically stronger than the often weak linear relaxation of compact formulations; (ii) symmetry issues that cause a poor performing BB algorithm occur less often compared to compact formulations; and (iii) non-linearities or complicated constraints may be hidden in the subproblem. Moreover, highly effective solution procedures for the subproblem often exist (Lübbecke and Desrosiers, 2005), e.g., in routing problems, the subproblem can usually be formulated as *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005) that can be solved effectively by dynamic-programming based labeling algorithms (Dijkstra, 1959; Ahuja *et al.*, 1993).

The main drawbacks of CG algorithms resulting from instability issues are: (i) a slow convergence (tailing-off effect); (ii) the generation of irrelevant columns in the first iterations because of poor dual information (heading-in effect); (iii) a constant RMP value for several iterations (plateau effect); (iv) instability in the dual solutions that are jumping from one extreme value to another (bang-bang effect); and (v) non-monotone convergence of Lagrangian dual bounds (yo-yo effect) (Vanderbeck, 2005). However, these drawbacks can be reduced by stabilization techniques (Lübbecke and Desrosiers, 2005). In this thesis, we exploit so-called dual optimal inequalities (DOIs) to stabilize the CG process (Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016).

1.2 Considered Packing and Routing Problems

The thesis at hand tackles four different routing and packing problems, namely the classical vector packing problem, a real-world two-dimensional vector packing problem with additional constraints, the soft-clustered vehicle-routing problem, and a multi-compartment vehicle-routing problem. The problems are introduced in the following.

In the classical bin packing and cutting stock problem, a given set of items needs to be packed into a minimum number of equally sized bins (Delorme *et al.*, 2016). The *vector packing problem* (VPP) is a multidimensional variant of these problems where the packing is constrained independently in all dimensions. A practical application is, for example, the transportation of goods in trucks or containers each of a fixed given volume, weight, and value capacity. Another example is the static resource allocation problem in which the number of required servers is minimized given a set of servers with known capacities and a set of services with known demands (Panigrahy *et al.*, 2011).

We also consider a real-world VPP with additional constraints that arises in a direct-shipping system in the food and beverage industry. In this problem, the packing is constrained by the two dimensions weight and volume. Products of the different categories standard, cooled, and frozen must be transported in separated zones in a truck. Moreover, product splitting should be avoided to simplify the truck (un-)loading. To take into account the cost of a truck that mainly depends on the loaded product categories, a cost-minimal packing is found by optimizing five objectives in a strictly lexicographic sense.

The *vehicle-routing problem* (VRP, Toth and Vigo, 2014) is one of the most important routing problems. Since Dantzig and Ramser (1959) introduced the problem 60 years ago, numerous variants have been studied. One of the most studied versions is the *capacitated vehicle-routing problem* (CVRP), in which homogeneous vehicles deliver goods from a single depot to customers and the objective is to minimize the total distance of all vehicle routes such that all customer demands are met and vehicle capacities are respected (Irnich *et al.*, 2014). In recent years, especially so-called rich VRPs have been studied that take into account various real-world problem extensions. This thesis considers two such rich VRPs.

In the *soft-clustered vehicle-routing problem* (SoftCluVRP, Defryn and Sörensen, 2017), customers are partitioned into clusters and all customers of the same cluster must be served by the same vehicle. In contrast to the hard-clustered variant (Sevaux and Sörensen, 2008), where a cluster must be served completely before the next cluster is served, visits to customers of the same cluster can be interrupted by visits to customers of another cluster. A practical example is parcel delivery in courier companies, where customers are divided into regional zones. The parcels are first sorted into containers according to a given districting and then delivered

to the recipients.

In a *multi-compartment vehicle-routing problem* (MCVRP, Henke, 2017), several product types are considered that must not be mixed during transportation. For this purpose, the considered vehicles can divide their capacity into a certain number of separated zones or compartments. In the literature, various different multi-compartment vehicle configurations are presumed, e.g., the compartment size can be fixed or flexible and the assignment of product types to compartments can be preset or arbitrary (Pollaris *et al.*, 2014). We consider two different MCVRP variants with flexible compartment sizes. In the *multi-compartment vehicle-routing problem with continuously flexible compartment sizes* (MCVRP-CFCS, Koch *et al.*, 2016), compartment sizes can be set arbitrarily within the limits of the vehicle capacity. This problem arises, in particular, when food of different levels of refrigeration has to be transported (Derigs *et al.*, 2010; Hübner and Ostermeier, 2019). In the *multi-compartment vehicle-routing problem with discretely flexible compartment sizes* (MCVRP-DFCS, Henke *et al.*, 2015), compartment sizes can only be set according to predefined, equally spaced positions. Practical applications are the collection of glass waste (Henke *et al.*, 2015) and the shipment of bulk products (Fagerholt and Christiansen, 2000).

1.3 Contribution and Outline

In this thesis, all four papers mainly contribute to exact branch-(cut-)and-price and branch-and-cut algorithms for packing and routing problems. In the following, the structure of this thesis and the contributions of all papers that are either published in or submitted to scientific journals are described.

Chapter 2 introduces a branch-and-price approach that tackles binary and non-binary formulations of the VPP. In these formulations, the pricing problems are variants of the multidimensional knapsack problem. We present a unified solution approach for the pricing problems based on dynamic programming and an SPPRC that can cope with Ryan and Foster (1981) and Vanderbeck (1999) branching schemes. Additionally, dual inequalities are added to stabilize the CG process. Computational results indicate that our unified approach is very competitive with the most recent exact VPP approaches presented in the literature.

Chapter 3 deals with the development of an exact BP and heuristic solution approach for a real-world VPP with five lexicographic objectives. The subproblem of the BP approach can be solved by one or several modified 2-dimensional knapsack problems. A non-trivial algorithmic component is the adaption of the branching scheme of Chapter 2. Moreover, stabilization with DOIs and other acceleration techniques speed up the CG process. For the heuristic approach, we first present constructive heuristics for solving a single level of the packing problem. Then,

these heuristics are integrated into a meta-scheme that is based on decomposing and regrouping. Computational results on real-world and difficult self-generated instances demonstrate the applicability of our approach.

In Chapter 4, a BC algorithm for the SoftCluVRP is presented that is based on a symmetric formulation with an asymmetric sub-model for the clustering part. The algorithm exploits valid inequalities for the CVRP that can be adapted, problem-specific cutting planes, and new heuristic and exact separation procedures. For square grid instances in the Euclidean plane, lower-bounding techniques and a reduction scheme are presented. The latter is also applicable to the respective traveling salesman problem. Computational results show that the algorithm is effective for instances that are truly clustered and several previously open instances can be solved to proven optimality.

Chapter 5 proposes exact algorithms for the MCVRP-DFCS and MCVRP-CFCS. Two BC algorithms and a route-based BPC algorithm are presented that can tackle both packing restrictions with mild adaptions. Extensive computational experiments show the effectiveness of the algorithms for instances with up to 50 customers. Moreover, cost savings of using continuously flexible compartment sizes instead of discretely flexible compartment sizes are analyzed.

Finally, Chapter 6 summarizes and draws final conclusions.

Chapter 2

Stabilized Branch-and-Price Algorithms for Vector Packing Problems

Katrin Heßler, Timo Gschwind, Stefan Irnich

Abstract

This paper considers packing and cutting problems in which a packing/cutting pattern is constrained independently in two or more dimensions. Examples are restrictions with respect to weight, length, and value. We present branch-and-price algorithms to solve these vector packing problems (VPPs) exactly. The underlying column-generation procedure uses an extended master program that is stabilized by (deep) dual-optimal inequalities. While some inequalities are added to the master program right from the beginning (static version), other violated dual-optimal inequalities are added dynamically. The column-generation subproblem is a multidimensional knapsack problem, either binary, bounded, or unbounded depending on the specific master problem formulation. Its fast resolution is decisive for the overall performance of the branch-and-price algorithm. In order to provide a generic but still efficient solution approach for the subproblem, we formulate it as a shortest path problem with resource constraints (SPPRC), yielding the following advantages: (i) Violated dual-optimal inequalities can be identified as a by-product of the SPPRC labeling approach and thus be added dynamically; (ii) branching decisions can be implemented into the subproblem without deteriorating its resolution process; and (iii) larger instances of higher-dimensional VPPs can be tackled with branch-and-price for the first time. Extensive computational results show that our branch-and-price algorithms are capable of solving VPP benchmark instances effectively.

2.1 Introduction

In this paper, we analyze different covering formulations and column-generation-based solution approaches for packing and cutting problems with two or more dimensions. While the 1-dimensional case includes the classical *bin packing problem* (BPP) and *cutting stock problem* (CSP) that are both well studied and recently surveyed by Delorme *et al.* (2016), we focus on the multidimensional case where packings/cutting patterns are constrained independently in all dimensions. Examples are restrictions with respect to weight, length, and value. In the following, we refer to these problems as *vector packing problems* (VPPs). The literature uses different names for VPPs such as p-dimensional vector (bin) packing (Buljubašić and Vasquez, 2016; Spieksma, 1994; Brandão and Pedroso, 2016), vector bin packing (Panigrahy *et al.*, 2011), or (for two dimensions) two-constraint bin packing (Monaci and Toth, 2006). In contrast to VPPs, there is another family of multi-dimensional packing problems where dimensions are not independent, e.g., when packing rectangles (Huang and Korf, 2012) or 3-dimensional items (Martello *et al.*, 2000). These problems are not addressed here.

VPPs have various practical applications. For example, consider a logistics company that has to transport items with different lengths and weights. The smallest amount of vehicles possible should be used for transportation. How can the items be packed into the vehicles taking into account the length of the vehicles and the maximum loading weight? Another example is the static resource allocation problem. Given a set of servers with known capacities and a set of services with known demands the aim is to minimize the number of required servers (Panigrahy *et al.*, 2011).

In the literature, several heuristic and exact methods have been introduced to solve VPPs. The first exact approach was proposed by Spieksma (1994). The author incorporated lower bounds into a branch-and-bound algorithm and, additionally, introduced a heuristic based on the first-fit decreasing heuristic. The approach was improved by Caprara and Toth (2001) who exploited on the one hand lower bounds combined with heuristics and on the other hand a branch-and-bound algorithm to find exact solutions for VPPs. Alves *et al.* (2014) further enhanced the approach by calculating lower bounds based on dual-feasible functions. Recently, Brandão and Pedroso (2016) adapted the arc-flow formulation with side constraints (Valério de Carvalho, 1999) to packing problems including VPPs. The solution was accelerated by means of graph compression to reduce symmetry and to combine sub-graphs. In contrast, Hu *et al.* (2017) used the covering formulation of Gilmore and Gomory (1961) and introduced a branching strategy based on dominance relations between cutting patterns. Besides exact approaches, heuristics have been applied to solve larger instances. Buljubašić and Vasquez (2016) and Vasquez and Buljubašić (2018) used consistent neighborhood search to find

heuristic solutions. Variants of the first-fit decreasing algorithm were presented by Panigrahy *et al.* (2011), yielding good heuristic VPP solutions.

We formally define the VPP as follows: Let $D \geq 1$ be an integer specifying the dimension. A given set $\mathcal{I}$ of items needs to be packed into a minimum number of equally sized bins. Bins are characterized by capacities given by a D-dimensional vector $\mathbf{W} = (W^1, \ldots, W^D)$. Items $i \in \mathcal{I}$ are characterized by weights/sizes which are given by D-dimensional vectors $\mathbf{w}_i = (w_i^1, \ldots, w_i^D) \leq \mathbf{W}$. Dyckhoff (1990) classified cutting and packing problems according to the categories dimensionality, kind of assignment, assortment of large objects (bins), and assortment of (small) items. Hence, the BPP is classified as 1/V/I/R (1: 1-dimensional/V: a selection of objects (bins) and all items/I: identical figure/R: many items of relatively few different (non-congruent) figures). In contrast, the CSP is classified as 1/V/I/M (M: many items of many different figures). Although classified differently in (Dyckhoff, 1990), the BPP and the CSP are essentially the same problem as pointed out in (Ben Amor and Valério de Carvalho, 2005, p. 132). The relationship between BPP and CSP is the following: In the BPP, all items are modeled as *individual objects*. As a consequence, each and every item $i \in \mathcal{I}$ has a demand of $q_i = 1$ and a model must include a corresponding covering/packing constraint. In contrast, in the CSP all items of the same weight are *aggregated*. We formally introduce the set of aggregated items as follows: Let $\mathcal{I}[\mathbf{w}]$ be the equivalence classes of items with identical weight $\mathbf{w}$. Then, the set I of *aggregated items* is the coset $\mathcal{I}/\mathcal{I}[\mathbf{w}]$ that contains one representative item for each of the different weights. As a consequence, any two different aggregated items $i_1, i_2 \in I$ have different weights $\mathbf{w}_{i_1} \neq \mathbf{w}_{i_2}$. The demand of an aggregated item $i \in I$ is $q_i = |\mathcal{I}[\mathbf{w}_i]| = |\{j \in \mathcal{I} : \mathbf{w}_j = \mathbf{w}_i\}| \geq 1$, and at least some aggregated items have a demand greater than 1.

The models that we compare are all variations of the famous covering formulation of Gilmore and Gomory (1961) for CSP that is based on cutting patterns. For the VPP, a *pattern* (or *packing*) describes how a subset of the items is packed into a bin. Assuming $I = \{1, 2, \ldots, m\}$, i.e., m is the number of aggregated items, the set of feasible patterns for the VPP, with different levels of aggregation, can be defined as

$$P = \left\{ (a_1, \ldots, a_m)^\top \in \mathbb{Z}_+^m : \sum_{i=1}^{m} a_i w_i^d \leq W^d \text{ for all } 1 \leq d \leq D \right\}. \tag{2.1}$$

In order to uniquely refer to a pattern $p \in P$ we write its coefficients as $\mathbf{a}^p = (a_1^p, \ldots, a_m^p)^\top$. Using integer decision variables x_p for the number of times pattern

$p \in P$ is used, the VPP can be formulated as

$$z^{\text{VPP}} = \min \sum_{p \in P} x_p \tag{2.2a}$$

$$\text{(VPP)} \qquad \text{s.t.} \sum_{p \in P} a_i^p x_p = q_i, \qquad i \in I \tag{2.2b}$$

$$x_p \geq 0 \text{ integer}, \qquad p \in P. \tag{2.2c}$$

The objective (2.2a) is the minimization of the patterns/bins that are used. Constraints (2.2b) ensure that all items are packed as often as their demand requires. As already mentioned by Gilmore and Gomory (1961), (2.2b) can be replaced by covering constraints, i.e., $\sum_{p \in P} a_i^p x_p \geq q_i$ for all $i \in I$. The domain of the decision variables is given by (2.2c).

The quality of the linear relaxation bound is crucial to solve mixed-integer problems. As in the BPP (i.e., $D = 1$ and $q_i = 1$ for all $i \in I$), patterns can be restricted to only have binary coefficients $\mathbf{a}^p \in \{0,1\}^m$ when modeling individual items. While such a constraint does not impact the validity of integer VPP solutions to (2.2) (also for $D \geq 2$), the linear relaxation is generally tighter than without the binary requirement. It means that a proper subset of patterns is used:

$$P_{01} = \left\{ (a_1, \ldots, a_{m_{01}})^\top \in \mathbb{Z}_+^{m_{01}} : \sum_{i=1}^{m_{01}} a_i w_i^d \leq W^d \right. \\ \left. \text{for all } 1 \leq d \leq D \text{ and } a_i \leq 1 \text{ for all } i \in I \right\} \tag{2.3}$$

We denote formulation (2.2) using only binary patterns P_{01} as *binary* VPP (01-VPP). In this case, optimal solutions to the 01-VPP have binary x_p-variables.

Also for VPP with non-unit demand, the patterns' coefficients can be further constrained. Whenever the demand q_i of some item $i \in I$ is smaller than $\lfloor W^d / w_i^d \rfloor$ for all $d \in D$, the pattern set

$$P_B = \left\{ (a_1, \ldots, a_{m_B})^\top \in \mathbb{Z}_+^{m_B} : \sum_{i=1}^{m_B} a_i w_i^d \leq W^d \right. \\ \left. \text{for all } 1 \leq d \leq D \text{ and } a_i \leq q_i \text{ for all } i \in I \right\} \tag{2.4}$$

is a proper subset of P. We denote formulation (2.2) using only bounded patterns P_B as *bounded* VPP (B-VPP). For completeness, formulation (2.2) with no additional bounds on pattern coefficients is referred to as *unbounded* VPP (U-VPP) and the pattern set is explicitly denoted by $P_U = P$ in the following.

Table 2.1: Comparison of covering formulations of the VPP

	01-VPP		B-VPP		U-VPP
Aggregation	none		partial or full		full
(Aggregated) Items I	$I = \mathcal{I}$		I		$I = \mathcal{I}/\mathcal{I}[\mathbf{w}]$
Weights $\mathbf{w}_i$	can be identical		can be identical		all different
Demand q_i	$q_i = 1$		$q_i \geq 1$		$q_i \geq 1$
Bound on Pattern Coefficients u_i	$a_i^p \leq 1 = u_i^{01}$		$a_i^p \leq u_i^B \leq q_i$		$a_i^p \leq u_i^U = \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$
Pattern Set	P_{01}		P_B		$P_U = P$
Domain of Variables	$x_p \in \{0,1\}$		$x_p \in \mathbb{Z}_+$		$x_p \in \mathbb{Z}_+$
Number of Constraints m	$m_{01} = \lvert\mathcal{I}\rvert$	$>$	m_B	$\geq$	$m_U = \lvert\mathcal{I}/\mathcal{I}[\mathbf{w}]\rvert$
LP Bound z_{LP}	$z_{LP}^{\text{01-VPP}}$	$=$	$z_{LP}^{\text{B-VPP}}$	$\geq$	$z_{LP}^{\text{U-VPP}}$
Subproblem	01-MKP		B-MKP		U-MKP
Branching	(Ryan and Foster, 1981)		(Vanderbeck, 1999)		(Vanderbeck, 1999)
Dual Inequalities					
PIs $\pi_h \geq \pi_i$	DDOIs		DDOIs		DOIs
PIs with $\mathbf{w}_h + \mathbf{w}_i \not\leq \mathbf{W}$	DOIs		DOIs		DOIs
SIs $\pi_h \geq \sum_{i \in S} \pi_i$	can be invalid		can be invalid		DOIs
SIs with $\mathbf{w}_h + \mathbf{w}_i \not\leq \mathbf{W}$ for all $i \in S$	DOIs		DOIs		DOIs
WSIs $\pi_h \geq \sum_i t_i \pi_i$	invalid		can be invalid		DOIs

Note: PIs=Pair Inequalities, SIs=Subset Inequalities, and WSIs=Weighted Subset Inequalities

Table 2.1 summarizes differences between the pure binary formulation 01-VPP, the bounded formulation B-VPP, and the unbounded formulation U-VPP. Note that for a given set of individual items $\mathcal{I}$, the formulations 01-VPP and U-VPP are unique. They represent the two extremes of aggregation (completely disaggregated vs. fully aggregated), while B-VPP is a family of formulations resulting from different types of aggregation and exploitation/disregarding of the coefficient bounds $a_i^p \leq q_i$ in (2.4).

Due to the huge number of variables, formulation (2.2) is typically solved with the help of a column-generation algorithm (Desaulniers *et al.*, 2005). One starts with a (small) subset of patterns $P' \subset P$ and the linear relaxation of (2.2) using only variables x_p with $p \in P'$. This so-called restricted master program (RMP) is then optimized. Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)^\top$ be the dual solution w.r.t. the covering constraints (2.2b). The task of the pricing subproblem is then to provide (at least) one new pattern p with negative reduced cost $\tilde{c}_p = 1 - \boldsymbol{\pi}^\top \mathbf{a}^p$ or to prove that no such pattern exists. In the former case, the resulting RMP (with the generated pattern/s) is re-optimized and the column-generation process is repeated. In the latter case, the linear relaxation of (2.2) is solved providing a lower bound z_{LP}^{VPP} on z^{VPP}. An optimal integer solution to (2.2) requires the integration of column generation into a branch-and-bound scheme a.k.a. branch-and-price (Lübbecke and Desrosiers, 2005).

The pricing problems of the different formulations 01-VPP, B-VPP, and U-VPP

seek to find a pattern with negative reduced cost $\tilde{c}_p$ that is feasible for the pattern set considered in the respective formulation. They are equivalent to solving one of the following integer programs:

$$\max \sum_{i=1}^{m} \pi_i y_i \tag{2.5a}$$

$$\text{s.t.} \sum_{i=1}^{m} w_i^d y_i \leq W^d, \qquad d \in D \tag{2.5b}$$

$$\text{(01-MKP)} \quad y_i \in \{0,1\}, \quad \text{(B-MKP)} \quad y_i \in \{0,1,\ldots,u_i^B\}, \quad \text{(U-MKP)} \quad y_i \in \mathbb{Z}_+, \qquad i \in I \tag{2.5c}$$

where $u_i^B = \min\{q_i, \min_d \lfloor \frac{W^d}{w_i^d} \rfloor\}$. The pricing subproblems are thus variants of *multidimensional knapsack problems* (MKPs, Kellerer *et al.*, 2004), either binary (01-MKP) for 01-VPP, bounded (B-MKP) for B-VPP, or unbounded (U-MKP) for U-VPP. For U-MKP, it is valid to bound the variables y_i in (2.5c) from above by $u_i^U = \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$. For convenience, we define a corresponding upper bound $u_i^{01} = 1$ for all $i \in I$ for 01-MKP.

We now briefly discuss the main advantages and disadvantages of the different covering formulations. The master program of 01-VPP is the largest of the three in terms of number of constraints, while the master of U-VPP is the smallest. Regarding the number of variables, i.e., the number of different feasible patterns, there is no general order between the three formulations, since it depends on the demands q_i and the share of items with identical weights in $\mathcal{I}$. To be precise, the number of feasible patterns for 01-VPP is $\mathcal{O}\left(\sum_{i=1}^{\min\{u,|\mathcal{I}|\}} \binom{|\mathcal{I}|}{i}\right)$, with $u = \max_i \min_d \lfloor \frac{W^d}{w_i^d} \rfloor$. For U-VPP, it is $\mathcal{O}\left(\sum_{i=1}^{u} \binom{|\mathcal{I}/\mathcal{I}[\mathbf{w}]|+i-1}{i}\right)$. In the special case of complete aggregation, i.e., $I = \mathcal{I}/\mathcal{I}[\mathbf{w}]$, the number of B-VPP patterns is by definition not greater than that of U-VPP, i.e., $|P_B| \leq |P_U|$.

Concerning the linear relaxation bounds, $z_{LP}^{\text{01-VPP}} = z_{LP}^{\text{B-VPP}} \geq z_{LP}^{\text{U-VPP}}$ holds. The equality $z_{LP}^{\text{01-VPP}} = z_{LP}^{\text{B-VPP}}$ results from the constraint aggregation approach discussed in (Ben Amor *et al.*, 2006). The following Example 1 shows a small instance with strict inequality between $z_{LP}^{\text{B-VPP}}$ and $z_{LP}^{\text{U-VPP}}$. Moreover, Rietz (2003) has shown that the 01-VPP with $D \geq 2$ does not possess the *modified integer round up property* (MIRUP) (see also Scheithauer and Terno, 1995; Caprara *et al.*, 2014). The property is also unclear for CSP, i.e., U-VPP with $D = 1$, and herewith also for U-VPP in higher dimension. Still, linear relaxation bounds seem to be rather tight for most instances and all three formulations (see Section 2.2.4.4).

Example 1. The following VPP example has the same LP bound for the covering formulations 01-VPP and B-VPP but different LP bounds for 01-VPP/B-VPP and

U-VPP. Consider a two-dimensional VPP with $\mathbf{W} = (6,6)^\top$ and items $\mathcal{I} = \{1,2,3\}$ with weights $\mathbf{w_1} = (3,3)^\top$ and $\mathbf{w_2} = \mathbf{w_3} = (2,2)^\top$. For 01-VPP, $z_{LP}^{\text{01-VPP}} = 1.5$. An optimal solution is 0.5 times $(1,1,0)^\top$, 0.5 times $(1,0,1)^\top$, and 0.5 times $(0,1,1)^\top$. For B-VPP, consider $I = \{1,[2]\}$ with $q_1 = 1$ and $q_2 = 2$. Then, $z_{LP}^{\text{B-VPP}} = 1.5$. An optimal solution is 1 time $(1,1)^\top$ and 0.5 times $(0,2)^\top$. For U-VPP, $I = \{[1],[2]\}$ and $z_{LP}^{\text{U-VPP}} = 7/6$. An optimal solution is 0.5 times $(2,0)^\top$ and $2/3$ times $(0,3)^\top$.□

Recall that the subproblems of formulations 01-VPP, B-VPP, and U-VPP are 01-MKPs, B-MKPs, and U-MKPs with $m_{01} \geq m_B \geq m_U$ items, respectively. Recall also that in the 1-dimensional case, all three versions can be solved in $\mathcal{O}(mW)$ time, where W is the 1-dimensional capacity. Moreover, the space requirement is (applying storage reduction techniques) $\mathcal{O}(m + W)$ for binary and bounded knapsack problems, while it is $\mathcal{O}(W)$ in the unbounded case (Kellerer *et al.*, 2004). In higher dimensions, the solution of the MKPs with a straightforward dynamic-programming algorithm requires $\mathcal{O}(m\mathcal{W})$ time, where $\mathcal{W} = \prod_d W^d$.

To obtain integer solutions, the well known branching rule of Ryan and Foster (1981), which has proven to be quite effective for set-partitioning in general, can be applied to 01-VPP. For B-VPP and U-VPP, however, Ryan-Foster branching is not applicable and one has to resort to alternative branching rules such as the branching scheme presented by Vanderbeck (1999). Note that Ryan-Foster branching is a special case of the Vanderbeck branching. All branching decisions taken in any of the three formulations have a non-trivial impact on the subproblem and have to be accounted for in its solution. Overall, branching is much more complex in the B-VPP and U-VPP cases compared to 01-VPP. On the other hand, formulation 01-VPP suffers from severe symmetry issues with respect to items with identical weights.

Another difference of the three formulations originates from efforts to stabilize the column-generation process using *(deep) dual-optimal inequalities* ((D)DOIs, see Section 2.2.3). This technique has already been successfully applied to BPP and CSP by Ben Amor *et al.* (2006) and Valério de Carvalho (2005). Additional classes of valid DOIs and DDOIs as well as the dynamic separation of violated (D)DOIs have been proposed and applied to different coloring and packing problems by Gschwind and Irnich (2016). It is shown in Section 2.2.3 that the dual inequalities used in (Gschwind and Irnich, 2016), namely *weighted subset inequalities* (WSIs) and *pair inequalities* (PIs), are DOIs for U-VPP. For B-VPP and 01-VPP, however, the PIs are DDOIs whereas the WSIs are generally neither DOIs nor DDOIs.

The contribution of the paper at hand is the following: We present a unified modeling and solution approach for the three MKP subproblems (binary, bounded, and unbounded) based on dynamic programming and the *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). From a worst-case complexity point of view, its temporal effort of $\mathcal{O}(m\mathcal{W})$ is equivalent to the

above-mentioned dynamic-programming approaches. It exploits that many of the dynamic-programming states are not reached in practice, giving it an attractive time and space consumption in the average case. Even more, the SPPRC-based solution approach can cope with the Ryan-Foster and the Vanderbeck branching schemes. In addition, the approach allows to identify violated (D)DOIs as a by-product of the pricing. Computational results indicate that our unified approach for solving 01-VPP, B-VPP, and U-VPP is very competitive with the most recent exact VPP approaches presented in the literature.

This paper is organized as follows. Section 2.2 presents the unified solution approach with subsections on the SPPRC formulation of the MKP subproblems, their resolution via dynamic-programming labeling, the discussion of stabilization using (D)DOIs, and branching. Section 2.3 presents the computational results. Final conclusions are drawn in Section 2.4.

2.2 Unified Branch-and-Price

We now describe the components of the branch-and-price algorithm. In Section 2.2.1 the column-generation pricing problem, the MKP, is modeled as an SPPRC. Two alternative labeling algorithms for its solution are presented in Section 2.2.2. The stabilization of the column-generation process via (D)DOIs is described in Section 2.2.3. Branching rules including the resulting modified MKP subproblem, its representation as an SPPRC, and the impact of branching decisions on (D)DOIs are discussed in Section 2.2.4.

2.2.1 SPPRC Formulation for MKP Subproblems

All three variants of the MKP (U-MKP, B-MKP, and 01-MKP) can be formulated as SPPRCs with $D + 1$ resources as follows: The underlying digraph $G = (V, E)$ consists of $m + 1$ vertices $V = \{0, \dots, m\}$ and $m + \sum_{i \in I} u_i$ arcs E. For each item $i \in I$, there exist $u_i + 1$ parallel arcs denoted by $E(i)$ connecting vertex $i - 1$ with vertex i. Recall that, depending on the MKP variant, the number u_i is defined differently, see above. We assume that the arcs $e \in E(i)$ are indexed by $a_i(e) \in \{0, 1, \dots, u_i\}$. Moreover, we define the weight of the $a_i(e)$th arc e of $E(i)$ as $\mathbf{w}(e) := a_i(e)\mathbf{w}$ and its profit as $\pi(e) := a_i(e)\pi_i$.

An example of the digraph G is shown in Figure 2.1. In general, every 0-m-path $(0, e_1, 1, e_2, 2, \dots, m-1, e_m, m)$ in G defines a pattern $(a_1(e_1), a_2(e_2), \dots, a_m(e_m))^\top$. This pattern is feasible if and only if $\sum_{i=1}^{m} \mathbf{w}(e_i) \leq \mathbf{W}$ holds. Hence, the MKP pricing problem is the problem of finding a maximum profit 0-m-path in G with a D-dimensional weight not exceeding $\mathbf{W}$. If the profit $\pi = \sum_{i=1}^{m} \pi(e_i)$ exceeds 1,

then the pattern $(a_1(e_1), a_2(e_2), \ldots, a_m(e_m))^\top$ has negative reduced cost $\tilde{c} = 1 - \pi$. This is an SPPRC with m independent capacity constraints.

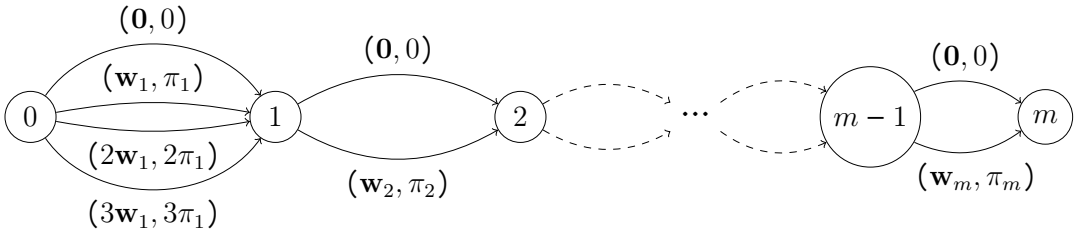


Figure 2.1: SPPRC digraph for a MKP with $u_1 = 3, u_2 = 1$, and $u_m = 1$. The arcs $e \in E$ are shown with their D-dimensional weight and profit component $(w(e), \pi(e)))$.

2.2.2 Subproblem Solution via Dynamic-Programming Labeling

Dynamic-programming labeling for solving SPPRCs offers several algorithmic choices such as mono-directional (pure forward or pure backward) or bidirectional labeling, and the dominance rules used to discard partial paths that provably lead to only non-optimal 0-m-paths. Forward dynamic programming starts at the trivial forward path $F = (0)$ with the initial label $(\mathbf{0}, 0)$ and labels are extended iteratively to $1, 2, \ldots, m$ to finally form feasible 0-m-paths. Accordingly, every forward partial path $F = (0, e_1, 1, \ldots, e_j, j)$ from 0 to some $j \in V$ is represented by the label $L(F) = (\sum_{i=1}^{j} \mathbf{w}(e_i), \sum_{i=1}^{j} \pi(e_i)) = (\mathbf{w}, \pi)$. Forward extension along an arc $e_{j+1} \in E(j+1)$ produces the label $L(F, e_{j+1}, j+1) = (\mathbf{w}+\mathbf{w}(e_{j+1}), \pi+\pi(e_{j+1}))$.

Conversely, backward dynamic programming starts with the trivial backward path $B = (m)$ with the initial label $(\mathbf{0}, 0)$ and labels are extended iteratively to $m-1, m-2, \ldots, 1, 0$ to finally form feasible 0-m-paths. Here, every backward partial path $B = (j, e_{j+1}, j+1, \ldots, e_m, m)$ from some $j \in V$ to m is represented by the label $L(B) = (\sum_{i=j+1}^{m} \mathbf{w}(e_i), \sum_{i=j+1}^{m} \pi(e_i)) = (\mathbf{w}, \pi)$. Backward extension along an arc $e_j \in E(j)$ produces the label $L(j-1, e_j, B) = (\mathbf{w} + \mathbf{w}(e_j), \pi + \pi(e_j))$.

Forward and backward labels are feasible if their weight component does not exceed $\mathbf{W}$. Every forward label at vertex m and every backward label at vertex 0 implies a negative reduced-cost pattern if and only if $1 - \pi$ is strictly negative.

A forward label $L(F) = (\mathbf{w}, \pi)$ and a backward label $L(B) = (\mathbf{w}', \pi')$ that end and start at the same vertex can be merged to form a complete feasible 0-m-path if $\mathbf{w} + \mathbf{w}' \leq \mathbf{W}$. The reduced-cost of the imposed pattern is $1 - (\pi + \pi')$.

Labeling with Weak Dominance. Our first dynamic-programming algorithm is a pure forward labeling algorithm and the one with the better worst-case time complexity. It uses the following simple dominance rule.

Rule 1. (Weak Dominance) A forward label $L(F) = (\mathbf{w}, \pi)$ of a forward partial path F dominates another forward label $L(F') = (\mathbf{w}', \pi')$ of a forward partial path F' ending at the same vertex if $\mathbf{w} = \mathbf{w}'$ and $\pi \geq \pi'$.

Dominated labels can be discarded except for cases of mutual dominance, i.e., identical costs, in which case one of the labels can be discarded while the other one must be kept.

The time complexity of a forward labeling algorithm that uses Rule 1 depends on the way how labels are stored. In our implementation, we use a hash table for the labels referring to the same vertex. For a label $(\mathbf{w}, \pi)$, $\mathbf{w}$ is the hash key and π the value to be stored in the hash table. Note that the number of possible keys is exactly $K := \prod_{d=1}^{D}(W^d + 1)$. If the hash table is designed appropriately, arbitrary insertions and lookups of key-value pairs can be guaranteed to take amortized constant time per operation (Cormen *et al.*, 2009, Chapter 11). In cases where the dimension D and the capacities $W^1, \ldots, W^D$ are very large, K may become so huge that a hash table of appropriate size does not fit into the main memory of the computer. Then, with a smaller hash table, its load factor becomes critical and constant-time operations can no longer be ensured. Hence, the overall worst-case complexity is bounded by $\mathcal{O}(mK)$ when computer memory of $\mathcal{O}(K)$ is available.

Labeling with Strong Dominance. Our second dynamic-programming algorithm is a bidirectional algorithm. The idea of bidirectional labeling is to alleviate the combinatorial explosion that typically occurs when partial paths grow longer. As a consequence, bidirectional labeling algorithms often, though not always, have a better average-case time effort than their monodirectional counterparts. For (E)SPPRC, the idea was first presented and successfully used by Righini and Salani (2006). They merge labels "in the middle", which in our context is at vertex $\lfloor m/2 \rfloor \in V$ (or $\lceil m/2 \rceil \in V$ if m is odd). Recent studies of Tilk *et al.* (2017) have shown that one can often benefit from choosing the merge point dynamically, comparing the (expected) workload in forward and backward direction.

In the bidirectional algorithm, we use the following stronger dominance rule whose application, however, produces additional effort in the worst case.

Rule 2. (Strong Dominance) A forward (backward) label $L(P) = (\mathbf{w}, \pi)$ of a partial path P dominates another forward (backward) label $L(P') = (\mathbf{w}', \pi')$ of a partial path P' ending at the same vertex if $\mathbf{w} \leq \mathbf{w}'$ and $\pi \geq \pi'$.

Note that there exist better dominance algorithms than straightforward pairwise comparison of labels. For dominance between Δ-dimensional vectors, Bentley

(1980) invented a multi-dimensional divide-and-conquer algorithm that requires $\mathcal{O}(n\log(n)^{\Delta-1})$ time to determine all Pareto-optimal out of n vectors. In our application, we have $\Delta = D + 1$ (labels have weights and profit). We use the algorithm of Bentley only for 2-dimensional VPPs but not for the 20-dimensional VPPs (see Section 2.3) even if $log(n)^D$ is in the worst case better than linear.

Figure 2.2 visualizes the differences between the weak and the strong dominance.

2.2.3 Dual Inequalities

For the VPP, any linear inequality in the dual variables $\pi_i, i \in I$, is a *dual inequality*. We denote by D^* the set of optimal solutions of the dual model to the linear relaxation of (2.2), i.e., the dual-optimal space.

Dual Inequalities for U-VPP. Following Ben Amor *et al.* (2006), a dual inequality $\mathbf{t}^\top\boldsymbol{\pi} \le t$ (with $\mathbf{t} \in \mathbb{Z}^I$ and $t \in \mathbb{Z}$ is a DOI if $D^* \subseteq \{\boldsymbol{\pi} : \mathbf{t}^\top\boldsymbol{\pi} \le t\}$. In (Gschwind and Irnich, 2016, Theorem 1(ii)) it is proven that for any $\mathbf{t} \in \mathbb{Z}_+^I$ and an item $h \in I$ with

$$\mathbf{w}_h \ge \sum_{i\in I} t_i\mathbf{w}_i \qquad \text{it follows that} \qquad \pi_h \ge \sum_{i\in I} t_i\pi_i \tag{2.6}$$

is a DOI for U-VPP. These DOIs are called *weighted subset inequalities* (WSIs). The interpretation of the precondition is that whenever item h is used (in a solution) this item can always be replaced by one or several occurrences of the items $S = S(\mathbf{t}) := \{i \in I : t_i > 0\}$, where the coefficient t_i specifies the number of occurrences. The DOI then says that covering one unit of demand of item h cannot be cheaper than covering t_i units of demand for all $i \in S$.

WSIs were first introduced in (Gschwind and Irnich, 2016) and generalize the *subset inequalities* (SI) of Ben Amor *et al.* (2006), in which it is required that all coefficients t_i are binary. Hence, dual inequalities $\pi_h \ge \sum_{i\in S}\pi_i$ with $h \in I$ and $S \subset I$ are DOIs for U-VPP if $\mathbf{w}_h \ge \sum_{i\in S}\mathbf{w}_i$ holds.

A special case of the SIs is two items $i, h \in I$ that fulfill $\mathbf{w}_i \le \mathbf{w}_h$. In this case, the so-called *pair inequality* $\pi_i \le \pi_h$ is a DOI for U-VPP. It means that covering one unit of demand of item h is not cheaper than one unit of demand of item i.

Dual Inequalities for 01-VPP. The situation of 01-VPP and B-VPP requires a more differentiated analysis. We start by analyzing dual inequalities for the 01-VPP. The formulation (2.2) for the 01-VPP is a pure binary formulation with binary constraint matrix A, right-hand side $\mathbf{1} \in \mathbb{R}^I$, and cost coefficients $\mathbf{1} \in \mathbb{R}^P$. Moreover, the coefficient matrix A has the (h, i)-row replacement property

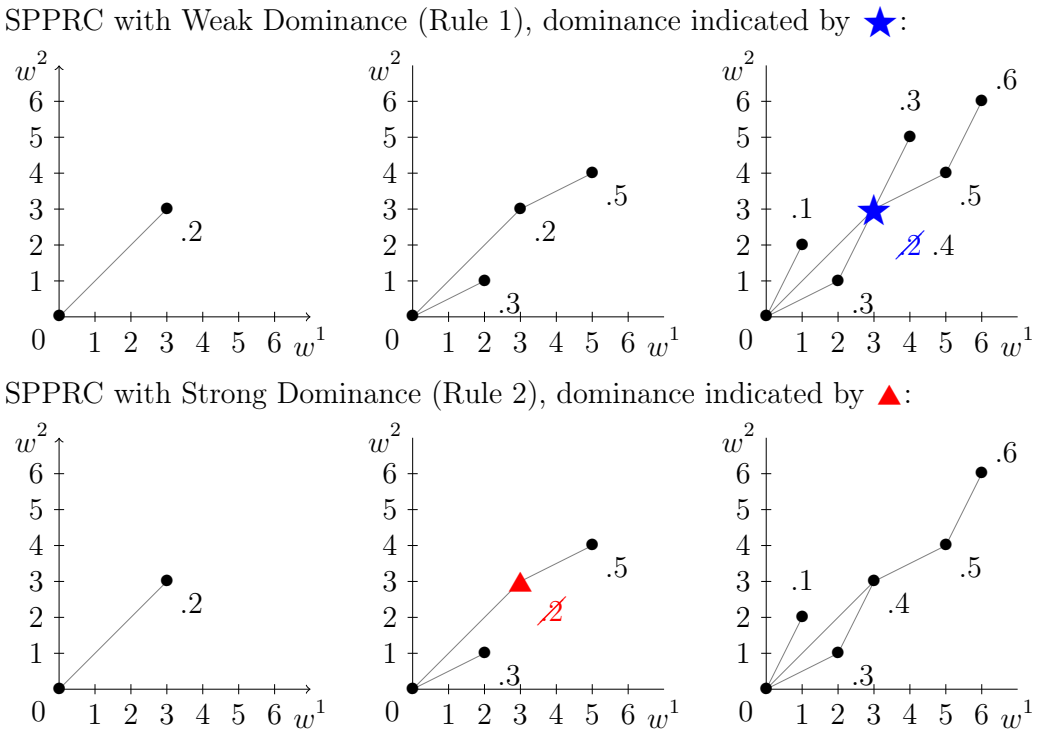


Figure 2.2: Comparison of two variants for solving 01-MKP with $\mathbf{w}_1 = (3,3)$, $\mathbf{w}_2 = (2,1)$, $\mathbf{w}_3 = (1,2)$, $\pi_1 = 0.2$, $\pi_2 = 0.3$ and $\pi_3 = 0.1$. For the SPPRC with Weak Dominance, the label consisting of item set $\{1\}$ with profit 0.2 and weight $(3,3)$ is discarded by Rule 1 at the final step. In the end, the undominated labels are $((0,0),0)$, $((1,2),0.1)$, $((2,1),0.3)$, $((3,3),0.4)$, $((4,5),0.3)$, $((5,4),0.5)$, $((6,6),0.6)$ with corresponding item sets $\varnothing, \{3\}, \{2\}, \{2,3\}, \{1,3\}, \{1,2\}$, and $\{1,2,3\}$, respectively. For the SPPRC with Strong Dominance, the label with profit 0.2 and weight $(3,3)$ is dominated by the label with profit 0.3 and weight $(2,1)$ and is therefore discarded at the second step. In the end, the undominated labels are $((0,0),0)$, $((1,2),0.1)$, $((2,1),0.3)$, $((3,3),0.4)$, $((5,4),0.5)$, $((6,6),0.6)$ with corresponding item sets $\varnothing, \{3\}, \{2\}, \{2,3\}, \{1,2\}$, and $\{1,2,3\}$, respectively.

(Gschwind and Irnich, 2016) for every two items $h, i \in I$ with $\mathbf{w}_h \geq \mathbf{w}_i$. The (h,i)-row replacement property means that any pattern $\mathbf{a} \in P$ with $a_h = 1$ and $a_i = 0$ can be modified into one with $a_h = 0$ and $a_i = 1$ (keeping all other coefficients unchanged) so that again a feasible pattern results. Hence, all preconditions of Proposition 5 of (Gschwind and Irnich, 2016) are fulfilled. Thus, the set of all PIs $\{\pi_h \geq \pi_i : \mathbf{w}_h \geq \mathbf{w}_i\}$ forms a set of *deep dual-optimal inequalities* (DDOIs) for

01-VPP. By definition (Ben Amor *et al.*, 2006) for a set of dual inequalities to be DDOIs, there must exist at least one dual-optimal solution $\boldsymbol{\pi}^* \in D^*$ that fulfills all inequalities of the set. In particular, all DOIs are DDOIs but the reverse is not necessarily true, because DDOIs may cut off some but not all dual-optimal solutions from D^*.

However, for the 01-VPP, specific pair inequalities that fulfill some additional requirements can be proven DOIs. For example, if in addition to $\mathbf{w}_h \geq \mathbf{w}_i$ the condition $\mathbf{w}_h^d + \mathbf{w}_i^d > W^d$ holds for at least one dimension $1 \leq d \leq D$, then this PI is a DOI for 01-VPP. Generalizing, for a SI defined by a subset $S \subset I$ and item $h \in I$, the additional conditions that $\mathbf{w}_h^d + \mathbf{w}_i^d > W^d$ for at least one dimension $1 \leq d \leq D$ for all $i \in S$ guarantee that this SI is a DOI for 01-VPP. These are restricted SIs; general SIs may neither be DOIs nor DDOIs for 01-VPP if only $\mathbf{w}_h \geq \sum_{i \in S} \mathbf{w}_i$ holds.

Dual Inequalities for B-VPP. Now we derive (D)DOIs for B-VPP by discussing their relationship to (D)DOIs of 01-VPP. Recall first that B-VPP is a family of formulations; every B-VPP can be derived from formulation 01-VPP via a (partial) constraint aggregation as already discussed in Section 2.1. Indeed, let h and i be two items of 01-VPP with identical weights $\mathbf{w}_h = \mathbf{w}_i$, i.e., items from the same equivalence class $\mathcal{I}[\mathbf{w}_h] = \mathcal{I}[\mathbf{w}_i]$. For these two items, the two pair inequalities $\pi_i \leq \pi_h$ and $\pi_h \leq \pi_i$ imply the dual equality $\pi_h = \pi_i$. Hence, $\bigcup_{i,h \in I} (\{\pi_h = \pi_i : \mathbf{w}_h = \mathbf{w}_i\} \cup \{\pi_h \geq \pi_i : \mathbf{w}_h \geq \mathbf{w}_i\})$ are DDOIs for 01-VPP. Now, any aggregation over some or all $\mathcal{I}[\mathbf{w}_h]$ with $h \in \mathcal{I}$ leads to a specific choice of aggregated items I and herewith to one specific B-VPP. In any case, Propositions 6 in (Gschwind and Irnich, 2016) ensures that all pair inequalities remain valid DDOIs for the B-VPP.

All theoretical results about the discussed dual inequalities for the three VPP formulations are summarized in the bottom part of Table 2.1.

Use of Dual Inequalities for Stabilization. Dual inequalities can be added as additional columns of (2.2) in order to stabilize the column-generation process for solving the linear relaxation. If the set of dual inequalities is DDOIs, this addition does not change the lower bound of the linear relaxation (this is shown in Proposition 1 in (Ben Amor *et al.*, 2006) and in Proposition 1 in (Gschwind and Irnich, 2016)). For a WSI (2.6) that replaces item $h \in I$ by multiple (aggregated) items $i \in S = S(\mathbf{t})$ each t_i times, the associated column in (2.2) has a cost of zero and the coefficients $\mathbf{t} - \mathbf{u}_h$, where $\mathbf{u}_h$ is the hth unit vector in $\mathbb{R}^I$. Assuming that the

set of DIs is given as $(h, \mathbf{t}) \in \mathcal{D}$, the expanded master becomes:

$$\min \sum_{p \in P} x_p \tag{2.7a}$$

$$\text{s.t.} \sum_{p \in P} a_i^p x_p + \sum_{(h,\mathbf{t}) \in \mathcal{D}} (\mathbf{t} - \mathbf{u}_h) y_{h\mathbf{t}} = q_i, \qquad i \in I \tag{2.7b}$$

$$x_p \geq 0 \text{ integer}, \qquad p \in P \tag{2.7c}$$

$$y_{h,\mathbf{t}} \geq 0 \text{ integer}, \qquad (h, \mathbf{t}) \in \mathcal{D} \tag{2.7d}$$

Note first that in general there is an exponential number of WSIs and SIs, while there can be at most m^2 PIs. In high dimensional VPPs, i.e., $D \gg 2$, it can happen that there are only very few PIs, if any. The question is therefore which dual inequalities one should add to the RMP at which point of the column-generation process. Different strategies have been described in the literature:

1. *Static Approach:* Valério de Carvalho (2005) and Ben Amor *et al.* (2006) preselect a subset of PIs and SIs (with S restricted to $|S| = 2$) for the BPP and CSP, respectively. These dual inequalities were then added to the initial RMP.
2. *Dynamic Approach:* Gschwind and Irnich (2016) were the first to suggest the addition of (the most) violated dual inequalities in every (or only some) column-generation iterations. This actually requires a separation algorithm. For PIs, separation can be done by inspection, while for SIs in the BPP and WSIs in the CSP they showed that separation is a by-product of solving the pricing problem. In the next paragraph, we describe how separation can also be accomplished efficiently for VPPs using the results of the SPPRC labeling algorithm.
3. *Mixed Approach:* It is obvious that the static and dynamic approaches can be combined in the sense that some dual inequalities are initially added to the RMP and others are separated in the column-generation iterations as done in (Gschwind and Irnich, 2016).

If additional columns for PIs, SIs, or WSIs are added to the RMP, the solution to the master program may consist not only of columns of patterns but of a mixture of these and columns of dual inequalities. Such a solution can however be transformed into a pure pattern-columns solution if the dual inequalities are DDOIs (both statements are in fact equivalent as proven in Gschwind and Irnich, 2016, Proposition 1). An iterative transformation procedure in the case that all added dual inequalities are WSIs is described in detail in (Gschwind and Irnich, 2016, Algorithm 1). The basic idea is as follows. In each iteration, it chooses from the current (partly transformed) solution a pattern column with coefficients $\mathbf{a}$ and a WSI column with coefficients $\mathbf{t} - \mathbf{u}_h$ that are compatible, meaning that $\mathbf{a} + \mathbf{t} - \mathbf{u}_h$ represents a feasible pattern. Let x^* and y^* be the solution value of the chosen pattern and WSI column, respectively. Then, a new solution is constructed from the current one by decreasing the solution values of the chosen pattern and WSI

columns by $\min\{x^*, y^*\}$ and increasing the solution value of the column corresponding to pattern $\mathbf{a}+\mathbf{t}-\mathbf{u}_h$ by the same amount. The procedure continues until no more compatible pattern and WSI column-pair exists. If all WSIs that have been added are DDOIs, it is guaranteed that none of them has positive value in the current transformed solution so that a pure pattern-columns solution is found.

As described and done in (Gschwind and Irnich, 2016) where, e.g., SIs are used for BPP, it is also possible to add to the RMP dual inequalities that are (generally) not DDOIs. The intention of this *overstabilization* when using appropriate families of dual inequalities is twofold: First, although they are generally not, these inequalities may in fact be DDOIs for most instances. Second, whether they are actually DDOIs or not, their addition has a stabilizing effect on the column-generation process. Moreover, if the added inequalities are no DDOIs for the given instance, i.e., all dual-optimal solutions are cut-off, this defect can be remedied typically without significant additional computational effort. The recovery procedure of Gschwind and Irnich (2016) can be used to detect and handle overstabilization if all added dual inequalities are WSIs. It proceeds by first trying to build from the RMP solution a pure pattern-columns solution. If this is not possible the RMP has been overstabilized. In this case, there exists a WSI column replacing item $h \in I$ with positive value, but no compatible pattern column exists in the solution. The recovery procedure then eliminates all WSIs replacing item h from the RMP, forbids their re-separation/generation, and restarts the column-generation process to optimize the RMP. The process iterates until a pure pattern-columns solution is found.

Separation of Violated Dual Inequalities. Since the number of valid PIs $\pi_h \geq \pi_i$ is at most quadratic, violated PIs can be found by simple inspection. For this purpose, the list of pairs $(h, i) \in I \times I$ with $\mathbf{w}_h \geq \mathbf{w}_i$ can be determined in a preprocessing step prior to the column-generation process and stored requiring $\mathcal{O}(m^2)$ time and space only. Then, in every separation step, this list needs to be parsed and $\pi_h < \pi_i$ is tested.

Direct inspection is clearly not possible for SIs and WSIs due to their exponential number. When solving the pricing problem with the SPPRC labeling algorithm, however, violated SIs and WSIs can be effectively identified. If Rule 2 is applied for dominance, the identification even is a direct by-product of the labeling procedure: Consider for each single item $h \in I$ its associated label $L_h = (\mathbf{w}_h, \pi_h)$ which results from the extension of $(\mathbf{0}, 0)$ along the second arc (with multiplicity 1) between $h-1$ and h. If L_h is dominated using the strong dominance Rule 2, with strict $>$ in the profit component, then a violated SI or WSI in which h is replaced is found. The item set S of the SI or WSI and, in case of the U-VPP the multiplicities $\mathbf{t} = (t_i)$, can be read from the dominating label $(\mathbf{w}, \pi)$. Indeed, the item set S is the item

set of the dominating label and (if needed) the multiplicities $\mathbf{t} = (t_i)$ are those counting the number of aggregated items in the dominating label. In Figure 2.2, item $h = 1$ with label $L_1 = ((3,3),0.2)$ is dominated by the label $((2,1),0.3)$. This latter label has item set $S = \{2\}$ (with multiplicity 1). Hence, the PI $\pi_1 \geq \pi_2$ is violated.

If in the example of Figure 2.2 a fourth item $h = 4$ with weight $\mathbf{w}_4 = (5,5)$ and profit $\pi_4 = 0.5$ existed, then its label $L_4 = ((5,5),0.5)$ would be dominated by the label $((5,4),0.5)$, which has item set $S = \{1,2\}$, so that the violated SI would be $\pi_4 \geq \pi_1 + \pi_2$.

When using the weak dominance Rule 1, however, this procedure is not applicable. Furthermore, the procedure does not produce most violated SIs/WSIs. Still, with some additional effort both cases can be effectively handled using information from the SPPRC as follows: Consider the set $\mathcal{P}$ of all Pareto-optimal labels of the final stage m and add the label $L_h = (\mathbf{w}_h, \pi_h)$ for each item $h \in I$ to $\mathcal{P}$. Then, invoke the dominance algorithm using the strong Rule 2 on the set $\mathcal{P}$ and track for each $L_h, h \in I$ the labels that dominate it with strict $>$ in the profit component. Finally, for each $h \in I$ this list needs to be scanned for (the most) violated SIs/WSIs. When using the algorithm of Bentley (1980) for the dominance algorithm, this separation procedure requires $\mathcal{O}(n\log(n)^D)$ and $\mathcal{O}(n)$ time and space, respectively.

2.2.4 Branching Schemes

Vanderbeck (1999) suggested a branching scheme for bin packing and cutting stock problems that we adapt to the multidimensional case. We first summarize the branching scheme and explain how the branching constraints are considered in the subproblem. Afterwards, we focus on how to treat DOIs along the branch-and-bound tree.

2.2.4.1 Subproblem Transformation and Branching Rule

The idea of the branching scheme of Vanderbeck (1999) is to formulate the knapsack subproblem as a pure binary problem. For the 01-VPP, the subproblem is already binary. For B-VPP and U-VPP, the subproblem is transformed into a binary multi-class knapsack problem. For a better understanding, we introduce an example of the transformation and present the general transformation afterwards.

Consider an instance of the B-VPP or U-VPP with $u_1 = 3$ and $u_2 = u_3 = u_4 = 1$. The coefficients $a_i^p \in \{0,1,\ldots,u_i\}$ of patterns for $i \in I = \{1,2,3,4\}$ are now written in binary code using $m_i = \lceil \log_2(u_i + 1) \rceil$ binary digits per item. Some

examples are:

$$p_1 = \begin{pmatrix}0\\1\\1\\1\end{pmatrix}_{10} = \begin{pmatrix}\boxed{\begin{smallmatrix}0\\0\end{smallmatrix}}\\1\\1\\1\end{pmatrix}_{2}, \; p_2 = \begin{pmatrix}1\\0\\1\\1\end{pmatrix}_{10} = \begin{pmatrix}\boxed{\begin{smallmatrix}1\\0\end{smallmatrix}}\\0\\1\\0\end{pmatrix}_{2},$$
$$p_3 = \begin{pmatrix}2\\0\\1\\0\end{pmatrix}_{10} = \begin{pmatrix}\boxed{\begin{smallmatrix}0\\1\end{smallmatrix}}\\0\\1\\1\end{pmatrix}_{2}, \; p_4 = \begin{pmatrix}3\\1\\1\\0\end{pmatrix}_{10} = \begin{pmatrix}\boxed{\begin{smallmatrix}1\\1\end{smallmatrix}}\\1\\1\\0\end{pmatrix}_{2}. \tag{2.8}$$

In these examples, the original 4-dimensional patterns are now represented using a 5-dimensional binary code because $\sum_{i\in I} m_i = 2 + 1 + 1 + 1 = 5$. For convenience, we define the binary places as pairs I_{01} of items and digits, here $I_{01} = \{(1,0),(1,1),(2,0),(3,0),(4,0)\}$. The binary coefficients are $\tilde{a}^p_{ib}$ for $(i,b) \in I_{01}$, e.g., $\tilde{a}^{p_1}_{1,0} = 0$ and $\tilde{a}^{p_1}_{2,0} = 1$.

Vanderbeck considers subsets of patterns that are defined by a pair (I^0, I^1) where I^0 and I^1 are disjoint subsets of I_{01}:

$$\hat{P}(I^0, I^1) = \{p \in P : \tilde{a}^p_{ib} = 0, (i,b) \in I^0 \text{ and } \tilde{a}^p_{ib} = 1, (i,b) \in I^1\}$$

Whenever a solution x^* to (2.2) is fractional, Vanderbeck has proven that there exists a pair (I^0, I^1) such that

$$\alpha(I^0, I^1) := \sum_{p\in\hat{P}(I^0,I^1)} x^*_p$$

is fractional. Consequently, one can create the two branches

$$\sum_{p\in\hat{P}(I^0,I^1)} x_p \le \lfloor \alpha(I^0, I^1) \rfloor \tag{2.9a}$$

and

$$\sum_{p\in\hat{P}(I^0,I^1)} x_p \ge \lceil \alpha(I^0, I^1) \rceil. \tag{2.9b}$$

Example 2. Consider an RMP with patterns $p_1, \ldots, p_4$ of equation (2.8), demands $q_1 = 5, q_2 = 2, q_3 = 3, q_4 = 4$ and the optimal solution $x^*_1 = 1.25, x^*_2 = 2.75, x^*_3 = 0$, and $x^*_4 = 0.75$. To branch with the pair $(I^0, I^1) = (\{(2,0)\}, \varnothing)$, find all patterns with value 0 at binary place $(2,0) \in I_{01}$, i.e., the pattern set $\hat{P}(\{(2,0)\}, \varnothing) = \{p_2, p_3\}$ and $\alpha(\{(2,0)\}, \varnothing) = 2.75$.

To branch with the pair $(I^0, I^1) = (\varnothing, \{(1,0),(3,0)\})$, find all patterns with value 1 at binary places $(1,0),(3,0) \in I_{01}$, i.e., the pattern set $\hat{P}(\varnothing, \{(1,0),(3,0)\}) = \{p_2, p_4\}$ and $\alpha(\varnothing, \{(1,0),(3,0)\}) = 3.5$.

For 01-VPP, the demand of items is 1 so that branching on single items is impossible. However, for any two items $i_1, i_2 \in I$, the relation $0 \leq \alpha(\varnothing, \{(i_1,0),(i_2,0)\}) \leq 1$ holds. Branching on $(I^0, I^1) = (\varnothing, \{(i_1,0),(i_2,0)\})$ means that the items are either not allowed to be packed together

$$\sum_{p \in \hat{P}(\varnothing, \{(i_1,0),(i_2,0)\})} x_p \leq 0 \tag{2.10a}$$

or forced to be packed together

$$\sum_{p \in \hat{P}(\varnothing, \{(i_1,0),(i_2,0)\})} x_p \geq 1. \tag{2.10b}$$

This is Ryan-Foster branching. □

When a constraint of type (2.9) is added to the RMP, the subproblem must take the dual price of this constraint into account. More generally, consider a branch-and-bound node n, and let L^n and G^n be sets of branching constraints of the form (2.9a) and (2.9b) with corresponding dual variables $\mu_j \leq 0, j \in L^n$ and $\nu_k \geq 0, k \in G^n$, respectively, that are active at n. Note that each inequality is defined on the basis of a unique pair (I_j^0, I_j^1) for $j \in L^n$ and a unique pair (I_k^0, I_k^1) for $k \in G^n$, respectively. Let the corresponding pattern sets be $\hat{P}(I_j^0, I_j^1)$ and $\hat{P}(I_k^0, I_k^1)$. The associated subproblem, i.e., the MKP written with binary pattern coefficients and branching constraints, can now be written as a pure binary problem. Herein, binary variables $\hat{y}_{ib}$ for $(i,b) \in I_{01}$ are binary representations of y-variables in the MKP (2.5), i.e., $y_i = \sum_{b=0}^{m_i-1} 2^b \hat{y}_{ib}$. The associated pattern to $\hat{y} \in \{0,1\}^{I_{01}}$ is

$$a(\hat{y}) = \left(\sum_{b=0}^{m_1-1} 2^b \hat{y}_{1b}, \sum_{b=0}^{m_2-1} 2^b \hat{y}_{2b}, \ldots, \sum_{b=0}^{m_m-1} 2^b \hat{y}_{mb} \right) \in P. \tag{2.11}$$

Moreover, additional binary variables $l_j, j \in L^n$, and $g_k, k \in G^n$, are needed to indicate whether the resulting pattern $a(\hat{y})$ occurs in the respective constraints

(2.9a) and (2.9b). The subproblem reads as follows:

$$\max \sum_{i \in I} \sum_{b=0}^{m_i-1} (2^b \pi_i) \hat{y}_{ib} \quad + \sum_{j \in L^n} \mu_j l_j + \sum_{k \in G^n} \nu_k g_k \tag{2.12a}$$

$$\text{s.t.} \sum_{i \in I} \sum_{b=0}^{m_i-1} (2^b w_i^d) \hat{y}_{ib} \leq W^d, \qquad d \in D \tag{2.12b}$$

$$\sum_{b=0}^{m_i-1} 2^b \hat{y}_{ib} \leq u_i, \qquad i \in I \tag{2.12c}$$

$$a(\hat{y}) \in \hat{P}(I_j^0, I_j^1) \implies l_j = 1, \qquad j \in L^n \tag{2.12d}$$

$$a(\hat{y}) \notin \hat{P}(I_k^0, I_k^1) \implies g_k = 0, \qquad k \in G^n \tag{2.12e}$$

$$\hat{y}_{ib} \in \{0,1\}, \qquad (i,b) \in I_{01} \tag{2.12f}$$

$$l_j \in \{0,1\}, \qquad j \in L^n \tag{2.12g}$$

$$g_k \in \{0,1\}, \qquad k \in G^n \tag{2.12h}$$

The objective (2.12a) consists of three terms, one for the regular profit achieved for the selected items, similar to (2.5a), and two other terms that can be interpreted as penalties imposed by the branching constraints (2.9); note that $\mu_j \leq 0, j \in L^n$ and $\nu_k \geq 0, k \in G^n$. The capacity constraints (2.12b) are reformulations of (2.5b). The upper bounds for the pattern coefficients are incorporated by (2.12c). The coupling between the $\hat{y}$-variables and the branching-related indicator variables l and g are given by (2.12d) and (2.12e). Their linearization is straightforward and omitted here for the sake of brevity. The domains of all variables are given by (2.12f)–(2.12h).

2.2.4.2 SPPRC Graph Adaption

The main advantage of solving the MKP pricing problems (2.12) via SPPRC labeling is that the above branching constraints can all be implemented by (i) modifying profits of arcs and/or (ii) aggregating stages of the SPPRC digraph defined in Section 2.2.1. Such modifications do not alter the general structure of the SPPRC. Hence, the same dynamic-programming labeling approach as presented in Section 2.2.2 remains applicable.

The type of graph modification needed to implement the given branching decisions depends on how many (original) items interact in the branching. For a single branching decision (I^0, I^1) with $I^0, I^1 \subset I_{01}$, the set of affected items is $I^p = proj_1(I^0 \cup I^1)$ (projection onto the first component, i.e., the item component). We explain the modifications required by the following three types of branching decisions:

(i) Branching decision (I^0, I^1) for single items, i.e., $|I^p| = 1$;
(ii) Branching decision (I^0, I^1) for more than one item, i.e., $|I^p| > 1$;
(iii) Several branching decisions (I^0_j, I^1_j) and/or (I^0_k, I^1_k).

In the following, we assume that a branch-and-bound node n with branching constraints $j \in L^n$ and $k \in G^n$ and dual prices μ_j to (2.9a) and ν_k to (2.9b) are given, respectively.

Branching Decision for Single Items. The case of branching decisions that each affect only a single item, i.e, $I^p = proj_1(I^0 \cup I^1) = \{i\}$ for an item $i \in I$ can be handled by modifying the profits of some arcs. The arcs affected by these constraints are those arcs $e_i = (i-1, i) \in E(i)$ of the stage referring to item i. An arc is affected by a branching constraint if and only if its pattern coefficients $a_i(e_i)$, written as $\tilde{a}_{ib}(e_i)$ in binary coding, fulfill

$$\tilde{a}_{ib}(e_i) = 0 \quad \text{for all } (i,b) \in I^0 \qquad \text{and} \qquad \tilde{a}_{ib}(e_i) = 1 \quad \text{for all } (i,b) \in I^1.$$

In this case, the dual price μ_j is added if $(I^0, I^1) = (I^0_j, I^1_j)$ for some $j \in L^n$ and/or the dual price ν_k is added if $(I^0, I^1) = (I^0_k, I^1_k)$ for some $k \in G^n$.

Example 3. We assume that there are two branching decisions active at a branch-and-bound node n_A, one of type (2.9a) with dual price $\mu_A \leq 0$ and one of type (2.9b) with dual price $\nu_A \geq 0$. We further assume that the (first) $\leq$-condition refers to $(I^0, I^1) = (\{(1,1)\}, \emptyset)$, i.e., to the item $i = 1$ and the first binary digit $b = 1$ with value 2^1. The (second) $\geq$-condition refers to $(I^0, I^1) = (\emptyset, \{(1,0)\})$, i.e., the same item $i = 1$ and the lowest binary digit $b = 0$ with value 2^0. Figure 2.3 shows the modification of the SPPRC digraph. □

Branching Decision for Several Items. Branching decisions that affect several items at the same time are handled by item aggregation. If $|I^p| = |proj_1(I^0 \cup I^1)| > 1$ for a branching constraint defined by (I^0, I^1), all items in I^p are aggregated into a single *virtual* item. Assuming $I^p = \{i_1, i_2, \ldots, i_r\}$, new arcs are built as combinations of all possible multiplicities of the items $i_1, i_2, \ldots, i_r$.

Example 4. (continued from Example 3) We assume that a son node n_B of the branch-and-bound node n_A of Example 3 has an additional active branching decision based on $(I^0, I^1) = (\emptyset, \{(3,0), (4,0)\})$ resulting from a $\geq$-condition of the form (2.9b) with dual price ν_B. Here, $I^p = \{3, 4\}$ requires an aggregation over the two items 3 and 4 with $u_3 = u_4 = 1$. The number of possible combinations is therefore $(u_3 + 1)(u_4 + 1) = 4$, where each of the two items can either be included or excluded. Due to $I^1 = \{(3,0), (4,0)\}$, the only combination affected by the additional branching constraint is the one in which both items 3 and 4 are included, i.e., for weight $\mathbf{w}_3 + \mathbf{w}_4$ and original profit $\pi_3 + \pi_4$. The modified arc now has profit $\pi_3 + \pi_4 + \nu_B$. Figure 2.4 shows the modification of the SPPRC digraph. □

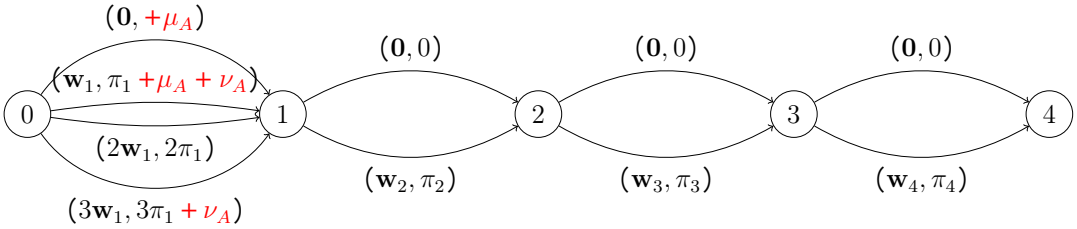


Figure 2.3: SPPRC digraph for a four-item MKP with $u_1 = 3, u_2 = u_3 = u_4 = 1$ (cf. Figure 2.1 and equation (2.8)). The arcs $e \in E$ are shown with their D-dimensional weight and profit component $(w(e), \pi(e))$ after implementing two branching constraints. The depicted situation refers to a branch-and-bound node with a $\leq$-constraint referring to $(I^0, I^1) = (\{(1,1)\}, \varnothing)$ with dual price μ_A and a $\geq$-constraint referring to $(I^0, I^1) = (\varnothing, \{(1,0)\})$ with dual price ν_A.

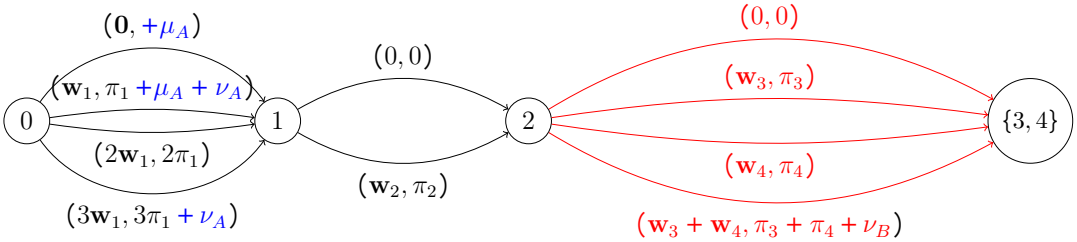


Figure 2.4: Modified SPPRC digraph resulting from three branching decisions. The first two decisions produce a digraph as in Figure 2.3. The additional third branching constraint, i.e., a $\geq$-condition on $(I^0, I^1) = (\varnothing, \{(3,0), (4,0)\})$ with dual price ν_B leads to the aggregation of the original items 3 and 4 indicated in red.

Several Branching Decisions. It can happen that several branching constraints overlap, i.e., the affected item sets are overlapping. More formally, consider several branching constraints affecting item sets $I_1^p, \ldots I_r^p$. These item sets are overlapping if they can be (re-)ordered such that $(I_1^p \cup \ldots \cup I_{j-1}^p) \cap I_j^p \neq \varnothing$ for all $j \in \{1, \ldots, r\}$. In this case, each such union $I^p = I_1^p \cup \ldots \cup I_r^p$ of the items must be aggregated. This works as described before by constructing all possible combinations of items in I^p.

Example 5. (continued from Example 4) We assume now that at a son node n_C of the branch-and-bound node n_B of Example 4 has an additional active branching decision on $(I^0, I^1) = (\{(2,0)\}, \{(3,0)\})$ resulting from a $\leq$-condition of the

form (2.9a) with dual price μ_C. At node n_C the sets of overlapping item sets are $I_1^p = \{1\}$ (resulting from the constraints active at the grandfather node n_A) and $I_2^p = \{3,4\} \cup \{2,3\} = \{2,3,4\}$ (resulting from the constraints defining n_B and n_C). The aggregation of items 2, 3, and 4 allows for eight combinations, each resulting from the inclusion or exclusion of the respective item. The modifications of nodes n_A and n_B remain and, additionally, μ_C is added to the profit of arcs excluding item 2 and including item 3. Figure 2.5 shows the modification of the SPPRC digraph.

In general, the item sets I^p may consist of non-consecutive items, e.g., $I_1^p = \{1,3\}$ and $I_1^p = \{2,4\}$, in which case the items/nodes of the SPPRC digraph have to be re-ordered. □

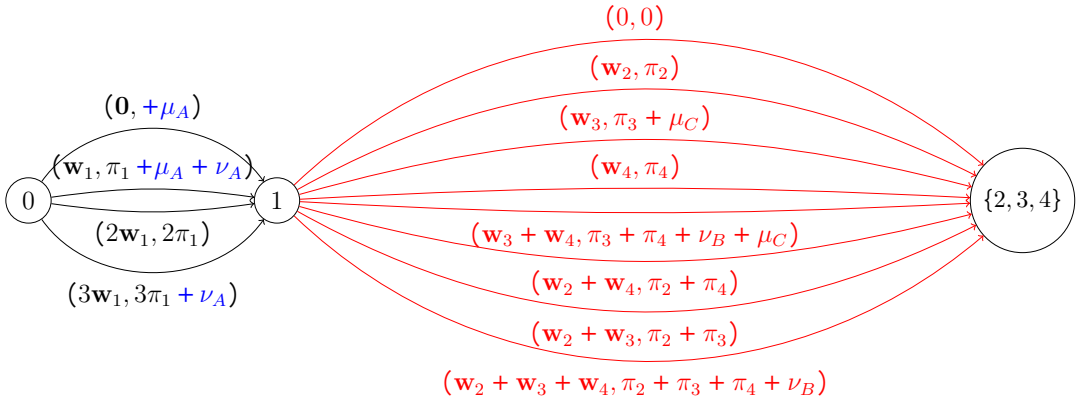


Figure 2.5: Fourth branching with $(I^0, I^1) = (\{(2,0)\}, \{(3,0)\})$ and $\leq$-condition (2.9a) with dual price μ_C. The profit of arcs excluding item 2 and including item 3 has to be modified by adding μ_C. The modification of the third branching decision (cf. Figure 2.4) remains. The items $I_2^p = \{2,3,4\}$ affected by the third and fourth branching decision are aggregated yielding $(u_2+1)(u_3+1)(u_4+1) = 8$ arcs indicated in red.

2.2.4.3 Handling of Dual-Optimal Inequalities

When using (D)DOIs within a branch-and-price algorithm, it needs to be ensured that the (D)DOIs are compatible with the branching decisions taken. Consider the following example.

Example 6. Consider again the RMP of Example 2 with patterns $p_1, \ldots, p_4$ and demands $q_1 = 5, q_2 = 2, q_3 = 3$, and $q_4 = 4$. Additionally, the PI $\pi_2 \geq \pi_1$ with associated primal variable x_5 is added to the RMP and the $\leq$-branching constraint

(2.9a) with $(I^0, I^1) = (\{(2,0)\}, \varnothing)$ and $\lfloor \alpha(\{(2,0)\}, \varnothing) \rfloor = 2$ has been imposed. The resulting constraints are

$$\begin{pmatrix}0\\1\\1\\1\\0\end{pmatrix} x_1 + \begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix} x_2 + \begin{pmatrix}2\\0\\1\\0\\1\end{pmatrix} x_3 + \begin{pmatrix}3\\1\\1\\0\\0\end{pmatrix} x_4 + \begin{pmatrix}1\\-1\\0\\0\\0\end{pmatrix} x_5 \quad \begin{matrix}\geq\\\geq\\\geq\\\geq\\\leq\end{matrix} \quad \begin{pmatrix}5\\2\\3\\4\\2\end{pmatrix}.$$

An optimal solution to the corresponding RMP is $x_1^* = x_2^* = 2, x_3^* = 0, x_4^* = x_5^* = 0.75$. It can be transformed into the pure patterns solution that uses pattern p_1 1.25 times, pattern p_2 2.75 times, and pattern p_4 0.75 times. This solution violates the branching constraint because

$$\sum_{p \in \hat{P}(\{(2,0)\}, \varnothing)} x_p^* = x_2^* + x_3^* = 2.75 \notin \lfloor \alpha(\{(2,0)\}, \varnothing) \rfloor = 2.$$

□

In essence, the issue with branching and the use of (D)DOIs in our case is the following: The branching decisions translate into RMP-constraints that are formulated on a set of pattern columns $\hat{P}(I^0, I^1)$. By replacing items in certain pattern columns $p \in P \setminus \hat{P}(I^0, I^1)$, (D)DOI-columns can then be used to implicitly represent pattern columns $p' \in \hat{P}(I^0, I^1)$, and vice versa, without any impact on the left-hand side of the corresponding RMP-constraint (2.9a) or (2.9b) so that a solution violating the branching decisions can result from the RMP.

One strategy to remedy this defect is to eliminate the respective (D)DOIs from the RMP and prevent their regeneration when using (D)DOIs in a dynamic approach, see Section 2.2.3. This strategy was also followed by Alves and Valério de Carvalho (2008) for solving the multiple length CSP with a stabilized column-generation approach. To the best of our knowledge, they were the first to use (D)DOIs within a fully-fledged branch-and-price algorithm. They derive conditions for the validity of the (D)DOIs that they use with respect to specific branching decisions present at a certain branch-and-bound node. All (D)DOIs that do not meet these conditions are then removed. Note that they use a different branching scheme that is compatible with their solution approach for the subproblem, so that their rules are not applicable in our case.

Consider a branch-and-bound node n in our algorithm. The set of items affected by the corresponding branching decisions is I^p. All (D)DOIs that are not related to any of the affected items, i.e., with $(\{h\} \cup S) \cap I^p = \varnothing$, are clearly valid at n. A straightforward way to ensure compatibility of the RMP solution with the

branching decisions is the elimination of all other (D)DOIs that are related to at least one affected item, i.e., with $(\{h\} \cup S) \cap I^p \neq \emptyset$. We can, however, also derive some additional conditions for the validity of (D)DOIs of the latter type. This enhancement is explained in the following.

Note first that given a branching decision on the patterns $\hat{P}(I^0, I^1)$, in the $\leq$-branch (2.9a) it has to be ensured that no pattern $p' \in \hat{P}(I^0, I^1)$ can be implicitly represented using (D)DOIs and patterns $p \in P \setminus \hat{P}(I^0, I^1)$, while in the $\geq$-branch (2.9b) no patterns $p' \in P \setminus \hat{P}(I^0, I^1)$ are allowed to be implicitly represented using (D)DOIs and patterns $p \in \hat{P}(I^0, I^1)$. Then, for 01-VPP, where coefficients are binary and the (D)DOIs we use are SIs, SIs that are related to at least one affected item are compatible in the $\leq$-branch if $h \notin proj_1(I^0)$ and $S \cap proj_1(I^1) = \emptyset$ hold. It is easy to verify that with the compatible SIs no patterns $p' \in \hat{P}(I^0, I^1)$ can be implicitly represented. In analogy, SIs are compatible in the $\geq$-branch if $S \cap proj_1(I^0) = \emptyset$ and $h \notin proj_1(I^1)$ hold.

For B-VPP and U-VPP, consider first only one of the affected items $i \in I^p$. A condition similar to the above condition for the binary case is as follows: In the $\leq$-branch, if *all* bits of i are fixed to 0 WSIs with $h \neq i$ are compatible with respect to item i, while if *all* bits of i are fixed to 1 WSIs with $i \notin S$ are compatible with respect to item i. The analog conditions in the $\geq$-branch are $i \notin S$ if all bits of i are fixed to 0 and $h \neq i$ if all bits of i are fixed to 1. Consider now the case that some of the m_i bits of item i are fixed to some value by the branching decisions and let $m_i' < m_i$ be the largest bit that is fixed. Then, WSIs with $i \in S$ and $t_i \bmod 2^{m_i'+1} \equiv 0$ are compatible with respect to item i. For a WSI to be compatible with the branching decisions at n, it has to fulfill one of the above conditions for each of the items $i \in (\{h\} \cup S) \cap I^p$.

2.2.4.4 Branching Strategy

For VPPs, the lower bound $\lceil z_{LP}^{VPP} \rceil$ is often the optimal number of patterns. In fact, this holds for all instances that we solved. The main issue is therefore to find such an integer solution by reducing the fractionality of the master solution. Hence, we conduct a depth-first search strategy and first explore the branch with greater-or-equal constraint (2.9b).

For 01-VPP, we adopt Ryan-Foster branching forcing to pack items together with $I^0 = \emptyset$ and $I^1 = \{(i_1, 0), (i_2, 0) : i_1, i_2 \in I\}$ (cf. Example 2). The branching priority is to select the most fractional corresponding $\alpha(I^0, I^1)$ value. For B-VPP, our priority is first to keep the subtrees balanced choosing the cardinality $|I^0| + |I^1|$ as low as possible, second forcing to pack items (and not to not-pack items), i.e., prohibit $I^0 = \{(i, 0) : i \in I\}$ and $I^1 = \emptyset$, and third to take the most fractional $\alpha(I^0, I^1)$ value. For U-VPP, it may happen that patterns p with a coefficient $a_{ip} > q_i$ are part of a solution. In this case, we first forbid these patterns creating

a single branch. Afterwards, the branching priority of B-VPP is adopted.

2.3 Computational Results

The branch-and-price algorithms were implemented in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2013. The experiments were conducted on a standard PC with an Intel(R) Core(TM) i7-5930k clocked at 3.5 GHz and 64 GB of RAM, by allowing a single thread for each run. CPLEX 12.6.2 was used to solve the RMP in the column-generation algorithms and as a primal MIP-based heuristic solver called after the solution of each branch-and-bound node using the so far generated columns. In the latter case, computation times are limited to a maximum of 60 seconds per call. CPLEX's default values were kept for all parameters.

2.3.1 Instances

In order to compare our algorithms with the leading exact approaches (Brandão and Pedroso, 2016; Hu *et al.*, 2017) from the literature, we base the computational experiments on the same set of instances which are those of Caprara and Toth (2001) who generated 2-dimensional VPPs. Their instances are grouped into ten classes with 40 instances each, according to the weights generation scheme. The ten classes are further divided into four subclasses each with ten instances of identical size: In the first nine classes, the instances have 25, 50, 100 and 200 items, respectively. In the tenth class, the instances have 24, 51, 99 and 201 items, respectively. In total, Caprara and Toth (2001) generated 400 benchmark instances.

In addition, Brandão and Pedroso (2016) combined each subclass with ten 2-dimensional instances into a single 20-dimensional instance, yielding another 40 instances. These instances have not been considered in the article of Hu *et al.* (2017).

For all 440 instances, the demand of each item is almost always one. Neither Hu *et al.* (2017) nor Brandão and Pedroso (2016) aggregated the (very few) items with identical weights. Therefore, we also solve these instances only with the 01-VPP model. Results on these binary VPPs are presented in Section 2.3.3.

Because the demand of the instances is almost always one, we generated new instances with larger demands to compare the three models 01-VPP, B-VPP, and U-VPP. The item sizes of the new instances are the same as already used by Caprara and Toth (2001) and Brandão and Pedroso (2016), respectively, but the demand of the items is larger: We generated the larger demand using a discrete, uniformly distributed random variable between 1 and 100. Section 2.3.4

discusses our results for these instances. The new instances are available on `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

2.3.2 Computational Setup

The initial RMP is constructed as follows. First, we add patterns for single items, i.e., unit vectors and columns with only one non-negative coefficient $a_i^p = u_i$ (where $u_i = u_i^{01}$ or $= u_i^B$ or $= u_i^U$ depending on the VPP formulation). Second, additional patterns are constructed with the help of the well-known first-fit (FF) and best-fit (BF) heuristics (Johnson, 1973). Both FF and BF are once run with the items sorted decreasingly by their 1-norm. Moreover, another 99 runs of FF and BF are performed in which the items are sorted randomly.

To measure the impact of (D)DOIs, all instances were solved with and without adding (D)DOIs. Pretests have shown that the *mixed approach* described in Section 2.2.3 of adding some (D)DOIs to the initial RMP and adding the most violated (D)DOIs later on dynamically on average dominates the other strategies. Since the VPP instances typically do not allow SI and WSI with $|S| > 2$, our DOI selection and separation strategy is straightforward: For the initialization of the first RMP, two different kinds of static (D)DOIs are added. On the one hand, for each item i the PI $\pi_i \geq \pi_j$ with $j = \arg\min_{k \in I}\{||\mathbf{w}_i - \mathbf{w}_k||_1 : \mathbf{w}_i \geq \mathbf{w}_k\}$ is added. These (D)DOIs are the multi-dimensional extension of cuts of type 1 and ranking constraints, respectively (Valério de Carvalho, 2005; Ben Amor *et al.*, 2006). On the other hand, all SIs with $|S| = 2$ are added. There could exist $\mathcal{O}(m^3)$ such inequalities but for the instances of the benchmark, the number is rather small. These (D)DOIs are the multi-dimensional extension of cuts of type 2 (Valério de Carvalho, 2005). Dynamic DOIs are only added while solving the root node. Separation of violated (D)DOIs is done with a straightforward enumeration procedure considering all PIs by inspection. The DOIs remain in the RMP except when becoming invalid due to some active branching decisions, see Section 2.2.4.3.

2.3.3 Results for Binary Vector Packing Problems

In this section, we report results of our branch-and-price algorithms when applied to the 01-VPP. According to Section 2.2.2, we compare two versions of dynamic-programming labeling for the solution of the column-generation subproblems, one is a monodirectional labeling using a weak but fast dominance rule and the other is a more sophisticated bidirectional labeling using the strong dominance rule. As the latter leads to less labels but a more time-consuming dominance algorithm, it is not clear a priori which of the two labeling approaches is superior. Moreover, we conduct experiments with non-stabilized and stabilized branch-and-price algorithms. This gives rise to four variants of branch-and-price.

Comparison with Other Exact Algorithms. We compare our algorithms against the branch-and-price of Hu *et al.* (2017) and the graph compression and arc-flow model based approach of Brandão and Pedroso (2016). While Hu *et al.* (2017) restrict computation times to a maximum of 600 seconds per instance, Brandão and Pedroso (2016) did not impose any time limit. To improve comparability of results, we let our branch-and-price algorithms run for a maximum of 600 as well as 3600 seconds.

The results are summarized in Tables 2.2 and 2.3, where the first table reports average results for the 400 instances with 2-dimensional items/bins and the second table reports instance-wise results for the 40 instances of dimension 20. Note that Hu *et al.* (2017) did not consider the latter instances. Hence, we do not report results for a time limit of 600 seconds in Table 2.3. The table entries have the following meanings:

Class: class of instances; weights generated using different distributions;
Items: number of items in the instance;
opt: number of instances (out of 10) solved to proven optimality within 1 hour (3600 seconds);
$\mathbf{opt}_{10}$: same for a time limit of 10 minutes (600 seconds);
T: computation time in seconds for a single instance;
T_{LP}: computation time in seconds for solving the linear relaxation of the master program for a single instance;
$\bar{T}$: average computation time in seconds over 10 instances; unsolved instances are taken into account with the time limit of 1 hour (3600 seconds);
$\bar{T}_{10}$: same for a time limit of 10 minutes (600 seconds);
LB: lower bound obtained by linear relaxation or Lagrangian lower bound if master program is terminated prematurely.

For all approaches, the instances of the classes 2, 3, and 8 are well solvable and computation times always remain below 15 seconds. The approach of Brandão and Pedroso is the only one that solves all 88 instances of the classes 1 and 10 (80 of dimension $D = 2$ and 8 of dimension $D = 20$). Instances of the classes 6 and 7 seem to be hard for the branch-and-price algorithm of Hu *et al.* as they solve only 58 of the 80 2-dimensional instances. In contrast, all 88 instances of classes 6 and 7 are solved with the algorithm of Brandão and Pedroso as well as all of our branch-and-price algorithms.

The classes 4, 5, and 9 comprise the most difficult instances for all approaches. The algorithm of Brandão and Pedroso can only cope with smaller-sized instances with 25 items of classes 4 and 5. However, it solves all 30+3 = 33 instances of class 9 with 25, 50, and 100 items. On the latter instances of class 9, the branch-and-price of Hu *et al.* solves only the 20 instances with 25 and 50 items and our

Table 2.2: Results and comparison for 2-dimensional 01-VPP

Class		Hu et al.		Brandao/P.		Monodirectional with weak dominance						Bidirectional with strong dominance					
						Stabilized			Non-Stabilized			Stabilized			Non-Stabilized		
	Items	opt_{10}	$\bar{T}_{10}$	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$
1	25	10	1.5	10	0.1	10	10	0.7	10	10	0.3	10	10	0.1	10	10	0.1
1	50	9	89.6	10	1.6	10	10	7.4	10	10	9.6	10	10	1.1	10	10	1.1
1	100	3	529.0	10	67.0	8	10	408.1	5	10	918.4	10	10	130.4	7	10	403.1
1	200	0	600.0	10	7601.4	0	3	2899.4	0	0	3600.0	0	3	2798.5	0	0	3600.0
2	25	10	0.7	10	0.0	10	10	0.1	10	10	0.2	10	10	<0.1	10	10	<0.1
2	50	10	1.0	10	0.0	10	10	0.1	10	10	0.3	10	10	<0.1	10	10	0.1
2	100	10	2.8	10	0.2	10	10	0.1	10	10	0.2	10	10	0.1	10	10	<0.1
2	200	10	11.0	10	6.9	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
3	25	10	0.5	10	0.0	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
3	50	10	1.0	10	0.0	10	10	0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
3	100	10	2.5	10	0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
3	200	10	8.1	10	0.2	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
4	25	10	7.9	10	22.0	10	10	14.1	10	10	17.1	10	10	0.6	10	10	0.5
4	50	10	10.9	–	–	10	10	46.9	10	10	61.2	10	10	6.9	10	10	6.3
4	100	6	419.8	–	–	10	10	305.2	3	3	2633.4	3	10	646.3	2	3	2681.0
4	200	0	600.0	–	–	0	3	2973.4	0	0	3600.0	0	0	3600.0	0	0	3600.0
5	25	10	20.3	10	10.6	10	10	16.5	10	10	24.4	10	10	0.3	10	10	0.3
5	50	10	31.4	–	–	10	10	58.5	10	10	81.5	10	10	34.2	10	10	39.3
5	100	10	93.9	–	–	10	10	386.3	8	10	530.4	0	0	3600.0	0	0	3600.0
5	200	7	346.0	–	–	0	7	2566.5	0	5	3235.5	0	0	3600.0	0	0	3600.0

(Table 2 cont'd)

Class		Hu et al.		Brandao/P.		Monodirectional with weak dominance						Bidirectional with strong dominance					
						Stabilized			Non-Stabilized			Stabilized			Non-Stabilized		
	Items	opt_{10}	$\bar{T}_{10}$	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$	opt_{10}	opt	$\bar{T}$
6	25	10	0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	0.4
6	50	10	0.5	10	0.1	10	10	0.2	10	10	0.1	10	10	0.1	10	10	0.4
6	100	6	308.5	10	0.3	10	10	1.4	10	10	1.4	10	10	0.9	10	10	1.0
6	200	0	600.0	10	4.8	10	10	14.2	10	10	17.2	10	10	13.3	10	10	23.3
7	25	10	0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
7	50	10	10.2	10	0.1	10	10	0.2	10	10	0.2	10	10	0.1	10	10	0.2
7	100	10	64.8	10	1.7	10	10	1.8	10	10	2.1	10	10	1.4	10	10	1.5
7	200	2	481.6	10	14.0	10	10	23.8	10	10	34.6	10	10	34.8	10	10	37.9
8	25	10	<0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
8	50	10	0.1	10	<0.1	10	10	<0.1	10	10	0.1	10	10	<0.1	10	10	0.1
8	100	10	0.2	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1	10	10	0.1
8	200	10	0.7	10	0.2	10	10	0.3	10	10	0.3	10	10	0.3	10	10	0.3
9	25	10	1.1	10	0.1	10	10	0.2	10	10	0.2	10	10	<0.1	10	10	<0.1
9	50	10	9.9	10	0.7	10	10	7.3	10	10	8.7	10	10	2.2	10	10	2.0
9	100	0	600.0	10	28.1	10	10	359.2	10	10	436.6	7	10	728.4	6	10	794.8
9	200	0	600.0	–	–	0	0	3600.0	0	0	3600.0	0	0	3600.0	0	0	3600.0
10	24	10	<0.1	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1	10	10	<0.1
10	51	10	0.2	10	0.1	10	10	0.2	10	10	0.2	10	10	0.1	10	10	0.1
10	99	10	0.5	10	0.7	10	10	2.4	10	10	3.3	10	10	2.1	10	10	2.6
10	201	8	222.4	10	155.4	3	7	1675.2	1	9	1113.5	4	7	1493.8	0	9	1191.2
Total																	
	600s	321	142.0	320	127.9	351		112.8	337		128.6	334		121.0	325		127.7
	3600s			320	727.9		370	384.3		357	498.3		350	507.4		342	579.7
	>3600s			330													

Table 2.3: Results for 20-dimensional 01-VPP

Class			Monodirectional with weak dominance				Bidirectional with strong dominance				
		Brandao/P.	Stabilized		Non-Stabilized		Stabilized		Non-Stabilized		
	Items	T	T_{LP}	T	T_{LP}	T	T_{LP}	T	T_{LP}	T	LB
1	25	0.1	0.1	0.1	0.1	0.1	<0.1	<0.1	<0.1	<0.1	8
	50	0.8	0.6	0.9	0.7	0.8	0.5	0.6	0.6	1.1	15
	100	36.3	37.9	39.3	53.6	57.1	480.6	482.4	558.1	559.5	29
	200	1374.2	1927.6	2141.8	2055.9	2210.4	3600.0	3600.0	3600.0	3600.0	57
2	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	23
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	48
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	98
	200	<0.1	<0.1	<0.1	<0.1	<0.1	0.1	0.1	0.1	0.1	193
3	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	23
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	48
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	98
	200	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	193
4	25	50.3	130.2	130.2	123.5	123.5	24.7	24.7	24.6	24.6	4
	50	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	100	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	200	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
5	25	73.2	2020.4	2020.7	2005.4	2006.0	17.2	17.2	17.1	17.2	2
	50	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	100	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
	200	–	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	
6	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	13
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	26
	100	0.1	<0.1	0.3	<0.1	<0.1	<0.1	0.4	<0.1	<0.1	52
	200	0.2	0.1	0.2	0.1	0.6	0.1	0.1	0.1	0.5	102
7	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	13
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	26
	100	0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	52
	200	0.2	0.1	0.6	0.1	0.8	0.1	0.6	0.1	0.7	102
8	25	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	21
	50	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	37
	100	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	63
	200	0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	104
9	25	0.1	<0.1	<0.1	<0.1	<0.1	<0.1	0.1	<0.1	<0.1	8
	50	0.4	0.2	0.4	0.2	0.4	0.1	0.3	0.1	0.4	16
	100	12.8	6.0	6.9	5.8	12.1	19.9	24.1	24.3	25.1	31
	200	–	1285.0	3399.9	1045.0	2868.3	3600.0	3600.0	3600.0	3600.0	58
10	24	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	10
	51	0.1	<0.1	0.2	<0.1	0.1	<0.1	0.1	<0.1	0.1	21
	99	0.9	4.5	4.9	4.8	5.1	10.7	11.4	8.5	9.1	39
	201	14.5	277.2	278.7	212.9	221.4	3284.1	3285.4	2754.4	2767.8	77
Total		669.1	682.3	740.6	677.7	727.7	816.0	816.2	804.7	805.2	
	opt	33		34		34		32		32	

best variant (with stabilization and monodirectional labeling) solves 30+4 = 34 instances.

Comparing all algorithms on the basis of a reduced computation time of 600 seconds, the approaches of Hu *et al.* and Brandão and Pedroso show a comparable performance solving 321 and 320 instances (of dimension 2). Our four branch-and-price algorithms deliver within the same time between 325 and 351 optimal solutions. With the longer computation time of 3600 seconds the algorithm of Brandão and Pedroso gives 320+33 = 353 optimal solutions, while our four branch-and-price algorithms provide between 342+32=374 and 370+34=414 optimal solutions.

Clearly, combining the proposed branch-and-price approaches with strong heuristics can further improve their performance. Buljubašić and Vasquez (2016) and Vasquez and Buljubašić (2018) provide 390 and 400 optimal solutions, respectively, for the 400 2-dimensional instances using consistent neighborhood search. Using the upper bounds of these heuristics and Lagrangian lower bounds in our algorithms, we are able to prove optimality for all 400 2-dimensional instances during the solution of the linear relaxation (399 of the 400 within 3600 seconds; only the instance `5.200.04` takes 9877 seconds; this latter result was computed in a separate run of the branch-and-price algorithm). Together with the results of Vasquez and Buljubašić, Hu *et al.* and Brandão and Pedroso, 0+6=6 instances of the benchmark remain open.

Comparison of the Four Variants of Branch-and-Price. Tables 2.2 and 2.3 clearly indicate that the stabilization of the column-generation process using (D)DOIs is always beneficial: (average) computation times with stabilization are always below those of the corresponding variant without stabilization, with very few exceptions. In these exceptional cases, the additional time for the approach with stabilization is always below 1 second. On the positive side, the speedup obtained with stabilization can reach a factor of more than 4, see, e.g., class 4 instances with 100 items in dimension 2. It is also obvious that in dimension 20 the acceleration caused by stabilization is significantly smaller than in dimension 2. This can be explained by the fact that there exist fewer valid (D)DOIs, e.g., pair inequalities (PI), in dimension 20 than in dimension 2. Detailed statistics (number of branch-and-bound nodes, column-generation iterations, and initially/dynamically added (D)DOIs) for all four variants are provided in the Appendix. As a consequence of all these observations, we restrict the remainder of the comparison to the two variants that use stabilization, i.e., monodirectional with stabilization vs. bidirectional with stabilization.

There is no clear winner between the monodirectional and the bidirectional labeling approach, as can be seen, e.g., from class 1 with up to 100 items where

the bidirectional approach is faster compared to class 5 with 100 and 200 items where the monodirectional approach outperforms the bidirectional one. Focussing on instances with 25 (or 24) items only, one can see that the bidirectional approach is always superior to the monodirectional approach. This may be explained by the fact that in these instances, the bidirectional approach guarantees that there are never more than $2^{12} = 4096$ and $2^{13} = 8192$ forward and backward labels, respectively, to consider when reaching the half-way point where the merge of forward and backward labels is performed. Typically, as only a few items fit into a single bin, the number of labels is much smaller than this worst-case estimation so that combinatorial explosion is effectively suppressed. For instances with more items (≥ 50), the number of undominated labels in the bidirectional approach grows quickly. Here, the execution of the more complex dominance algorithm becomes predominant. Even using the multi-dimensional divide-and-conquer algorithm of Bentley (1980) for the dominance, which has an attractive worst-case complexity of $\mathcal{O}\left(n \log(n)^2\right)$ in the number n of labels for $D = 2$ (compared to quadratic with a straightforward pairwise dominance test), does not help here: The reduction in the number of labels through the dominance is eaten up by the time-consuming dominance algorithm. The general recommendation is therefore to use monodirectional labeling with the fast hash-table based weaker dominance Rule 1 for solving SPPRC subproblems whenever the number of items is larger.

Finally, we give some comments on the linear relaxation results that we obtained with our strongest branch-and-price variant, i.e, the one with stabilization and monodirectional labeling. For the 2-dimensional instances, there are only ten instances for which the column-generation process was not terminated before the time limit of 3600 seconds was reached (these are the instances `1.200.04`, `4.200.01`, `4.200.03`, `4.200.04`, `5.200.03`, `5.200.04`, `9.200.03`, `9.200.04`, `9.200.08`, and `9.200.09`; using the short notation `Class.Items.Instance`). For nine of these instances, however, the Lagrangian lower bound coincides with the best known upper bound that we did not find with our simple initialization heuristics. As mentioned before, the linear relaxation for the instance `5.200.04` takes 9877 seconds. All lower bounds are reported on our website `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

For the 20-dimensional instances, Lagrangian lower bounds are reported in Table 2.3 in column **LB**. Among the six unsolved instances, a lower bound is only available for the instance `4.50` (notation `Class.Items`).

Table 2.4: Results for VPP instances with larger demands.

Dimension			01-VPP		B-VPP		U-VPP	
	Stabilization		Monodir.	Bidir.	Monodir.	Bidir.	Monodir.	Bidir.
2	Stab.	opt	119	117	324	307	323	306
		$\bar{T}_{LP}$	1312.1	1312.4	64.6	344.1	219.5	485.9
		$\bar{T}$	2593.3	2611.7	863.2	1017.6	865.3	990.0
		$\bar{N}^{bb}$	11.5	12.3	5.9	4.7	5.2	4.4
	Non-Stab.	opt	113	111	294	290	291	290
		$\bar{T}_{LP}$	1785.9	1655.4	227.3	532.1	228.2	492.4
		$\bar{T}$	2629.3	2640.8	1117.8	1156.1	1142.2	1161.5
		$\bar{N}^{bb}$	9.1	10.7	8.0	5.1	7.9	5.1
20	Stab.	opt	22	21	30	30	30	30
		$\bar{T}_{LP}$	1294.9	1163.9	824.7	925.1	824.7	900.6
		$\bar{T}$	1704.4	1713.3	971.8	1110.1	971.7	1077.8
		$\bar{N}^{bb}$	3.6	5.8	2.3	1.7	2.3	1.7
	Non-Stab.	opt	21	21	30	30	30	30
		$\bar{T}_{LP}$	1484.8	1354.1	824.7	907.5	824.7	906.5
		$\bar{T}$	1712.8	1713.0	980.6	1127.0	980.6	1124.8
		$\bar{N}^{bb}$	2.8	4.1	1.9	1.5	1.9	1.6

2.3.4 Results for Vector Packing Problems with Larger Demands

We now compare the three models 01-VPP, B-VPP, and U-VPP and their solution with different branch-and-price algorithms. Table 2.4 summarizes the results in a very condensed way, numbers are aggregated over all instances of dimension 2 and all instances of dimension 20, respectively. The meaning of the row entries is analog to the column entries described at the beginning of this section. In addition, the rows labeled with $\bar{N}^{bb}$ give the average number of branch-and-bound nodes solved by the respective algorithm.

Table 2.4 shows that, for larger demands, the branch-and-price algorithms that are based on formulations B-VPP and U-VPP behave very similarly, and they clearly outperform the algorithms based on the binary formulation 01-VPP. This is not only true for the presented averages but for every single instance in the benchmark. The inferiority of the algorithms based on formulation 01-VPP can be attributed to two characteristics: First, for a larger demand, the 01-VPP model has $\sum_{i \in I} q_i$ constraints making the RMP a really huge IP with dense columns. Hence, RMP reoptimization is really time-consuming, cf. entries $\bar{T}_{LP}$ in Table 2.4. Second, the 01-VPP-based approaches suffer for the inherent symmetry when it comes to branching, see entries N^{bb} in the last row of Table 2.4. As a consequence, we

Table 2.5: Results for B-VPP instances with larger demands

		2-dimensional										20-dimensional						
Class		Brandao/P.		Monodir./weak dom.				Bidir./strong dom.				Brandao/P.	Monodir./weak dom.			Bidir./strong dom.		
	Items	opt	$\bar{T}$	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	T	T_{LP}	T	N^{bb}	T_{LP}	T	N^{bb}
1	25	10	0.2	10	0.4	2.3	1.0	10	0.2	2.0	1.0	0.2	0.3	2.2	1	0.1	2.4	1
	50	10	3.9	10	1.6	223.2	2.6	10	0.6	208.1	2.6	1.2	0.7	6.3	1	0.7	9.0	1
	100	9	2149.1	10	39.8	669.2	2.7	10	121.6	754.8	2.6	57.5	77.5	113.9	1	646.4	651.5	1
	200	0	3600.0	1	465.9	3311.2	34.4	1	729.6	3296.6	38.8	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
2	25	10	<0.1	10	0.2	0.8	1.0	10	0.2	0.7	1.0	<0.1	0.8	1.6	1	0.3	0.6	1
	50	10	<0.1	10	0.4	3.5	1.0	10	0.4	3.2	1.0	<0.1	0.7	1.5	1	0.8	1.5	1
	100	10	0.3	10	0.9	233.6	1.0	10	0.8	221.2	1.0	<0.1	1.3	1.3	1	1.6	1.6	1
	200	10	7.8	4	2.1	2165.9	1.0	4	2.1	2166.2	1.0	<0.1	2.8	2.8	1	4.0	4.0	1
3	25	10	<0.1	10	0.2	0.4	1.0	10	0.2	0.4	1.0	<0.1	0.3	0.8	1	0.3	0.6	1
	50	10	<0.1	10	0.3	1.9	1.0	10	0.4	1.8	1.0	<0.1	0.7	1.5	1	0.7	1.5	1
	100	10	<0.1	10	0.9	3.9	1.0	10	0.9	3.8	1.0	<0.1	1.3	1.3	1	1.7	1.7	1
	200	10	0.1	9	2.4	367.7	1.0	9	2.3	367.3	1.0	<0.1	2.8	2.8	1	4.2	4.2	1
4	25	6	2175.0	10	8.6	21.0	1.0	10	14.7	28.1	1.0	736.0	124.7	125.2	1	2618.9	2619.1	1
	50	0	3600.0	10	24.1	207.5	3.1	10	35.3	256.0	3.1	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
	100	0	3600.0	2	76.1	2972.0	28.2	2	359.7	3013.1	17.1	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
	200	0	3600.0	0	352.5	3600.0	19.3	0	942.4	3600.0	3.0	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
5	25	0	3600.0	9	32.6	431.5	6.3	7	1140.9	1562.9	1.3	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
	50	0	3600.0	10	62.6	361.8	3.3	0	2602.8	3600.0	0.9	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
	100	0	3600.0	4	161.2	2511.8	13.5	0	3350.0	3600.0	0.1	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
	200	0	3600.0	1	635.8	3346.4	11.1	0	3600.0	3600.0	0.0	3600.0	3600.0	3600.0	0	3600.0	3600.0	0
6	25	10	<0.1	10	0.1	1.5	1.0	10	0.1	1.3	1.0	<0.1	0.3	0.3	1	0.2	0.2	1
	50	10	<0.1	10	0.4	120.2	1.3	10	0.3	141.7	1.3	<0.1	0.5	0.5	1	0.5	0.5	1
	100	10	0.3	10	0.9	664.8	1.4	10	0.8	768.4	1.4	0.1	0.9	0.9	1	1.1	1.1	1
	200	10	2.7	10	94.7	137.0	1.2	10	86.3	126.6	1.3	0.4	1.6	1.6	1	2.4	2.4	1
7	25	10	<0.1	10	0.1	2.1	1.0	10	0.1	1.9	1.0	<0.1	0.3	0.3	1	0.2	0.2	1
	50	10	0.2	10	1.2	142.9	1.5	10	1.1	149.8	1.5	<0.1	0.5	0.5	1	0.5	0.5	1
	100	10	1.1	10	12.4	711.0	1.5	10	11.4	933.6	1.6	0.1	0.9	0.9	1	1.1	1.1	1
	200	10	9.1	10	73.0	83.7	1.0	10	68.2	80.3	1.0	0.4	1.7	1.7	1	2.1	2.1	1
8	25	10	<0.1	10	<0.1	10.5	1.0	10	<0.1	8.8	1.0	<0.1	0.3	0.3	1	0.3	0.3	1
	50	10	<0.1	10	0.4	4.5	1.0	10	0.4	4.0	1.0	<0.1	0.6	0.6	1	0.6	0.6	1
	100	10	<0.1	10	<0.1	501.4	1.0	10	<0.1	605.9	1.0	<0.1	1.0	1.0	1	1.2	1.2	1
	200	10	0.2	3	<0.1	2523.8	1.0	3	<0.1	2523.1	1.0	0.2	1.7	1.7	1	2.4	2.4	1
9	25	10	0.3	10	0.4	1.9	1.0	10	0.3	1.6	1.0	0.1	0.3	1.7	1	0.1	1.7	1
	50	10	1.7	10	4.0	39.6	1.2	10	1.9	21.8	1.2	0.4	0.5	5.9	1	0.4	9.5	1
	100	8	1212.3	10	119.2	448.0	1.7	10	103.7	341.7	1.7	9.6	3.0	3600.0	22	15.7	3600.0	7
	200	0	3600.0	0	404.0	3600.0	42.7	0	577.0	3600.0	47.4	1226.5	3600.0	3600.0	0	3600.0	3600.0	0
10	24	10	<0.1	10	0.1	5.7	1.0	10	0.1	6.4	1.0	<0.1	<0.1	1.5	1	<0.1	1.3	1
	51	10	0.1	10	0.3	198.4	1.7	10	0.3	195.1	1.7	0.1	0.4	3.0	1	0.4	4.3	1
	99	10	1.8	10	1.7	1651.8	2.7	10	1.7	1662.5	2.7	1.1	0.8	907.1	2	1.0	1482.2	2
	201	10	230.0	1	4.0	3243.0	36.7	1	3.8	3243.0	36.9	21.8	359.0	1680.6	39	1293.5	3593.8	31
Total			954.9		64.6	863.2	5.9		344.1	1017.6	4.7	771.4	824.7	971.8	2.3	925.1	1110.1	1.7
	opt	303		324				307				32		30			30	

discuss and present more detailed results only for our branch-and-price algorithms based on formulation B-VPP in the following.

Comparing the four variants of branch-and-price using model B-VPP, i.e., results presented in the middle block of Table 2.4, we see a similar behavior as already observed for 01-VPP: Stabilization of the column-generation process is clearly beneficial and there is no clear winner between monodirectional and bidirectional labeling for solving the MKP subproblems using the SPPRC model.

Finally, Table 2.5 provides detailed results for B-VPP and the instances with larger demand. In order to have an alternative algorithm to compare with, we downloaded the code (VPSolver) of the graph compression procedure and the arc-flow formulation of Brandão and Pedroso (2016) in version 3.1.2 and compiled it using a Virtual Machine (Oracle Virtual Box) under Ubuntu 16.04 LTS using CPLEX 12.7.1.0. Columns headed with *Brandao/P.* in Table 2.5 refer to results obtained with their implementation run on our PC (single thread). Overall, Table 2.5 does not provide surprising new insights. The general behavior is similar to the 01-VPP case. Regarding the comparison of the monodirectional and bidirectional approaches, the results for the classes 4 and 5 with 25 and 50 items in dimension 2 differ from those presented in Table 2.2. In the B-VPP case, the bidirectional labeling is not longer superior. The reason is that in the B-MKP subproblem and the associated SPPRC digraph, the number of arcs per stage is $u_i^B + 1$. For example, this number of arcs is on average 8 for class 4 and 16 for class 5. Already for instances with 25 items, the bidirectional labeling is no longer able to effectively limit the number of labels generated up to the half-way point. The dominance algorithm based on the stronger Rule 2 becomes too slow. In comparison to the algorithm of Brandão and Pedroso (2016), the arc-flow formulation delivers 303+32=335 optimal solutions on the 400+40=440 instances of dimension 2 and 20, respectively, while our best-performing branch-and-price, i.e., the one with monodirectional labeling and weak dominance, delivers 324+30=354 optima.

In total, the four branch-and-price algorithms solve 328+30=358 of the 400+40=440 instances with larger demands to proven optimality.

2.4 Conclusions

In this paper, we proposed branch-and-price algorithms for the solution of different formulations of the VPP. To the best of our knowledge, this is the first branch-and-price approach that also tackles non-binary formulations of the VPP, namely the B-VPP and U-VPP variants. To test the impact of using these formulations, we introduced new benchmark instances with demands $\gg 1$ that were generated based on the standard benchmark sets for the VPP from the literature (Caprara and Toth, 2001; Brandão and Pedroso, 2016). Moreover, the presented approach

constitutes the first branch-and-price for VPPs with more than two dimensions.

Our column-generation subproblems are variants of the multidimensional knapsack problem, either binary, bounded, or unbounded for 01-VPP, B-VPP, or U-VPP, respectively. In order to obtain a generic approach, the subproblems were formulated as SPPRCs. Two different labeling algorithms (one monodirectional, one bidirectional) with different dominance rules were derived for their solution. Additional dual inequalities (not necessarily (deep) dual optimal) are added to stabilize the column-generation process. This is the first time that such dual inequalities are used in a branch-and-price for the VPP. For specific families of dual inequalities we showed that they are (deep) dual optimal inequalities for some of the formulations. The dual inequalities are used in a combined static/dynamic fashion, where the dynamic generation of violated dual inequalities is a by-product of solving the SPPRC representation of the subproblem with our labeling methods. Even more, our solution approach for the subproblem is also compatible with our branching scheme (the branching scheme of Vanderbeck, 1999, which is equal to the Ryan-Foster rule in the 01-VPP case) meaning that branching decisions can be implemented into it without deteriorating the resolution process. In order to use the dual inequalities also in the branch-and-bound tree, different rules for their validity with respect to the branching decisions taken had to be derived.

In an extensive computational study, we demonstrated that our branch-and-price algorithms are competitive with respect to the state of the art. For the 2-dimensional benchmark set, our strongest algorithm provides 368 provably optimal solutions compared to the 330 and 321 optima found by Brandão and Pedroso (2016) and Hu *et al.* (2017), respectively. Of the 20-dimensional instances, we can solve 34 instances vs. 33 optima computed by Brandão and Pedroso (2016). With all branch-and-price variants, we solve to proven optimality 404 out of the 440 standard benchmark instances. Combining our algorithms with strong heuristics can further improve their performance: Using the results of Vasquez and Buljubašić (2018) as upper bounds, we are able to prove optimality for 434 instances. Together with the results of Vasquez and Buljubašić (2018), 6 20-dimensional instances remain open.

Finally, our computational results indicate the usefulness of all the different algorithmic parts: (i) the stabilized algorithms are clearly superior to their non-stabilized counterparts, (ii) neither of the labeling algorithms for the subproblems completely dominates the other, and (iii) for instances with larger demands one has to resort to the non-binary formulations B-VPP and U-VPP, since the 01-VPP is too large and suffers from severe symmetry issues.

For the future, it might be interesting to combine graph reduction techniques such as those invented by Brandão and Pedroso (2016) with VPP-specific exact approaches based on column generation. In addition, strong valid inequalities

might help to strengthen the linear relaxation and to reduce overall computation times.

Acknowledgment

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/6-1.

Bibliography

Alves, C. and Valério de Carvalho, J. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, **35**(4), 1315–1328.

Alves, C., Valério de Carvalho, J., Clautiaux, F., and Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, **233**(1), 43–63.

Ben Amor, H. and Valério de Carvalho, J. (2005). Cutting stock problems. In Desaulniers *et al.* (2005), chapter 5, pages 131–161.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, **23**(4), 214–229.

Brandão, F. and Pedroso, J. P. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers & Operations Research*, **69**, 56–67.

Buljubašić, M. and Vasquez, M. (2016). Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, **76**, 12–21.

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Dell'Amico, M., Díaz-Díaz, J. C., Iori, M., and Rizzi, R. (2014). Friendly bin packing instances without integer round-up property. *Mathematical Programming*, **150**(1), 5–17.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, **44**(2), 145–159.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Hu, Q., Zhu, W., Qin, H., and Lim, A. (2017). A branch-and-price algorithm for the two-dimensional vector packing problem with piecewise linear cost function. *European Journal of Operational Research*, **260**(1), 70–80.

Huang, E. and Korf, R. E. (2012). Optimal rectangle packing: An absolute placement approach. *Journal of Artificial Intelligence Research*, **46**, 47–87.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

Johnson, D. S. (1973). *Near-Optimal Bin Packing Algorithms*. Ph.D. thesis, Massachusetts Institute of Technology.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, **48**(2), 256–267.

Monaci, M. and Toth, P. (2006). A Set-Covering-Based heuristic approach for Bin-Packing problems. *INFORMS Journal on Computing*, **18**(1), 71–85.

Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. (2011). Heuristics for vector bin packing. Technical report, Microsoft Research.

Rietz, J. (2003). *Untersuchungen zu MIRUP für Vektorpackprobleme*. Dissertation, Faculty of Mathematics und Computer Science, Technischen Universität Bergakademie Freiberg, Germany.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, chapter 17, pages 269–280. Elsevier, North-Holland.

Scheithauer, G. and Terno, J. (1995). The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, **84**, 562–571.

Spieksma, F. C. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers & operations research*, **21**(1), 19–25.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Vasquez, M. and Buljubašić, M. (2018). Consistent Neighbourhood Search for Two-Dimensional Vector Packing. Presented at ROADEF conference, Lorient, France.

Appendix

In Tables 2.6 to 2.9, we present detailed results for the 2- and 20-dimensional 01-VPP instances from Caprara and Toth (2001) and Brandão and Pedroso (2016), respectively. The column entries have the following meaning:

Class:	class of instances; weights generated using different distributions;
Items:	number of items in the instance;
opt:	number of instances (out of 10) solved to proven optimality within 1 hour (3600 seconds);
T**:**	computation time in seconds for a single instance;
$\bar{T}$**:**	average computation time in seconds over 10 instances; unsolved instances are taken into account with the time limit of 1 hour (3600 seconds);
T_{LP}**:**	computation time in seconds for solving the linear relaxation of the master program for a single instance;
$\bar{T}_{LP}$**:**	average computation time in seconds for solving the linear relaxation of the master program for a single instance; unsolved linear relaxations are taken into account with the time limit of 1 hour (3600 seconds);
N^{bb}**:**	number of solved branch-and-bound nodes;
$\bar{N}^{bb}$**:**	average number of solved branch-and-bound nodes;
#**CG:**	(average) number of column-generation iterations;
SP/RMP:	(average) ratio of the times spent for solving the subproblem and re-optimizing the RMP; only stated if the solution time is greater than 1 second.
#**static:**	(average) number of static DOIs that are generated at the beginning;
#**dynamic:**	number of DOIs that are generated dynamically during the column-generation process.

Table 2.6: Results for 2-dimensional 01-VPP solved monodirectional with weak dominance

Class		Stabilized								Non-Stabilized					
	Items	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	#CG	SP/RMP	#static	#dynamic	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	#CG	SP/RMP
1	25	10	0.5	0.7	1.0	12.9	1.8	22.4	0.4	10	0.3	0.3	1.0	13.2	
	50	10	7.2	7.4	1.0	21.0	77.8	49.8	1.7	10	9.4	9.6	1.0	21.8	411.8
	100	10	157.0	408.1	9.0	82.7	1669.8	102.5	1.8	10	170.4	918.4	22.9	167.2	1755.5
	200	3	1799.0	2899.4	8.5	112.6	2042.5	258.6	3.3	0	1854.8	3600.0	13.5	134.6	1957.5
2	25	10	0.0	0.1	1.0	1.0		22.1	0.0	10	0.2	0.2	1.0	1.0	
	50	10	0.0	0.1	1.0	1.0		49.0	0.0	10	0.2	0.3	1.0	1.0	
	100	10	0.0	0.1	1.0	1.0		109.7	0.0	10	0.1	0.2	1.0	1.0	
	200	10	0.0	0.1	1.0	1.2		312.3	0.0	10	0.0	0.1	1.0	1.1	
3	25	10	0.0	0.0	1.0	1.0		22.1	0.0	10	0.0	0.0	1.0	1.0	
	50	10	0.0	0.1	1.0	1.0		48.5	0.0	10	0.0	0.0	1.0	1.0	
	100	10	0.0	0.0	1.0	1.0		101.0	0.0	10	0.0	0.0	1.0	1.0	
	200	10	0.0	0.1	1.0	1.0		246.7	0.0	10	0.0	0.1	1.0	1.1	
4	25	10	14.1	14.1	1.0	11.1	968.8	22.6	1.4	10	17.1	17.1	1.0	11.2	710.0
	50	10	46.9	46.9	1.0	7.0	1886.2	49.0	2.8	10	61.2	61.2	1.0	7.1	3959.5
	100	10	300.4	305.2	1.0	11.5	4165.6	107.3	3.5	3	377.8	2633.4	17.3	50.3	5470.2
	200	3	2436.1	2973.4	3.3	41.7	3199.3	278.2	6.6	0	2699.3	3600.0	4.8	43.1	2904.5
5	25	10	16.5	16.5	1.0	8.6	1111.2	22.5	2.0	10	24.4	24.4	1.0	10.4	1548.0
	50	10	58.5	58.5	1.0	6.4	3143.1	50.2	3.7	10	81.5	81.5	1.0	6.8	4508.6
	100	10	386.3	386.3	1.0	11.6	4318.5	109.5	12.0	10	530.4	530.4	1.0	13.0	4282.4
	200	7	2301.5	2566.5	1.6	29.3	2765.3	299.0	20.4	5	3031.5	3235.5	1.2	30.7	2323.1
6	25	10	0.0	0.0	1.0	1.5		23.5	0.0	10	0.0	0.0	1.0	1.4	
	50	10	0.0	0.2	1.0	4.2		49.9	9.0	10	0.0	0.1	1.0	4.5	
	100	10	0.8	1.4	1.0	22.0	21.6	118.2	22.5	10	1.0	1.4	1.0	22.0	31.4
	200	10	11.8	14.2	1.0	79.9	32.0	352.3	54.1	10	13.9	17.2	1.0	80.9	40.2
7	25	10	0.0	0.0	1.0	1.8		28.0	0.8	10	0.0	0.0	1.0	1.9	
	50	10	0.1	0.2	1.0	6.0		74.0	3.1	10	0.1	0.2	1.0	6.5	
	100	10	1.3	1.8	1.0	30.4	19.3	296.7	33.4	10	1.7	2.1	1.0	34.0	28.5
	200	10	17.8	23.8	1.0	123.6	26.0	190.0	35.6	10	24.4	34.6	1.0	145.4	36.3
8	25	10	0.0	0.0	1.0	1.0		10.7	0.0	10	0.0	0.0	1.0	1.0	
	50	10	0.0	0.0	1.0	1.0		27.5	0.0	10	0.0	0.1	1.0	1.0	
	100	10	0.0	0.1	1.0	1.0		67.9	0.0	10	0.0	0.1	1.0	1.0	
	200	10	0.0	0.3	1.0	1.0		143.0	0.0	10	0.0	0.3	1.0	1.0	
9	25	10	0.2	0.2	1.0	11.0		22.1	2.1	10	0.2	0.2	1.0	11.2	
	50	10	7.3	7.3	1.0	40.7	228.1	49.5	0.3	10	8.6	8.7	1.0	39.1	287.4
	100	10	357.8	359.2	1.0	110.2	1781.4	102.5	0.1	10	428.9	436.6	1.0	114.8	2078.7
	200	0	3332.2	3600.0	1.8	189.8	2355.5	258.6	1.2	0	3571.3	3600.0	0.2	161.2	3028.0
10	24	10	0.0	0.0	1.0	2.5		22.6	0.5	10	0.0	0.0	1.0	3.1	
	51	10	0.2	0.2	1.0	15.4		49.9	8.0	10	0.2	0.2	1.0	16.5	
	99	10	2.4	2.4	1.0	68.8	15.3	106.1	3.8	10	3.3	3.3	1.0	79.1	24.0
	201	7	8.8	1675.2	98.9	470.1	23.3	297.5	23.1	9	12.0	1113.5	95.0	441.6	26.1
Total		370	281.6	384.3	3.9	38.7	1421.5	114.3	6.4	357	323.1	498.3	4.7	42.1	1770.6

Table 2.7: Results for 2-dimensional 01-VPP solved bidirectional with strong dominance

Class		Stabilized								Non-Stabilized					
	Items	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	#CG	SP/RMP	#static	#dynamic	opt	$\bar{T}_{LP}$	$\bar{T}$	$\bar{N}^{bb}$	#CG	SP/RMP
1	25	10	0.1	0.1	1.0	13.5		22.4	0.4	10	0.1	0.1	1.0	13.5	
	50	10	1.0	1.1	1.0	21.9	41.8	49.8	1.6	10	0.9	1.1	1.0	21.7	40.2
	100	10	44.8	130.4	14.7	117.3	451.7	102.5	2.8	10	45.4	403.1	19.8	164.0	360.7
	200	3	2162.6	2798.5	6.2	74.0	3372.0	258.6	3.2	0	2127.0	3600.0	14.4	107.7	3419.2
2	25	10	0.0	0.0	1.0	1.0		22.1	0.0	10	0.0	0.0	1.0	1.0	
	50	10	0.0	0.0	1.0	1.0		49.0	0.0	10	0.0	0.1	1.0	1.0	
	100	10	0.0	0.1	1.0	1.0		109.7	0.0	10	0.0	0.0	1.0	1.0	
	200	10	0.0	0.1	1.0	1.2		312.3	0.0	10	0.0	0.1	1.0	1.1	
3	25	10	0.0	0.0	1.0	1.0		22.1	0.0	10	0.0	0.0	1.0	1.0	
	50	10	0.0	0.0	1.0	1.0		48.5	0.0	10	0.0	0.0	1.0	1.0	
	100	10	0.0	0.0	1.0	1.0		101.0	0.0	10	0.0	0.0	1.0	1.0	
	200	10	0.0	0.1	1.0	1.0		246.7	0.0	10	0.0	0.1	1.0	1.1	
4	25	10	0.6	0.6	1.0	10.6		22.6	1.3	10	0.5	0.5	1.0	10.7	265.8
	50	10	6.9	6.9	1.0	6.8	496.7	49.0	2.5	10	6.3	6.3	1.0	6.6	371.4
	100	10	641.4	646.3	1.0	9.2	10982.0	107.3	3.1	3	611.5	2681.0	26.1	85.2	3608.0
	200	0	3600.0	3600.0	0.0	2.8	18497.9	278.2	2.5	0	3600.0	3600.0	0.0	3.6	13571.3
5	25	10	0.3	0.3	1.0	8.3		22.5	1.8	10	0.3	0.3	1.0	9.7	206.4
	50	10	34.2	34.2	1.0	6.5	1857.7	50.2	3.9	10	39.3	39.3	1.0	7.1	2172.6
	100	0	3600.0	3600.0	0.0	4.3	75601.8	109.5	6.4	0	3600.0	3600.0	0.0	5.6	60196.0
	200	0	3600.0	3600.0	0.0	0.2	1774.3	28.5	1.1	0	3600.0	3600.0	0.0	0.9	20384.8
6	25	10	0.0	0.0	1.0	1.5		23.5	0.0	10	0.2	0.4	1.0	1.5	
	50	10	0.0	0.1	1.0	4.6		49.9	9.8	10	0.1	0.4	1.0	4.7	
	100	10	0.4	0.9	1.0	19.4		118.2	17.9	10	0.4	1.0	1.0	22.2	14.4
	200	10	9.2	13.3	1.0	76.4	27.0	352.3	43.7	10	9.2	23.3	1.1	79.7	28.2
7	25	10	0.0	0.0	1.0	1.9		28.0	0.8	10	0.0	0.0	1.0	2.0	
	50	10	0.0	0.1	1.0	4.9		74.0	2.7	10	0.1	0.2	1.0	5.7	
	100	10	0.9	1.4	1.0	28.0	37.6	296.7	29.5	10	1.0	1.5	1.0	32.5	26.5
	200	10	27.5	34.8	1.0	84.5	59.8	190.0	25.4	10	31.1	37.9	1.0	98.5	67.7
8	25	10	0.0	0.0	1.0	1.0		10.7	0.0	10	0.0	0.0	1.0	1.0	
	50	10	0.0	0.0	1.0	1.0		27.5	0.0	10	0.0	0.1	1.0	1.0	
	100	10	0.0	0.1	1.0	1.0		67.9	0.0	10	0.0	0.1	1.0	1.0	
	200	10	0.0	0.3	1.0	1.0		143.0	0.0	10	0.0	0.3	1.0	1.0	
9	25	10	0.0	0.0	1.0	11.0		22.1	1.9	10	0.0	0.0	1.0	10.9	
	50	10	2.1	2.2	1.0	42.4	73.7	49.5	0.3	10	1.9	2.0	1.0	38.4	72.6
	100	10	727.0	728.4	1.0	110.2	4135.3	102.5	0.2	10	758.2	794.8	1.1	111.8	3859.9
	200	0	3600.0	3600.0	0.0	60.8	6094.2	258.6	1.1	0	3600.0	3600.0	0.0	32.4	2946.3
10	24	10	0.0	0.0	1.0	2.7		22.6	0.5	10	0.0	0.0	1.0	3.3	
	51	10	0.1	0.1	1.0	13.4		49.9	3.2	10	0.1	0.1	1.0	14.7	
	99	10	2.1	2.1	1.0	53.6	19.4	106.1	5.0	10	2.6	2.6	1.0	63.2	22.2
	201	7	12.9	1493.8	108.3	422.9	19.4	297.5	14.5	9	15.5	1191.2	126.2	515.8	20.4
Total		350	451.9	507.4	4.1	30.6	7267.2	107.6	4.7	342	451.3	579.7	5.5	37.1	5582.7

Table 2.8: Results for 20-dimensional 01-VPP solved monodirectional with weak dominance

Class	Items	Stabilized							Non-Stabilized				
		T_{LP}	T	N^{bb}	#RMP	SP/RMP	#static	#dynamic	T_{LP}	T	N^{bb}	#RMP	SP/RMP
1	25	0.1	0.1	1	6		0	0	0.1	0.1	1	6	
	50	0.6	0.9	1	6		2	0	0.7	0.8	1	6	
	100	37.9	39.3	1	24	789.0	5	0	53.6	57.1	1	28	1139.93617
	200	1927.6	2141.8	3	74	4513.6	10	0	2055.9	2210.4	2	77	3914.471243
2	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	200	0.0	0.0	1	1		10	0	0.0	0.0	1	1	
3	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		5	0	0.0	0.0	1	1	
	200	0.0	0.0	1	1		10	0	0.0	0.0	1	1	
4	25	130.2	130.2	1	38	4199.5	0	0	123.5	123.5	1	38	7714.625
	50	3600.0	3600.0	0	4	240000.0	3	0	3600.0	3600.0	0	1	
	100	3600.0	3600.0	0	1		5	0	3600.0	3600.0	0	1	
	200	3600.0	3600.0	0	1		11	0	3600.0	3600.0	0	1	
5	25	2810.2	2810.3	1	14	3581.2	0	0	2005.4	2006.0	1	14	8794.429825
	50	3600.0	3600.0	0	1		3	0	3600.0	3600.0	0	1	
	100	3600.0	3600.0	0	1		5	0	3600.0	3600.0	0	1	
	200	3600.0	3600.0	0	1		2	0	3600.0	3600.0	0	1	
6	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.3	1	1		5	0	0.0	0.0	1	1	
	200	0.1	0.2	1	2		10	0	0.1	0.6	1	2	
7	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		8	0	0.0	0.0	1	1	
	200	0.1	0.6	1	1		29	0	0.1	0.8	1	1	
8	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	200	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
9	25	0.0	0.0	1	2		0	0	0.0	0.0	1	2	
	50	0.2	0.4	1	3		3	0	0.2	0.4	1	3	
	100	6.0	6.9	1	4	186.5	5	0	5.8	12.1	1	4	362.3125
	200	1285.0	3399.9	16	107	3527.8	10	0	1045.0	2868.3	11	94	4637.883817
10	24	0.0	0.0	1	1		18	0	0.0	0.0	1	1	
	51	0.0	0.2	1	2		37	0	0.0	0.1	1	2	
	99	4.5	4.9	1	31	55.6	75	0	4.8	5.1	1	31	147.4375
	201	277.2	278.7	1	88	680.0	163	0	212.9	221.4	1	62	546.2287918
Total		702.0	760.4	1.3	10.8	28614.8	11.2	0.0	677.7	727.7	1.1	9.9	3407.2
	opt							34					34

Table 2.9: Results for 20-dimensional 01-VPP solved bidirectional with strong dominance

Class		Stabilized							Non-Stabilized				
	Items	T_{LP}	T	N^{bb}	#RMP	SP/RMP	#static	#dynamic	T_{LP}	T	N^{bb}	#RMP	SP/RMP
1	25	0.0	0.0	1	5		0	0	0.0	0.0	1	5	
	50	0.5	0.6	1	4		2	0	0.6	1.1	1	6	20.3
	100	480.6	482.4	1	24	7508.1	5	0	558.1	559.5	1	27	8999.8
	200	3600.0	3600.0	0	2	38297.9	10	0	3600.0	3600.0	0	0	
2	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	200	0.1	0.1	1	1		10	0	0.1	0.1	1	1	
3	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		5	0	0.0	0.0	1	1	
	200	0.0	0.0	1	1		10	0	0.0	0.0	1	1	
4	25	24.7	24.7	1	42	1540.4	0	0	24.6	24.6	1	42	1638.7
	50	3600.0	3600.0	0	1		3	0	3600.0	3600.0	0	0	
	100	3600.0	3600.0	0	1		5	0	3600.0	3600.0	0	1	
	200	3600.0	3600.0	0	1		11	0	3600.0	3600.0	0	1	
5	25	17.2	17.2	1	14	571.5	0	0	17.1	17.2	1	14	2855.0
	50	3600.0	3600.0	0	1		3	0	3600.0	3600.0	0	1	
	100	3600.0	3600.0	0	1		5	0	3600.0	3600.0	0	1	
	200	3600.0	3600.0	0	1		2	0	3600.0	3600.0	0	1	
6	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.4	1	1		5	0	0.0	0.0	1	1	
	200	0.1	0.1	1	2		10	0	0.1	0.5	1	2	
7	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		3	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		8	0	0.0	0.0	1	1	
	200	0.1	0.6	1	1		29	0	0.1	0.7	1	1	
8	25	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	50	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	100	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
	200	0.0	0.0	1	1		0	0	0.0	0.0	1	1	
9	25	0.0	0.1	1	2		0	0	0.0	0.0	1	2	
	50	0.1	0.3	1	3		3	0	0.1	0.4	1	3	
	100	19.9	24.1	1	4	1416.4	5	0	24.3	25.1	1	4	1426.9
	200	3600.0	3600.0	0	2	51428.6	10	0	3600.0	3600.0	0	1	85714.3
10	24	0.0	0.0	1	1		18	0	0.0	0.0	1	1	
	51	0.0	0.1	1	2		37	0	0.0	0.1	1	2	
	99	10.7	11.4	1	35	212.2	75	0	8.5	9.1	1	26	166.1
	201	3284.1	3285.4	1	91	7726.3	163	0	2754.4	2767.8	1	77	6464.6
Total		816.0	816.2	0.8	6.5	13587.7	11.2	0.0	804.7	805.2	0.8	5.9	13410.7
	opt							32					32

Chapter 3

Lexicographic Bin-Packing Optimization for Loading Trucks in a Direct-Shipping System

Katrin Heßler, Stefan Irnich, Tobias Kreiter, Ulrich Pferschy

Abstract

We consider a packing problem that arises in a direct-shipping system in the food and beverage industry: Trucks are the containers and products to be distributed are the items. The packing is constrained by two independent quantities, weight (e.g., measured in kg) and volume (number of pallets). Additionally, the products are grouped into the three categories standard, cooled, and frozen (the latter two require refrigerated trucks). Products of different categories can be transported in one truck using separated zones, but the cost of a truck depends on the transported product categories. Moreover, product splitting should be avoided so that (un-)loading is simplified. As a result, we seek for a feasible packing optimizing the following objective functions in a strictly lexicographic sense: minimize the (1) total number of trucks; (2) number of refrigerated trucks; (3) number of refrigerated trucks which contain frozen products; (4) number of refrigerated trucks which also transport standard products; (5) and minimize product splitting. This is a real-world application of a bin-packing problem with cardinality constraints a.k.a. the two-dimensional vector packing problem, with additional constraints. We provide a heuristic and an exact solution approach. The heuristic meta-scheme considers the multi-compartment and item-fragmentation features of the problem and applies various problem-specific heuristics. The exact solution algorithm covering all five stages is based on branch-and-price using stabilization techniques exploiting dual-optimal inequalities. Computational results on real-world and difficult self-generated instances prove the applicability of our approach.

3.1 Introduction

In this paper, we present a system of bin-packing problems that arise in a direct-shipping system in the food and beverage industry. More than 1.2 million units of different products leave the factory of our industry partner every day. By far the largest share of the products is transported by trucks from the factory to the distribution centers of supermarket chains (in the following denoted as warehouses). As shipping is done directly from the factory to individual warehouses with full-truck volumes, the routing of trucks plays no role. For each warehouse, the use of the utilized trucks has to be optimized by assigning shipments to trucks. Hence, the overall problem naturally decomposes by warehouse.

The transportation of products is standardized and uses euro-pallets (which are typically packed with a set of uniform boxes of a product). In our application, quantities are large so that pallets are not mixed, i.e., each pallet contains only one product. Moreover, it is legitimate to assume that products are equally distributed on pallets such that each pallet loaded with a certain product has the same weight. Let $K = \{1, 2, \ldots, m\}$ denote the set of products (to be shipped to a particular warehouse on a specific day). As the range of food products includes a huge variety of commodities, e.g., fruit juices, spirits, baked goods, jams, wafers, and ketchup, the weight and space requirements for the transport differ substantially by product. For each product $k \in K$, these requirements are therefore described by two *attributes*:

- the *unit weight* ρ_k (in *kg/pallet*) of a pallet loaded with product k and
- the *number* n_k of pallets that need to be shipped.

In addition, all products can be uniquely grouped into three *categories*:

- *Standard products* I^S require no cooling;
- *Cooled products* I^C require cooling;
- Shipping of *frozen products* I^F requires deep cooling and is the most costly transport.

As a result, the set of products is partitioned into $I = I^S \cup I^C \cup I^F$.

The fleet consists of *standard trucks* and *refrigerated trucks*, i.e., trucks without and with a cooling system. Frozen and cooled products must be transported in refrigerated trucks, which are more costly than standard trucks. Frozen and cooled products can be mixed (using different zones), but costs are higher if frozen products are transported. Also, standard products can be transported by refrigerated trucks (again using separated zones). There are no conflicts between products, i.e., any subset of products can be packed together into a truck. All trucks are homogeneous regarding capacity. Let W be the *capacity* (in *kg*) for the physical weight and B be the *capacity* for the number of pallets that fit into one truck (usually $B = 33$ *pallets*).

Moreover, depending on the customer or warehouse, two *policies* are applied regarding the *splitting* of pallets of the same product:

forbidden: Splitting is forbidden so that all n_k pallets loaded with product $k \in K$ must be transported by the same truck. This is the standard case because it simplifies the handling and processing in the loading and unloading phases.

allowed: Splitting is allowed so that the n_k pallets of a product $k \in K$ may be distributed arbitrarily over the trucks. This splitting allowed policy can help to reduce the total number of used trucks. However, distributing a product over several trucks causes inconveniences. It should not occur more often than necessary (see objective 5 below).

For a given solution, we denote a product assigned to two or more trucks as *split product*.

We want to present both policies, splitting forbidden and splitting allowed, as variants of the same packing problem. For this purpose, we introduce *items* which are the atomic, not-divisible objects in the bin-packing model. The set K of products and the set I of items are identical, i.e., $K = I$. However, we write $k \in K$ to refer to products and $i \in I$ to refer to items. Depending on the policy, we define two weights for each item: w_i refers to the physical weight and b_i to the volume expressed as number of pallets. Table 3.1 and Example 7 clarify the correspondence.

Product $k \in K$	Corresponding item $i \in I$		
	Policy	splitting forbidden	splitting allowed
Unit weight ρ_k	Weight 1	$w_i = \rho_k \cdot n_k$	$w_i = \rho_k$
Number of pallets n_k	Weight 2	$b_i = n_k$	$b_i = 1$
	Demand	$q_i = 1$	$q_i = n_k$

Table 3.1: Relationship between products and items for the policies splitting forbidden and splitting allowed.

Example 7. Consider a set of products consisting of three standard products ($n_1 = 5, n_2 = 5, n_3 = 3, \rho_1 = \rho_2 = \rho_3 = 1$), (gray in Figure 3.1), one cooled product ($n_4 = 2, \rho_4 = 2$) (green in Figure 3.1), and one frozen product ($n_5 = 3, \rho_5 = 3$) (blue in Figure 3.1), i.e., $K = I = \{1, \ldots, 5\}$ with $I^S = \{1, 2, 3\}, I^C = \{4\}, I^F = \{5\}$. Then, the corresponding instance of the splitting forbidden policy has parameters $\boldsymbol{w} = (5, 5, 3, 4, 9), \boldsymbol{b} = (5, 5, 3, 2, 3), \boldsymbol{q} = \mathbb{1}$, whereas the corresponding instance of the splitting allowed policy has parameters $\boldsymbol{w} = (1, 1, 1, 2, 3), \boldsymbol{b} = \mathbb{1}, \boldsymbol{q} = (5, 5, 3, 2, 3)$.

With these definitions of items $i \in I$ with two weights $(w_i, b_i)_{i \in I}$ and two capacities (W, B), the problem of assigning products to trucks becomes a two-dimensional bin/vector-packing problem (see Section 3.2 for references to the pertinent literature). Trivially, a feasible packing exists only if $w_i \leq W$ and $b_i \leq B$ holds for all items $i \in I$.

What also makes our real-world problem interesting is the non-standard objective. We are seeking for a feasible product-truck assignment optimizing the following five objectives in a strictly *lexicographic* sense:

1. minimize the total number of trucks;
2. minimize the number of refrigerated trucks;
3. minimize the number of refrigerated trucks transporting frozen products;
4. minimize the number of refrigerated trucks which also transport standard products; and
5. minimize product splittings, which is either
   - 5a. minimize the overall number of splits; or
   - 5b. minimize the number of products which are split (over any number of trucks).

The fifth objective is only relevant for the policy splitting allowed. Altogether, we show that this lexicographic objective can be optimized by solving a sequence of extended bin-packing problems. Note that it is not a-priori given how to measure the impact of splitting products. The corresponding objective will depend on the actual handling of split products after their delivery at the destination warehouse. According to our industry partner, there are two different strategies observed with its customers: One customer handles products separately for every truck. This means that for every truck the processing effort depends on the number of different products (but also on other factors, such as the number of pallets). Therefore, it is cost efficient to minimize the overall number of splits, i.e., packing one product separated into two trucks is preferred over packing it separated into three trucks, as stated in objective 5a.A second customer collects all pallets carrying the same product, but arriving on different trucks, in the unloading zone and then inserts them into the warehouse in one batch. Thus, the main effort of splitting is due to the necessity of reserving a collection area, which happens whenever a product is split. Whether pallets are collected from two or more trucks hardly plays a role, as stated in objective 5b.

Note that for both splitting policies, objectives 1.–4. have to be observed in a lexicographic sense. This means that, e.g., a solution with 20 trucks and many refrigerated trucks is preferred over a solution with 21 trucks and less refrigerated trucks. An example of three different optimal solutions (splitting forbidden, splitting allowed and minimize either the number of splits or the number of split products) is given in Figure 3.1.

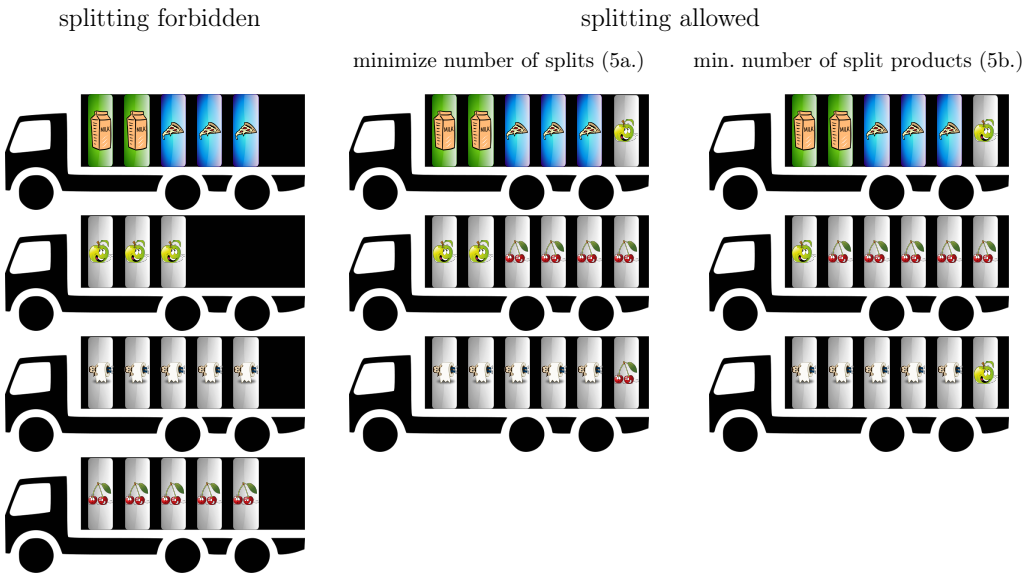


Figure 3.1: Optimal solutions of three objectives for Example 7 with capacity $B = 6$ pallets.

The methodological contribution of our paper is the development of an exact and heuristic solution approach. The exact approach is based on *branch-and-price* (BaP) (Desaulniers *et al.*, 2005; Lübbecke and Desrosiers, 2005). For the different objectives and corresponding extended bin-packing problems, we develop a unified description of the respective column-generation subproblems. We show that, in turn, each subproblem can be solved by one or several modified 2-dimensional knapsack problems (Martello and Toth, 2003), (Kellerer *et al.*, 2004, ch. 9.6). These are especially involved in the case of the policy splitting allowed. Moreover, an important and here non-trivial algorithmic component is the branching scheme. We adopt the branching scheme originally developed and successfully applied by Heßler *et al.* (2018) to the vector-packing problem. Stabilization with dual-optimal inequalities (Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016) helps to increase the performance of the BaP approach.

The heuristic approach works in two levels: In the first level, several constructive heuristics are employed to reach a packing of items from the same category. They strive for a balanced utilization of weight and volume constraints based on weight-per-pallet, i.e., *density* considerations and on the surrogate weights as introduced by Caprara and Toth (2001). In the second level, a more complicated meta-scheme considers the multi-compartment and product-fragmentation features of the problem at hand and applies various problem-specific heuristics to search for good solutions. They are based on decomposing and regrouping the best solution

obtained from the first level to improve the lexicographic objectives for the different product categories. At first, only the splitting forbidden policy is considered. Then, for the splitting allowed policy we try to improve the given solution by separating certain pallets of the same product and possibly reduce the number of trucks by reassigning these items. Throughout the heuristic process, we compare the current solutions to lower bounds as a stopping criterion. It should be noted that the heuristics were also incorporated in the SAP ERP software system of our industrial partner, a European producer of foodstuffs and beverages, and are used in the daily planning process.

We conducted a series of computational experiments with the exact and heuristic algorithms based on real-world data sets with up to 430 items and on more difficult test instances derived from these. We compare the performance of our heuristic to the previously employed strategy of the company and to the proven optimal solutions. It turns out that the heuristic saves on average one truck per day compared to the previously used approach for real-world instances, with an even higher advantage for more difficult instances. At the same time, running times are moderate, often less than one second. The gap to the optimal solution is fairly moderate for real-world instances but increases for the self-generated hard test cases. Regarding the performance of the exact branch-and-price algorithm, most of the test instances can be solved to optimality within the time limit of 10 minutes for every subproblem. Notably exceptions arise for some of the generated, difficult instances in case of the splitting allowed policy. We can also illustrate the highly positive effect of strengthening the ILP-models by lower bounds and of solving pricing problems on a reduced graph. Finally, combining the two approaches it turns out that adding the best heuristic solution to the set of columns of the branch-and-price approach has hardly any effect for the splitting forbidden policy, but it gives a major improvement for the splitting allowed policy.

The remainder of the paper is organized as follows: In Section 3.2, a brief overview of the literature on two-dimensional bin-packing problems with and without item fragmentation and multi-compartment vehicle-routing problems is given. Analytically computed lower bounds are presented in Section 3.3. Details on the exact BaP algorithm are provided in Section 3.4, including the sequence of extensive formulations used for the different objectives, the unified column-generation algorithm, and stabilization techniques. The heuristic approach is presented in Section 3.5. In Section 3.6, the results of our computational experiments are presented and discussed. Final conclusions are drawn in Section 3.7.

3.2 Literature Review

The classical one-dimensional *bin-packing problem* (BP) is one of the most fundamental problems in combinatorial optimization: A given set of items has to be packed into a minimum number of bins such that the total weight of the items in each bin does not exceed the bin-capacity. Its mathematical foundations were first studied in the early 1970s (Garey *et al.*, 1972). Classical heuristics for BP, such as *first-fit* (FF), *best-fit* (BF) and *worst-fit* (WF), and their variants based on a decreasing ordering of item weights, i.e., FFD, BFD, and WFD (see, e.g., Johnson, 1973), will be assumed common knowledge throughout this paper.

For the exact solution of BP, two alternative and competing solution principles can be observed in the recent literature (see Delorme *et al.*, 2016, for an overview of BP formulations). The first type of solution approach relies on pseudo-polynomial arc-flow formulations (Valério de Carvalho, 1999) solved with general-purpose *mixed-integer linear programming* (MIP) solvers. Important refinements are network/graph compression techniques (Brandão and Pedroso, 2016) and the use of *reflect networks* modeling packings with only half of the bin capacity (Côté and Iori, 2018). The first half and the second half of a packed bin are represented and feasibly combined in the reflect network. The second type of solution approach uses a pattern-based formulation a.k.a. the Gilmore-Gomory model (Gilmore and Gomory, 1961), which is solved by column-generation techniques. While early works focused on the solution of the linear relaxation in order to quickly compute a strong lower bound, the necessary theory providing a complete branching scheme leading to a fully-fledged BaP algorithm has been provided with the work of Vanderbeck (1999). A state-of-the-art branch-price-and-cut approach for BP is presented by Wei *et al.* (2019). A hybrid of both approaches, denoted as *reflect+*, has been recently coined by Delorme and Iori (2020) and seems to be a very competitive approach making use of the excellent dual bounds provided by column generation, heuristically and exactly truncated reflect networks, and the power of modern MIP solvers.

Due to the high practical relevance of packing problems many variations of the classical one-dimensional BP have been investigated. For a comprehensive overview, we refer to the surveys by Christensen *et al.* (2017) and Coffman Jr *et al.* (2013). In the context of our real-world bin packing variant especially publications dealing with two-dimensional, multi-compartment, and item fragmentation aspects are of particular relevance.

Two-dimensional BPs represent a well-studied generalization and, hence, a large number of publications in this field appeared over the last decades. We do *not* consider packing problems in which geometric items have to be placed into bins, known as rectangle packing (Lodi *et al.*, 2002). Relevant for our application are p-dimensional BPs with $p = 2$ independent dimensions (such as physical weight and

volume, or value and weight). In the literature these problems are given different names, e.g., (2-dimensional) *vector (bin) packing problem* (VPP, 2D-VPP) and *two-constraint bin packing problem*. Our problem is a variant of the 2D-VPP for which an early exact approach is given by Spieksma (1994). Recent exact approaches are based on BaP and are described in Heßler *et al.* (2018) and Wei *et al.* (2020). General approximation results are given in Bansal *et al.* (2016). The case, where weights and volume are strictly ordered is considered in Caprara *et al.* (2003). Simple greedy-type heuristics are presented in Aringhieri *et al.* (2018). A comprehensive survey of vector packing problems can be found in Christensen *et al.* (2017).

Multi-compartment problems arise when different item types can be packed into the same bin but have to be separated from each other (incompatibility constraints). Those settings are mainly studied in vehicle-routing problems, see the surveys (Pollaris *et al.*, 2014; Henke, 2018). In our application, there are no incompatibilities between products/items. However, the use of compartments for cooled and frozen products imposes additional costs.

The introduction of *item fragmentation* generalizes the classical BP by allowing items to be split among several bins at a cost. Item fragmentation for BPs is introduced and proved NP-hard by Mandal *et al.* (1998). More recent publications provide exact solution algorithms based on column generation and dual cuts techniques as well as employing heuristics and implicit enumeration (Casazza and Ceselli, 2016). LeCun *et al.* (2015) presents models for practical applications of item fragmentation like minimizing the number of money transfers after a group trip and present approximation algorithms for identical as well as for the more generic case of non-equal-sized bins. The generalization to bins that differ in both cost and capacity is tackled in the contribution Casazza (2019) by proposing new mathematical programming models that even avoid the use of fractional variables. A worst-case analysis for BP with item fragmentation is performed by Bertazzi *et al.* (2019). General models of valuations for fragmented items are considered for the closely related knapsack problem in the recent work Malaguti *et al.* (2019). Recall from Section 3.1 that in our application, products might be divisible (depending on the splitting policy) but that by definition our items are never split. Our definition of an item has implicitly solved issues related to product fragmentation by discretely splitting demand along the volume (=pallets) dimension.

3.3 Lower bounds

Various lower bounds for subsets of items or all items are used in different parts of the algorithms. For convenience, we introduce these lower bounds in this brief section.

A straightforward lower bound on the number of bins needed to pack a subset $\mathcal{I} \subseteq I$ can be obtained by dividing the problem along the two dimensions. For each dimension one can compute a bound on the respective one-dimensional bin packing problems and then use the better bound of the two:

$$\mathrm{LB}_1(\mathcal{I}) = \max\left\{\left\lceil \frac{\sum_{i\in\mathcal{I}} w_i}{W} \right\rceil, \left\lceil \frac{\sum_{i\in\mathcal{I}} b_i}{B} \right\rceil\right\}.$$

Caprara (1998) proved that this lower bound equals the rounded up value of the linear relaxation to the standard assigment-type ILP model for the two-dimensional bin-packing problem, see e.g. (1)–(6) in Caprara and Toth (2001).

Another lower bound is inspired by the lower bounding procedure of Martello and Toth (1990) for bin-packing problems. We use two distinct subsets of $\mathcal{I}$, namely the subset

$$\mathcal{I}_1 = \{i \in \mathcal{I} : w_i > W/2 \text{ and } b_i > B/2\}$$

of items consuming more than half of a bin's capacity, and the disjoint subset

$$\mathcal{I}_2 = \{i \in \mathcal{I} \setminus \mathcal{I}_1 : w_i \geq W/2 \text{ or } b_i \geq B/2\}$$

of items that do not fit together into a bin with any item of the subset $\mathcal{I}_1$. We obtain the lower bound

$$\mathrm{LB}_2(\mathcal{I}) = |\mathcal{I}_1| + \mathrm{LB}_1(\mathcal{I}_2).$$

In the following, the lower bound $\mathrm{LB}(\mathcal{I}) = \max\{LB_1(\mathcal{I}), LB_2(\mathcal{I})\}$ is used for various subsets $\mathcal{I}$. For the sake of brevity, we define the following shorthand notation for the lower bounds

$$\mathrm{LB} = \mathrm{LB}(I), \quad \mathrm{LB}^S = \mathrm{LB}(I^S), \quad \mathrm{LB}^F = \mathrm{LB}(I^F), \quad \text{and} \quad \mathrm{LB}^{C,F} = \mathrm{LB}(I^C \cup I^F).$$

3.4 Branch-and-price algorithm

We propose the exact solution of the overall lexicographic minimization problem using a stage-wise modified BaP algorithm (Desaulniers *et al.*, 2005; Lübbecke and Desrosiers, 2005). While the first objective is minimized by solving a standard 2D-VPP, the following four stages first restrict the solution space to previously computed optimal value(s) and then apply BaP to the restricted formulation. Since the initial formulation is of the Gilmore-Gomory type (Gilmore and Gomory, 1961; Heßler *et al.*, 2018), the linear programs can be characterized as *extensive*, meaning that they typically exhibit an enormous number of variables. BaP is tailored to

solving such extensive optimization problems. Technically, BaP is a branch-and-bound algorithm in which the linear relaxation at each node of the search tree is solved by *column generation* (CG). CG is an iterative method that solves a linear program by decomposing it into a *restricted master problem* (RMP) and a pricing *subproblem* (SP).

In Section 3.4.1, we start by describing the set of all feasible packing patterns and define some subsets of patterns which are later used for presenting the restricted formulations that address the objectives 2.–4., 5a., and 5b. at subsequent stages. The unifying approach for the pricing subproblems presented in Section 3.4.2 is followed by the discussion of stabilization, branching schemes, and acceleration techniques in Sections 3.4.3–3.4.5.

3.4.1 Pattern-based formulations

The models of all stages are variations of the covering formulation of Gilmore and Gomory (1961) that are based on *patterns*. A pattern describes a feasible packing of a bin with items. For each item $i \in I$ the value

$$u_i = \min\left\{q_i, \left\lfloor \frac{W}{w_i} \right\rfloor, \left\lfloor \frac{B}{b_i} \right\rfloor\right\}$$

is an upper bound on the number of times that item i may occur in a pattern. The set of all feasible patterns is therefore

$$P = \left\{(a_1, \ldots, a_m)^\top \in \mathbb{Z}_+^m : \sum_{i=1}^m a_i w_i \le W, \sum_{i=1}^m a_i b_i \le B \text{ and } a_i \le u_i \text{ for all } i \in I\right\}.$$

Recall that for the standard case of the splitting forbidden policy, we have $q_i = 1$ and thus $u_i = 1$ so that all feasible patterns are binary, i.e., $(a_1, \ldots, a_m) \in \{0,1\}^m$. Thus, the resulting VPP is a *binary VPP* (01-VPP). For the splitting allowed policy, the coefficients a_i are upper bounded by integers u_i so that the resulting VPP is a *bounded VPP* (B-VPP).

For Stages 2.–5., pattern subsets taking different product types into account have to be considered. We use the superscripts S, C, and F to refer to standard, cooled, and frozen products/items, respectively. A combination of superscripts refers to patterns that only include items of the respective categories. If a superscript is supplemented with > 0, at least one item of the category must be included. Table 3.2 exactly defines the relevant pattern subsets.

Stage 1.: Minimize the total number of trucks

At Stage 1., the minimization of the number of trucks is equivalent to the minimization of the total number of bins that are used. Therefore, the problem is a

Pattern subset	Specification regarding categories of items: standard $\sum_{i\in I^S} a_i$		cooled $\sum_{i\in I^C} a_i$		frozen $\sum_{i\in I^F} a_i$
P^S			$= 0$	and	$= 0$
P^C	$= 0$				$= 0$
P^F	$= 0$	and	$= 0$		
$P^{C,F}$	$= 0$				
$P^{S,C>0}$			≥ 1	and	$= 0$
$P^{C,F>0}$	$= 0$			and	≥ 1
$P^{S>0,C>0}$	≥ 1	and	≥ 1	and	$= 0$
$P^{S,C,F>0}$					≥ 1
$P^{S>0,C,F>0}$	≥ 1			and	≥ 1
$P^{S,C\cup F>0}$			$\sum_{i\in I^C} a_i + \sum_{i\in I^F} a_i \geq 1$		
$P^{S>0,C\cup F>0}$	≥ 1	and	$\sum_{i\in I^C} a_i + \sum_{i\in I^F} a_i \geq 1$		

Table 3.2: Overview of pattern subsets.

standard 2D-VPP. The Gilmore-Gomory formulation comprises non-negative integer variables x_p for all patterns $p \in P$ and is given by

$$z^{1.} = \min \sum_{p\in P} x_p \qquad \text{duals:} \tag{3.1a}$$

$$\text{(Stage 1.)} \quad \text{subject to} \sum_{p\in P} a_i^p x_p \geq q_i, \qquad i \in I \qquad [\pi_i] \tag{3.1b}$$

$$x_p \geq 0 \text{ integer}, \qquad p \in P. \tag{3.1c}$$

The objective (3.1a) minimizes the number of bins, i.e., patterns used. Constraints (3.1b) ensure that the demand of all items is covered. Note that for an RMP, i.e., the linear relaxation of a model with a restricted variable/pattern set $P' \subset P$, we also show the associated dual variables $(\pi_i)_{i\in I}$ of constraints (3.1b) that we later use in Section 3.4.2 to describe the CG process. The domain of the pattern variables is defined by (3.1c). The first stage can be exactly solved with the BaP algorithm presented by Heßler *et al.* (2018) without any adaptation.

Stage 2.: Minimize the number of refrigerated trucks

The secondary objective of minimizing the number of refrigerated trucks is equivalent to minimizing the number of patterns of the subset $P^{S,C\cup F>0}$ (see Table 3.2). The first objective imposes the additional condition that the total number of all

trucks/bins is restricted to $z^{1.}$. It can be incorporated into model (3.1) with a single additional constraint, leading to:

$$z^{2.} = \min \sum_{p \in P^{S,C \cup F>0}} x_p \qquad \text{duals:} \qquad (3.2a)$$

$$\text{(Stage 2.)} \quad \text{subject to (3.1b)–(3.1c)}$$

$$\sum_{p \in P} x_p \leq z^{1.} \qquad [\delta_{3.2b}] \qquad (3.2b)$$

$$\sum_{p \in P^{S,C \cup F>0}} x_p \geq \mathrm{LB}^{C,F}. \qquad [\delta_{3.2c}] \qquad (3.2c)$$

The objective (3.2a) minimizes the number of refrigerated trucks. Constraint (3.2b) restricts the total number of trucks to the minimum number resulting from the solution of formulation (3.1). A lower bound on the number of refrigerated trucks is imposed by constraint (3.2c). Note that constraint (3.2c) is not mandatory for the correctness of the model but its addition often yields a better lower bound of the linear relaxation.

Stage 3.: Minimize the number of refrigerated trucks which contain frozen products

The tertiary objective of minimizing the number of refrigerated trucks that contain frozen products is equivalent to minimizing the number of patterns of the subset $P^{S,C,F>0}$ (see Table 3.2). At Stage 3., we additionally have to bound the number of refrigerated trucks by $z^{2.}$, which yields to the model

$$z^{3.} = \min \sum_{p \in P^{S,C,F>0}} x_p \qquad \text{duals:} \qquad (3.3a)$$

$$\text{(Stage 3.)} \quad \text{subject to (3.1b)–(3.1c), (3.2b)}$$

$$\sum_{p \in P^{S,C \cup F>0}} x_p \leq z^{2.} \qquad [\delta_{3.3b}] \qquad (3.3b)$$

$$\sum_{p \in P^{S,C,F>0}} x_p \geq \mathrm{LB}^{F}. \qquad [\delta_{3.3c}] \qquad (3.3c)$$

The objective (3.3a) is the minimization of the number of trucks transporting frozen products. Constraint (3.3b) fixes the total number of refrigerated trucks. Since this fixation makes constraint (3.2c) redundant, it is omitted. We add the constraint (3.3c) imposing a lower bound on the number of frozen trucks to strengthen the linear relaxation.

Stage 4.: Minimize the number of refrigerated trucks which also contain standard products

The fourth rate objective of minimizing the number of refrigerated trucks which also contain standard products is equivalent to the minimization of patterns of the subset $P^{S>0,C\cup F>0}$. Optimal values from the Stages 1.–3. are again incorporated as additional constraints:

$$z^{4.} = \min \sum_{p\in P^{S>0,C\cup F>0}} x_p \qquad \text{duals:} \quad (3.4a)$$

(Stage 4.) subject to (3.1b)–(3.1c), (3.2b), (3.3b)

$$\sum_{p\in P^{S,C,F>0}} x_p \le z^{3.} \qquad [\delta_{3.4b}] \quad (3.4b)$$

$$\sum_{p\in P^{S>0,C\cup F>0}} x_p \ge \mathrm{LB}^S + z^{2.} - z^{1.}. \qquad [\delta_{3.4c}] \quad (3.4c)$$

The objective (3.4a) minimizes the number of refrigerated trucks which also transport at least one standard product. In the following, such a truck is denoted as *mixed truck*. A lower bound on the total number of mixed trucks is $\mathrm{LB}^S - (z^{1.} - z^{2.})$. The difference $z^{1.} - z^{2.}$ gives the number of trucks containing only standard products. Subtracting this number from the lower bound for the number of trucks required to pack all standard products, gives a lower bound on the number of mixed trucks. Constraint (3.4b) restricts the number of frozen trucks. As constraint (3.3c), constraint (3.4c) is not mandatory but helps to strengthen the linear relaxation.

Stage 5.: Minimize product splitting

Recall from Section 3.1 that the last objective of minimizing product splitting applies only to the splitting allowed policy and can be put into effect in two different ways using objective 5a. or 5b.

In the case of objective 5a., minimizing the overall number of splits is equivalent to minimizing the number of different products packed into all bins. To this end, let $s_p = \left|\{a_i^p > 0 : i \in I\}\right|$ denote this number of different products/items in a pattern $a_i^p \in P$. Then, the actual number of splits is given by the difference of the total number of different products in all used bins and the total number m of

products. Therefore, minimizing the number of splits can be formulated as

$$z^{5a.} = \min \sum_{p \in P} s_p x_p \qquad \text{duals:} \quad (3.5a)$$

$$\text{(Stage 5a.)} \quad \text{subject to (3.1b)–(3.1c), (3.2b), (3.3b), (3.4b)}$$

$$\sum_{p \in P^{S>0, C \cup F>0}} x_p \leq z^{4.}. \qquad [\delta_{3.5b}] \quad (3.5b)$$

Constraint (3.5b) restricts the number of mixed trucks as determined at Stage 4.

In the case of objective 5b., it only counts whether a product is split or not, but the number of trucks a split product occupies is irrelevant. Minimizing the number of split products is equivalent to maximizing the number of *integrally packed products*. A product is integrally packed if all its pallets are packed on the same truck. Accordingly, for a pattern $p \in P$, we define for our minimization problem the negative number of integrally packed products as $d_p = -\left|\left\{i \in I : a_i^p = q_i\right\}\right|$. Hence, our objective is expressed by the minimization of the total value of d_p over all patterns used:

$$z^{5b.} = \min \sum_{p \in P} d_p x_p \qquad \text{duals:} \quad (3.5c)$$

$$\text{(Stage 5b.)} \quad \text{subject to (3.1b)–(3.1c), (3.2b), (3.3b),}$$
$$\text{(3.4b), (3.5b)}$$

$$\sum_{p \in P} \left(\sum_{i \in I} a_i^p \right) x_p \leq \sum_{i \in I} q_i. \qquad [\gamma] \quad (3.5d)$$

Constraint (3.5d) ensures that each item $i \in I$ is packed at most q_i times. Note that the latter constraint is indeed necessary because otherwise one might artificially improve the objective function by adding additional integrally packed products more often than demanded. In combination with the covering constraint (3.1b), both constraints (3.5d) and (3.1b) are always binding. Therefore, adding disaggregated constraints of the form $\sum_{p \in P} a_i^p x_p = q_i$ for all $i \in I$ does not yield a tighter formulation.

3.4.2 Column generation

With the models (3.1), (3.2), (3.3), (3.4), (3.5a)–(3.5b), and (3.5c)–(3.5d) we have presented six different extensive formulations that all require a CG-based solution approach. In principle, pattern generation is algorithmically simple to manage, since we will show that the subproblem solution finally boils down to solving binary or bounded *2-dimensional knapsack problems*, see (Martello and Toth, 2003),

(Kellerer *et al.*, 2004, ch. 9.6). In the following, we assume that a linear relaxation of the RMP is solved and has produced dual prices $(\pi_i)_{i\in I}$ and (δ_{con}) for the respective constraint(s) *con*. The task of the SP is to determine at least one negative reduced-cost pattern or to prove that no negative reduced-cost pattern exists.

A first difficulty for precisely describing the associated SPs lies in the fact that each of the six SPs associates different dual prices to groups of patterns depending on whether they include standard, cooled, or frozen products/items. Handling these multiple cases is indeed confusing. A fundamental step towards a concise and unified handling is the idea to further divide each SP into 2D-KPs for subsets of patterns. Solving the SP then amounts to solving the associated 2D-KPs. The decisive point is that the reduced cost of a pattern $p = (a_i)$ can be described as (at least for the first four stages, see below)

$$\tilde{c}_p = \delta - \sum_{i\in I} \sigma_i a_i,$$

where the coefficients $\delta = \delta^X$ and $\sigma_i = \sigma_i^X$ for $i \in I$ depend of the particular subcase X. Table 3.3 formalizes the different (sub)cases: Depending on the stage/objective, there are between one and five different pattern subsets (see Table 3.2 for their formal definition) for which independent 2D-KPs need to be solved. For each pattern subset, the columns σ_i and δ describe how these coefficients can be computed from the given dual solution $(\pi_i)_{i\in I}$ and (δ_{con}) (using only dual prices of constraints *con* that exist at the actual stage). Moreover, the 2D-KP can be restricted to an item subset $\mathcal{I} \subset I$. Finally, some pattern subsets require the presence of one or two subsets of items. For example, $P^{S>0,C,F>0}$ requires that at least one item of a standard product and one item of a frozen product are present in the pattern. For this purpose, let $\mathcal{R}$ be the *set of required subsets of items*, i.e., for $P^{S>0,C,F>0}$ there is $\mathcal{R} = \{I^S, I^F\}$. The elements of the respective sets $\mathcal{R}$ are displayed in the last column of Table 3.3.

A second difficulty stems from the fact that, for the objectives 5a. and 5b., the reduced cost of a pattern is no longer completely linear in the coefficients of the pattern. However, all 2D-KPs can be formalized as follows. Nonnegative bounded integer (binary for $u_i = 1$) variables y_i for $i \in I$ describe the unknown pattern coefficients a_i. Moreover, for these variables $\mathbf{y} = (y_i)_{i\in I}$, the function $f(\mathbf{y})$ captures the non-linear part of the reduced cost (described in detail in the next paragraphs). For a given subset of admissible items $\mathcal{I}$ (in the sense of the second

Stage	Pattern subsets	Dual prices σ_i	δ	Item subset $\mathcal{I}$	Elements of subsets $\mathcal{R}$ of required items
1.	P	π_i	1	I	
2.	P^S	π_i	$1 - \delta_{3.2b}$	I^S	I^S
	$P^{S,C\cup F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c}$	I	$I^C \cup I^F$
3.	P^S	π_i	$1 - \delta_{3.2b}$	I^S	I^S
	$P^{S,C>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b}$	$I^S \cup I^C$	I^C
	$P^{S,C,F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c}$	I	I^F
4.	P^S	π_i	$1 - \delta_{3.2b}$	I^S	
	P^C	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b}$	I^C	
	$P^{C,F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b}$	$I^C \cup I^F$	I^F
	$P^{S>0,C>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.4c}$	$I^S \cup I^C$	I^S, I^C
	$P^{S>0,C,F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b} - \delta_{3.4c}$	I	I^S, I^F
5a.	P^S	π_i	$1 - \delta_{3.2b}$	I^S	
	P^C	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b}$	I^C	
	$P^{C,F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b}$	$I^C \cup I^F$	I^F
	$P^{S>0,C>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.4c} - \delta_{3.5b}$	$I^S \cup I^C$	I^S, I^C
	$P^{S>0,C,F>0}$	π_i	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b} - \delta_{3.4c} - \delta_{3.5b}$	I	I^S, I^F
5b.	P^S	$\pi_i + \gamma$	$1 - \delta_{3.2b}$	I^S	
	P^C	$\pi_i + \gamma$	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b}$	I^C	
	$P^{C,F>0}$	$\pi_i + \gamma$	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b}$	$I^C \cup I^F$	I^F
	$P^{S>0,C>0}$	$\pi_i + \gamma$	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.4c} - \delta_{3.5b}$	$I^S \cup I^C$	I^S, I^C
	$P^{S>0,C,F>0}$	$\pi_i + \gamma$	$1 - \delta_{3.2b} - \delta_{3.2c} - \delta_{3.3b} - \delta_{3.3c} - \delta_{3.4b} - \delta_{3.4c} - \delta_{3.5b}$	I	I^S, I^F

Table 3.3: Overview of pricing subproblems.

last column of Table 3.3), the extended 2D-KP can now be described as

$$z^{\mathrm{KP}(\mathcal{I},\sigma_i,\delta,\mathcal{R})} = \quad \delta - \max\left(\sum_{i\in\mathcal{I}} \sigma_i y_i - f(\mathbf{y})\right) \tag{3.6a}$$

$$(\mathrm{KP}(\mathcal{I},\sigma_i,\delta,\mathcal{R})) \quad \text{subject to} \sum_{i\in\mathcal{I}} w_i y_i \le W \tag{3.6b}$$

$$\sum_{i\in\mathcal{I}} b_i y_i \le B \tag{3.6c}$$

$$\sum_{i\in J} y_i \ge 1 \qquad J \in \mathcal{R} \tag{3.6d}$$

$$y_i \in \{0,1,\ldots,u_i\}, \qquad i \in \mathcal{I}. \tag{3.6e}$$

The objective (3.6a) is the minimization of the reduced cost of the pattern. The constraints (3.6b) and (3.6c) are the capacity constraints. The defining property of some pattern subsets to contain at least one item of one or two particular categories is enforced by condition(s) (3.6d). The domain of the decision variables is defined by (3.6e).

At Stages 1.–4., the function f vanishes, i.e., $f \equiv 0$, so that $\mathrm{KP}(\mathcal{I},\sigma_i,\delta,\mathcal{R})$ reduces to a standard binary or bounded 2D-KP with the additional constraints (3.6d). A straightforward solution approach could be to solve several standard binary 2D-KP that previously pack one item of each set $J \in \mathcal{R}$. Instead of enumerating all combinations, the selection of previously packed items could also be embedded into a branch-and-bound algorithm. However, we formulate and solve this subproblem as a variant of the shortest-path problem with resource constraints (see next paragraph).

For the first splitting variant that minimizes the total number of splits, i.e., objective 5a., the function f must count the number of products the constructed pattern contains. This is done with

$$f(\boldsymbol{y}) = |\{i \in I : y_i > 0\}|.$$

For the second splitting variant that minimizes the number of split products, i.e., objective 5b., the function f must count the (negative) number of integrally packed products the constructed pattern contains. The definition

$$f(\boldsymbol{y}) = -|\{i \in I : y_i = q_i\}|$$

gives the correct contribution. As an example, the two 2D-KPs arising at Stage 2. are specified in the following example.

Example 8. At Stage 2., we consider two 2D-KPs that generate patterns of the subsets P^S and $P^{S,\overline{C\cup F>0}}$. Recall that $\delta_{3.2b}$ and $\delta_{3.2c}$ are the dual prices of constraints (3.2b) and (3.2c), respectively. Both 2D-KPs have $\sigma_i = \pi_i$ for all $i \in I$ and

$f \equiv 0$. The first 2D-KP generates patterns of set P^S and is defined by $\delta = 1 - \delta_{3.2b}$, $\mathcal{I} = I^S$, and $\mathcal{R} = \{I^S\}$, while the second generates patterns of the set $P^{S,C \cup F>0}$ and is defined by $\delta = 1 - \delta_{3.2b} - \delta_{3.2c}$, $\mathcal{I} = I$, and $\mathcal{R} = \{I^C \cup I^F\}$.

SPPRC-based solution. The different problems $\mathrm{KP}(\mathcal{I}, \sigma_i, \delta, \mathcal{R})$ can be formulated as *shortest path problems with resource constraints* (SPPRCs, Irnich and Desaulniers, 2005). SPPRCs can model intricate relationships between attributes and many variants of the SPPRC can be effectively solved by dynamic-programming labeling algorithms. Such an approach has been successfully chosen for the one-dimensional BP by Wei *et al.* (2019) and for the VPP by Heßler *et al.* (2018). The advantage of a labeling-based approach is that also subsets $\mathcal{R}$ of required items and the objectives 5a. and 5b. defining the function f can be considered as well as involved branching decisions (discussed later in Section 3.4.4).

For the sake of simplicity, we start with the case that any item can be contained in a pattern, i.e., $\mathcal{I} = I = \{1, 2, \ldots, m\}$. The underlying digraph $G = (V, E)$ then consists of $m + 1$ vertices $V = \{0, \ldots, m\}$. The arc set E consists of $m + \sum_{i \in I} u_i = \sum_{i \in I}(u_i + 1)$ arcs partitioned into m subsets $E(i)$ associated with item $i \in I$. The set $E(i)$ contains $u_i + 1$ parallel arcs connecting vertex $i - 1$ with vertex i. Thus, we have an arc set $E(i)$ consisting of arcs $e_j(i)$ with $j \in \{0, 1, \ldots, u_i + 1\}$.

Each arc $e_j(i)$ has three attributes, two for the weights and one for the profit. For the j-th arc $e_j(i)$ of $E(i)$, these are defined as $w(e_j(i)) = j \cdot w_i$, $b(e_j(i)) = j \cdot b_i$, and $\sigma(e_j(i)) = j \cdot \sigma_i$, respectively.

An example of the digraph is sketched in Figure 3.2. Each 0-m-path $(0, e_{j_1}(1), 1, e_{j_2}(2), 2, \ldots, m-1, e_{j_m}(m), m)$ uniquely defines a pattern (a_i) with $a_i = j_i$. Necessary conditions for its feasibility are that $\sum_{i=1}^{m} w(e_{j_i}(i)) \leq W$ and $\sum_{i=1}^{m} b(e_{j_i}(i)) \leq B$ holds. Partial paths that do not fulfill these conditions can be directly discarded.

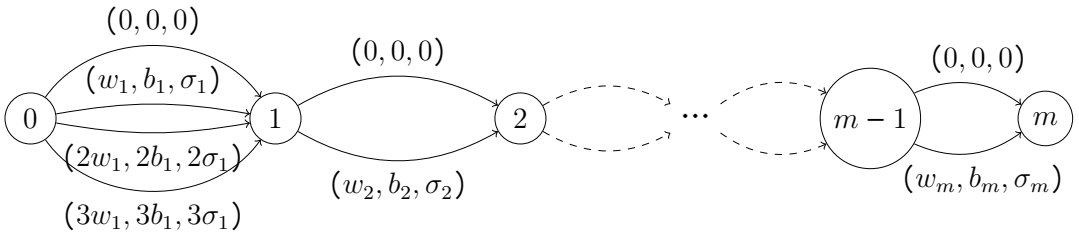


Figure 3.2: SPPRC digraph with items having $u_1 = 3, u_2 = 1$, and $u_m = 1$. The arcs $e \in E$ are shown with their weights and profit $(w(e), b(e), \sigma(e))$.

If $\mathcal{I} \subsetneq I$, i.e., some categories of items are not in $\mathcal{I}$, the corresponding digraph can be obtained by reducing the arc set so that for each $i \in I \setminus \mathcal{I}$ only the null-arc

$(0,0,0)$ is in $E(i)$. Subsequently, the digraph can be shrunk by merging vertices $i-1$ and i for all $i \in I \setminus \mathcal{I}$.

At Stages 2.–5., three additional boolean attributes $\boldsymbol{v} = (v_S, v_C, v_F)$ for the three categories standard, cooled, and frozen are added to indicate whether a specific product category has been packed. For an item $i \in I$, we define $\boldsymbol{v}(e_j(i)) = (1,0,0)$, $(0,1,0)$, or $(0,0,1)$ for all j according to the category product/item i belongs to.

The SPPRC is solved by a forward dynamic-programming labeling algorithm that works as follows: The starting point is the trivial partial path (0) with the initial label $(0,0,0)$. All labels at a vertex $i-1$ are extended to vertex i iteratively for $i = 1,2,\ldots,m$. A forward extension along an arc $e(i) \in E(i)$ of label $L(F) = (w, b, \boldsymbol{v}, \sigma)$ (omitting the index j) produces the label $(w + w(e(i)), b + b(e(i)), \max\{\boldsymbol{v}, \boldsymbol{v}(e(i))\}, \sigma + \sigma(e(i)))$, where the maximum operator is applied component-wise. The labels at m and the associated 0-m-paths have to be filtered at the end: Only labels $(w, b, \boldsymbol{v}, \sigma)$ with correct $\boldsymbol{v}$-values provide feasible patterns, e.g., $v_C + v_F \geq 1$ for the pattern set $P^{S,C\cup F>0}$ used at Stage 2. in the second subcase. Moreover, the reduced cost of the associated pattern is $\delta - \sigma$.

The work of Heßler *et al.* (2018) has pointed out that different dominance principles between labels are possible and worth to be investigated. There clearly exists a tradeoff between the strength of a dominance rule and its computational burden resulting from the pairwise comparison of labels. A strong dominance rule can help to drastically reduce the overall number of labels that remain at termination of the labeling algorithm. The result is also fewer labels at intermediate vertices, which reduces the computational effort associated with the label extension step and later dominance comparisons. On the downside, the application of the following strong dominance rule may require the pairwise comparison of a possibly huge number of labels.

Rule 3. (Strong Dominance) A forward label $L(F) = (w, b, \boldsymbol{v}, \sigma)$ of a forward partial path F dominates another forward label $L(F') = (w', b', \boldsymbol{v}', \sigma')$ of a forward partial path F' ending at the same vertex if $w \leq w'$, $b \leq b'$, $\boldsymbol{v} \geq \boldsymbol{v}'$, and $\sigma \geq \sigma'$.

The following weaker dominance rule allows a direct $\mathcal{O}(1)$ comparison of labels, immediately at the time when a label is created, if labels are stored in a lookup table.

Rule 4. (Weak Dominance) A forward label $L(F) = (w, b, \boldsymbol{v}, \sigma)$ of a forward partial path F dominates another forward label $L(F') = (w', b', \boldsymbol{v}', \sigma')$ of a forward partial path F' ending at the same vertex if $w = w'$, $b = b'$, $\boldsymbol{v} = \boldsymbol{v}'$, and $\sigma \geq \sigma'$.

As in Heßler *et al.* (2018), also our pretests have confirmed that the weak dominance Rule 4 performs better than the strong Rule 3 for most benchmark instance used in the computational experiments. Therefore, we do not use Rule 3 and also omit its proof. The validity of the weak dominance Rule 4 is obvious.

Finally, we discuss the incorporation of the functions f at Stage 5. into the SPPRC modelling and solution approach. For the first objective 5a., the function f counts the number of products the pattern contains, i.e., $f(\boldsymbol{y}) = |\{i \in I : y_i > 0\}|$. Consequently, the profit attribute on the arcs of the SPPRC digraph must consider the number of different products in the resulting pattern. Thus, a cost of 1 is subtracted for every item that is packed, i.e., every non-zero arc. An example is given in Figure 3.3.

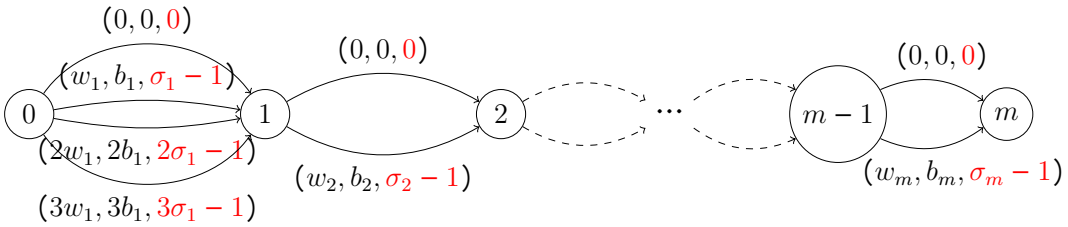


Figure 3.3: SPPRC digraph for objective 5a. with otherwise identical attributes as depicted in Figure 3.2. The arcs $e \in E$ are shown with their weights and profit $(w(e), b(e), \sigma(e))$.

For the second objective 5b., the function f counts the (negative) number of integrally packed products the pattern contains, i.e., $f(\boldsymbol{y}) = -|\{i \in I : y_i = q_i\}|$. Hence, arc profits are modified if an item i is packed completely, i.e., the additional profit of 1 is added to the q_ith arc of item i (present only if $q_i \leq u_i$). An example is given in Figure 3.4. Note that the dual price γ of constraint (3.5d) is already included in the definition of σ_i for all $i \in I$.

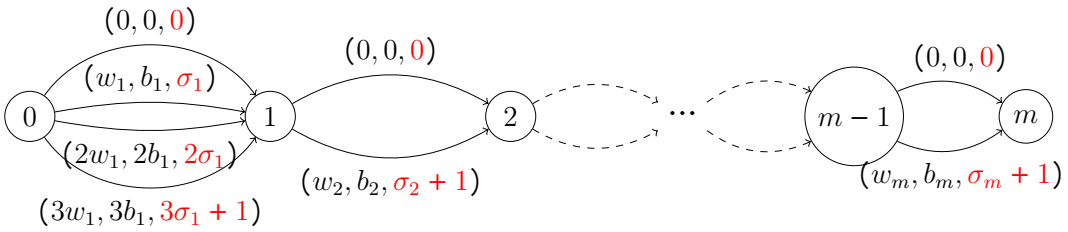


Figure 3.4: SPPRC digraph for objective 5b. with otherwise identical attributes as depicted in Figure 3.2. The arcs $e \in E$ are shown with their weights and profit $(w(e), b(e), \sigma(e))$.

3.4.3 Stabilization

The CG process can be stabilized by using dual inequalities (Valério de Carvalho, 2005; Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016). In the following, we restrict ourselves to inequalities that are formulated only in the dual variables π_i that correspond to the demand fulfillment constraints (3.1b) of the model solved at Stage 1. (for the subsequent stages we consider the projection onto the m-dimensional space defined by the π-variables). Let D^* be the set of dual optimal solutions to the linear relaxation. For $\boldsymbol{t} \in \mathbb{Z}^m$ and $t \in \mathbb{Z}$, the *dual inequality* (DI) $\boldsymbol{t}^\top \pi \leq t$ is a *dual-optimal inequality* (DOI) if $D^* \subseteq \{\pi : \boldsymbol{t}^\top \pi \leq t\}$. A set of DIs is a set of *deep dual-optimal inequalities* (DDOIs) if at least one optimal dual solution $\pi^* \in D^*$ fulfills *all* inequalities of this set.

We first focus on Stage 1. and the B-VPP and 01-VPP. According to Heßler *et al.* (2018), *pair inequalities* (PIs) $\pi_h \geq \pi_i$ are DDOIs for all $h, i \in I$ with $w_h \geq w_i$ and $b_h \geq b_i$. In the primal formulation (3.1), these additional columns stand for the exchange of the larger item h by the smaller item i, i.e., a zero-cost column in the RMP with entries -1 and 1 for h and i, respectively. Moreover, *subset inequalities* (SIs) $-\pi_h + \sum_{i \in S} \pi_i \leq 0$ are defined for any item $h \in I$ and subset $S \subseteq I \setminus \{h\}$ with $\sum_{i \in S} w_i \leq w_h$ and $\sum_{i \in S} b_i \leq b_h$. We refer to a SI as (S, h). In general, SIs are not necessarily DOIs or DDOIs for the B-VPP and the 01-VPP. Nevertheless, their associated primal columns can be added to stabilize the RMP at the risk of a possible over-stabilization. Over-stabilization can be restored with a recovery procedure of low computational effort as described in detail in Gschwind and Irnich (2016).

At Stages 2.–4., the use of DIs is still possible when categories of items are kept separate, i.e., a SI for (S, h) can only be used if all items $i \in S$ and also item h all belong to the same, stage-dependent *subset of exchangeable items*. An overview of the applicable DIs at the different stages is provided in Table 3.4. For example, at Stage 2., the PIs and SIs exclusively exchange either items of standard products I^S or of refrigerated products $I^C \cup I^F$. At Stages 3. and 4., exchanges are only allowed among items of the same category.

At Stage 5., we have to take into account the following speciality. The transformation of a solution with packing and DI columns into a pure packing column solution may change the number of packed products in a pattern (to be taken into account at Stage 5a.). Likewise, an integrally packed product may be transformed into a split product and vice versa (to be taken into account at Stage 5b.). In order to ensure a valid objective value after the replacement, we have to modify the right-hand side of a DI from 0 to some integer value c, i.e., add a cost c to the corresponding primal DI column. Table 3.4 summarizes the different cases. For example, let $p \in P$ be a pattern and let $\sum_{i \in I} t_i \pi_i \leq c$ be a DI. If at Stage 5a. in the primal model both corresponding variables are positive, the following replace-

Stage/ objective	Subsets of exchangeable items	RHS of a SI defined by (S, h), i.e., cost c of the associated primal column
1.	I	0
2.	$I^S, I^C \cup I^F$	0
3. and 4.	I^S, I^C, I^F	0
5a.	I^S, I^C, I^F	$\lvert\{i \in I : t_i > 0\}\rvert - \chi_1(u_h)$
5b.	I^S, I^C, I^F	1

Table 3.4: Overview of DIs.

ment can be performed: Exchange item h by items $\{i \in I : t_i > 0\}$ in pattern p with original associated cost s_p (i.e., the number of packed products) yields a new pattern p' with associated cost $s_{p'} \leq s_p + \lvert\{i \in I : t_i > 0\}\rvert - \chi_1(u_h)$, where χ is the indicator function. The new pattern p' might consist of added items $\{i \in I : t_i > 0\}$ compared to pattern p and if $u_h = 1$ then item h vanishes in pattern p'.

Similarly, at Stage 5b., the (negative) number of integrally packed items for the new pattern p' can be estimated by $d_{p'} \leq d_p + 1$ because item h might change from an integrally packed item to a split item.

As discussed in Gschwind and Irnich (2016) in more detail, one may experiment with different strategies which DIs and at what point in time the DI columns are added to the RMP. On the one hand, DI columns can be added at the very beginning to the initial RMP and stay there during the entire CG process (*static*). On the other hand, only violated DIs can be added during the CG process (*dynamic*). We combine these two strategies which yields a *mixed* approach that adds some DI columns at the start and adds violated DIs later on.

We employ this mixed strategy in the following form: For the initialization of the first RMP, for each item i, the PI $\pi_i \geq \pi_j$ with $j = \arg\min_{k \in I \setminus \{i\}} \{|w_i - w_k| + |b_i - b_k| : w_i \geq w_k, b_i \geq b_k\}$ is added and all SIs with $|S| = 2$ are added. While solving the root node, violated PIs are added dynamically. At Stages 2.–5., we only keep DIs belonging to the subset of exchangeable items according to Table 3.4 (and remove the others). In the same spirit, we only dynamically add violated PIs belonging to these subsets.

3.4.4 Branching

We apply a single or a two-level branching scheme depending on the stage. At Stage 1., there is only one level where we apply the branching scheme suggested by Vanderbeck (1999). The idea is to formulate the 2D-VPP as a pure binary problem so that all pattern coefficients are also binary. Branching is then per-

formed choosing two disjoint subsets I^0, I^1 of items. In any fractional solution there exists a pair (I^0, I^1) for which the number of patterns having coefficients a_i equal to 0 (1) for all $i \in I^0$ ($\in I^1$) is fractional. For a detailed explanation and examples, we refer to Heßler *et al.* (2018).

At Stages 2.–4., there are two branching levels. First, we branch on the number of refrigerated trucks (3.2c), the number of frozen trucks (3.3c), and the number of mixed trucks (3.4c), respectively. Second, we apply the above described branching rules of Vanderbeck (1999).

At Stage 5., there are also two branching levels. First, we branch on the number of integrally packed products, i.e., when the value

$$\sum_{p=(a_i)\in P: a_i^p = q_i} a_i^p x_p \tag{3.7}$$

is fractional for an item $i \in I$. Note that this is a special case of Vanderbeck branching, which we apply subsequently at the second level.

At all levels, the branching variable/term is selected as one with fractional part closest to 0.5. Ties are resolved randomly.

Note that all first-level and second-level branching decisions can be implemented by either adding some linear constraint(s) to the RMP or by removing certain pattern subsets. Since the branching rule of Vanderbeck (1999) ensures integrality, the overall branching scheme is complete for all stages.

Finally, we use a depth-first tree exploration strategy. The intention is to find an integer solution as soon as possible. Since the lower bounds of the linear relaxations of formulations (3.1a)-(3.5) are often tight, we quickly obtain good and sometimes optimal solutions.

3.4.5 Acceleration techniques

When applying the policy splitting allowed, patterns with fewer split products are preferred over patterns with more split products. Therefore, it is beneficial to already generate patterns with very few split products (if any) at the Stages 1.–4. so that the initial RMP at Stage 5. mainly consists of variables representing patterns with many integrally packed products. To foster that predominantly patterns with only integrally packed products are generated, the SP is always first solved heuristically over a reduced SPPRC digraph. In this reduced digraph, each item $i \in I$ has only two associated parallel arcs $(0, 0, 0)$ and $(u_i w_i, u_i b_i, u_i \pi_i)$ so that for $u_i = q_i$ the item is either integrally packed or not packed at all. An example of a reduced graph is given in Figure 3.5. Only if no pattern with negative reduced cost is found in the reduced digraph, SPPRC SPs are solved again using the complete digraph.

This approach has two advantages: First, the 2D-KPs at the Stages 1.–4. are solved faster because in most iterations the reduced network provides negative reduced cost patterns. Secondly, Stage 5. is solved faster because the RMP already consists of beneficial columns. Computational results show that this approach significantly accelerates the CG process. For a computational analysis, we refer to Section 3.6.4.

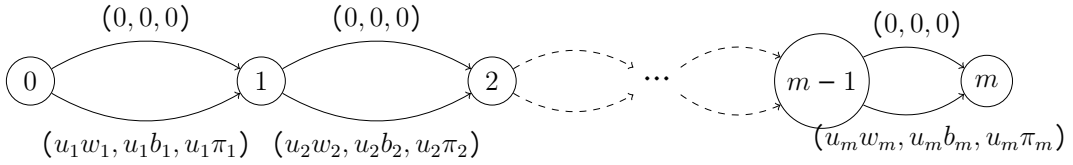


Figure 3.5: Reduced SPPRC digraph for objective 1. with otherwise identical attributes as depicted in Figure 3.2. The arcs $e \in E$ are shown with their weights and profit $(w(e), b(e), \sigma(e))$.

3.5 Heuristics

For real-world applications, user acceptance is a crucial success factor, especially for constructive heuristics, where the key users want to get some insights into the decision mechanism. Therefore, we developed various heuristic ideas and presented their underlying rationals as well as their strengths and weaknesses to the decision makers of the affected company. For comparison, we also re-implemented the strategy previously employed by the dispatchers, the so-called *Business Today* logic.

For our own development, we started with the well-known BFD, FFD, and WFD heuristics, all of them employing the concept of surrogate weight (Section 3.5.1). Then five additional heuristic approaches were developed in accordance to the real-world situation of our industrial partner (Section 3.5.2). Comparisons to results of the Business Today logic can be found in Section 3.6.

3.5.1 Basic single-class heuristics

At the beginning, we will introduce a number of heuristics which consider only a single class of items and do not distinguish between products of different categories. Also splitting products is not taken into account. At the end of this section, we will have obtained one combined heuristic which consists of running all heuristics described so far and taking the best solution. This combined heuristic

will serve as building block for a more complicated heuristic framework described in Section 3.5.4.

As pointed out in the introduction, we are facing a highly heterogeneous product range where the unit weight ρ_k of product $k \in K$ varies considerably. It is obvious that truck loads containing mainly heavy goods ($\rho_k \gg W/B$) will leave excess pallet spaces even if the weight capacity is exhausted. On the other hand, truck loads containing mainly lightweight goods ($\rho_k \ll W/B$) still provide excess weight capacity even if all pallet spaces are used. Thus, a "good" loading pattern should utilize both capacity dimensions by mixing heavy and lightweight products in a suitable way.

Based on this simple observation the transport planners previously created loading patterns filling one truck after the other in a FF manner. Each truck is manually assigned its load following a strict rule that alternately packs the heaviest and the lightest unpacked item as described in the following *Business Today* Algorithm 1.

Algorithm 1: Business Today

```
Sort item set I by increasing ρ_i.
Open truck 1 and set t ← 1.
while I ≠ ∅ do
  Take the last (first) item i from I in an odd (even) iteration.
  if i fits into truck t then
    Insert i into truck t.
  else
    Open a new truck t + 1 and insert i.
    t ← t + 1.
  end if
  Remove i from I.
end while
```

Practical experience showed: If each product fills only one pallet, i.e., $b_i = 1$ for all $i \in I$, then this procedure amounts to a pairwise combination of items with large and small unit weight ρ_i and shows reasonably good results. However, this is a rare case, and for all other, more general instances, inefficient solutions may occur. In the worst case, many large items (high w_i and b_i) with medium density ρ_i remain until the end, which leads to unnecessary additional trucks.

Naturally, Algorithm 1 suffers from neglecting the absolute weights of items and from the FF strategy of handling trucks. However, sorting items by decreasing weight w_i or decreasing number of pallets b_i would only be useful for instances with a clear dominance in one dimension. Our real-world test instances showed

that both constraints, weight and pallet capacity, can become active, i.e., there is no clear dominance of one of the two dimensions.

To overcome this dilemma, we suggest to apply the concept of a *surrogate weight* proposed by Caprara and Toth (2001). The surrogate weight of an item $i \in I$ constitutes a convex combination of relative weight and relative pallet number. For each $i \in I$, it is defined as

$$s_i = \lambda \frac{w_i}{W} + (1 - \lambda)\frac{b_i}{B}, \quad \text{where } \lambda = \frac{\sum_{i \in I} \frac{w_i}{W}}{\sum_{i \in I} \left(\frac{w_i}{W} + \frac{b_i}{B} \right)}.$$

Based on these surrogate weights, we can (almost) transform our packing problem into a classical one-dimensional bin packing problem and apply well-known standard heuristics. In particular, we take FFD, BFD, and WFD, and adapt them for the solution of our truck loading problem by using the surrogate weight s_i. In the following, we present further heuristic approaches based on the concept of a surrogate weight.

3.5.2 Block-based single-class heuristics

In this section, we introduce three heuristics called A, B, and C, which are all based on a fixed pattern of assigning blocks of items to trucks. Therefore, the lower bound of the number of trucks LB (see Section 3.3) is computed and LB trucks are opened. Then, the item set is partitioned into blocks of equal size LB. In each iteration, one block is selected and matched to the set of trucks, i.e., every item of the block is assigned to one of the trucks. The selection of blocks and the matching patterns differ between the three heuristics. If an item does not fit into the preferred truck, the item is moved to the next truck in the sequence (following a FF logic) or, if necessary, a new truck is opened, see Algorithm 2. Note that any new trucks opened during the execution of the heuristic only serve as a backup for loading items exceeding the capacity, while the matching of subsets remains restricted to the originally opened LB trucks.

All three heuristics are based on sorting the items by increasing surrogate weights s_i. Their details are informally described below followed by pseudocode. An illustrative example is given in Table 3.5 further below.

Heuristic A assigns the LB items with largest surrogate weights to the LB trucks and then adds the LB items with smallest surrogate weights in a reversed order, such that the items with $s_{\max}$ and $s_{\min}$ end up in the same truck number 1. These two steps are repeated iteratively for the remaining items as described in Algorithm 3.

Heuristic B also starts by assigning the LB items with largest surrogate weights to the LB trucks, but then proceeds by assigning the next largest (w.r.t. s_i) LB

items to the trucks in a mirrored order such that items with LBth and (LB + 1)-largest surrogate weight end up being assigned to the same truck number LB, see Algorithm 4.

Finally, Heuristic C repeats the first step of Heuristic A and iteratively assigns the items with largest surrogate weights to the LB trucks, always maintaining the same order of trucks, see Algorithm 5.

Algorithm 2: Subprocedure **Assign item i to truck t**

Try to insert i into an open truck by a FF strategy,
i.e., try all trucks $t, t+1, \ldots, \text{LB}$ and then trucks $1, 2, \ldots, t-1$ until item i fits.
If no such truck is found, insert i into a truck $\text{LB}+1, \text{LB}+2, \ldots$ by a FF strategy, opening a new truck if necessary.

Algorithm 3: Heuristic A

Calculate lower bound LB and open LB trucks.
Sort item set I by increasing s_i.
Partition set I into $H := \lceil m/\text{LB} \rceil$ subsets I_j of cardinality LB according to the sorting. {For simplicity of notation we assume that H is even.}
$j \leftarrow 1$, $h \leftarrow H$
while $j < h$ **do**
  for $i \leftarrow$ LB downto 1 **do**
    Assign item i in set I_h to truck $\text{LB} - i + 1$.
  end for
  $h \leftarrow h - 1$
  for $i \leftarrow 1$ to LB **do**
    Assign item i in set I_j to truck i.
  end for
  $j \leftarrow j + 1$
end while

To illustrate the different heuristics, an example with 20 items is given in Table 3.5. Obviously, Heuristic A exhibits more similarities to the Business Today logic, including the feature of not keeping the items with lowest s_i until the end. Heuristics B and C overcome that issue. Besides that, Heuristic B aims at uniformly filling all trucks. According to the basic intuition of how good solutions might look like, this should reduce the probability that any smaller item does not fit into its *preferred truck*. On the other hand, Heuristic C provides gaps of different sizes in the LB different trucks, which may offer more room to maneuver if an item does not fit into its assigned truck.

Algorithm 4: Heuristic B

Calculate lower bound LB and open LB trucks.
Sort item set I by increasing s_i.
Partition set I into $H := \lceil m/\text{LB} \rceil$ subsets I_j of cardinality LB according to the sorting. {For simplicity of notation we assume that H is even.}
$j \leftarrow H$
while $j \geq 1$ **do**
  for $i \leftarrow$ LB downto 1 **do**
    Assign item i in set I_j to truck LB $- i + 1$.
  end for
  $j \leftarrow j - 1$
  for $i \leftarrow 1$ to LB **do**
    Assign item i in set I_j to truck i.
  end for
  $j \leftarrow j - 1$
end while

Algorithm 5: Heuristic C

Calculate lower bound LB and open LB trucks.
Sort item set I by increasing s_i.
Partition set I into $H := \lceil m/\text{LB} \rceil$ subsets I_j of cardinality LB according to the sorting.
for $j \leftarrow H$ downto 1 **do**
  for $i \leftarrow$ LB downto 1 **do**
    Assign item i in set I_j to truck LB $- i + 1$.
  end for
end for

	Business Today					Heuristic A				Heuristic B				Heuristic C			
truck \ iter.						1	2	3	4	1	2	3	4	1	2	3	4
$t = 1$	20	1	19	2	18	20	1	15	6	20	11	10	1	20	15	10	5
$t = 2$	3	17	4	16		19	2	14	7	19	12	9	2	19	14	9	4
$t = 3$	5	15	6	14		18	3	13	8	18	13	8	3	18	13	8	3
$t = 4$	7	13	8	12	9	17	4	12	9	17	14	7	4	17	12	7	2
$t = 5$	11	10				16	5	11	10	16	15	6	5	16	11	6	1

Table 3.5: Comparison of solutions produced by the Business Today algorithm and Heuristics A, B, and C. The example has 20 items $I = \{1, 2, \ldots, 19, 20\}$ and LB = 5. Items are numbered according to their surrogate weight in ascending order, i.e., item 1 is the smallest item and item 20 is the largest item.

3.5.3 Density-based single-class heuristics

The third group of heuristics for items of the same class is based on the unit weight ρ considerations discussed at the beginning of Section 3.5.1. Recall that "good" loading patterns, in general, utilize both capacity dimensions and do not leave large excess capacity of pallet space if weight capacity is used to the limit, and vice versa. Therefore, we can define a desired *average density* for an ideal load of a truck, namely the ratio $optavg_\rho = W/B$. Clearly, a loading pattern P_t of a truck t with an average density $avgr_t := \sum_{j \in P_t} w_j / \sum_{j \in P_t} b_j$ close to $optavg_\rho$ and total weight close to the capacity W also fills the available pallet space close to the limit B, and vice versa.

Based on this simple observation, we introduce two heuristics D and E. In contrast to heuristics A, B, and C, they both consider trucks one after the other and for each considered truck t they add the item yielding a new average density $avgr_t$ as close as possible to the target value $optavg_\rho$. Of course, the item also has to fit into the truck at hand. Ties are broken by choosing the item with the largest weight, because smaller items can be expected to fit more easily in later iterations.

Heuristics D and E differ from each other only by the weight measure for the initial sorting step. Note that the sorting only determines the first item put into every truck. Heuristic D sorts by the surrogate weight s_i ensuring that "large" items are assigned first and will not give rise to trucks with low utilization towards the end of the packing procedure. On the contrary, heuristic E focuses on outliers with an unusual density that are potentially hard to assign. Therefore, items are sorted in decreasing order of unit weight ρ_i. By the same rationale also the reverse ordering could be a reasonable choice, but for our test data, pretests showed that this ordering is less effective.

Algorithm 6: Heuristic D (E)

Sort items I by decreasing s_i (ρ_i).
Let $P_t \leftarrow \varnothing$ be the loading of all trucks t.
$t \leftarrow 0$
while $I \neq \varnothing$ **do**
  $t \leftarrow t + 1$
  Take the first i from I and insert i into P_t.
  Set $avgr_t = \rho_i$ and remove i from I.
  while items exist that fit into P_t **do**
    $i' \leftarrow \arg\min_{i \in I} \left\{ \left| \frac{w_i + \sum_{j \in P_t} w_j}{b_i + \sum_{j \in P_t} b_j} - optavg_\rho \right| : i \text{ fits into } P_t \right\}$
    Insert i' into P_t and remove i' from I.
  end while
end while

Heuristics D and E could be immediately extended to consider all trucks instead of only the current truck t for inserting items. In that case, the computation of i' would have to be extended to taking the arg min over all currently open trucks, which means that an item is inserted into a truck where the minimal deviation from $optavg_\rho$ can be realized. Indeed, we also implemented this extension with the initialization that the first LB items (according to the given ordering) are packed on LB trucks. However, since improvements were only marginal and occurred only for a small subset of instances, we decided to stick to the simpler standard version described above.

3.5.4 Multi-class heuristic scheme of policy splitting forbidden

All heuristics described in the previous sections focus on creating good patterns utilizing both capacity dimensions, but they take neither items of different product categories packed into different compartments of a truck nor item fragmentation into account, while the latter is considered in the following Section 3.5.5. Accordingly, we now present a heuristic scheme which is based on the single-class heuristics presented before. These are embedded into a more complicated algorithmic framework consisting of various decomposition and re-merging steps. Since the running times of the single-class heuristics are rather low, we

perform the task of solving a single-class problem for a certain subset of item $\mathcal{I} \subseteq I$ as a subroutine by running all nine heuristics from Sections 3.5.1, 3.5.2, and 3.5.3 and taking the best solution found. More precisely, we run the Business Today algorithm, FFD, BFD, WFD, and Heuristics A to E. The result (=number of trucks) derived by this combined single-class heuristic is denoted by $z_{\text{heu}}(\mathcal{I})$. Additionally, this combined heuristic is executed with a given starting solution of prepacked trucks, i.e., some trucks have reduced residual capacities. We refrain from listing all the details of adapting the single-class heuristics to this situation.

In Section 3.3, the lower bounds $\text{LB}, \text{LB}^S, \text{LB}^{C,F}$, and LB^F for the corresponding subsets of items are introduced. Applying a single-class heuristic only for standard items and then separately only for refrigerated items yielding heuristic solution values z^S_{heu} and $z^{C,F}_{\text{heu}}$, respectively, we can conclude the following: If $z^{1.}_{\text{sep}} := z^S_{\text{heu}} + z^{C,F}_{\text{heu}} = LB$, then we have reached an optimal solution w.r.t. the objectives of Stages 1. and 2. with $z^{1.} = z^{1.}_{\text{sep}}$ and $z^{4.} = 0$.

Otherwise, if $z^S_{\text{heu}} + z^{C,F}_{\text{heu}} > LB$, we cannot guarantee optimality of the merged solution without mixed trucks. In this case, it makes sense to gradually introduce mixed trucks while trying to reduce the total number of trucks. As already used in Section 3.4, constraint (3.4c), the lower bound for mixed trucks ($z^{4.}$) is given by $\text{LB}^{\text{mixed}} = \text{LB}^S + \text{LB}^{C,F} - \text{LB}$ for solutions that are optimal w.r.t. Stages 1. to

3.. Therefore, $LB^{mixed} = 0$, if $LB^S + LB^{C,F} = LB$ and $LB^{mixed} > 0$, otherwise.

The heuristic scheme striving for a good solution of the lexicographic optimization problem defined by Stages 1.–4. with splitting forbidden works in three phases which are described in Algorithm 7. Phase 1 builds up a solution starting with frozen products. After packing I^F by the combined single-class heuristic (following the Stage 4. objective), the cooled items are added which might require additional trucks (Stage 3. objective). Then a separate solution for the standard item set I^S is determined. Taking the union of the two sets of trucks gives a first solution, where standard and refrigerated items are packed in separate trucks (Stage 2. objective). If the resulting number of trucks z_{best} equals the overall lower bound LB we stop (**STOP 1**), since we have found a solution which is optimal w.r.t. Stages 1. and 2.

Otherwise, we try to improve the currently best solution in Phase 2: Therefore, we allow mixed trucks and pack standard products also into refrigerated trucks. To reach a better starting position for adding also large standard items to the refrigerated trucks given from Phase 1, but keeping the number of mixed trucks small, the load of the refrigerated trucks is *regrouped* at the beginning of Phase 2 with the goal of obtaining some refrigerated trucks with a low load and others which are almost fully packed with cooled products. This is done by subprocedure **Regroup**, which tries to empty the least loaded refrigerated truck by moving its items (in decreasing order of surrogate weight) to other refrigerated trucks selected by a BF strategy. If the number of frozen products is small, we try to keep them together in the regrouping in the spirit of the Stage 3. objective. Therefore, all frozen products contained in the current truck are considered as a single product and moved to a new truck together. Only if this artificial product does not fit, the frozen products are reassigned individually. This regrouping is applied to all trucks in an increasing order of surrogate load. Then the standard items I^S are added to the regrouped refrigerated trucks by the combined single-class heuristic. In this way, we aim to find a packing with a lower number of trucks (Stage 1.), but without increasing the number of refrigerated trucks (Stage 2.). Again, we stop if the new solution matches the lower bound LB^I (**STOP 1**).

A further reduction of the total number of trucks is sought in Phase 3 at the cost of increasing the number of refrigerated trucks. Again we try to improve the possibility of adding standard products to refrigerated trucks. Therefore, we take the refrigerated truck with lowest load w.r.t. the surrogate weight and split its content on two empty trucks, thereby incrementing the number of refrigerated trucks by one. The splitting is performed in a balanced way by assigning the items by a WFD strategy (w.r.t. surrogate weights). Again, all frozen items are treated as one product, if the truck contains both frozen and cooled products. Then the standard items are again added by the single-class heuristic. If the total number of

trucks reaches the lower bound LB we stop the procedure. Otherwise, we unpack all standard items again and iterate, taking the refrigerated truck with lowest load that was not split before. If all refrigerated trucks given at the beginning of Phase 3 were split, another iteration is started permitting a second split operation for each refrigerated truck given at the end of the first iteration. Further iterations along this rule are possible, until at some point the number of refrigerated trucks exceeds the lower bound for all items (**STOP 2**). In this case, we also run the single-class heuristic for all items I to reach a possible improvement of the Stage 1. objective.

3.5.5 Multi-class heuristic scheme for policy splitting allowed

If the multi-class heuristic of Section 3.5.4 reaches a solution where all lower bounds for the objectives of Stages 1.–4. are reached, obviously the result cannot be improved by splitting products. However, if this is not the case, item fragmentation offers potential for improvements.

In the following, we consider the two different splitting objectives introduced in the Introduction.

3.5.5.1 Stage 5a.: Minimizing the number of splits

In this setting it is not a-priori clear which items should be split and in which way to gain the largest improvement of the solution. We have chosen to incorporate item splitting as a post-processing step applied to a solution derived from the multi-class heuristic of Section 3.5.4. In every iteration of the main loop of our Algorithm 8 we try to empty the truck with lowest surrogate load and thus reduce the total number of trucks by one. If this turns out to be impossible, the procedure is stopped. Otherwise we proceed to the next iteration, unless we have reached a solution with LB_1 trucks, which proves optimality of objective 1. and also lets us stop the procedure.

Every iteration for the truck t with minimal surrogate weight consists of two parts: At first (**for-loop**) we try to further reduce the load of truck t by moving items (without splitting them) to other trucks by a BF strategy. Then (in the **while-loop**) we iteratively take the largest (w.r.t. surrogate weight) remaining item i' in truck t and split it, i.e., we pack the largest possible fractional part of it, expressed as number of pallets, to one of the other trucks. Formally, the largest pallet number h' is calculated, such that h' pallets out of the $b_{i'}$ pallets constituting i' can be loaded on some truck t'. Ties among trucks are broken by a BF rule. The remaining part of item i' consisting of $b_{i'} - h'$ pallets is treated like an individual item in t. By moving the largest possible number of pallets, it can be expected that the total number of splits remains small. If no single pallet can be moved

Algorithm 7: Multi-class heuristic

Compute LB.
Phase 1: {*pack refrigerated and standard items separately*}
Compute z^F_{heu}
Compute $z_{\text{refrig}} \leftarrow z^C_{\text{heu}}$ starting from the packing implied by z^F_{heu}
Compute z^S_{heu} {*for empty trucks*}
$z_{\text{best}} \leftarrow z_{\text{refrig}} + z^S_{\text{heu}}$ {*current best solution*}
if $z_{\text{best}} = \text{LB}$ **then**
  STOP 1.
end if
Phase 2: {*reshuffle the refrigerated items and then add standard items*}
Regroup(all z_{refrig} refrigerated trucks)
Compute z^S_{heu} starting from the packing of the z_{refrig} refrigerated trucks returned by **Regroup**.
$z_{\text{best}} \leftarrow \min\{z_{\text{best}}, z^S_{\text{heu}}\}$
if $z_{\text{best}} = \text{LB}$ **then**
  STOP 1.
end if
Phase 3: {*Iteration: expand the number of refrigerated trucks by distributing the load of one truck on two trucks and add standard items*}
Remove all standard items I^S from the z_{refrig} refrigerated trucks returned by **Regroup**.
repeat
  $z_{\text{current}} \leftarrow z_{\text{refrig}}$
  for all z_{current} refrigerated trucks t opened so far in increasing order of load w.r.t. surrogate weights **do**
    Open a new (empty) truck $z_{\text{refrig}} + 1$
    $z_{\text{refrig}} \leftarrow z_{\text{refrig}} + 1$
    Take all (refrigerated) items currently loaded on truck t and pack them on the two trucks t and z_{refrig} by a WFD strategy. {*gives a balanced bipartition*}
    Compute z^S_{heu} starting from the current packing of the z_{refrig} refrigerated trucks.
    $z_{\text{best}} \leftarrow \min\{z_{\text{best}}, z^S_{\text{heu}}\}$
    if $z_{\text{best}} = \text{LB}$ **then**
      STOP 1.
    end if
    if $z_{\text{refrig}} \geq \text{LB}$ **then**
      Compute z_{heu} {*for empty trucks*}
      $z_{\text{best}} \leftarrow \min\{z_{\text{best}}, z_{\text{heu}}\}$
      STOP 2.
    end if
    Remove all standard items I^S from the trucks
  end for
until false

Subprocedure **Regroup**$(P_1, \ldots, P_T)$
Sort the T trucks in increasing order of load w.r.t. surrogate weights
for $t = 1$ **to** T **do**
  for all items i in P_t in decreasing order of surrogate weights **do**
    Try to pack item i in trucks $\{P_{t+1}, \ldots, P_T\}$ by a BF strategy (w.r.t. surrogate weights)
  end for
end for

Algorithm 8: Multi-class heuristic: Stage 5a., minimizing the number of splits

Compute LB_1
Execute Algorithm 7 returning $T := z_{best}$ trucks with loads $P_1, \ldots, P_T$
Sort the T trucks in increasing order of load w.r.t. surrogate weights
$t \leftarrow 1$
repeat
  for all items i in P_t in decreasing order of surrogate weights **do**
    Try to pack item i into another truck by a BF strategy (w.r.t. surrogate weights)
  end for
  while $P_t \neq \emptyset$ **do**
    Let i' be the item in P_t with largest surrogate weight
    {*find the truck t' where the largest number of pallets of i' can be added*}
    $h' \leftarrow \max_\tau\{h : h \cdot w_{i'}/b_{i'} + \sum_{j \in P_\tau} w_j \leq W,\ h + \sum_{j \in P_\tau} b_j \leq B, \tau \in \{1, \ldots, T\} \setminus \{t\}\}$
    attained for t'
    if $h' = 0$ **then**
      {*Truck t cannot be emptied*}
      Return the solution with z_{best} trucks (as given at the start of the current iteration of **repeat**)
      STOP.
    end if
    Move h' pallets of product i' from P_t to $P_{t'}$
  end while
  Remove P_t
  $z_{best} \leftarrow z_{best} - 1$
  if $z_{best} = LB_1^I$ **then**
    Return current solution with z_{best} trucks
    STOP.
  end if
  $t \leftarrow t + 1$
until false

Algorithm 9: Multi-class heuristic: Stage 5b., minimizing the number of split products

Compute LB_1, LB_1^S, $LB_1^{C,F}$ and LB_1^{mixed}
Sort all items $i \in I$ in decreasing order of b_i
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  Split item i into b_i separate items
  Execute Algorithm **Multi-class heuristic** returning z_{best}
until all lower bounds LB_1, LB_1^S, $LB_1^{C,F}$ and LB_1^{mixed} are reached or $b_{i+1} = 1$ or $i = m$
Return z_{best}

from truck t (i.e., $h' = 0$), the whole procedure is stopped. Otherwise, truck t is completely empty and can be removed.

3.5.5.2 Stage 5b.: Minimizing the number of split products

In this case it does not matter whether a large or small item i is split and whether i is split into two fragments or the maximum number of b_i fragments. It is intuitively clear that the largest potential for improvement is given by splitting large products completely into separate pallets. Therefore, in our Algorithm 9 we iterate over all items in decreasing order of b_i. Every considered item is split into separate pallets and then the multi-class heuristic from Section 3.5.4 is executed. The iterative process is stopped in any of the following three cases: (i) the multi-class heuristic stops with a solution matching all lower bounds for the objectives of Stages 1.–4.. Since we consider split products, we can only utilize the volume based lower bound LB_1; (ii) all products consisting of more than one pallet were split; (iii) all products were considered and split.

3.6 Computational results

In this section we will report computational results for all algorithms described before. The experiments are based on real-world instances and on more difficult instances generated from these by reducing their data to more difficult item sets (see Section 3.6.2). We will first report in Section 3.6.3 results for the heuristic approaches described above in Section 3.5. To analyze the performance of the algorithm we compare the heuristic with the exact solutions in detail. Then we will illustrate the performance of the exact BaP algorithm in Section 3.6.4. Furthermore, we also analyze in that section the impact of adding the heuristic solutions as columns to the initial RMP.

3.6.1 Details of the implementation

All heuristic algorithms presented in Section 3.5 are implemented and tested in Python 3.3.7 on a standard PC equipped with an Intel® Core™ i5-3210M processor with 2.5 GHz and 4 GB of RAM. Note that the heuristics are also used in the daily planning task of our industrial partner. For this application they were originally implemented within the SAP ERP system of the company in the SAP-internal programming language ABAP. However, for test purposes we re-implemented the heuristics in Python.

The BaP algorithm described in Section 3.4 is implemented in C++ and compiled in release mode into a single-thread code under MS Visual Studio 2015.

The experiments are conducted on a standard PC with an Intel® Core™ i7-5930k processor clocked at 3.5 GHz and 64 GB of RAM. The RMP is solved at each column-generation iteration by means of CPLEX 12.9. Moreover, CPLEX is called after the solution of each branch-and-bound node as a primal MIP-based heuristic solver using the so far generated columns. CPLEX's default values are kept for all parameters. The time limit to solve each stage is set to 10 minutes (600 seconds). In case a stage cannot be solved within the time limit, the computation is stopped and following stages are not considered.

3.6.2 Benchmark instances

We consider 80 real-world instances with 35 to 430 items provided by our partner from the European food and beverage industry. The cardinality bound of each truck is 33 pallets (a common industry standard) and the weight capacity is 22.8 or 24.5 tons. The weight of each item is rounded up with an accuracy of 10 kg. Preliminary tests show that both constraints, weight and cardinality, can become active, i.e., there is no clear dominance of one of the two dimensions. Not all instances contain all types of products. Especially, frozen products often appear in rather small quantities or even miss completely. A detailed summary of our test instances can be found in Table 3.6.

Instance type	real-world	generated
number of instances	80	30
average number of items	144	153.9
average weight of items (in 10 kg)	93.7	114.6
average pallet number of items	3.8	6.7
number of instances with standard products	80	30
number of instances with cooled products	73	30
number of instances with frozen products	31	16

Table 3.6: Overview of benchmark instances.

During the computational study it turned out that for most of the instances the optimal solutions for Stages 1.–4. do not require any splitting and thus the objective of Stage 5. would be optimized by default. The reason for this behavior is that the instances also contain rather small items which fill up trucks without being split. Therefore, we distilled more difficult new instances from the given real-world instances avoiding this effect in the following way. The 80 real-world instances are divided into 10 groups each with 8 instances. For each group, we selected 5, 10 or 15 items with the highest weight and/or the highest number of used pallets of

each instance forming a new instance, respectively. In total, we generated 30 new instances with up to 80, 160 or 240 items, respectively. All instances are available on the website `http://logistik.bwl.uni-mainz.de/benchmarks.php`.

3.6.3 Results of the heuristic approach

In this section, we report results of the heuristic approach. Table 3.7 shows the key results for the pure heuristic approach. For 56 out of the 80 real-world instances, all lower bounds for objectives 1.–4. are already achieved without splitting. The 24 remaining instances provide at least a theoretical improvement potential, which is exploited in 18 cases. For 13 of them, even the lower bounds for objectives 1.–4. are achieved if items are allowed to be split. While for the real-world instances exactly 70 % could be solved to the lower bounds for objectives 1.–4., this was possible only for 10 % (3 out of 30) of the generated instances, all without splitting.

Instance type	real-world	generated
number of instances	80	30
Business Today logic achieved LB	21	0
heuristics outperform Business Today logic	59	30
heuristics achieve LB without split	56	3
splitting improved heuristic solution	18	25
improvements led to reaching LB	13	21
no improvement by splitting	6	2
heuristic runtime < 1 sec.	46	3
heuristic runtime 1-10 sec.	27	8
heuristic runtime 10-30 sec.	4	7
heuristic runtime > 30 sec.	3	12
average number of trucks Business Today	23.1	45.8
average number of trucks heuristic unsplit	22.2	42.2
average number of trucks heuristic split	22.1	40.6

Table 3.7: Results of the heuristic approach.

However, the share of solutions that could be improved by applying splitting is far higher (25 out of 27) and the same is valid for the number of instances where that improvements even led to the lower bounds for objectives 1.–4. This higher improvement is not surprising, as the average pallet size of items is more than 75 % larger for generated instances and thus the effect of splitting items into

separate pallets is much stronger. The higher difficulty of generated instances is also reflected in the running times of the heuristics.

For none of the 110 instances, the original Business Today logic outperformed the heuristic approach, but for 89 instances, the heuristics performed better. For the other 21 (all real-world) instances, both the heuristics and the Business Today logic reached the lower bounds.

For a further performance analysis, we compare the objective values of the heuristic approach with the optimal objective values computed by the BaP approach. Detailed results can be found in Table 3.8. The table entries have the following meaning:

#opt: number of instances solved to proven optimality at the specific stage;

$z_H = z^*$: number of instances in which the heuristic approach finds the optimal solution; instances with non-optimal heuristic solutions in former stages are not considered;

$z_H > z^*$: number of instances in which the heuristic approach does not find the optimal solution; instances with non-optimal heuristic solutions in former stages are not considered;

$\mathbf{gap}_H$: average gap between heuristic and exact solution, we use the gap $z_H - z^*$ instead of the relative gap because the optimal solution can be $z^* = 0$ in Stages 4.–5.; recall that solution values for Stages 1.–4. represent number of trucks, but split parameters for Stage 5.

#implicit: number of instances for which the heuristic and optimal solution implicitly coincide, or where the current stage is not active (e.g., no Stage 3. for instances without any frozen items)

Σ-$z_H = z^*$: total number of instances in which the heuristic approach finds the optimal solution up to the specific stage: $(\Sigma\text{-}z_H = z^*) := (z_H = z^*) + (\#\text{implicit})$.

For a better understanding, we describe the entries in the second row of Table 3.8 in detail. At Stage 2., the branch-and-price algorithm solves 64 instances to proven optimality. For 60 (3) instances, the heuristic approach could (not) find the optimal solution. For one instance, we cannot compare the heuristic and exact approach at Stage 2. because the solution at Stage 1. differs for this instance. For 6 instances without any refrigerated items, the heuristic approach finds the optimal solution at Stage 1. so that Stage 2. is solved implicitly. Including instances without refrigerated products, the heuristic and exact solution coincide for in total 66 instances.

For the policy splitting forbidden, the heuristic solution is optimal for 58+6 of 68+18 instances. Interestingly, in most of the cases, the heuristic approach fails to

splitting	class	Stage	#opt	$z_H = z^*$	$z_H > z^*$	gap_H	#implicit	Σ-$z_H = z^*$
forbidden	real-world	1	76	71	5	0.1	0	71
		2	64	60	3	<0.1	6	66
		3	26	24	1	<0.1	41	65
		4	61	52	4	0.3	6	58
	generated	1	30	15	15	0.6	0	15
		2	22	9	5	0.9	0	9
		3	11	4	0	<0.1	5	9
		4	18	6	2	1.3	0	6
allowed	real-world	1	73	73	0	0.0	0	73
		2	64	64	0	0.0	7	71
		3	27	27	0	0.0	44	71
		4	60	60	0	0.0	7	67
5a.		5	62	50	6	98.3	0	50
5b.		5	63	48	6	3.9	0	48
	generated	1	28	28	0	0.0	0	28
		2	27	25	2	0.2	0	25
		3	15	15	0	0.0	10	25
		4	25	24	0	<0.1	0	24
5a.		5	7	3	4	80.8	0	3
5b.		5	8	3	5	32.2	0	3
	Total		767	661	58	9.5	126	

Table 3.8: Comparison between the heuristic and the exact solution.

find the optimal solution at the first stage. Especially for the generated instances, only 15 of 30 instances are solved to optimality. For the policy splitting allowed, the heuristic approach performs well with 67+24 of 67+25 instances solved optimally up to Stage 4. The absolute deviance gap_H is low up to Stage 4. with a maximum of 0.2, but the values are big for Stage 5. With 6/6+4/5 of 62/63+7/8 non-optimal instances, the heuristics fail for most of the instances at Stage 5.

3.6.4 Results of the branch-and-price algorithm

In this section, we report the results of our BaP algorithm. In general, reaching a stage of the lexicographic optimization task triggers the execution of a BaP algorithm solving the respective model (3.1)–(3.4), and possibly (3.5a)–(3.5b) or (3.5c)–(3.5d). However, for some instances, not all stages become *active*, e.g., if there are no frozen products in the instance or if a stage is solved implicitly during the preceding stage. Therefore, the total number of instances per stage to be reported in the computational experiments may be different for every stage. Moreover, the different BaP algorithms of the stages may have different success rates and thus will lead to a different number of instances remaining for the successive stages. This makes the comparison of different algorithmic approaches a delicate task. The following analyses may, therefore, seem somewhat less transparent but our focus is on the correct interpretation of the results we obtained.

The construction of the initial RMP works as follows. As the first stage is a vector packing problem without additional constraints, we start with the same initial RMP as described in (Heßler *et al.*, 2018) by adding unit vectors, columns with only one non-negative coefficient $a_i^p = u_i$, and 200 additional columns resulting from variants of the FF and BF heuristics. At Stage 2., we add other 200 columns resulting from the FF and BF heuristics that first assign items to bins that already contain the same product type. Moreover, the same FF and BF variants used in Stage 1. are performed on restricted item sets by considering standard or refrigerated products separately. Analogously, Stage 3. applies the FF and BF variants for cooled or frozen products separately. At Stage 4.–5., no additional columns are added initially to the RMPs.

For each stage, the computation time is restricted to a maximum of 10 minutes (600 seconds). If a stage cannot be solved within the time limit, we cannot solve the instance exactly. Therefore, the computation is stopped at this stage and the following stages are not considered. If an instance has no frozen or no refrigerated products at all, Stage 3. or Stages 2.–4. are skipped, respectively.

At first, the BaP algorithm is tested employing the results of the heuristic approach, i.e., the corresponding columns are added to the initial RMP. We refer to this variant as `base`. To analyze the impact of the heuristic solution, we compare the `base` variant to a variant without using the heuristic solution. Moreover,

the impact of the lower bounds enforced by constraints (3.2c), (3.3c), and (3.4c), and the graph reduction presented in Section 3.4.5 are tested. We refer to these variants as `without heuristic solution`, `without lb`, and `without reduced graph`, respectively. The results are summarized in Table 3.9. The table entries have the following meaning:

class: class of instances; real-world or self-generated; for details see Section 3.6.2;

splitting: allowed or forbidden; in case of splitting allowed, either the number of splits (objective 5a.) or the number of split products (objective 5b.) is minimized, see Section 3.4.1;

#inst: number of active instances;

#inst_cons: number of active instances considered at the current stage; instances that are not solved to proven optimality in a previous stage are not considered;

#opt: number of instances (out of #inst_cons) solved to proven optimality at the specific stage within 10 minutes (600 seconds);

#implicit: number of instances for which the current stage is already implicitly solved by the result of a previous stage, or where the current stage is not active (e.g., at Stage 3. for instances without any frozen items);

Σ-#opt: number of instances solved to proven optimality at the current stage: Σ-#opt:= #opt+#implicit

$\bar{T}$**:** average computation time in seconds over #inst_cons instances; unsolved instances are taken into account with the time limit of 10 minutes (600 seconds);

$\bar{T}_{LP}$**:** average computation time of the linear relaxation in seconds; unsolved linear relaxations are taken into account with the time limit of 10 minutes (600 seconds);

gap: average gap between the upper and lower bound; we use the gap $UB - LB$ instead of the relative gap because the objective value can be zero at Stages 4.–5.

We summarize and interpret the results as follows: For the policy splitting forbidden, 68 of 80 real-world and 18 of 30 self-generated instances are solved to optimality. In particular, the algorithm performs well at Stage 3. with an average computation time of around one minute. In comparison, the policy splitting allowed is more difficult than splitting forbidden. The performance at Stage 1. is similar for both policies with 101 solved instances (splitting allowed) instead of 106 (splitting forbidden). Up to Stage 4., 67 of the 80 real-world and 25 of the 30 generated instances are solved which is $(-1) + 3$ instances more compared to the policy splitting forbidden. The performance of the algorithm at Stage 5., where

62/63+7/8 instances are solved, is almost the same for both objectives 5a. and 5b. with similar average solution time. As can be expected, Stage 5. is particularly difficult for the generated instances. Especially, the average deviation (gap) at Stage 5a. is very high because in many cases the root node cannot be solved or the solution of the preceding stage is poor.

To analyze the impact of the heuristic solution, we compare the variants `base` and `without heuristic solution`. For the policy splitting forbidden, the results hardly change when the initial columns from the heuristic are omitted. In total, only one instance less can be solved and the average running times differ by less than 5 seconds. Therefore, the use of the heuristic solution provides only a small advantage for the policy splitting forbidden. In contrast, for the policy splitting allowed, the effect of adding the heuristic solution is more significant, as it helps to solve 6/7+2/3 additional instances, respectively.

Adding the additional constraints (3.2c), (3.3c), and (3.4c) to ensure lower bounds on the number of refrigerated, frozen, and mixed trucks, respectively, has a strong impact on the performance of the BaP algorithm. Without adding these lower bounds, only 45+13 instances of the policy splitting forbidden and 38/39+3 instance of the policy splitting allowed can be solved. In total, around one-third of the instances cannot be solved without these bounds. Moreover, the running time increases significantly for all stages and problem variants.

For the policy splitting allowed, using a reduced graph as described in Section 3.4.5 has a positive impact for the Stages 1.–4. Already at Stage 1., 10+9 instances less are solved if the graph reduction is omitted. Overall, 8/9+1/2 instances less are solved.

3.7 Conclusion

Packing pallets of products from industrial producers into trucks for shipment to wholesalers is an important daily task for supply chain management. Reducing the number of trucks, even if only by one, or rearranging the package plan to diminish the number of the more costly trucks carrying cooled or frozen products, gives a valuable impact on costs as well as emissions, since the supply operation is usually performed daily or several times a week. Thus, the effort spent on optimizing the packing operation yields savings that will add up over a year to numbers of considerable economic relevance.

In this contribution, we consider a real-word packing problem where products should be packed into trucks such that a five-level lexicographic objective function is minimized. Feasibility is defined by a weight and a volume (=number of pallets) constraint for every truck. This special variant of a multi-objective optimization problem considers the total number of required trucks as main objective dominat-

			base						without heuristic solution								without 1b								without reduced graph							
splitting	class	Stage	#inst	#inst_cons	#opt	#implicit	Σ-#opt	$\bar{T}$	$\bar{T}_{LP}$	gap	#inst_cons	#opt	#implicit	Σ-#opt	$\bar{T}$	$\bar{T}_{LP}$	gap	#inst_cons	#opt	#implicit	Σ-#opt	$\bar{T}$	$\bar{T}_{LP}$	gap	#inst_cons	#opt	#implicit	Σ-#opt	$\bar{T}$	$\bar{T}_{LP}$	gap
forbidden	real-world	1	80	80	76	0	76	65.7	30.1	0.1	80	**77**	0	**77**	59.1	32.6	<0.1	80	75	0	75	91.4	58.9	0.1								
		2	73	69	**64**	7	**71**	79.8	21.3	0.9	70	62	7	69	99.9	21.3	1.6	68	63	7	70	136.2	88.8	0.6								
		3	31	26	**26**	45	**71**	61.4	40.0	0.0	25	25	44	69	58.3	38.0	0.0	26	25	44	69	101.9	68.4	0.3								
		4	73	64	**61**	7	**68**	65.3	32.6	0.1	62	59	7	66	69.6	23.4	<0.1	62	38	7	45	261.3	34.5	0.4								
	generated	1	30	30	**30**	0	**30**	47.2	40.2	0.0	30	**30**	0	**30**	49.5	39.9	0.0	30	**30**	0	**30**	80.2	73.6	0.0								
		2	30	30	22	0	22	192.4	33.7	3.3	30	**23**	0	**23**	186.4	33.1	2.5	30	**23**	0	**23**	187.2	79.9	4.5								
		3	16	11	11	11	22	22.3	13.1	0.0	12	**12**	11	**23**	45.6	16.6	0.0	11	11	12	**23**	28.0	20.0	0.0								
		4	30	22	18	0	18	132.6	18.0	0.2	23	**19**	0	**19**	130.9	18.8	0.2	23	13	0	13	283.4	29.3	0.5								
allowed	real-world	1	80	80	**73**	0	**73**	131.4	107.9	1.4	80	71	0	71	139.7	106.5	1.4	80	**73**	0	**73**	131.3	107.9	1.4	80	63	0	63	235.8	225.0	6.5
		2	73	66	**64**	7	**71**	114.8	75.8	0.5	64	61	7	68	117.8	75.7	1.2	66	61	7	68	179.8	147.0	0.9	56	55	7	62	94.3	57.3	0.1
		3	31	27	**27**	44	**71**	170.5	139.5	0.0	25	21	43	64	227.2	137.9	0.4	24	19	44	63	230.6	213.5	0.5	23	22	39	61	141.3	119.6	0.2
		4	73	64	**60**	7	**67**	140.3	95.5	0.2	57	52	7	59	144.6	81.8	0.1	56	34	7	41	285.0	61.1	0.4	54	51	7	58	107.9	69.5	0.2
	5a.	5	80	67	**62**	0	**62**	157.3	91.2	5.8	59	56	0	56	140.4	79.1	8.4	41	38	0	38	140.9	68.7	7.3	58	54	0	54	137.2	74.2	12.6
	5b.	5	80	67	**63**	0	**63**	148.3	88.2	1.4	59	56	0	56	139.1	87.1	4.0	41	39	0	39	122.4	64.3	1.7	58	54	0	54	121.8	58.9	1.6
	generated	1	30	30	**28**	0	**28**	181.6	159.2	5.4	30	22	0	22	226.9	175.9	9.0	30	29	0	29	169.8	146.0	3.0	30	19	0	19	289.4	282.3	19.8
		2	30	28	**27**	0	**27**	192.5	143.1	0.8	22	11	0	11	355.2	139.2	10.2	29	24	0	24	312.1	276.1	2.2	19	17	0	17	197.6	162.3	1.9
		3	16	15	**15**	12	**27**	240.3	201.6	0.0	6	5	5	10	231.1	121.0	0.2	13	11	11	22	280.2	245.7	0.4	9	9	8	17	201.0	172.4	0.0
		4	30	27	**25**	0	**25**	229.4	175.9	0.4	10	10	0	10	124.2	112.8	0.0	22	13	0	13	339.3	153.2	0.6	17	16	0	16	190.2	157.6	0.2
	5a.	5	80	25	**7**	0	**7**	465.8	317.3	347.3	10	5	0	5	360.3	249.2	177.6	13	3	0	3	497.6	366.9	275.7	16	6	0	6	413.2	382.4	336.9
	5b.	5	80	25	**8**	0	**8**	459.2	320.5	32.0	10	5	0	5	399.1	251.9	39.1	13	3	0	3	498.0	366.9	23.5	16	6	0	6	441.8	384.9	19.1
	Total		1046	853	**767**	140	**907**	127.9	78.5	12.3	764	682	131	813	133.6	73.0	5.0	758	625	139	764	188.8	106.9	6.4	436	372	61	433	182.1	145.6	17.6

Table 3.9: Results of the exact BaP-based approach.

ing all other goals due to the high cost of labor. However, after fixing the overall number of trucks (and thus the number of required personal) secondary objectives come into play, which concern the cost consuming operations of cooling and freezing devices of a truck. Finally, for practical handling it is clearly convenient for the customer to receive all pallets carrying the same product in the same truck. But if the total number of trucks could be reduced, the customer will agree to a splitting of products onto several trucks. Nevertheless, such a splitting of products should be kept as low as possible, which opens the question how to measure the inconvenience of splitting. We believe that such a lexicographic setting has not be considered for one- or two-dimensional bin-packing problems before.

The first part of this paper gives a branch-and-price framework for computing exact solutions of our problem. While the upper-level problem of minimizing the number of trucks can be modelled as a two-dimensional vector packing problem, with a special structure in one dimension, the lexicographic objective requires five iterative steps of successive optimization problems, in each of which columns are generated by solving appropriate instances of a shortest path problem with resource constraints (SPPRC). It is a major contribution of the paper to put the five different problems arising for the different levels into a uniform framework of column generation. Furthermore, advanced concepts of stabilization and some acceleration techniques are employed.

In the second part, we describe the heuristic solution method previously employed in practice and then develop a number of new constructive heuristics for solving a single level of the packing problem. These try to balance the two constraints and generate solutions with a favorable mix of weight and volume. Then, the single-level heuristics are integrated into a more complex heuristic framework for building solutions of high quality with respect to the multi-level lexicographic objective function.

Computational experiments for real-world benchmark instances as well as structurally more difficult instances extracted from these show that the branch-and-price algorithm is highly effective in computing optimal solutions. Within 10 minutes of computation time it reaches proven optimality for 68 out of 80 real-world scenarios without splitting products. Allowing splitting leads to more difficult problems, but still 62 of 80 are solved to optimality.

It also turns out that the heuristic framework performs much better than the approach previously applied in practice and matches the optimal solutions in 86 % of all subproblems. Note that it is also very helpful to use the heuristic solution as a starting column for the branch-and-price algorithm. Based on this evaluation, our heuristic framework has been successfully incorporated into the SAP ERP software system of our industrial partner and is currently used for the daily planning process.

In the future, it would be interesting to extend our single-source single-sink shipping problem to a more complex supply network serving multiple customers. This would open up the possibility of delivering less-than-truckload amounts to one customer and using the residual capacity of a truck for serving a second (or third) customer, if the routing costs support such a decision.

Acknowledgements

We would like to thank *scc EDV-Beratung AG, Vienna,* and its industrial partner for the fruitful collaboration and for providing all the practical insights.

Ulrich Pferschy was supported by the Field of Excellence "COLIBRI" at the University of Graz.

Bibliography

Aringhieri, R., Duma, D., Grosso, A., and Hosteins, P. (2018). Simple but effective heuristics for the 2-constraint bin packing problem. *Journal of Heuristics*, **24**(3), 345–357.

Bansal, N., Eliáš, M., and Khan, A. (2016). Improved approximation for vector bin packing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 3, pages 1561–1579. SIAM.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bertazzi, L., Golden, B., and Wang, X. (2019). The bin packing problem with item fragmentation: A worst-case analysis. *Discrete Applied Mathematics*, **261**, 63–77.

Brandão, F. and Pedroso, J. P. (2016). Bin packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research*, **69**, 56–67.

Caprara, A. (1998). Properties of some ILP formulations of a class of partitioning problems. *Discrete Applied Mathematics*, **87**(1-3), 11–23.

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Kellerer, H., and Pferschy, U. (2003). Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, **50**(1), 58–69.

Casazza, M. (2019). New formulations for variable cost and size bin packing problems with item fragmentation. *Optimization Letters*, **13**(2), 379–398.

Casazza, M. and Ceselli, A. (2016). Exactly solving packing problems with fragmentation. *Computers & Operations Research*, **75**, 202–213.

Christensen, H. I., Khan, A., Pokutta, S., and Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, **24**, 63–79.

Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., and Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. *Handbook of combinatorial optimization*, pages 455–531.

Côté, J.-F. and Iori, M. (2018). The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, **30**(4), 646–661.

Delorme, M. and Iori, M. (2020). Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, **32**(1), 101–119.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Garey, M. R., Graham, R. L., and Ullman, J. D. (1972). Worst-case analysis of memory allocation algorithms. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 143–150. ACM.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Henke, T. (2018). *Multi-compartment vehicle routing problems in the context of glass waste collection*. Ph.D. thesis, Otto-von-Guericke University Magdeburg, Magdeburg, Germany.

Heßler, K., Gschwind, T., and Irnich, S. (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, **271**(2), 401–419.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

Johnson, D. S. (1973). *Near-Optimal Bin Packing Algorithms*. Ph.D. thesis, Massachusetts Institute of Technology.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

LeCun, B., Mautor, T., Quessette, F., and Weisser, M.-A. (2015). Bin packing with fragmentable items: Presentation and approximations. *Theoretical Computer Science*, **602**, 50–59.

Lodi, A., Martello, S., and Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, **123**(1-3), 379–396.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Malaguti, E., Monaci, M., Paronuzzi, P., and Pferschy, U. (2019). Integer optimization with penalized fractional values: The Knapsack case. *European Journal of Operational Research*, **273**(3), 874–888.

Mandal, C. A., Chakrabarti, P. P., and Ghose, S. (1998). Complexity of fragmentable object bin packing and an application. *Computers & Mathematics with Applications*, **35**(11), 91–97.

Martello, S. and Toth, P. (1990). Bin-packing problem. In *Knapsack problems: Algorithms and Computer Implementations*, Wiley Series in Discrete Mathematics and Optimization, pages 221–245. Wiley.

Martello, S. and Toth, P. (2003). An exact algorithm for the two-constraint 0–1 knapsack problem. *Operations Research*, **51**(5), 826–835.

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., and Limbourg, S. (2014). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, **37**(2), 297–330.

Spieksma, F. C. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers & Operations Research*, **21**(1), 19–25.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2019). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*.

Wei, L., Lai, M., Lim, A., and Hu, Q. (2020). A branch-and-price algorithm for the two-dimensional vector packing problem. *European Journal of Operational Research*, **281**(1), 25–35.

Chapter 4

A Branch-and-Cut Algorithm for the Soft-Clustered Vehicle-Routing Problem

Katrin Heßler, Stefan Irnich

Abstract

The soft-clustered vehicle-routing problem is a variant of the classical capacitated vehicle-routing problem (CVRP) in which customers are partitioned into clusters and all customers of the same cluster must be served by the same vehicle. We introduce a novel symmetric formulation of the problem in which the clustering part is modeled with an asymmetric sub-model. We solve the new model with a branch-and-cut algorithm exploiting some known valid inequalities for the CVRP that can be adapted. In addition, we derive problem-specific cutting planes and new heuristic and exact separation procedures. For square grid instances in the Euclidean plane, we provide lower-bounding techniques and a reduction scheme that is also applicable to the respective traveling salesman problem. In comprehensive computational test on standard benchmark instances, we compare the different formulations and separation strategies in order to determine a best performing algorithmic setup. The computational results with this branch-and-cut algorithm show that several previously open instances can now be solved to proven optimality.

4.1 Introduction

The *soft-clustered vehicle-routing problem* (SoftCluVRP) is a variant of the classical *capacitated vehicle-routing problem* (CVRP, Toth and Vigo, 2014) in which customers are partitioned into clusters, and all customers of the same cluster must be served by the same vehicle. In contrast to the hard-clustered variant, where a cluster must be served completely before the next cluster is served, we consider the variant in which visits to customers of the same cluster can be interrupted by visits to customers of another cluster.

Both the CVRP and SoftCluVRP are defined over a complete undirected graph $G = (V, E)$ with the vertex set $V = \{0, 1, 2, \dots, n\}$ and the edge set E. The vertex 0 denotes the *depot* and the other vertices $C = \{1, 2, \dots, n\}$ denote the *customers*. A homogeneous fleet of m vehicles with capacity Q is hosted at the depot 0. For each edge $\{i, j\} \in E$, non-negative routing costs c_{ij} are given. A route $r = (i_0, i_1, \dots, i_s, i_{s+1})$ is a cycle in G passing through the depot, i.e., $i_0 = i_{s+1} = 0$ and $i_1, \dots, i_s$ are customers ($s \geq 1$). In the CVRP, a route is feasible if (i) all customers $i_1, \dots, i_s$ are different and (ii) capacity constraint $\sum_{j=1}^{s} d_{i_j} \leq Q$ hold for given positive customer-specific demands d_i for $i \in C$.

The clustered variants require the definition of a partitioning of the vertex set: Let $V = V_0 \cup V_1 \cup V_2 \cup \dots \cup V_N$ be such a partitioning, where $V_0 = \{0\}$ denotes the *depot cluster*. The *customer clusters* $V_h \subset C$ are index by $h \in H = \{1, 2, \dots, N\}$ and are disjoint, i.e., $V_h \cap V_{h'} = \emptyset$ for all $h \neq h' \in H$. For any customer $i \in C$, the associated cluster is denoted by $h(i) \in H$, i.e., $i \in V_{h(i)}$. Each cluster V_h has an associated positive demand d_h and we define $d_0 = 0$ for the depot cluster V_0. The demand d_h can be thought of as the sum of demands of all customers V_h belonging to that cluster. Since all customers V_h must be visited together in a route, it is however sufficient to specify the total demand d_h. For a route $r = (i_0, i_1, \dots, i_s, i_{s+1})$, it is convenient to define $H(r)$ as the customer clusters *touched* by the route r, i.e., $H(r) = \{h \in H : V_h \cap \{i_1, \dots, i_s\} \neq \emptyset\}$.

In the SoftCluVRP, a route is feasible if (i) all customers $i_1, \dots, i_s$ are different and (ii) the capacity constraint $\sum_{h \in H(r)} d_h \leq Q$ holds (summing over the customer clusters touched by the route), and (iii) the soft-cluster constraints are fulfilled, i.e.,

$$V_h \subseteq \{i_1, \dots, i_s\} \qquad \forall h \in H(r).$$

Note that the *clustered vehicle-routing problem* (CluVRP, Battarra *et al.*, 2014) requires that the hard-cluster constraints hold, i.e., for each $h \in H(r)$ there must exist an index $k \in \{1, 2, \dots, s - |V_h| + 1\}$ such that $V_h = \{i_k, i_{k+1}, \dots, i_{k+|V_h|-1}\}$.

In all cases (CVRP, CluVRP, SoftCluVRP), the task is to determine a cost-minimal set of m routes that together visit all customers, and herewith also serve all clusters, exactly once.

A practical application for clustered VRPs is parcel delivery in courier companies. First, the parcels are sorted into containers according to a given districting, for example, by postal code. The districts therefore divide customers into clusters. For technical reasons, the sorting policy is typically fixed for more than just one delivery day and altered only once in a while. In a second step, the containers filled with parcels are assigned to vehicles and drivers who deliver them to the recipients (Sevaux and Sörensen, 2008). In such a setting, the weight of individual packages is unknown and thus the demand of individual customers. However, at least good estimates for the demand of a cluster, i.e, the total weight of all shipments assigned to the postal zone, can be given.

Note that the accessibility of all parcels within a delivery vehicle determines whether the resulting problem is either soft-clustered or hard-clustered: Indeed, if the delivery person has always access to all containers in the vehicle, deliveries to customers of the same postal zone/cluster can be interrupted by deliveries to customers of another zone/cluster. Thus, a SoftCluVRP results; otherwise it is a CluVRP.

The very recent paper by Hintsch and Irnich (2020) surveys the pertinent literature. Therefore, we omit a comprehensive survey on clustered VRPs but briefly mention the most important findings: For the exact solution of the CluVRP, Battarra *et al.* (2014) developed a very competitive two-level exact algorithm computing optimal Hamiltonian paths through clusters for several entry and exit points in the subproblem level and combining them to route plans in the master level. The most powerful metaheuristics for the CluVRP are those of Vidal *et al.* (2015) and Hintsch and Irnich (2018) and they follow a similar two-level principle. For the problem considered in the paper at hand, the SoftCluVRP, the only tailored exact solution approach is the branch-and-price algorithm of Hintsch and Irnich (2020) using a MIP-based approach in the pricing subproblem instead of a shortest-path dynamic programming labeling algorithm. These results will serve as a benchmark for the newly developed branch-and-cut algorithm.

The only newer results not surveyed in (Hintsch and Irnich, 2020) are the following: Hintsch *et al.* (2019) have developed a branch-price-and-cut for the soft-clustered variant of the capacitated arc-routing problem. Again, the pricing subproblem is solved by branch-and-cut as well as metaheuristics. Additional cutting planes on the route-based master formulation help to strengthen the linear relaxation. Hintsch (2019) has developed a *large-neighborhood search* (LNS) metaheuristic, which is the currently best-performing heuristic approach for the SoftCluVRP. We later compare against these results.

The contributions of the paper at hand are the following:

1. We introduce the first two-index formulation for the exact solution of the SoftCluVRP. The formulation decomposes into a standard routing part, a novel

part ensuring vehicle-capacity and soft-cluster feasible solutions using a directed cluster graph, and a simple coupling between the two parts. For solving intricate VRPs with general-purpose MIP solvers, finding a compact formulation without variables having a vehicle index (e.g., a two-index formulation) is an essential first step towards an effective branch-and-cut approach, because vehicle-indexed formulations typically suffer from the inherent symmetry when it comes to branching. Symmetry-breaking constraints can only partially mitigate the often observed plateau-effect of the dual bound values. An example helping practitioners to quickly implement effective formulations for the pickup-and-delivery problem with time windows in MIP solvers is the model provided in (Furtado *et al.*, 2017). Recently, a vehicle-index free model for the split-delivery VRP with time windows has been introduced and very successfully used by Munari and Savelsbergh (2019).

2. We analyze the impact of the fleet-size constraint on the validity of the formulation. Some additional constraints are mandatory to cope with non-minimal fleets. In addition, we exploit some non-trivial redundancies of the basic formulation.
3. The soft-cluster requirement leads to a new type of capacity cuts. These cuts are known for the CVRP and we derive a variant of the capacity cuts valid for the SoftCluVRP that are stronger than the straightforward adaptation of the capacity cuts for the CVRP.
4. Some SoftCluVRP instances from the benchmark of Golden *et al.* (1998) (adopted by Battarra *et al.* (2014)) are constructed on a square grid network. We provide lower-bounding techniques for SoftCluVRP instances that are especially strong for grid-based instances. For square grid instances in the Euclidean plane, we develop preprocessing techniques that allow to substantially reduce the edge set and the corresponding routing variables of the formulation.
5. With comprehensive computational test, we determine a competitive setup combining the multiple branch-and-cut components. The computational results on benchmark instances using SoftCluVRP benchmark instances from the literature show that the resulting branch-and-cut algorithm is competitive. While our branch-and-cut implementation does not generally outperform the branch-and-price algorithm of Hintsch and Irnich (2020), it is certainly much simpler to implement using the callable library of modern MIP solvers such as CPLEX, Gurobi, SCIP, or similar softwares. Moreover, the branch-and-cut algorithm seems to perform consistently better on instances where the customers are located in clusters.

The remainder of this paper is structured as follows. In the next section, we present the new formulation for the SoftCluVRP. The components of the branch-and-cut algorithm including the families of valid inequalities and their heuristic

and exact separation are detailed in Section 4.3. Lower-bounding and reduction techniques for the grid-based instances are presented in Section 4.4. In Section 4.5, we present the computational experiments, in which we configure the final branch-and-cut algorithm, analyze the influence of reduction for the grid instances on computational performance, and compare the results on all benchmark instances against those from the literature. Final conclusions are drawn in Section 4.6.

4.2 Two-Index Formulation

Since the SoftCluVRP is a relatively new problem, only a few models can be found in the scientific literature. A three-index formulation for the symmetric SoftCluVRP was recently presented by Hintsch and Irnich (2020), while another three-index formulation was presented by Defryn and Sörensen (2017) for the asymmetric version of the SoftCluVRP. These formulations use routing variables with a third index, say $k \in K = \{1, 2, \ldots, m\}$, to refer to a specific vehicle (the first two indices describe the endpoints of a direct connection). The major drawback of three-index formulations is that they grow linearly with the fleet size and, more severely, they are inherently symmetric with respect to the numbering of the vehicles. Indeed, for a given solution, any permutation of the vehicle indices $k \in K$ produces one of $|K|!$ equivalent solutions. This makes a branch-and-bound-based approach as used in MIP solvers ineffective (Fischetti *et al.*, 1995). The addition of symmetry-breaking constraints can only very partially mitigate the ineffectiveness of the MIP solver's branching decisions (Adulyasak *et al.*, 2014).

Hintsch and Irnich (2020) also derive an extensive route-based formulation (a set-partitioning type of model) from the aforementioned three-index formulation via Dantzig-Wolfe decomposition and subsequent aggregation over vehicles. The drawback of this type of formulation is that a sophisticated branch-and-price algorithm is needed to cope with the huge number of route variables.

The idea of the new formulation we present in the following is to exploit that already well-performing standard models for the CVRP are known. We use the symmetric formulation of Laporte *et al.* (1985) that has non-negative integer routing variables x_{ij} for all edges $\{i, j\} \in E$. Note that all benchmark sets for the SoftCluVRP comprise symmetric instances.

To enforce the clustering constraints, we assume that the elements of the cluster index set H are completely ordered, which holds true if, e.g., $H \subset \mathbb{N}$. We introduce the directed acyclic *cluster graph* $D = (H, A)$ with the arc set $A = \{(h, h') : h, h' \in H, h < h'\}$. The new formulation uses additional binary variables y_a for each $a = (h, h') \in A$ that indicates whether clusters V_h and $V_{h'}$ are served by the same vehicle ($y_a = 1$), or not ($y_a = 0$). While routing variables are symmetric, we intentionally model with asymmetric y-variables. Each component in the subgraph

of D spanned by the positive y-variables, i.e., by $A_{y=1} = \{a \in A : y_a = 1\}$, represents a subset of clusters served by one vehicle.

The orientation in the cluster graph enables an *Miller-Tucker-Zemlin* (MTZ)-based (Miller *et al.*, 1960) modeling approach with continuous resource variables u_h for $h \in H$ that accumulate the demand served by each route. The new formulation is:

$$\min \quad \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{4.1a}$$

$$\text{subject to} \quad \sum_{\{i,j\}\in\delta(i)} x_{ij} = 2 \qquad \forall i \in C \tag{4.1b}$$

$$\sum_{\{0,j\}\in\delta(0)} x_{0j} = 2m \tag{4.1c}$$

$$\sum_{\{i,j\}\in\delta(S)} x_{ij} \geq 2r(S) \qquad \forall S \subseteq C, S \neq \varnothing \tag{4.1d}$$

$$x_{ij} \in \{0,1\} \qquad \forall \{i,j\} \in E \setminus \delta^*(0) \tag{4.1e}$$

$$x_{0j} \in \{0,1,2\} \qquad \forall \{0,j\} \in \delta^*(0) \tag{4.1f}$$

$$x_{ij} \leq y_{h(i),h(j)} \qquad \forall \{i,j\} \in E \setminus \delta(0) : h(i) < h(j) \tag{4.1g}$$

$$u_h - u_{h'} + Qy_{hh'} \leq Q - d_{h'} \qquad \forall (h,h') \in A \tag{4.1h}$$

$$d_h \leq u_h \leq Q \qquad \forall h \in H \tag{4.1i}$$

$$y_{hh'} \geq y_{hh''} + y_{h'h''} - 1 \qquad \forall (h,h'),(h',h'') \in A \tag{4.1j}$$

$$y_{h'h''} \geq y_{hh'} + y_{hh''} - 1 \qquad \forall (h,h'),(h',h'') \in A \tag{4.1k}$$

$$y_{hh''} \geq y_{hh'} + y_{h'h''} - 1 \qquad \forall (h,h'),(h',h'') \in A \tag{4.1l}$$

$$y_a \in \{0,1\} \qquad \forall a \in A \tag{4.1m}$$

The first part (4.1a)–(4.1f) is the CVRP formulation of Laporte *et al.* (1985): The objective (4.1a) minimizes the routing costs. Constraints (4.1b) ensure that each customer is visited once, and constraints (4.1c) ensure that exactly m vehicles leave and return to the depot. In the *capacity cuts* (4.1d), the set $\delta(S)$ is the set of edges with exactly one endpoint in S, and the number $r(S)$ describes the minimum number of vehicles needed to feasibly serve the customer subset S. In the CVRP, it suffices to bound $r(S)$ from below by computing $\lceil d(S)/Q \rceil$, where $d(S)$ is the sum of the demands of all customers in S. Since in the SoftCluVRP the demand is associated with clusters, we can arbitrarily distribute the demand d_h of every cluster V_h onto its customers, e.g., defining $d_i = d_h/|V_h|$ for all $i \in V_h$ and $h \in H$. We discuss the role of the capacity cuts in more detail in Section 4.3.1. The capacity cuts prohibit subtours not including the depot as well as subtours that serve more customers than possible when respecting vehicle capacity. The domains

of the routing variables are given by (4.1e) and (4.1f). Note that a back-and-forth route $(0, j, 0)$ is only feasible if $j \in C$ is a customer that forms a singleton cluster, i.e., $H_{h(j)} = \{j\}$. Therefore, we define $\delta^*(0) = \{\{0, j\} \in \delta(0) : H_{h(j)} = \{j\}\}$, where $\delta(i)$ denotes set of all edges incident to a vertex $i \in V$.

The last part (4.1h)–(4.1m) of the model provides a description of feasible combinations of clusters to be served by the same vehicle. The MTZ-like constraints (4.1h) impose $u_{h'} \geq u_h + d_{h'}$ for $y_{hh'} = 1$. It is crucial here that the set H is ordered and that (4.1h) is imposed only for one direction, i.e., for $(h, h') \in A$ and not for $(h', h) \notin A$, because otherwise the two constraints and $y_{hh'} = y_{h'h} = 1$ would directly imply the contradiction $u_{h'} > u_h$ and $u_h > u_{h'}$. The constraints (4.1i) describe the domain of the u-variables and guarantee that the capacity Q is not exceeded. The constraints (4.1j)–(4.1l) are transitivity-enforcing constraints for the y-variables.

The coupling between the x- and y-variables is established via constraints (4.1g).

Any feasible solution to the SoftCluVRP can be represented in the model (4.1). The opposite is not necessarily true, i.e., a solution to model (4.1) may not be a feasible solution to the SoftCluVRP. In the following, we specify conditions and formulate additional constraints so that (4.1) is a valid model for the SoftCluVRP.

Proposition 1. *If m is the minimum number of vehicles needed to serve all customers C, then every feasible solution to formulation* (4.1) *is a feasible solution to the given SoftCluVRP instance.*

Proof. Note first that the minimum number of vehicles needed to serve C can be obtained as the solution value $m_{\min}$ of a bin-packing instance with bins of capacity Q and items with weights $(d_h)_{h \in H}$.

A feasible solution to model (4.1) may have the following defect: The y-variables can impose a connected component $O \subset H$ of the cluster graph D that is served by more than one vehicle, i.e., the x-variables impose more than one route in $\{0\} \cup \bigcup_{h \in O} V_h$. However, the connected component O respects the capacity constraint, i.e., $d(O) \leq Q$ holds true due to (4.1h)–(4.1m). As all components of D imposed by the y-variables together partition the set H into a feasible bin-packing solution, the number of connected components cannot be smaller than $m_{\min}$. If $m = m_{\min}$, the pigeonhole principle tells us that only one vehicle can serve each connected component. Therefore, a feasible SoftCluVRP solution results. □

A trivial improvement to formulation (4.1) is to add

$$y_{hh'} = 0 \qquad \forall (h, h') \in A : d_h + d_{h'} > Q. \tag{4.1n}$$

This can also be established by eliminating the y-variable that are set to zero from formulation (4.1), modifying the affected constraints (4.1g) and (4.1j)–(4.1l), and eliminating redundant constraints (4.1h).

Non-minimal Fleet. In many vehicle-routing problems, the primary objective is to minimize the number of vehicles. Therefore, Proposition 1 shows that formulation (4.1) is valid and relevant for the standard application, i.e., when $m = m_{\min}$.

Minimizing the number of vehicles and minimizing routing costs are in general conflicting objectives. With the focus on the second objective (routing costs), a relaxed version of the SoftCluVRP is one in which the minimum fleet-size constraint $\sum_{\{0,j\}\in\delta(0)} x_{0j} = 2m_{\min}$ is replaced by $\sum_{\{0,j\}\in\delta(0)} x_{0j} \leq 2m$ with a fleet-size limit $m > m_{\min}$. We denote this relaxation by SoftCluVRP$^{\leq m}$. The following proposition states that formulation (4.1) is also valid for the SoftCluVRP$^{\leq m}$ under some mild assumptions.

Proposition 2. *If the* depot-triangle inequality *holds for the routing costs of a given SoftCluVRP$^{\leq m}$ instance, i.e., $c_{ij} \leq c_{0i} + c_{0j}$ for all $i, j \in C$, then there exists, for every feasible solution to formulation* (4.1)*, a feasible solution to the SoftCluVRP$^{\leq m}$ with identical or lower cost.*

Proof. As shown in the proof of Proposition 1, a feasible solution to formulation (4.1) may only have the defect that more than one route/vehicle serves a connected component O in the subgraph of D spanned by the positive y-variables. In this case, a pair of edges $\{0, i\}$ and $\{0, j\}$ belonging to two different routes can be replaced by the edges $\{i, j\}$ so that the two routes are merged into one. This route is feasible, because constraints (4.1h)–(4.1m) ensure $d(O) \leq Q$. Moreover, the depot-triangle inequality implies that after the replacement the new route is never more costly than the two merged routes. Iterative replacements finally lead to a single feasible route per component. The constructed new solution is then feasible for the SoftCluVRP$^{\leq m}$. □

If neither the assumptions of Proposition 1 nor of Proposition 2 are fulfilled, it is still possible to use formulation (4.1) for the SoftCluVRP$^{\leq m}$. In this case, the following class of *single-route inequalities* must be added to model (4.1):

$$\sum_{\substack{\{0,i\}\in\delta(0),\\ h(i)\in H(T)}} x_{0i} + 2 \sum_{a\in A(T)} y_a \leq 2|H(T)| \qquad \forall T = (H(T), A(T)) \text{ tree in } D \tag{4.2}$$

Regarding the validity of (4.2), consider an integer feasible solution $(\bar{x}, \bar{y})$ to the SoftCluVRP or SoftCluVRP$^{\leq m}$. For an arbitrary tree $T = (H(T), A(T))$ in D, the customers $\bigcup_{h\in H(T)} V_h$ are served by a certain number of vehicles, say $m(T)$ vehicles. This implies

$$\sum_{\substack{\{0,i\}\in\delta(0),\\ h(i)\in H(T)}} \bar{x}_{0i} \leq 2m(T).$$

Moreover, the subgraph $T_{\bar{y}}$ of the tree spanned by arcs a with $\bar{y}_a = 1$ must decompose into $m(T)$ or more components. The latter implies that

$$\sum_{a \in A(T)} \bar{y}_a \leq |H(T)| - m(T) \tag{4.3}$$

holds true. Adding the first and twice the second inequality yields the inequality (4.2) for $(\bar{x}, \bar{y})$.

Note that inequalities (4.3), for all trees $(H(T), A(T))$, can be used to ensure capacity-feasible solutions, i.e., they can replace the MTZ-part (4.1h)–(4.1i) of formulation (4.1). We denote (4.3) as *tree-capacity constraints* in the following.

Proposition 3. *For every feasible solution to model* (4.1)–(4.2) *with a relaxed fleet-size constraint* $\sum_{\{0,j\} \in \delta(0)} x_{0j} \leq 2m$ *instead of* (4.1c) *defined by an arbitrary value* $m \geq m_{\min}$*, the solution is also feasible for the SoftCluVRP or SoftCluVRP*$^{\leq m}$*.*

Proof. Consider a feasible solution $(\bar{x}, \bar{y})$ to model (4.1)–(4.2) with a relaxed fleet-size constraint. The positive $\bar{y}$-values decompose D into a number of components. Consider an arbitrary component $O = \{h_1, h_2, \ldots, h_{|O|}\} \subset H$ and its ordered elements $h_1 < h_2 < \cdots < h_{|O|}$. Since O is a single connected component, the transitivity constraints (4.1j) and (4.1k) impose $\bar{y}_{h_1,h_2} = \bar{y}_{h_2,h_3} = \cdots = \bar{y}_{h_{|O|-1},h_{|O|}} = 1$. The path $(h_1, h_2, \ldots, h_{|O|})$ is also a tree T. For this tree $T = (O, \{(h_k, h_{k+1}) : k = 1, \ldots, |O| - 1\})$, inequality (4.2) imposes $\sum_{(0,i) \in \delta(0), h(i) \in O} \bar{x}_{0i} \leq 2$ showing that the $\bar{x}$-values define only a single route serving component O. □

Redundancy. Formulation (4.1) has some redundant transitivity constraints. We denote by (R) the model (4.1) without constraints (4.1l). The relationship between the two formulations is characterized in the following two propositions:

Proposition 4. *Formulations* (4.1) *and* (R) *have the same set of integer solutions regarding the projection onto the* x*-variables.*

For the sake of clarity, all longer proofs (such as the one for Proposition 4) have been moved to the Appendix section.

Proposition 5. *The linear relaxations of formulations* (4.1) *and* (R) *have the same set of solutions regarding the projection onto the* x*-variables.*

4.3 Branch-and-Cut Algorithm

Formulations (4.1) and (4.1)–(4.2) are not directly solvable with a MIP solver, because they contain some large-sized families of constraints. This section describes how constraints of these families can be added dynamically using separation procedures. We distinguish between (possibly infeasible) integer solutions $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$ and fractional solutions $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$ for

- capacity cuts (4.1d) (exponential in $|V|$),
- single-route inequalities (4.2) (exponential in $|H|$),
- tree-capacity constraints (4.3) (exponential in $|H|$),
- transitivity constraints (4.1j) and (4.1k) (cubic in $|H|$).

In the branch-and-cut implementation, we consider an inequality as violated only if the degree of violation (difference between right-hand and left-hand side) exceeds $\varepsilon = 10^{-4}$.

4.3.1 Capacity Cuts

For an integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$, we determine the graph $G_{\bar{x}}$ spanned by the positive $\bar{x}$-variables. Subsequently, we remove the depot vertex 0 leading to the induced graph $G_{\bar{x}}[V \setminus \{0\}] = G_{\bar{x}}[C]$. The connected components of $G_{\bar{x}}[C]$ are subsets of customers served together. Each such subset $S \subset V \setminus \{0\}$ may violate the associated capacity constraint, i.e., if the component forms a cycle (=subtour not including the depot). Since the left-hand side of the capacity cut is equal to zero in this case, the capacity cut (4.1d) is violated independent of the value of $r(S)$.

However, we strive for a tight value $r(S)$ in order to add a strong valid inequality. It was already mentioned in Section 4.2 that different lower bounds on the minimum number of vehicles needed to serve some customers can be computed by arbitrarily distributing the cluster's demands onto the associated customers. For a given customer subset $S \subset V \setminus \{0\}$, we define the *clusters touched* as

$$H(S) = \{h \in H : S \cap V_h \neq \emptyset\}. \tag{4.4}$$

An optimal distribution of the demand is one that assigns the entire cluster demand for $h \in H(S)$ to only a single customer $i_h \in S \cap V_h$. This is summarized in the following proposition:

Proposition 6. *For any $S \subset V \setminus \{0\}$ with $S \neq \emptyset$, a largest lower bound $r(S)$ on the minimum number of vehicles needed to serve the customers S results from considering the demands $(d_h)_{h \in H(S)}$:*

(i) the exact value $r(S)$ results from solving a bin-packing problem with bins of capacity Q and weights $(d_h)_{h \in H(S)}$;

(ii) a valid lower bound for $r(S)$ sufficient for formulation (4.1) is given by $\left\lceil \sum_{h \in H(S)} d_h / Q \right\rceil$.

For a fractional solution $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$, we apply the following series of heuristic separation procedures. First, we again consider $G_{\bar{x}}[C]$ and its components. For each component $S \subset C$, we test $\bar{x}(\delta(S)) < 2\left\lceil \sum_{h \in H(S)} d_h / Q \right\rceil$ (and herewith $\bar{x}(\delta(S)) < 2r(S)$) and, if true, a violated capacity cut (4.1d) is found.

Algorithm 10: Karger's contraction algorithm

input : graph $G = (V, E)$
output: sets to check S

for $e \in E$ **do**
  $p_e \leftarrow \bar{x}_e / \sum_{f \in E} \bar{x}_f$;
for $|V|$ *iterations* **do**
  $G' \leftarrow G$;
  while G' *has more than two vertices* **do**
    Choose an edge $e \in E(G')$ with probability p_e;
    Contract e into a single vertex, i.e., $G' \leftarrow G'/e$;
  $(S, \bar{S}) \leftarrow$ subsets of vertices represented by the two remaining vertices of G', $0 \in \bar{S}$;
  Compute $r(S)$;
  Check S regarding $\bar{x}(\delta(S)) < 2r(S)$;

Second, we apply two heuristic procedures that work according to the subset-first check-second principle, i.e., promising subsets S are computed first and the violation $\bar{x}(\delta(S)) < 2r(S)$ is checked for each computed subset S. The first heuristic is the probabilistic graph contraction algorithm of Karger (1993), summarized in Algorithm 10. Starting with $G = G_{\bar{x}}[V]$ as an input, the idea is to iteratively contract edges of G where edges $e \in E$ with a higher value $\bar{x}_e$ are chosen with a higher probability (proportional to $\bar{x}_e$). In the contraction step for an edge $\{i, j\} = e \in E$, the vertices i and j, and later subsets S_i and S_j containing i and j, respectively, are replaced by the union $S_i \cup S_j$. Edges $\{k, S_i\}$ and $\{k, S_j\}$ with $k \in V \setminus (S_i \cup S_j)$ are merged into a single edge with weight $\bar{x}_{k,S_i} + \bar{x}_{k,S_j}$. The result of $|V| - 1$ contraction steps is a partition $S \cup \bar{S} = V$ where we assume that $\bar{S}$ contains the depot 0. Therefore, S is a subset of $V \setminus \{0\}$ and thus a valid candidate set for testing the violation of a capacity cut (4.1d).

The graph contraction algorithm and the testing of violated SEC is repeated $|V|$ times, and a most-violated capacity cut is added.

The second heuristic uses the heuristic separation procedures for the CVRP publicly available in the library of Lysgaard *et al.* (2004). Potential subsets S result from the solution of some max-flow/min-cut problems. As the library is tailored to the CVRP, we distribute cluster demands d_h equally onto the customers $i \in V_h$. Both heuristic procedures (Karger, Lysgaard) are used independently.

If no violated capacity cut has been found with the heuristics, we apply the following exact MIP-based separation algorithm: the MIP simultaneously determines

the subset $S \subset C$, computes the lower bound on $r(S)$ given by Proposition 6(ii), and maximizes the violation (if any) $2r(S) - \bar{x}(\delta(S))$. The MIP generalizes ideas first presented by Ahr (2004) and later refined by Martinelli *et al.* (2013) for exactly separating capacity cuts for the capacitated arc-routing problem. The formulation of the separation problem uses five types of variables: Binary variables s_i for $i \in V$ are indicator variables describing whether the vertex i belongs to the unknown set S. Similarly, binary variables y_h describe the clusters $H(S)$ touched by S, i.e., $H(S) = \{h \in H : y_h = 1\}$. Variables z_e for $e \in E$ are indicators for the cut set, i.e., $z_e = 1$ iff $e \in \delta(S)$. Moreover, the integer variable r describes (the lower bound on) $r(S)$ and the non-negative continuous variable $f < 1$ describes the fractional difference between $\lceil d(S)/Q \rceil$ and $d(S)/Q$.

$$\begin{aligned}
\max \quad & 2r - \sum_{e \in E} \bar{x}_e z_e && && (4.5a)\\
\text{subject to} \quad & s_0 = 0 && && (4.5b)\\
& z_e - s_i + s_j \geq 0 && \forall e = \{i,j\} \in E, i \neq 0 && (4.5c)\\
& z_e - s_j + s_i \geq 0 && \forall e = \{i,j\} \in E, j \neq 0 && (4.5d)\\
& s_i + s_j - z_e \geq 0 && \forall e = \{i,j\} \in E && (4.5e)\\
& s_i + s_j + z_e \leq 2 && \forall e = \{i,j\} \in E \setminus \delta(0) && (4.5f)\\
& \sum_{i \in V_h} s_i - y_h \geq 0 && \forall h \in H && (4.5g)\\
& r = \sum_{h \in H} (d_h/Q) y_h + f && && (4.5h)\\
& s_i \in \{0,1\} && \forall i \in V \setminus \{0\} && (4.5i)\\
& 0 \leq z_e \leq 1 && \forall e \in E && (4.5j)\\
& y_h \in \{0,1\} && \forall h \in H && (4.5k)\\
& r \geq 0, \text{integer} && && (4.5l)\\
& 0 \leq f \leq 1 - 1/Q && && (4.5m)
\end{aligned}$$

The objective (4.5a) maximizes the violation of a capacity cut described by $S = \{i \in V : s_i = 1\}$. Forcing $s_0 = 0$ ensures that $S \subset V \setminus \{0\} = C$ holds. The coupling of the z-variables with the s-variables is established via (4.5c)–(4.5f), where (4.5c) and (4.5d) force z_e to one if $s_i \neq s_j$, i.e., $e = \{i,j\} \in \delta(S)$, while (4.5e) and (4.5f) force z_e to zero if $s_i = s_j$. To correctly consider that a cluster V_h for some $h \in H$ is touched, constraints (4.5g) couple the vertex and cluster indicator variables. The correct value of r is guaranteed with constraint (4.5h). The domains of the variables are described by (4.5i)–(4.5m).

Repeatedly solving the exact separation formulation (4.5) with a MIP solver consumes considerable computation time. Therefore, the exact separation is only

used at the root node of the branch-and-cut algorithm. Moreover, we do not call the exact separation routine if the lower bound is not improved by more than 0.01 % within the last ten iterations.

4.3.2 Single-Route Inequalities

Recall that single-route inequalities (4.2) are mandatory only if the fleet is larger than needed and the depot-triangle inequality does not hold (cf. Propositions 1 and 2). As we found them violated only rarely, we inspect only integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$. For every connected component O of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$, let $H(O)$ be the vertex set of the component O. We can take any spanning tree $T = (H(O), A(O))$ spanning $H(O)$ in $D_{\bar{y}}$ and check whether (4.2) is violated which is the case if more than two edges $\{0, i\} \in \delta(0)$ are chosen.

4.3.3 Tree-Capacity Constraints

The tree-capacity constraints (4.3) can replace the MTZ constraints (4.1h)–(4.1i). Note that there are a quadratic number of MTZ constraints, while the number of tree-capacity constraints is exponential in $|H|$. Therefore, the latter family of constraints must be added dynamically.

For an integer solution $(\bar{x}, \bar{y}) \in \mathbb{Z}^{|E|+|A|}$, we consider all connected components O of $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$ and spanning trees $T = (H(O), A(O))$ (as in the previous Section 4.3.2). For the sake of acceleration, the value $m(T)$ is approximated by $\lceil d(O)/Q \rceil$ instead of solving a bin-packing problem. As the number of components is small, all violated tree-capacity constraints (4.3) are added at the same time.

For fractional solutions $(\bar{x}, \bar{y}) \in \mathbb{R}^{|E|+|A|}$, we use a heuristic inspired by the procedure for integer solutions. Also here we consider all connected components O of $D_{\bar{y}}$. Within each component O, the tree T maximizing the left-hand-side of constraints (4.3) is computed as a maximum spanning tree using Kruskal's algorithm.

A special case of the tree-capacity constraints are constraints

$$\sum_{h' \in H : h < h'} y_{hh'} + \sum_{h' \in H : h' < h} y_{h'h} \leq N - 1 \qquad \text{for all } h \in H$$

because all ingoing and outgoing arcs of h form a spanning tree in D. We add these $N = |H|$ constraints at initialization to accelerate the solution process.

Overall, later computational experiments will compare three setups: (1) only MTZ constraints statically added at the beginning, (2) only tree-capacity constraints added dynamically, and (3) the combination of static MTZ and dynamic

tree-capacity constraints. The last strategy may lead to a faster branch-and-cut algorithm as neither MTZ dominate tree-capacity constraints on fractional solutions, nor vice versa.

4.3.4 Transitivity Constraints

Propositions 4 and 5 have proven that the transitivity constraints (4.1l) are completely redundant. However, the two groups (4.1j) and (4.1k) of transitivity constraints are each of cubic size in $|H|$. Therefore, one may either add transitivity constraints to the model right from the beginning or add them only when violated. In the later computational experiments, we test the following two strategies: Either, all transitivity constraints (4.1j) and (4.1k) are present in formulation (4.1). This is the *static* case.

Alternatively, formulation (4.1) is initialized without constraints (4.1j) and (4.1k), and only violated constraints are separated and added dynamically. The identification of violated transitivity constraints can be done by straightforward direct inspection consuming $|H|^3$ time.

In pretests we found that many transitivity constraints are violated at the same time. In order to not add too many constraints simultaneously, we separate them in batches. In every round, only those violated transitivity constraints (4.1j) and (4.1k) defined by $h < h' < h''$ are added for which the distance in D between h and h'' is minimal (counting the number of arcs). We use this selection rule for integer as well as fractional solutions, where in the latter case the distance-based rule does not at all consider the degree of violation (except for the cut tolerance ε).

4.4 Square Grid Instances

In the VRP benchmark set of Golden *et al.* (1998) that is also used for the SoftClu-VRP, the instances have a specific structure. The customer vertices are located in a systematic and non-random fashion (see also Section 4.5). In the groups `Golden1` to `Golden8`, customers are located on concentric circles. In the groups `Golden9` to `Golden16`, customers are located on a square grid. Finally, in the groups `Golden17` to `Golden20`, the customers form a star.

For grid-based instances, we prove in Section 4.4.1 some properties of optimal solutions that allow the reduction of the edge set E without losing optimality. Moreover, we derive in Section 4.4.2 simple-to-compute lower bounds that are especially effective for the grid-based instances.

4.4.1 Reduction of the Edge Set

The set of edges E of the grid-based instances can be reduced using the following theorem.

Theorem 1. *Let an instance of a* Euclidean traveling salesman problem *(EucTSP) be given, where all vertices are points of a square grid.*

If there exists a (3×3)*-vertex block, see Figure 4.1, then the vertex in the middle of the block (depicted as a diamond* $\diamond$ *in Figure 4.2) is connected to two block neighbors in every optimal Euclidean EucTSP tour. Therefore, in any optimal solution, two of the eight blue edges in Figure 4.2 are selected.*

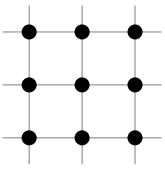


Figure 4.1: A (3×3)-vertex block.

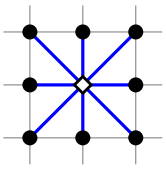


Figure 4.2: Edges of $\delta(\diamond)$ of an optimal TSP tour.

The proof of Theorem 1 can be found in the Appendix.

In the following, a vertex in a Euclidean SoftCluVRP defined over a square grid is denoted as a *middle vertex* if there exists a (3×3)-vertex block that is completely contained in one of the clusters. All edges $\{i, j\} \in E$ that connect a middle vertex i with a non-neighboring vertex j (different from one of the eight neighbors depicted in Figure 4.2) are denoted as *long edges*.

Corollary 1. *An optimal solution of a Euclidean SoftCluVRP defined over a square grid does not contain long edges.*

Proof. Follows directly from Theorem 1. □

The instances with vertices located on a circle (classes 1–8 of the benchmark of Golden *et al.* (1998)) cannot be reduced as suggested in Corollary 1. A counterexample is given in Figure 4.3.

4.4.2 Lower Bounds

We present two different lower-bounding techniques for grid-based instances constructed as those of classes 9–16 of the benchmark set of Golden *et al.* (1998).

First, we exploit that any cluster is connected to the depot with not more than two edges. Therefore, connecting the depot to the closest customer vertices (allowing double connections for edges in $\delta^*(0)$) with the additional restriction

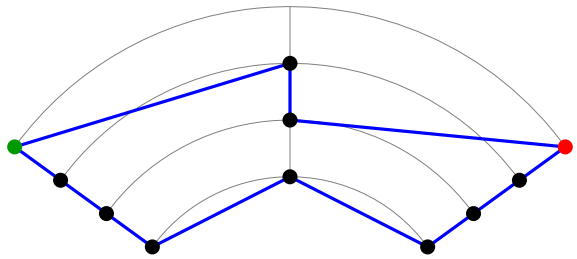


Figure 4.3: Instance with an optimal EucTSP tour with vertices on a concentric circles. The black vertices • form a 3×3-vertex block. The vertex in the middle of the (3×3)-vertex block is connected to the red vertex • that is not part of the block.

that no cluster is connected more than two times yields a lower bound on the cost of depot-edges. Moreover, each customer vertex is connected to other vertices with distance of at least 1. Hence, the sum of connections to the depot plus the sum of distance 1 connections provides a valid lower bound, in the following referred to as the *grid lower bound*.

Second, we reuse the same idea that every cluster is connected to the depot with not more than two edges to formulate a relaxed model. The model relaxes formulation (4.1) and adds the condition on the depot connections. It reads as follows:

$$\min \quad \sum_{\{i,j\} \in E} c_{ij} x_{ij} \tag{4.6a}$$

$$\text{subject to} \quad (4.1\text{b})\text{–}(4.1\text{f})$$

$$\sum_{i \in V_h} x_{0i} \leq 2 \qquad \forall h \in H \tag{4.6b}$$

For the grid instances, formulation (4.6) provides the same or a better lower bound as the simple grid lower bound explained first, but the computational effort is higher. We refer to the second bound as the *relax lower bound*. Note that this latter bound is generally applicable to all SoftCluVRP instances.

4.5 Computational Results

The computational experiments are based on the same benchmark instances as considered by Hintsch and Irnich (2020). All benchmarks use CVRP instances and define an additional parameter θ that specifies the average cluster size. Clusters are then constructed in various way (for details see Fischetti *et al.*, 1997; Bektaş *et al.*, 2011).

The first set of 158 small- and medium-sized instances is based on the GVRP benchmarks A, B, P, and GC with $\theta \in \{2, 3\}$. The instances with 16 to 262 vertices and 6 to 131 clusters were generated by Bektaş *et al.* (2011). The second set of 220 large-scale instances is based on the Golden instances of Golden *et al.* (1998) with $\theta \in \{5, \ldots, 15\}$ and were generated by Battarra *et al.* (2014). The instances are divided into 20 groups denoted by Golden1 to Golden20 with 201 to 484 vertices and 14 to 97 clusters.

Unfortunately, the distances in the grid-based instances (groups Golden9 to Golden16) are computed as Euclidean distances rounded to the next integer value. As a consequence, neither distances fulfill the triangle inequality nor Corollary 1 is directly applicable. To anyway test the reduction techniques described in Section 4.4.1, we generated 90 additional but smaller Grid instances as follows: Six instances were generated for each combination of grid size $d \times d$ with $d \in \{10, 12, 14\}$ and number $N \in \{6, 8, 10, 12, 14\}$ of clusters. In all instances, the minimal distance between vertices is 1. The depot is either in the middle or at the corner of the grid. To define the clusters, one vertex is randomly assigned to each cluster at initialization. As long as vertices are unassigned, such a vertex with at least one already assigned vertex in the neighborhood is chosen at random. This vertex is then assigned to a randomly chosen cluster of its neighborhood (see vertex $\diamond$ with neighbors $\bullet$ in Figure 4.2 for the definition of the neighborhood). Demands of the clusters are equally distributed on the interval $[10, 50]$ and the vehicle capacity is set to $Q = 100$. The number of vehicles is defined as $m = \lceil \sum_h d_h / Q \rceil$. All distances are computed as nontruncated Euclidean distances. The instances are online available at https://logistik.bwl.uni-mainz.de/benchmarks/.

The branch-and-cut algorithm was implemented in C++ using CPLEX 12.8.0 with Concert Technology and compiled into 64-bit single-thread code with Microsoft Visual Studio 2015. Experiments are carried out on a 64-bit Microsoft Windows 7 personal computer with an Intel® Core™ i7-5930k CPU clocked at 3.5 GHz and 64 GB of RAM. CPLEX's default values are kept for all parameters. Unless stated otherwise, computation times are limited to a maximum of 3600 seconds (1 hour).

4.5.1 Comparison of Cutting Strategies

In a first experiment, we try to find a reasonable cutting strategy for the final branch-and-cut algorithm. In Section 4.3, we discussed possible separation strategies for capacity cuts, single-route inequalities, tree-capacity constraints, and transitivity constraints. A synopsis of the strategies is presented in Table 4.1. We distinguish between constraints for model (4.1) (first section of the table) and the mandatory single-route inequalities for model (4.1)–(4.2) needed for fixed non-minimal fleets and when the depot-triangle inequality is not satisfied (second sec-

tion). Note that in several instances of the `Golden` benchmark the fleet is larger than the minimum fleet size.

Table 4.1: Summary of Cutting Strategies

Constraint type	Abbrev.	Separation strategies	Remark
Transitivity constraints (4.1j)–(4.1k)	Trans	(2): `static`, `dyn`	int: mandatory
MTZ constraints (4.1h)–(4.1i)	MTZ	(2): `none`, `static`	int: mandatory, redundant if Tree:`dyn`
Capacity cuts (4.1d) separated			int: mandatory
with Lysgaard library	LysCC	(3): `none`, `root only`, `dyn`	fract: optional
with Karger's contr. alg.	ProbCC	(3): `none`, `root only`, `dyn`	fract: optional
with MIP (4.5)	MIP CC	(3): `none`, `root only`, `dyn`	fract: optional
Tree-capacity constraints (4.3)	Tree	(2): `none`, `dyn`	int: mandatory, redundant if MTZ:`static`
Single-route inequalities (4.2)	Single	(1): `dyn`	int: mandatory only for SoftCluVRP$^{\leq m}$ w/o depot-triangle inequality

Note: Default strategies are underlined.

Table 4.1 offers $2^3 \cdot 3^3 = 216$ possible configurations by combining separation strategies that either completely switch off a separation procedure (denoted by "`none`"), add all inequalities at initialization ("`static`"), or add inequalities dynamically using separation procedures. In the latter case, we compare dynamic separation at the root node of the branch-and-bound tree *only* ("`root only`") and separation at all tree nodes ("`dyn`"). There are some invalid configurations, e.g., combining MTZ:`none` and Tree:`none`, that cannot ensure capacity feasible solutions. We omit all invalid configurations.

Empirically testing all possible configurations is hardly possible given that the benchmark sets are large and the previous literature allowed one hour of computation time per instance. We address this issue in the following way: First, we define some *default separation strategies* (denoted by "`default`") that worked well in pretests. We did however not find good default strategies for the transitivity constraints and MTZ constraints, but default strategies for the remaining inequalities (those underlined in Table 4.1). As a consequence, we compare separation strategies that consider:

- all four combinations of either Trans:`static` or Trans:`dyn` and MTZ:`none` or MTZ:`static`; (4 strategies)
- use all default strategies (`default`) or vary exactly one of the default parameters for all other valid inequalities and their separation. (8 strategies)

In the latter case, the parameter altered is either LysCC:`none`, LysCC:`root only`, ProbCC:`none`, ProbCC:`root only`, MIP CC:`none`, MIP CC:`dyn`, or Tree:`none`.

Overall, there remain $4 \cdot 8 - 1 = 31$ separation strategies to compare (one strategy is invalid). Second, we restrict the instance set to `GVRP` instances of the groups `A`, `B`, `P`, the `GC` instances with $n = 100$ customers, and the two smallest `Golden` instances of each group `Golden1` to `Golden20`. The selection comprises 190 instances for this experiment.

We compare the different B&C algorithms resulting from the 31 different separation strategies with the help of performance profiles as suggested by Dolan and Moré (2002). For a set of algorithms $\mathcal{A}$ (the 31 B&C algorithms in our case), the performance profile $\rho_A(\tau)$ of an algorithm $A \in \mathcal{A}$ describes the ratio of instances that can be solved by A within a factor τ compared to the fastest algorithm, i.e.

$$\rho_A(\tau) = \frac{\left|\left\{I \in \mathcal{I} : t_I^A / t_I^* \leq \tau\right\}\right|}{|\mathcal{I}|},$$

in which $\mathcal{I}$ is the set of instances, t_I^A is the computation time of algorithm A when applied to instance $I \in \mathcal{I}$, and t_I^* is the smallest computation time among all algorithms of set $\mathcal{A}$ for instance I. Unsolved instances are taken into account with $t_I^A = \infty$ (assuming also that $t_I^* = \infty$ gives $t_I^A / t_I^* = \infty$). Note that $\rho_A(1)$ is the percentage of instances for which algorithm A is the fastest, and $\rho_A(\infty)$ is the percentage of instances that are solved by algorithm A within the time limit.

Figure 4.4 displays the performance profiles of the B&C algorithms using the 31 different cutting strategies. For the sake of clarity, we have decided to group the profiles in different way. In the upper part, the four Subfigures 4.4a–4.4d group by values of Trans and MTZ, i.e., Trans:`static` or Trans:`dyn` and MTZ:`none` or MTZ:`static`. Within each of these subfigures, the compared eight (seven in Subfigure 4.4b) strategies that result from the default (`default`) and the variation of exactly one of the default parameters.

The result from each subfigure is simple to summarize. In all four subfigures, the strategy $A =$ MIP CC:`none` is the one that is the fastest for the highest number of instances (compare the values $\rho_{\text{MIP CC:none}}(1)$ and $\rho_A(1)$). The reason is that completely switching off the MIP-based separation of the capacity cuts accelerates the B&C substantially, but it comes at the cost of providing also less optimality proofs (compare the values $\rho_{\text{MIP CC:none}}(\tau)$ and $\rho_A(\tau)$ for $\tau \approx 15$). Therefore, the strategy MIP CC:`none` is not always the winning strategy. Having said this, we can identify the strategy ProbCC:`none` as the best one in Subfigure 4.4a, ProbCC:`root only` in Subfigure 4.4b, Tree:`none` in Subfigure 4.4c, and MIP CC:`none` in Subfigure 4.4d.

These four strategies are finally compared in Subfigure 4.4e: There is no strategy that dominates all other strategies. The strategy Trans:`dyn`, MTZ:`static`, Tree:`none` solves the largest number of instances (within the time limit). This strategy is (compare the figures) not the fastest algorithm most of the time. Analyzing results in more detail, we find that the strategy Trans:`dyn`, MTZ:`none`,

MIP CC:`none` is the fastest variant most of the time for small-sized instances. It is however inferior for larger-sized instances. Therefore, we have decided to choose the strategy

$$\begin{array}{rc} & \text{Trans:}\texttt{dyn}\text{, MTZ:}\texttt{static}\text{, MIP CC:}\texttt{root only} \\ \text{with default values} & \text{LysCC:}\texttt{dyn}\text{, ProbCC:}\texttt{dyn}\text{, and Tree:}\texttt{none}. \end{array} \tag{B\&C}$$

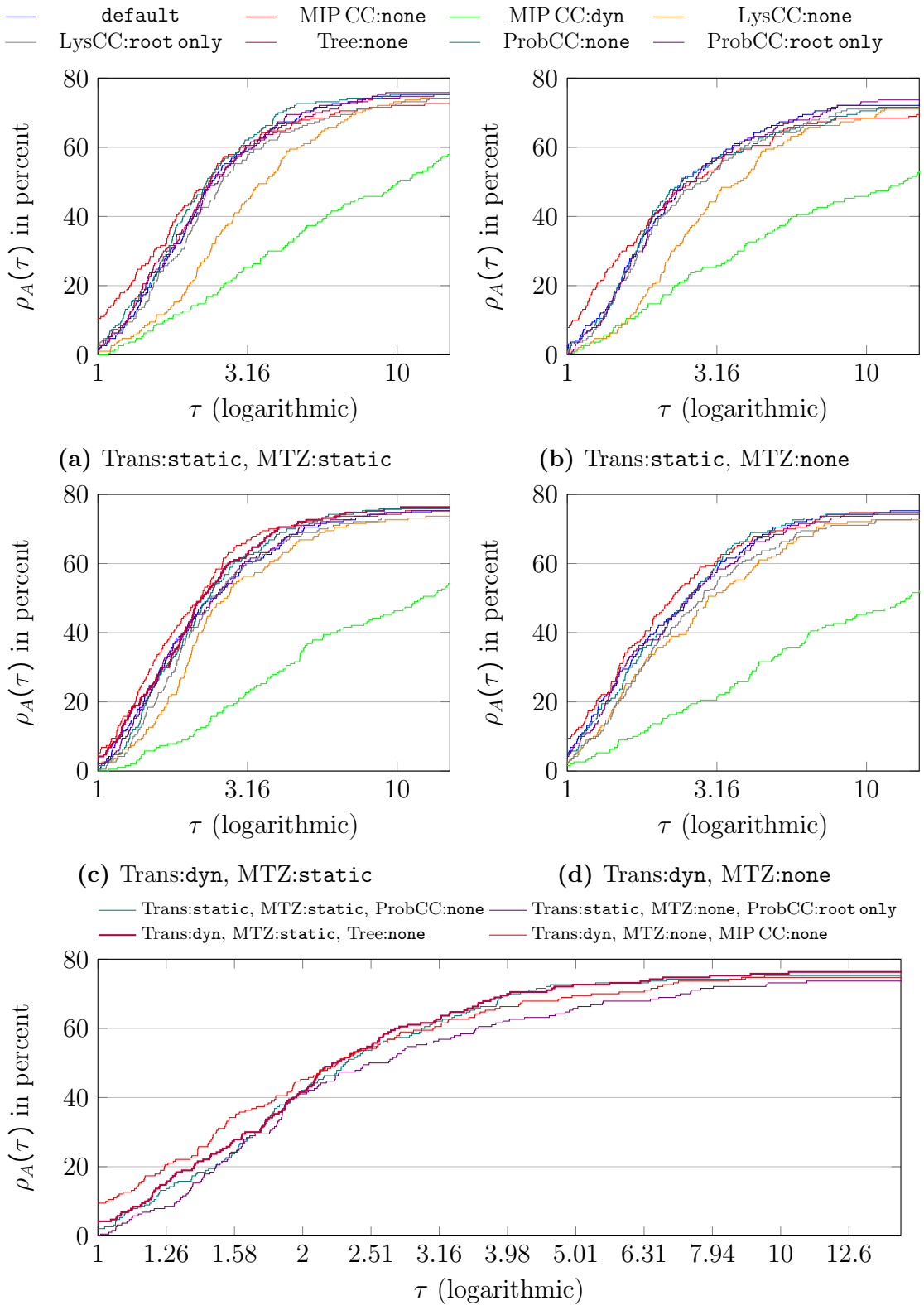
as the one that we use as the *reference B&C algorithm* in the following computational experiments. This is the algorithm that we refer to as *B&C*.

4.5.2 Savings-based Upper Bounds

Pretests have shown that tight upper bounds are very helpful for initializing the B&C algorithm. We employ the savings-based algorithm of Hintsch (2019, p. 6), which is tailored to the SoftCluVRP. It works as follows: In contrast to the classical savings algorithm, there is one initial route for each cluster visiting all its customers. For each $h \in H$, this route is a TSP tour over the vertex set $V_h \cup \{0\}$ computed with the help of a combined *iterated local search* and *variable neighborhood decent* (ILS/VND) (Hintsch, 2019, Section 2.1). Additionally, the same is done for each pair $(g, h) \in H \times H$. Such a route visits all customers $V_g \cup V_h$ and the depot 0. Let the cost of the resulting routes be $\hat{c}_h$ and $\hat{c}_{g,h}$, respectively. Savings values are calculated for each pair $(g, h) \in H \times H$ as $sav_{g,h} = \hat{c}_g + \hat{c}_h - \hat{c}_{g,h}$. Subsets of clusters to be served by one route are constructed now as in the classical savings algorithm: the largest (feasible) savings value $sav_{g,h}$ is chosen first. The two clusters are then merged, i.e., their demand is added. A saving becomes infeasible if either the vehicle capacity Q is exceeded by the total demand of both routes or both clusters are already part of the same route. The savings algorithm repeats these steps until the number of routes matches the number of vehicles or all remaining savings become infeasible.

In case the number of routes exceeds the number of vehicles m, we compute alternative subsets of clusters to be served by one route as a bin-packing solution with items of weight $d_h, h \in H$, and capacity Q. We use the arc-flow model of Valério de Carvalho (1999) for this purpuse.

Finally, for each set of clusters served by a route, we construct a route with the combined ILS/VND. Such a route visits all customers belonging to the given clusters and the depot 0.



(a) Trans:`static`, MTZ:`static`

(b) Trans:`static`, MTZ:`none`

(c) Trans:`dyn`, MTZ:`static`

(d) Trans:`dyn`, MTZ:`none`

(e) Performance profile for the different settings.

Figure 4.4: Performance profiles for different settings. `default` is defined as the version in which MIP CC:`root only`, LysCC:`dyn`, Tree:`dyn`, ProbCC:`dyn`. The legend specifies which parameter of the `default`-version is changed.

4.5.3 Results for the GVRP Instances

In this section, we compare the new B&C algorithm against the branch-and-price algorithm of Hintsch and Irnich (2020) using the `GVRP` benchmark. As mentioned before, the B&C algorithm benefits from tight upper bounds available at initialization. To highlight this effect, we used the reference B&C initialized with the savings-based upper bound of Section 4.5.2 and the same B&C algorithm but with the *best known solution* (BKS) as upper bound.

The results are summarized in Table 4.2, where the table entries have the following meaning:

#opt:	number of instances solved to proven optimality within 1 hour (3600 seconds);
time $\bar{T}$:	average computation time in seconds; unsolved instances are taken into account with the time limit TL of 1 hour (3600 seconds);
gap:	gap $100 \cdot (UB - LB)/LB$, i.e., gap in percent;
Total:	Aggregated result of Hintsch and Irnich (2020) and our default B&C algorithm (see B&C below);
Total*:	Best known results (including other algorithms and higher computation times);
HI20:	branch-and-price of Hintsch and Irnich (2020);
B&C:	reference B&C with upper bound of the savings-based heuristic (see Section 4.5.2);
⋄B&C:	reference B&C, but with BKS provided as upper bound UB;
simple LB	grid or relax lower bound; the latter with model (4.6) run for a maximum of 600 seconds (see Section 4.4.2); BKS provided as UB.

Overall, the branch-and-price of Hintsch and Irnich (2020) computes the largest number of proven optima and it is slightly faster than the B&C algorithm. However, results are clearly mixed over the different groups of benchmark instances. Both versions of the B&C algorithm are much faster for class `B`, which can be attributed to the special structure of the `B` instances: the customer vertices are truly *clustered* and not scattered, i.e., almost uniformly distributed like all others (Augerat, 1995).

The results show that the branch-and-price and B&C algorithms are complementing each other. The column *Total* underlines this statement, because the reference B&C algorithm computes optima for five non-solved instances of HI20. Comparing results with extended computation times and other algorithms employed in Hintsch and Irnich (2020) another four (five) previously open instances are solved now with the B&C (⋄B&C) algorithm. At the end, only seven of the 158 `GVRP` instances remain unsolved.

Table 4.2: Results for the GVRP instances.

			Reference B&C algorithm							
	HI20		B&C			⋄B&C			Total	Total*
Set (#inst.)	#opt	time $\bar{T}$	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	#opt
GVRP-2										
A (27)	**26**	**713.2**	21	1000.3	1.4	26	605.2	0.1	26	27
B (23)	17	1104.5	**22**	**342.2**	2.9	23	156.5	0.0	22	23
P (24)	**23**	**474.5**	15	1482.3	4.1	16	1355.4	0.8	23	24
GC (5)	**1**	**2993.8**	0	*TL*	50.9	0	*TL*	3.6	1	1
GVRP-3										
A (27)	**27**	**83.0**	26	256.1	0.2	26	207.9	0.1	27	27
B (23)	**23**	160.1	**23**	**10.8**	0.0	23	7.5	0.0	23	23
P (24)	**24**	**106.6**	23	253.7	0.1	23	244.9	0.0	24	24
GC (5)	**1**	3221.4	**1**	3084.8	9.7	1	2937.0	2.5	1	2
Total (158)	**142**	**605.1**	131	741.3	8.7	138	612.8	0.9	147	151

We provide instance-by-instance results with additional information in the Appendix.

4.5.4 Results for the Golden Instances

In this section, we use the Golden benchmark to compare the new B&C algorithm against the branch-and-price algorithm of Hintsch and Irnich (2020). The results are summarized in Table 4.3.

The branch-and-price of Hintsch and Irnich (2020) is for all twenty classes better than the B&C algorithms. We have two possible explanations for the rather poor performance of the B&C algorithm on the Golden benchmark: First, the instances are much larger than the GVRP instances with vertices symmetrically distributed (on a circle, square grid, and star, see Section 4.4). The results of the previous section showed that the B&C algorithm is strong for truly clustered instances. The Golden instances are however not nicely clustered.

Second, the capacity restriction is most of the time not tight. In fact, for almost all instances, the number of required vehicles can be reduced. Indeed, instances of the SoftCluVRP$^{\leq m}$ most of the time have optimal solutions with strictly lower costs. To show this, we perform an additional run for the Golden instances where the number of vehicles is limited but not fixed to m. For the SoftCluVRP$^{\leq m}$, more instances are solved optimally (26 instead of 21) in less time (on average) compared to the SoftCluVRP with fixed fleet size. Detailed results can be found in the Appendix.

On the positive side, the ⋄B&C algorithm always provides a valid lower bound

Table 4.3: Results for the `Golden` instances.

			Reference B&C algorithm									
	HI20		B&C			⋄B&C			simple LB		Total	Total*
Set (#inst.)	#opt	time $\bar{T}$	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	gap	#opt	#opt
`Golden`												
1 (11)	**1**	**3300.4**	0	*TL*	5.4	0	*TL*	2.1	0	4.6	1	1
2 (11)	**1**	**3530.5**	0	*TL*	5.7	0	*TL*	2.1	0	3.3	1	1
3 (11)	0	*TL*	0	*TL*	6.1	0	*TL*	2.0	0	2.8	0	0
4 (11)	0	*TL*	0	*TL*	5.7	0	*TL*	1.9	0	2.1	0	0
5 (11)	**10**	**832.4**	6	2486.6	3.7	6	2216.4	1.3	0	5.8	10	11
6 (11)	**5**	**2702.1**	1	3558.2	5.5	1	3415.7	2.1	0	3.2	5	5
7 (11)	0	*TL*	0	*TL*	4.9	0	*TL*	1.9	0	2.6	0	4
8 (11)	0	*TL*	0	*TL*	6.1	0	*TL*	1.8	0	2.6	0	0
9 (11)	**8**	**1445.5**	0	*TL*	4.8	0	*TL*	4.4	7	0.4	9	10
10 (11)	**6**	**2453.1**	0	*TL*	8.8	0	*TL*	3.6	0	0.7	6	7
11 (11)	**1**	**3514.6**	0	*TL*	5.2	0	*TL*	4.0	2	0.4	3	9
12 (11)	0	*TL*	0	*TL*	8.5	0	*TL*	3.1	0	0.8	0	0
13 (11)	**9**	**1118.8**	4	2963.2	1.6	4	3021.6	0.3	6	0.2	11	11
14 (11)	**4**	**2860.9**	0	*TL*	3.4	0	*TL*	0.6	0	0.2	4	8
15 (11)	**1**	**3561.5**	0	*TL*	3.2	0	*TL*	0.6	6	0.1	6	8
16 (11)	0	*TL*	0	*TL*	3.4	0	*TL*	0.8	3	0.2	3	4
17 (11)	**8**	**1314.4**	**8**	1629.7	1.0	8	1654.5	0.2	0	2.0	8	9
18 (11)	**6**	**2338.4**	2	3259.3	4.1	3	3165.8	0.9	0	2.0	6	8
19 (11)	**5**	**2741.5**	0	*TL*	6.6	0	*TL*	3.4	0	2.8	5	8
20 (11)	**3**	**3032.7**	0	*TL*	6.5	0	*TL*	5.1	0	2.6	3	4
Total (220)	**68**	**2817.3**	21	3394.9	5.0	22	3373.7	2.1	24	2.0	81	108

so that the overall average gap for the `Golden` instances is approximately 2 %. In contrast, the branch-and-price algorithm failed to provide a valid lower bound in approximately one third of the cases (only 137 of 220 with LB). This is clearly a point in favor of the B&C algorithm.

What we can also see from Table 4.3 and the *simple LB* section is that sometimes the lower bounding procedures of Section 4.4.2 are very effective. For three classes, the *simple LB* computation is successful and provides a proof of optimality for more instances than could be solved with either branch-and-price or B&C. Overall, the new lower bounds allow proving optimality for nine previously open instances.

4.5.5 Results for Square Grid Instances

In this final experiment, we analyze and quantify the impact that the reduction technique of Section 4.4.1 has on the performance of the B&C algorithm. We use the 90 self-generated instances with customers located on a square grid and nontruncated Euclidean distances.

Both B&C algorithms that are compared use the default B&C once for the reduced and once for the non-reduced edge set. The number of columns and rows of the model is reduced on average by 17.8% and 16.7%, respectively. As before, we compare both variants with the help of performance profiles, see Figure 4.5.

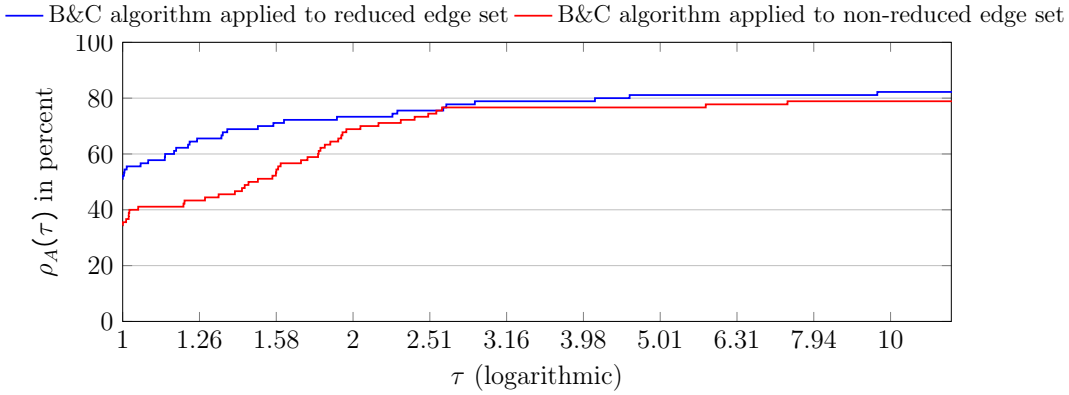


Figure 4.5: Performance profile for the reduced and non-reduced edge set.

The profiles clearly indicate that the reduction procedure is effective and has a relatively strong impact. Comparing both algorithms at $\tau = 1$, the algorithm with reduced (non-reduced) edge set is in 51% (34%) of the cases the fastest. For higher τ-values, the difference between both algorithms decreases but in total three more instances can be solved by the B&C algorithm that is applied to the reduced graph.

Results summarized in Table 4.4 also confirm the positive effect of the edge reduction: Average time and gap are overall smaller for the reduced version. In total 75 of 90 instances can be solved to proven optimality. Detailed results including number of columns and rows of the model can be found in the Appendix.

Table 4.4: Results for square `Grid` instances.

		reduced			non-reduced			Total	
Set (#inst.)	n+1	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	gap
`Grid`									
1-30 (30)	121	**28**	**605.8**	**0.4**	27	643.1	0.5	28	0.3
31-60 (30)	169	**24**	1091.6	**1.8**	23	**1084.2**	2.8	25	1.6
61-90 (30)	225	**22**	**1470.3**	5.4	21	1487.2	**5.1**	22	4.6
Total (90)		**74**	**1055.9**	**2.6**	71	1071.5	2.8	75	2.2

4.6 Conclusions

In this paper, we provided a first two-index formulation for the soft-clustered vehicle-routing problem (SoftCluVRP). The novelty of the model is the very clear separation of the standard routing part as known for the capacitated VRP from the part that ensures routes that respect the soft-cluster constraints. This latter part of the model uses an asymmetric and directed cluster graph so that MTZ-like constraints can be used here. Overall, the formulation is rather simple to use with modern MIP solvers, because only capacity constraints have to be added dynamically, i.e., in a cutting-plane fashion. All other valid inequalities that we presented are not mandatory when the fleet is not made artificially larger than necessary.

On the theoretical side, there are several contributions: First, we analyzed the impact of the fleet-size constraint and derived new constraints that allow to cope with non-minimal fleets. Second, we proved that one third of the transitivity constraints are actually redundant, for the complete integer formulation as well as for its linear relaxation. Third, we presented new capacity cuts that are stronger than the straightforward adaptation of the well-known version for the capacitated VRP. Finally, we prove a deep result that SoftCluVRPs defined on a square grid are reducible.

On the algorithmic side, the new branch-and-cut (B&C) algorithm is complementary to the only other exact solution approach from the literature that is based

on branch-and-price. While not competitive on all instance classes, the B&C algorithm is particularly useful for SoftCluVRP instances that are truly clustered. A B&C approach is also much simpler to implement compared to the sophisticated branch-and-price of Hintsch and Irnich (2020). Overall, the B&C algorithm and lower-bounding strategies deliver new provably optimal solutions to five `GVRP` and nine `Golden` benchmark instances that were unsolved before.

For the future, soft-clustered capacitated arc-routing problems are worthwhile to study (as motivated by Hintsch *et al.*, 2019). For problems in arc routing, branch-and-cut-based approaches have been found very competitive compared to branch-price-and-cut algorithms. One could therefore develop and analyze branch-and-cut algorithms for soft-clustered capacitated arc-routing problems. Moreover, additional real-world constraints could be integrated such as limits on the length and duration of a route as well as balancing constraints ensuring a similar workload on all constructed routes. This is relevant in both the node-routing and arc-routing context. Finally, we feel that more integrated approaches are needed to shed light on the interplay between a predetermined clustering/districting and the later vehicle routing, i.e., between typically separately considered tactical and operational problems in parcel delivery.

Bibliography

Adulyasak, Y., Cordeau, J.-F., and Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, **26**(1), 103–120.

Ahr, D. (2004). *Contributions to Multiple Postmen Problems*. Ph.d. dissertation, Department of Computer Science, Heidelberg University, Heidelberg, Germany.

Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. Ph.D. thesis, Institut National Polytechnique de Grenoble, Grenoble, France.

Battarra, M., Erdoğan, G., and Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Operations Research*, **62**(1), 58–71.

Bektaş, T., Erdoğan, G., and Røpke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, **45**(3), 299–316.

Defryn, C. and Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, **83**, 78–94.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Fischetti, M., Salazar González, J. J., and Toth, P. (1995). Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*.

Fischetti, M., Salazar Gonzáles, J. J., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, **45**(3), 378–394.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, **4**(1), 61–75.

Furtado, M. G. S., Munari, P., and Morabito, R. (2017). Pickup and delivery problem with time windows: A new compact two-index formulation. *Operations Research Letters*, **45**(4), 334–341.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In *Fleet Management and Logistics*, pages 33–56. Springer US.

Hintsch, T. (2019). Large multiple neighborhood search for the soft-clustered vehicle-routing problem. Technical Report LM-2019-01, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hintsch, T. and Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, **270**(1), 118–131.

Hintsch, T. and Irnich, S. (2020). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, **280**(1), 164–178.

Hintsch, T., Irnich, S., and Kiilerich, L. (2019). Branch-price-and-cut for the soft-clustered capacitated arc-routing problem. Technical Report LM-2019-02, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Karger, D. R. (1993). Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Laporte, G., Nobert, Y., and Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, **33**(5), 1050–1073.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, **100**(2), 423–445.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulations of traveling salesman problems. *Journal of the Association for Computing Machinery (JACM)*, **7**, 326–329.

Munari, P. and Savelsbergh, M. (2019). Compact formulations for split delivery routing problems. Technical Report MS01/2019, Operations Research Group, Production Engineering Department, Federal University of SÃ£o Carlos, Brazil.

Sevaux, M. and Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME*, volume 8, pages 411–417.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Vidal, T., Battarra, M., Subramanian, A., and Erdoğan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, **58**, 87–99.

Appendix

Proof of Theorem 1:

Our proof is by contradiction, i.e., we assume an optimal EucTSP tour in which the vertex in the middle of the (3×3)-vertex block (the vertex $\diamond$) is connected to a vertex $x = (x_1, x_2)$ that is not part of the (3×3)-vertex block.

According to Flood [The traveling-salesman problem. *Operations Research*, **4**(1), 1956, 61–75.], an optimal EucTSP tour is without any crossings in the Euclidean plane. Therefore, the vertex $\diamond$ cannot be connected to any other vertex on the horizontal or vertical line crossing $\diamond$ outside the (3×3)-vertex block. For the same reason, the vertex $\diamond$ cannot be connected to any points on the diagonals (45° and 135°) outside the 3×3-vertex block (the diagonals cross $\diamond$ and the top-left and bottom-right vertex of the (3×3)-vertex block or the top-right and bottom-left vertices of the block).

Exploiting symmetry, we can assume that x is one of the green vertices $\bullet$ depicted in Table 4.5. Moreover, the vertex $\star$, right of $\diamond$, must be connected to two other vertices; these are denoted by $y = (y_1, y_2)$ and $z = (z_1, z_2)$. Possible y-coordinates have a gray background. The coordinates of x, y, and z are measured according to a coordinate system in which $\star$ is the origin. W.l.o.g. the minimal distance is one so that $\diamond = (-1, 0)$, $\star = (0, 0)$, and $x, y, z \in \mathbb{Z}^2$.

Because crossings are prohibited, the vertex $\star$ must be connected to green or purple vertices (depicted as $\bullet$ and $\bullet$, see Cases 1–3 in Table 4.5) or to the vertex $\diamond$ (see Cases 4–7 in Table 4.5).

For all seven cases, we can construct a (strictly) shorter EucTSP tour proving that the original tour was not optimal. Table 4.5 summarizes all cases, where below each grid the corresponding distances are calculated and y-vertices are marked with a gray background.

In the following, the inequalities of all cases are derived in detail. Simple calculations (square and estimate) show that

$$\sqrt{(x_1 + 1)^2 + x_2^2} \geq \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} \qquad (*)$$

holds true for $x_1 \geq x_2 \geq 0$. This inequality is helpful for some intermediate steps. Note that in general $x_1, x_2 \geq 1$ and $x_1 \geq x_2$.

Case 1: Note first that in case of $x_1 \geq y_1$ and $x_2 \leq y_2$ two lines of x and y would cross. Hence, this case does not need to be considered. Three other cases remain: First, if $x_1 \geq y_1$ and $x_2 \geq y_2$ then $\sqrt{y_1^2 + y_2^2} \geq 1$ and $\sqrt{(x_1 + 1)^2 + x_2^2} > \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$. Second, if $x_1 \leq y_1$ and $x_2 \leq y_2$ then $\sqrt{y_1^2 + y_2^2} > \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $\sqrt{(x_1 + 1)^2 + x_2^2} > 1$.

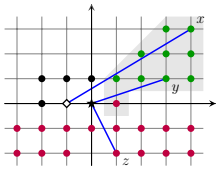
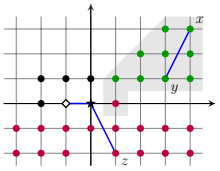
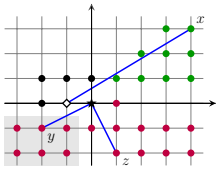
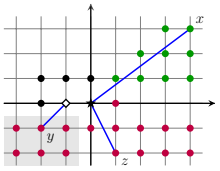
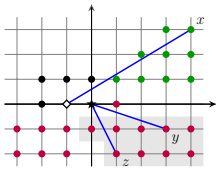
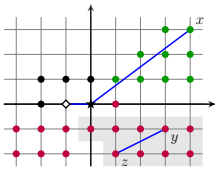
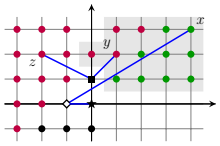
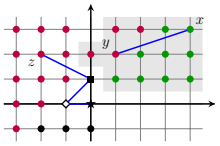
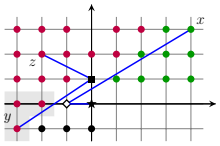
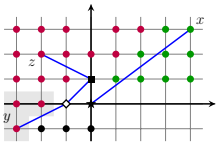
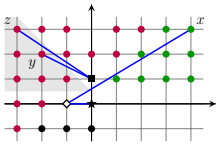
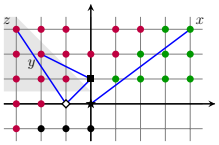
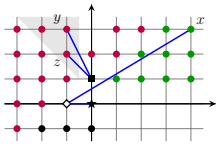
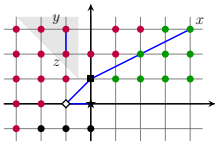
Case	Assumption	TSP tour	Improved TSP tour
1	$z - \star$, $\star - y$ $y_1 \geq 1, y_2 \geq 0$		
	$\sqrt{y_1^2 + y_2^2} + \sqrt{(x_1+1)^2 + x_2^2} > 1 + \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$		
2	$z - \star$, $\star - y$ $y_1 \leq -1, y_2 \leq -1$		
	$\sqrt{y_1^2 + y_2^2} + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$		
3	$z - \star$, $\star - y$ $z_1 \geq 1, z_2 \leq -1$ $y = (0, -1)$ or $y_1 \geq 1$, $y_2 \leq -1$		
	$\sqrt{(x_1+1)^2 + x_2^2} + \sqrt{y_1^2 + y_2^2} + \sqrt{z_1^2 + z_2^2} > 1 + \sqrt{x_1^2 + x_2^2} + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$		
4	$\diamond - \star$ $y_1, y_2 \geq 1$ or $y = (0, 2)$		
	$\sqrt{y_1^2 + (y_2 - 1)^2} + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2} + \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$		
5	$\diamond - \star$ $y_1 \leq -2$, $y_2 \leq 0$		
	$1 + \sqrt{y_1^2 + (\lvert y_2\rvert + 1)^2} + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2} + \sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$		
6	$\diamond - \star$ $y_1 \leq -1, y_2 \geq 1$ $\lvert y_1\rvert \geq y_2$		
	$1 + \sqrt{y_1^2 + (y_2 - 1)^2} + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2} + \sqrt{(\lvert y_1\rvert - 1)^2 + y_2^2} + \sqrt{x_1^2 + x_2^2}$		
7	$\diamond - \star$ $y_1 \leq -1, y_2 \geq 1$ $\lvert y_1\rvert < y_2$ $\lvert z_1\rvert < z_2$		
	$\sqrt{y_1^2 + (y_2 - 1)^2} + \sqrt{z_1^2 + (z_2 - 1)^2} + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2} + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2} + \sqrt{x_1^2 + x_2^2}$		

Table 4.5: Original and improved TSP tours.
Note: The horizontal axis is the first axis for x_1, y_1, and z_1, while the vertical axis is the second axis for x_2, y_2, and z_2.

Third, if $x_1 \leq y_1$ and $x_2 \geq y_2$ then we distinguish three subcases: If $y = (1, 0)$ the inequality holds true because $1 + \sqrt{(x_1 + 1)^2 + x_2^2} > 1 + \sqrt{(x_1 - 1)^2 + x_2^2}$. If $\sqrt{(x_1 + 1)^2 + x_2^2} \geq \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $y_1, y_2 \geq 1$, the inequality holds true because $\sqrt{y_1^2 + y_2^2} > 1$. Otherwise, let $\sqrt{(x_1 + 1)^2 + x_2^2} < \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ and $y_1, y_2 \geq 1$. We consider the triangle defined by the vertices $\mathbf{0}$, x, and y depicted in Figure 4.6. Let β be the angle between the edges (x, y) and $(\mathbf{0}, x)$. Since $x_1 \geq x_2$ and $y_2 \geq 1$, the angle is obtuse, i.e., $\beta > 90$. Let $v = (v_1, v_2)$ be the intersection of the line crossing $\mathbf{0}$ and y and the line forming a 90-degree angle with vertices x and y. It follows that $\sqrt{y_1^2 + y_2^2} > \sqrt{(v_1 - y_1)^2 + (v_2 - y_2)^2} > \cos(\beta)\sqrt{(v_1 - y_1)^2 + (v_2 - y_2)^2} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

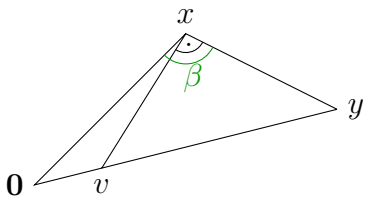


Figure 4.6: Triangle described in Case 1.

Case 2: Obviously, $\sqrt{y_1^2 + y_2^2} > \sqrt{(|y_1| - 1)^2 + y_2^2}$ and $\sqrt{(x_1 + 1)^2 + x_2^2} > \sqrt{x_1^2 + x_2^2}$.

Case 3: For the x-part, we find again $\sqrt{(x_1 + 1)^2 + x_2^2} > \sqrt{x_1^2 + x_2^2}$.

For the y-part, we consider the four different cases that distinguish the positions of y relative to z: First, if $y_1 \geq z_1$ and $y_2 \leq z_2$ then $\sqrt{y_1^2 + y_2^2} \geq \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$. Second, if $y_1 \leq z_1$ and $y_2 \geq z_2$ then $\sqrt{z_1^2 + z_2^2} \geq \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$. Third, if $y_1 \geq z_1$ and $y_2 \geq z_2$, it follows that $\sqrt{y_1^2 + y_2^2} + \sqrt{z_1^2 + z_2^2} \geq y_1 + \sqrt{1 + z_2^2} \geq 1 + \sqrt{(y_1 - 1)^2 + z_2^2} \geq 1 + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$. Fourth and finally, if $y_1 \leq z_1$ and $y_2 \leq z_2$, it follows that $\sqrt{y_1^2 + y_2^2} + \sqrt{z_1^2 + z_2^2} \geq y_2 + \sqrt{z_1^2 + 1} \geq 1 + \sqrt{z_1^2 + (y_2 - 1)^2} \geq 1 + \sqrt{(y_1 - z_1)^2 + (y_2 - z_2)^2}$. Note that the inequalities in the third and fourth case result again from squaring and estimating (as done for (*)).

Case 4: The only possibility for $y_1 = 0$ is the point $y = (0, 2)$. We consider two subcases for x_2: On the one hand, we assume that $x_2 = 1$. It follows that $1 + \sqrt{(x_1 + 1)^2 + 1} > \sqrt{2} + \sqrt{x_1^2 + 1}$ [it is simple to show that the larger x_1, the smaller is the difference between left-hand side and ride-hand side; the most critical

case is therefore $x_1 = 1$, in which case the two sides evaluate to $1 + \sqrt{5} > \sqrt{2} + \sqrt{2}$]. On the other hand, we assume $x_2 \geq 2$. Then, $1 + \sqrt{(x_1+1)^2 + x_2^2} \overset{(*)}{\geq} 1 + \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} > \sqrt{2} + \sqrt{x_1^2 + (x_2-2)^2}$.

If $y_1, y_2 \geq 1$, we consider four subcases according to the positions of x relative to y: First, in case of $x_1 \geq y_1$ and $x_2 \leq y_2$ two lines of x and y would cross so that this case does not need to be considered. Second, if $x_1 \geq y_1$ and $x_2 \geq y_2$ then $\sqrt{y_1^2 + (y_2-1)^2} \geq 1$ and $\sqrt{(x_1+1)^2 + x_2^2} \overset{(*)}{\geq} \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} > \frac{1}{\sqrt{2}} + \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$. Third, if $x_1 \leq y_1$ and $x_2 \leq y_2$ then $\sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2}$ and $\sqrt{y_1^2 + (y_2-1)^2} > \sqrt{(y_1-x_1)^2 + (y_2-x_2)^2}$. Fourth and finally, if $x_1 \geq y_1$ and $x_2 \leq y_2$ then it follows by simple calculations that $\sqrt{y_1^2 + (y_2-1)^2} + \sqrt{(x_1+1)^2 + x_2^2} \geq \sqrt{1 + (y_2-1)^2} + x_1 + 1 > \sqrt{2} + \sqrt{(x_1-1)^2 + (y_2-1)^2} \geq \sqrt{2} + \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$.

Case 5: Obviously, $\sqrt{y_1^2 + (|y_2|+1)^2} > \sqrt{(|y_1|-1)^2 + y_2^2}$ and $\sqrt{(x_1+1)^2 + x_2^2} \overset{(*)}{\geq} \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} > \sqrt{2} - 1 + \sqrt{x_1^2 + x_2^2}$.

Case 6: For the x-part we know $1 + \sqrt{(x_1+1)^2 + x_2^2} > \sqrt{2} + \sqrt{x_1^2 + x_2^2}$. For the y-part, we exploit the precondition $|y_1| \geq y_2 \geq 1$. It follows by simple estimations that $\sqrt{y_1^2 + (y_2-1)^2} \geq \sqrt{(|y_1|-1)^2 + y_2^2}$.

Case 7: If $y_1 \geq 1$ and $y_2 \geq 2$ then $\sqrt{y_1^2 + (y_2-1)^2} \geq 1 + \sqrt{(y_1-1)^2 + (y_2-2)^2}$. It follows that $\sqrt{y_1^2 + (y_2-1)^2} + \sqrt{z_1^2 + (z_2-1)^2} + \sqrt{(x_1+1)^2 + x_2^2} \overset{(*)}{\geq} 2 + \sqrt{(y_1-1)^2 + (y_2-2)^2} + \sqrt{(z_1-1)^2 + (z_2-2)^2} + \frac{1}{\sqrt{2}} + \sqrt{x_1^2 + x_2^2} > \sqrt{2} + \sqrt{(y_1-z_1)^2 + (y_2-z_2)^2} + \sqrt{x_1^2 + x_2^2}$.

Two final remarks are due: The difference between the left-hand and right-hand side of the $>$-inequalities is in all the (sub)cases greater than 0.2. Hence, the tours depicted on the right-hand side are always strictly improving.

Moreover, the improved tours of Cases 5 and 6 are certainly not optimal: The middle vertex $\diamond$ of the (3×3)-vertex block is still connected to an outer vertex. Hence, these tours can be further shortened using one of the Cases 1, 2, or 3.

Proof of Proposition 4:

Let $(\bar{x}, \bar{y}, \bar{u})$ be an integer feasible solution to model (R). We consider the connected components of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a = 1$. Let $(H(O), A(O))$ be one of these components and let $O = \{h_1, h_2, \ldots, h_{|O|}\}$ with

$h_1 < h_2 < \cdots < h_{|O|}$. In a first step, we show that

$$(h_i, h_{i+1}) \in A(O), \text{ i.e., } \bar{y}_{h_i,h_{i+1}} = 1 \qquad \text{for all } i = 1, 2, \ldots, |O| - 1 \tag{4.7}$$

holds true. If the opposite were true, i.e., $(h_i, h_{i+1}) \notin A(O)$, we consider a h_i-h_{i+1}-path P in $(H(O), A(O))$ with a minimum number of arcs (disregarding the direction of the arcs). Note that P must exist by definition of a connected component and that P and (h_i, h_{i+1}) form a cycle (again disregarding the direction of the arcs). We now consider the minimum and maximum vertices $h_{\min} = \min H(P)$ and $h_{\max} = \max H(P)$ w.r.t. the $<$-relation, respectively. Note that $h_{\min} = h_i$ and $h_{\max} = h_{i+1}$ at the same time is not possible.

If $h_{\min} < h_i$, the vertex $h_{\min}$ has an out-degree of 2 in P, i.e., there exist two arcs $(h_{\min}, h)$ and $(h_{\min}, h') \in A(O)$ with $h < h'$. Transitivity constraints (4.1k) for $h_{\min} < h < h'$ then imply that also $\bar{y}_{hh'} = 1$, i.e, $(h, h') \in A(O)$. This however contradicts with the minimality of P, because replacing (h_{min}, h) and (h_{min}, h') by (h, h') in P would create a shorter h_i-h_j-path.

If $h_{\max} > h_{i+1}$, the vertex $h_{\max}$ has an in-degree of 2 in P. The same type of argument can now be used together with the transitivity constraints (4.1j) leading to a shorter h_i-h_j-path contradicting with the minimality of P. Therefore, (4.7) must hold true.

In a second step, we show that the $\bar{y}$-values can be set to one within each component without violating any constraint. Let $(h_i, h_j) \in A(O)$ be an arc with $\bar{y}_{h_i h_j} = 0$. Because of (4.7), we know that $j > i + 1$ holds true. Summing up (4.1h) over the arcs $(h_i, h_{i+1}), (h_{i+1}, h_{i+2}), \ldots, (h_{j-1}, h_j)$ gives $\bar{u}_{h_i} - \bar{u}_{h_j} \leq -\sum_{k=i+1}^{j} d_{h_k} \leq -d_{h_j}$. This proves that (4.1h) for the arc $(h_i, h_j) \in A$ is also fulfilled even after increasing $\bar{y}_{h_i,h_j}$ to 1. Hence, all arcs with both endpoints in O can be increased without violating (4.1i).

Note that the new values $\bar{y}$ for the y-variables do not violate any transitivity constraint (4.1j)–(4.1k) because the positive $\bar{y}$ represent the transitive closure over O.

Proof of Proposition 5:

Let $(\bar{x}, \bar{y}, \bar{u})$ be a feasible solution to the linear relaxation of model (R). The feasible solution $(\bar{x}, \bar{\bar{y}}, \bar{u})$ to the linear relaxation of model (4.1) is constructed as follows. We consider the connected components of the digraph $D_{\bar{y}}$ spanned by the arcs a with $\bar{y}_a > 0$. Note that $\bar{y}_{h,h'} = 0$ for h and h' in different components. Accordingly, we also set $\bar{\bar{y}}_{h,h'} = 0$.

Within each component, the $\bar{\bar{y}}$-values are defined recursively. For this purpose, let $(H(O), A(O))$ be one of these components and let $C = \{h_1, h_2, \ldots, h_{|O|}\}$ with

$h_1 < h_2 < \cdots < h_{|O|}$. We define

$$\bar{\bar{y}}_{h_i h_{i+1}} := \bar{y}_{h_i h_{i+1}} \qquad \text{for all } i = 1, \ldots, |O| - 1;$$
$$\bar{\bar{y}}_{h_i h_{i+2}} := \max\{\bar{y}_{h_i h_{i+2}}, \bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1\} \qquad \text{for all } i = 1, \ldots, |O| - 2;$$

and for all $k = 3, \ldots, |O| - 1$

$$\bar{\bar{y}}_{h_i h_{i+k}} := \max\{\bar{y}_{h_i h_{i+k}}, \max_{j \in \{1,\ldots,k-1\}} \{\bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1\}\} \qquad \text{for all } i = 1, \ldots, |O| - k.$$

We have to show that $(\bar{x}, \bar{\bar{y}}, \bar{u})$ is a feasible solution to the linear relaxation of (4.1). Since $\bar{\bar{y}} \geq \bar{y}$, it suffices to show that this solution fulfills the MTZ-constraints (4.1h) and transitivity constraints (4.1j)–(4.1l). Obviously, it suffices to check the validity of these constraints for arcs or pairs of arcs $(h_i, h_{i+k}) \in A(O)$.

Regarding MTZ-constraints (4.1h), if $\bar{\bar{y}}_{h_i, h_{i+k}} = \bar{y}_{h_i, h_{i+k}}$ then (4.1h) is fulfilled because $(\bar{x}, \bar{y}, \bar{u})$ is feasible. In addition, this solves the subcases $k = 1$, i.e., arcs (h_i, h_{i+1}). Otherwise, we first consider the case of $k = 2$ and $\bar{\bar{y}}_{h_i h_{i+2}} = \bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1$. It follows

$$\begin{aligned} u_{h_i} - u_{h_{i+2}} + Q\bar{\bar{y}}_{h_i h_{i+2}} &= u_{h_i} - u_{h_{i+1}} + u_{h_{i+1}} - u_{h_{i+2}} + Q(\bar{y}_{h_i h_{i+1}} + \bar{y}_{h_{i+1} h_{i+2}} - 1) \\ &= (u_{h_i} - u_{h_{i+1}} + Q\bar{y}_{h_i h_{i+1}}) + (u_{h_{i+1}} - u_{h_{i+2}} + Q\bar{y}_{h_{i+1} h_{i+2}}) - Q \\ &\overset{(4.1h)}{\leq} Q - d_{h_{i+1}} + Q - d_{h_{i+2}} - Q \leq Q - d_{h_{i+2}}. \end{aligned}$$

where we exploit (4.1h) for the $\bar{y}$-values.

For $k > 2$, we assume that the values of $\bar{\bar{y}}_{h_i h_{i+k}}$ results from an index $j \in \{1, \ldots k-1\}$ with $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1$. By induction over k, we get

$$\begin{aligned} u_{h_i} - u_{h_{i+k}} + Q\bar{\bar{y}}_{h_i h_{i+k}} &= u_{h_i} - u_{h_{i+j}} + u_{h_{i+j}} - u_{h_{i+k}} + Q(\bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1) \\ &= (u_{h_i} - u_{h_{i+j}} + Q\bar{\bar{y}}_{h_i h_{i+j}}) + (u_{h_{i+j}} - u_{h_{i+k}} + Q\bar{\bar{y}}_{h_{i+j} h_{i+k}}) - Q \\ &\overset{(4.1h)}{\leq} Q - d_{h_{i+j}} + Q - d_{h_{i+k}} - Q \leq Q - d_{h_{i+k}}. \end{aligned}$$

where we exploit (4.1h) for smaller k by induction hypothesis. This completes all case for the MTZ-constraints (4.1h).

Next we prove that the first two classes of transitivity constraints (4.1j) and (4.1k) hold true for the $\bar{\bar{y}}$-values. The proof is also by induction over $k \geq 2$ for pairs of arcs (h_i, h_{i+j}) and $(h_{i+j}, h_{i+k}) \in A(O)$, i.e., with $h_i < h_{i+j} < h_{i+k}$. For $k = 2$, the arc pair is (h_i, h_{i+1}) and (h_{i+1}, h_{i+2}) so that the result directly follows because $\bar{\bar{y}}_{h_i h_{i+l}} = \bar{y}_{h_i h_{i+l}}$ and $\bar{\bar{y}}_{h_{i+1}, h_{i+2}} = \bar{y}_{h_{i+1}, h_{i+2}}$.

For $k > 2$, we show w.l.o.g. that transitivity constraints (4.1j) are fulfilled (the proof for the second class of transitivity constraints (4.1k) works analogously). Let $l \in \{1, \ldots, k-1\}$. We distinguish three cases.

The first case is $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{y}_{h_i h_{i+k}}$ and $\bar{\bar{y}}_{h_{i+l} h_{i+k}} = \bar{y}_{h_{i+l} h_{i+k}}$. Then,

$$\bar{\bar{y}}_{h_i h_{i+l}} \geq \bar{y}_{h_i h_{i+l}} \overset{(4.1j)}{\geq} \bar{y}_{h_i h_{i+k}} + \bar{y}_{h_{i+l} h_{i+k}} - 1 = \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1. \tag{4.8}$$

In the second case, let $\bar{\bar{y}}_{h_i h_{i+k}} = \bar{y}_{h_i h_{i+k}}$ and $\bar{\bar{y}}_{h_{i+l} h_{i+k}} > \bar{y}_{h_{i+l} h_{i+k}}$. The latter implies (using the definition of the $\bar{\bar{y}}$-values) that there exist indices $l_1 < l_2 < \cdots < l_s$ with $l_1 > l$ and $l_s < k$ for an $s \in \{1, \ldots, k-l\}$ such that $\bar{\bar{y}}_{h_{i+l} h_{i+k}}$ can be expressed by $\bar{y}$-values in the following form:

$$\bar{\bar{y}}_{h_{i+l} h_{i+k}} = \bar{y}_{h_{i+l} h_{i+l_1}} + \bar{y}_{h_{i+l^1} h_{i+l_2}} + \cdots + \bar{y}_{h_{i+l_{s-1}} h_{i+k}} + \bar{y}_{h_{i+l_s} h_{i+k}} - s + 1 \tag{4.9}$$

Iteratively exploiting constraints (4.1j) for the $\bar{y}$-values yields

$$\begin{aligned}
\bar{\bar{y}}_{h_i h_{i+l}} &\geq \bar{y}_{h_i h_{i+l}} \overset{(4.1j)}{\geq} \bar{y}_{h_i h_{i+l_1}} + \bar{y}_{h_{i+l} h_{i+l_1}} - 1 \overset{(4.1j)}{\geq} \bar{y}_{h_i h_{i+l_2}} + \bar{y}_{h_{i+l_1} h_{i+l_2}} + \bar{y}_{h_{i+l} h_{i+l_1}} - 2 \\
&\overset{(4.1j)}{\geq} \ldots \overset{(4.1j)}{\geq} \bar{y}_{h_i h_{i+k}} + \bar{y}_{h_{i+l_s} h_{i+k}} + \bar{y}_{h_{i+l_{s-1}} h_{i+l_s}} + \ldots + \bar{y}_{h_{i+l_1} h_{i+l_2}} + \bar{y}_{h_{i+l} h_{i+l_1}} - s \\
&\overset{(4.9)}{\geq} \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1.
\end{aligned}$$

In the third and last case, let $\bar{\bar{y}}_{h_i h_{i+k}} > \bar{y}_{h_i h_{i+k}}$. Again, using the definition of the $\bar{\bar{y}}$-values, we know that there exists a $j \in \{1, \ldots, k-1\}$ such that

$$\bar{\bar{y}}_{h_i h_{i+k}} = \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+k}} - 1. \tag{4.10}$$

By induction hypothesis, constraints

$$\bar{\bar{y}}_{h_i h_{i+l}} \geq \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+l}} - 1, \tag{4.11a}$$

$$\bar{\bar{y}}_{h_{i+j} h_{i+l}} \geq \bar{\bar{y}}_{h_{i+j} h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1 \tag{4.11b}$$

are satisfied, yielding

$$\begin{aligned}
\bar{\bar{y}}_{h_i h_{i+l}} &\overset{(4.11a)}{\geq} \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_{i+j} h_{i+l}} - 1 \overset{(4.11b)}{\geq} \bar{\bar{y}}_{h_i h_{i+j}} + \bar{\bar{y}}_{h_j h_{i+k}} + \bar{\bar{y}}_{h_l h_{i+k}} - 2 \\
&\overset{(4.10)}{=} \bar{\bar{y}}_{h_i h_{i+k}} + \bar{\bar{y}}_{h_{i+l} h_{i+k}} - 1.
\end{aligned}$$

This completes the proof for the transitivity constraints (4.1j) and (4.1k).

Finally, the validity of the third class of transitivity constraints (4.1l) directly results from the definition of the $\bar{\bar{y}}$-values.

Detailed Computational Results

In this Appendix, we present instance-by-instance results. The entries in the Tables 4.6–4.8 have the following meaning:

Group:	Subset of instances: `A`, `B`, `P`, `C`, and `G` for the `GVRP` instances; `Golden1`-`Golden20` for the `Golden` instances
No.:	Number of the instance for the self-generated, smaller-sized `Grid` instances
n:	Number of customers
k:	Number of vehicles in the original CVRP instance
N:	Number of customer clusters
m:	Number of vehicles for the SoftCluVRP
UB:	Upper bound; bold if $LB = UB$, i.e., optimality is proven
LB:	Lower bound; bold ditto

Moreover, we indicate which SoftCluVRP algorithm has computed/proven a(n) upper/lower bound for the first time ("first found by"):

HI20:	branch-and-price of Hintsch and Irnich (2020)
H19:	metaheuristic of Hintsch (2019)
DS17:	metaheuristic of Defryn and Sörensen (2017)
BEV19:	exact algorithm of Battarra *et al.* (2014)
B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none`
*B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none` with a time limit of 36,000 seconds (10 hours)
⋄B&C:	Trans:`dyn`, MTZ:`static`, Tree:`none`, BKS provided as UB
•B&C:	Trans:`static`, MTZ:`none`
†B&C:	Trans:`static`, MTZ:`static`, ProbCC:`none`
pretest:	Pretests with one of the algorithms described in Section 4.5.1
grid:	grid lower bound from Section 4.4.2
relax:	relax lower bound from model (4.6) with a time limit of 600 seconds, see Section 4.4.2

Table 4.6 displays the results for the `GVRP` instances, Table 4.7 for the `Golden` instances, and Table 4.8 for the self-generated `Grid` instances.

Table 4.6: Detailed results for the GVRP instances.

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
GVRP-2										
A	32	5	16	2	**595**	**595**	HI20	HI20	7	140
A	33	5	17	3	**528**	**528**	HI20	HI20	12	4
A	33	6	17	3	**561**	**561**	HI20	HI20	5	5
A	34	5	17	3	**568**	**568**	HI20	HI20	13	3
A	36	5	18	2	**596**	**596**	HI20	HI20	65	101
A	37	5	19	3	**573**	**573**	HI20	HI20	6	4
A	37	6	19	3	**660**	**660**	HI20	HI20	4	10
A	38	5	19	3	**547**	**547**	HI20	HI20	1	2
A	39	5	20	3	**659**	**659**	HI20	HI20	78	9
A	39	6	20	3	**676**	**676**	HI20	HI20	78	375
A	44	6	22	3	**723**	**723**	HI20	HI20	23	42
A	45	6	23	4	**679**	**679**	HI20	HI20	4	1
A	45	7	23	4	**774**	**774**	HI20	HI20	242	1856
A	46	7	23	4	**708**	**708**	HI20	HI20	209	49
A	48	7	24	4	**784**	**784**	HI20	HI20	1431	149
A	53	7	27	4	**732**	**732**	HI20	HI20	285	56
A	54	7	27	4	**806**	**806**	HI20	HI20	265	1147
A	55	9	28	5	**778**	**778**	HI20	HI20	84	68
A	60	9	30	5	**877**	**877**	HI20	HI20	2010	262
A	61	9	31	5	**749**	**749**	HI20	HI20	142	432
A	62	8	31	4	**849**	**849**	HI20	HI20	839	692
A	63	9	32	5	**1043**	**1043**	HI20	HI20	3159	*TL*
A	63	10	32	5	**895**	**895**	HI20	HI20	512	*TL*
A	64	9	32	5	**895**	**895**	HI20	HI20	1132	*TL*
A	65	9	33	5	**825**	**825**	HI20	HI20	2544	*TL*
A	69	9	35	5	**857**	**857**	HI20	HI20	2506	*TL*
A	80	10	40	5	**1115**	**1115**	HI20	HI20	*TL*	*TL*
B	31	5	16	3	**451**	**451**	HI20	HI20	7	1
B	34	5	17	3	**495**	**495**	HI20	HI20	26	<1
B	35	5	18	3	**654**	**654**	HI20	HI20	27	<1
B	38	6	19	3	**479**	**479**	HI20	HI20	3	2
B	39	5	20	3	**378**	**378**	HI20	HI20	5	<1

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
B	41	6	21	3	**514**	**514**	HI20	HI20	13	50
B	43	6	22	3	**522**	**522**	HI20	HI20	897	44
B	44	7	22	4	**562**	**562**	HI20	HI20	363	3
B	45	5	23	3	**542**	**542**	HI20	HI20	7	1
B	45	6	23	4	**506**	**506**	HI20	HI20	141	246
B	50	7	25	4	**495**	**495**	HI20	HI20	1	<1
B	50	8	25	5	**954**	**954**	HI20	B&C	*TL*	502
B	51	7	26	4	**672**	**672**	HI20	HI20	123	2
B	52	7	26	4	**485**	**485**	HI20	HI20	224	2
B	56	7	28	4	**520**	**520**	HI20	B&C	*TL*	61
B	57	7	29	4	**776**	**776**	H19	B&C	*TL*	7
B	57	9	29	5	**983**	**983**	HI20	HI20	251	1204
B	63	10	32	5	**865**	**865**	HI20	HI20	1402	722
B	64	9	32	5	**550**	**550**	HI20	HI20	21	5
B	66	9	33	5	**849**	**849**	H19	B&C	*TL*	207
B	67	10	34	5	**721**	**721**	H19	B&C	*TL*	1192
B	68	9	34	5	**745**	**745**	HI20	HI20	293	21
B	78	10	39	5	**842**	**842**	HI20	HI20	*TL*	*TL*
P	16	8	8	5	**299**	**299**	HI20	HI20	<1	<1
P	19	2	10	2	**195**	**195**	HI20	HI20	<1	<1
P	20	2	10	2	**208**	**208**	HI20	HI20	<1	1
P	21	2	11	2	**208**	**208**	HI20	HI20	<1	<1
P	22	2	11	2	**209**	**209**	HI20	HI20	<1	1
P	22	8	11	5	**397**	**397**	HI20	HI20	<1	1
P	23	8	12	5	**369**	**369**	HI20	HI20	2	5
P	40	5	20	3	**401**	**401**	HI20	HI20	10	13
P	45	5	23	3	**443**	**443**	HI20	HI20	5	24
P	50	7	25	4	**464**	**464**	HI20	HI20	119	173
P	50	8	25	4	**501**	**501**	HI20	HI20	230	*TL*
P	50	10	25	5	**512**	**512**	HI20	HI20	56	358
P	51	10	26	6	**548**	**548**	HI20	HI20	21	114
P	55	7	28	4	**477**	**477**	HI20	HI20	8	284
P	55	8	28	4	**484**	**484**	HI20	HI20	40	2096
P	55	10	28	5	**514**	**514**	HI20	HI20	5	105
P	55	15	28	8	**684**	**684**	HI20	HI20	70	*TL*

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
P	60	10	30	5	**575**	**575**	HI20	HI20	725	*TL*
P	60	15	30	8	**700**	**700**	HI20	HI20	216	*TL*
P	65	10	33	5	**616**	**616**	HI20	HI20	291	*TL*
P	70	10	35	5	**643**	**643**	HI20	HI20	934	*TL*
P	76	4	38	2	**557**	**557**	HI20	HI20	2745	*TL*
P	76	5	38	3	**571**	**571**	HI20	HI20	2311	*TL*
P	101	4	51	2	**645**	**645**	H19	HI20	*TL*	*TL*
G	101	10	51	5	**628**	**628**	HI20	HI20	569	*TL*
C	121	7	61	4	799	782	H19	*B&C	*TL*	*TL*
C	151	12	76	6	805	793	H19	HI20	*TL*	*TL*
C	200	16	100	8	944	910	H19	◇B&C	*TL*	*TL*
C	262	25	131	12	3655	3355	H19	B&C	*TL*	*TL*
GVRP-3										
A	32	5	11	2	**515**	**515**	DS17	HI20	1	<1
A	33	5	11	2	**461**	**461**	DS17	HI20	1	1
A	33	6	11	2	**554**	**554**	DS17	HI20	2	3
A	34	5	12	2	**538**	**538**	DS17	HI20	6	1
A	36	5	12	2	**543**	**543**	DS17	HI20	6	1
A	37	5	13	2	**545**	**545**	HI20	HI20	11	2
A	37	6	13	2	**605**	**605**	DS17	HI20	6	8
A	38	5	13	2	**507**	**507**	BEV19	HI20	5	<1
A	39	5	13	2	**588**	**588**	DS17	HI20	13	3
A	39	6	13	2	**603**	**603**	DS17	HI20	6	4
A	44	6	15	2	**691**	**691**	DS17	HI20	27	246
A	45	6	15	3	**652**	**652**	DS17	HI20	2	4
A	45	7	15	3	**661**	**661**	DS17	HI20	29	11
A	46	7	16	3	**642**	**642**	DS17	HI20	12	18
A	48	7	16	3	**680**	**680**	DS17	HI20	57	34
A	53	7	18	3	**627**	**627**	DS17	HI20	14	<1
A	54	7	18	3	**699**	**699**	DS17	HI20	143	41
A	55	9	19	3	**645**	**645**	DS17	HI20	9	13
A	60	9	20	3	**762**	**762**	DS17	HI20	29	1127
A	61	9	21	4	**671**	**671**	DS17	HI20	23	75
A	62	8	21	3	**771**	**771**	DS17	HI20	404	48
A	63	9	21	3	**837**	**837**	DS17	HI20	43	432

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
A	63	10	21	4	**779**	**779**	DS17	HI20	13	61
A	64	9	22	3	**767**	**767**	DS17	HI20	585	934
A	65	9	22	3	**693**	**693**	DS17	HI20	14	17
A	69	9	23	3	**794**	**794**	DS17	HI20	603	*TL*
A	80	10	27	4	**944**	**944**	DS17	HI20	178	228
B	31	5	11	2	**375**	**375**	BEV19	HI20	4	<1
B	34	5	12	2	**415**	**415**	DS17	HI20	5	<1
B	35	5	12	2	**557**	**557**	DS17	HI20	18	<1
B	38	6	13	2	**427**	**427**	DS17	HI20	3	1
B	39	5	13	2	**317**	**317**	DS17	HI20	<1	<1
B	41	6	14	2	**469**	**469**	DS17	HI20	12	<1
B	43	6	15	2	**405**	**405**	DS17	HI20	8	1
B	44	7	15	3	**443**	**443**	DS17	HI20	7	2
B	45	5	15	2	**489**	**489**	DS17	HI20	3	<1
B	45	6	15	2	**386**	**386**	DS17	HI20	4	5
B	50	7	17	3	**464**	**464**	DS17	HI20	16	2
B	50	8	17	3	**661**	**661**	DS17	HI20	5	8
B	51	7	17	3	**578**	**578**	DS17	HI20	17	1
B	52	7	18	3	**427**	**427**	BEV19	HI20	11	1
B	56	7	19	3	**420**	**420**	DS17	HI20	16	3
B	57	7	19	3	**622**	**622**	DS17	HI20	437	1
B	57	9	19	3	**746**	**746**	DS17	HI20	1606	24
B	63	10	21	3	**685**	**685**	BEV19	HI20	21	3
B	64	9	22	4	**524**	**524**	DS17	HI20	32	6
B	66	9	22	3	**683**	**683**	DS17	HI20	252	74
B	67	10	23	4	**619**	**619**	DS17	HI20	72	7
B	68	9	23	3	**582**	**582**	DS17	HI20	25	40
B	78	10	26	4	**704**	**704**	DS17	HI20	1109	68
P	16	8	6	4	**251**	**251**	DS17	HI20	<1	<1
P	19	2	7	1	**170**	**170**	DS17	HI20	<1	<1
P	20	2	7	1	**177**	**177**	DS17	HI20	<1	<1
P	21	2	7	1	**179**	**179**	DS17	HI20	<1	<1
P	22	2	8	1	**183**	**183**	DS17	HI20	<1	<1
P	22	8	8	4	**365**	**365**	BEV19	HI20	<1	<1
P	23	8	8	3	**270**	**270**	DS17	HI20	1	1

Continued on next page

Instance					Results					
							first found by		time T	
Group	$n+1$	k	N	m	UB	LB	UB	LB	HI20	B&C
P	40	5	14	2	**381**	**381**	DS17	HI20	8	2
P	45	5	15	2	**422**	**422**	DS17	HI20	2	3
P	50	7	17	3	**430**	**430**	DS17	HI20	4	9
P	50	8	17	3	**441**	**441**	DS17	HI20	3	23
P	50	10	17	4	**471**	**471**	DS17	HI20	5	40
P	51	10	17	4	**493**	**493**	DS17	HI20	2	11
P	55	7	19	3	**454**	**454**	HI20	HI20	21	94
P	55	8	19	3	**454**	**454**	HI20	HI20	9	51
P	55	10	19	4	**481**	**481**	DS17	HI20	4	21
P	55	15	19	6	**572**	**572**	DS17	HI20	9	86
P	60	10	20	4	**534**	**534**	HI20	HI20	61	703
P	60	15	20	5	**591**	**591**	DS17	HI20	39	967
P	65	10	22	4	**575**	**575**	HI20	HI20	8	43
P	70	10	24	4	**602**	**602**	DS17	HI20	30	240
P	76	4	26	2	**556**	**556**	HI20	HI20	382	50
P	76	5	26	2	**556**	**556**	DS17	HI20	71	143
P	101	4	34	2	**649**	**649**	DS17	HI20	1899	*TL*
G	101	10	34	4	**598**	**598**	DS17	HI20	1707	1024
C	121	7	41	3	680	673	H19	*B&C	*TL*	*TL*
C	151	12	51	4	**756**	**756**	HI20	HI20	*TL*	*TL*
C	200	16	67	6	865	858	H19	HI20	*TL*	*TL*
C	262	25	88	9	3178	2974	H19	◇B&C	*TL*	*TL*

Table 4.7: Detailed results for the Golden instances.

Instance				Results						SoftCluVRP$^{\leq m}$		
						first found by		time T				
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden1	241	17	4	**4640**	**4640**	HI20	HI20	304	*TL*	**4531**	**4531**	2709
Golden1	241	18	4	4645	4636	HI20	HI20	*TL*	*TL*	4539	4535	*TL*
Golden1	241	19	4	4650	4647	HI20	HI20	*TL*	*TL*	4597	4523	*TL*
Golden1	241	21	4	4650	4640	HI20	HI20	*TL*	*TL*	4582	4518	*TL*
Golden1	241	22	4	4650	4638	H19	HI20	*TL*	*TL*	4595	4517	*TL*
Golden1	241	25	4	4650	4614	H19	HI20	*TL*	*TL*	4628	4505	*TL*
Golden1	241	27	4	4652	4624	H19	HI20	*TL*	*TL*	4652	4478	*TL*
Golden1	241	31	4	4665	4632	H19	HI20	*TL*	*TL*	4665	4474	*TL*
Golden1	241	35	4	4619	4583	H19	HI20	*TL*	*TL*	4619	4441	*TL*
Golden1	241	41	4	4619	4525	H19	B&C	*TL*	*TL*	4619	4470	*TL*
Golden1	241	49	4	4607	4525	H19	B&C	*TL*	*TL*	4607	4470	*TL*
Golden2	321	22	4	7394	7393	H19	HI20	*TL*	*TL*	7394	7135	*TL*
Golden2	321	23	4	**7369**	**7369**	HI20	HI20	2836	*TL*	7369	7129	*TL*
Golden2	321	25	4	7367	7366	H19	HI20	*TL*	*TL*	7367	7126	*TL*
Golden2	321	27	4	7333	7329	H19	HI20	*TL*	*TL*	7333	7080	*TL*
Golden2	321	30	4	7329	7162	H19	B&C	*TL*	*TL*	7329	7107	*TL*
Golden2	321	33	4	7311	7162	H19	B&C	*TL*	*TL*	7311	7107	*TL*
Golden2	321	36	4	7293	7162	H19	•B&C	*TL*	*TL*	7293	7107	*TL*
Golden2	321	41	4	7283	7161	H19	B&C	*TL*	*TL*	7283	7106	*TL*
Golden2	321	46	4	7284	7161	H19	B&C	*TL*	*TL*	7284	7106	*TL*
Golden2	321	54	4	7274	7160	H19	B&C	*TL*	*TL*	7274	7105	*TL*
Golden2	321	65	4	7261	7160	H19	B&C	*TL*	*TL*	7261	7105	*TL*
Golden3	401	27	4	10077	10064	H19	HI20	*TL*	*TL*	10077	9733	*TL*
Golden3	401	29	4	10018	9795	H19	pretest	*TL*	*TL*	10018	9727	*TL*
Golden3	401	31	4	10002	9783	H19	⋄B&C	*TL*	*TL*	10002	9723	*TL*
Golden3	401	34	4	9995	9772	H19	†B&C	*TL*	*TL*	9995	9713	*TL*
Golden3	401	37	4	9986	9762	H19	⋄B&C	*TL*	*TL*	9986	9708	*TL*
Golden3	401	41	4	9926	9763	H19	•B&C	*TL*	*TL*	9926	9712	*TL*
Golden3	401	45	4	9936	9759	H19	B&C	*TL*	*TL*	9936	9704	*TL*
Golden3	401	51	4	9916	9742	H19	B&C	*TL*	*TL*	9916	9687	*TL*
Golden3	401	58	4	9910	9741	H19	B&C	*TL*	*TL*	9910	9686	*TL*
Golden3	401	67	4	9901	9741	H19	B&C	*TL*	*TL*	9901	9686	*TL*
Golden3	401	81	4	9868	9740	H19	B&C	*TL*	*TL*	9868	9685	*TL*
Golden4	481	33	4	12741	12409	H19	pretest	*TL*	*TL*	12741	12331	*TL*
Golden4	481	35	4	12740	12427	H19	pretest	*TL*	*TL*	12740	12325	*TL*
Golden4	481	37	4	12645	12376	H19	⋄B&C	*TL*	*TL*	12645	12323	*TL*
Golden4	481	41	4	12568	12375	H19	relax	*TL*	*TL*	12568	12310	*TL*
Golden4	481	44	4	12566	12375	H19	relax	*TL*	*TL*	12566	12315	*TL*
Golden4	481	49	4	12566	12375	H19	relax	*TL*	*TL*	12566	12324	*TL*
Golden4	481	54	4	12525	12367	H19	relax	*TL*	*TL*	12525	12307	*TL*
Golden4	481	61	4	12558	12367	H19	relax	*TL*	*TL*	12558	12294	*TL*
Golden4	481	69	4	12573	12347	H19	B&C	*TL*	*TL*	12573	12292	*TL*
Golden4	481	81	4	12555	12339	H19	B&C	*TL*	*TL*	12555	12282	*TL*
Golden4	481	97	4	12528	12337	H19	B&C	*TL*	*TL*	12528	12282	*TL*
Golden5	201	14	4	**6970**	**6970**	HI20	HI20	212	533	**6742**	**6742**	377
Golden5	201	15	3	**6742**	**6742**	HI20	HI20	280	582	**6742**	**6742**	241
Golden5	201	16	3	**6742**	**6742**	HI20	HI20	142	922	**6742**	**6742**	261
Golden5	201	17	3	**6862**	**6862**	HI20	HI20	380	1731	**6862**	**6862**	783
Golden5	201	19	4	**6874**	**6874**	HI20	HI20	180	2907	**6735**	**6735**	2485
Golden5	201	21	4	**6816**	**6816**	HI20	HI20	666	2679	**6637**	**6637**	1136
Golden5	201	23	4	**6750**	**6750**	HI20	HI20	260	*TL*	**6637**	**6637**	3352
Golden5	201	26	4	**6704**	**6704**	HI20	HI20	647	*TL*	**6521**	**6521**	1959

Continued on next page

Instance				Results						SoftCluVRP$^{\leq m}$		
						first found by		time T				
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden5	201	29	4	**6704**	**6704**	HI20	HI20	779	*TL*	**6521**	**6521**	*TL*
Golden5	201	34	4	**6684**	**6684**	HI20	HI20	*TL*	*TL*	6567	6389	*TL*
Golden5	201	41	4	**6557**	**6557**	HI20	HI20	2010	*TL*	6557	6317	*TL*
Golden6	281	19	3	**8115**	**8115**	HI20	HI20	1110	3140	**8115**	**8115**	1824
Golden6	281	21	3	**8119**	**8119**	HI20	HI20	901	*TL*	8119	8068	*TL*
Golden6	281	22	3	**8107**	**8107**	HI20	HI20	1053	*TL*	8107	8047	*TL*
Golden6	281	24	4	**8316**	**8316**	HI20	HI20	2491	*TL*	8267	8008	*TL*
Golden6	281	26	4	**8249**	**8249**	HI20	HI20	2568	*TL*	8225	7987	*TL*
Golden6	281	29	4	8244	8234	H19	HI20	*TL*	*TL*	8244	7966	*TL*
Golden6	281	32	4	8179	8175	H19	HI20	*TL*	*TL*	8179	7955	*TL*
Golden6	281	36	4	8179	8178	H19	HI20	*TL*	*TL*	8179	7947	*TL*
Golden6	281	41	4	8204	7995	H19	•B&C	*TL*	*TL*	8204	7938	*TL*
Golden6	281	47	4	8179	7970	H19	relax	*TL*	*TL*	8179	7913	*TL*
Golden6	281	57	4	8204	7960	H19	B&C	*TL*	*TL*	8204	7908	*TL*
Golden7	361	25	3	**9318**	**9318**	HI20	HI20	*TL*	*TL*	9318	9173	*TL*
Golden7	361	26	3	**9295**	**9295**	HI20	HI20	*TL*	*TL*	9295	9173	*TL*
Golden7	361	28	3	9271	9150	H19	†B&C	*TL*	*TL*	9271	9151	*TL*
Golden7	361	31	4	**9418**	**9418**	HI20	HI20	*TL*	*TL*	9418	9159	*TL*
Golden7	361	33	4	9395	9215	H19	•B&C	*TL*	*TL*	9395	9155	*TL*
Golden7	361	37	4	**9395**	**9395**	HI20	HI20	*TL*	*TL*	9395	9161	*TL*
Golden7	361	41	4	9386	9198	H19	⋄B&C	*TL*	*TL*	9386	9142	*TL*
Golden7	361	46	4	9368	9177	H19	†B&C	*TL*	*TL*	9368	9111	*TL*
Golden7	361	52	4	9365	9173	H19	⋄B&C	*TL*	*TL*	9365	9118	*TL*
Golden7	361	61	4	9316	9157	H19	B&C	*TL*	*TL*	9316	9102	*TL*
Golden7	361	73	4	9302	9157	H19	B&C	*TL*	*TL*	9302	9102	*TL*
Golden8	441	30	4	10409	10190	H19	pretest	*TL*	*TL*	10409	10122	*TL*
Golden8	441	32	4	10409	10197	H19	relax	*TL*	*TL*	10409	10120	*TL*
Golden8	441	34	4	10409	10177	H19	⋄B&C	*TL*	*TL*	10409	10121	*TL*
Golden8	441	37	4	10360	10198	H19	⋄B&C	*TL*	*TL*	10360	10142	*TL*
Golden8	441	41	4	10360	10219	H19	relax	*TL*	*TL*	10360	10155	*TL*
Golden8	441	45	4	10385	10198	H19	•B&C	*TL*	*TL*	10385	10141	*TL*
Golden8	441	49	4	10399	10195	H19	B&C	*TL*	*TL*	10399	10139	*TL*
Golden8	441	56	4	10371	10195	H19	B&C	*TL*	*TL*	10371	10139	*TL*
Golden8	441	63	4	10361	10195	H19	B&C	*TL*	*TL*	10361	10139	*TL*
Golden8	441	74	4	10356	10195	H19	B&C	*TL*	*TL*	10356	10139	*TL*
Golden8	441	89	4	10281	10195	H19	B&C	*TL*	*TL*	10281	10139	*TL*
Golden9	256	18	4	**281**	**281**	HI20	HI20	1287	*TL*	276	269	*TL*
Golden9	256	19	4	**279**	**279**	HI20	HI20	209	*TL*	276	269	*TL*
Golden9	256	20	4	**276**	**276**	HI20	HI20	112	*TL*	273	269	*TL*
Golden9	256	22	4	**276**	**276**	HI20	HI20	217	*TL*	276	269	*TL*
Golden9	256	24	4	**276**	**276**	HI20	HI20	175	*TL*	270	269	*TL*
Golden9	256	26	4	**273**	**273**	HI20	HI20	465	*TL*	273	266	*TL*
Golden9	256	29	4	**273**	**273**	HI20	HI20	985	*TL*	273	266	*TL*
Golden9	256	32	4	**273**	**273**	HI20	HI20	1650	*TL*	273	266	*TL*
Golden9	256	37	4	**273**	**273**	HI20	HI20	*TL*	*TL*	273	266	*TL*
Golden9	256	43	4	**270**	**270**	H19	HI20	*TL*	*TL*	270	262	*TL*
Golden9	256	52	4	269	268	H19	HI20	*TL*	*TL*	269	262	*TL*
Golden10	324	22	4	**346**	**346**	HI20	HI20	923	*TL*	346	345	*TL*
Golden10	324	24	4	**346**	**346**	HI20	HI20	1014	*TL*	346	345	*TL*
Golden10	324	25	4	**346**	**346**	HI20	HI20	1114	*TL*	346	345	*TL*
Golden10	324	27	4	**346**	**346**	HI20	HI20	1360	*TL*	346	345	*TL*
Golden10	324	30	4	**347**	**347**	HI20	HI20	1848	*TL*	347	345	*TL*
Golden10	324	33	4	**344**	**344**	HI20	HI20	2725	*TL*	344	341	*TL*
Golden10	324	36	4	**344**	**344**	HI20	HI20	*TL*	*TL*	344	341	*TL*

Continued on next page

Instance				Results						SoftCluVRP$^{\leq m}$		
						first found by		time T				
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden10	324	41	4	346	340	H19	grid	*TL*	*TL*	346	340	*TL*
Golden10	324	47	4	344	340	H19	grid	*TL*	*TL*	344	340	*TL*
Golden10	324	54	4	340	338	H19	grid	*TL*	*TL*	340	338	*TL*
Golden10	324	65	4	335	334	H19	grid	*TL*	*TL*	335	334	*TL*
Golden11	400	27	5	**434**	**434**	HI20	HI20	*TL*	*TL*	434	423	*TL*
Golden11	400	29	5	**434**	**434**	HI20	HI20	*TL*	*TL*	434	423	*TL*
Golden11	400	31	5	**433**	**433**	HI20	HI20	2661	*TL*	433	423	*TL*
Golden11	400	34	5	**427**	**427**	HI20	HI20	*TL*	*TL*	427	416	*TL*
Golden11	400	37	5	**427**	**427**	H19	HI20	*TL*	*TL*	427	416	*TL*
Golden11	400	40	5	**425**	**425**	H19	HI20	*TL*	*TL*	425	415	*TL*
Golden11	400	45	5	**425**	**425**	H19	HI20	*TL*	*TL*	425	415	*TL*
Golden11	400	50	5	423	422	H19	grid	*TL*	*TL*	423	415	*TL*
Golden11	400	58	5	**422**	**422**	H19	grid	*TL*	*TL*	422	415	*TL*
Golden11	400	67	5	**422**	**422**	H19	grid	*TL*	*TL*	422	415	*TL*
Golden11	400	80	5	417	416	H19	grid	*TL*	*TL*	417	410	*TL*
Golden12	484	33	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	35	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	38	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	41	5	512	507	H19	grid	*TL*	*TL*	512	500	*TL*
Golden12	484	44	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	49	5	511	507	H19	grid	*TL*	*TL*	511	500	*TL*
Golden12	484	54	5	510	507	H19	grid	*TL*	*TL*	510	500	*TL*
Golden12	484	61	5	510	507	H19	grid	*TL*	*TL*	510	500	*TL*
Golden12	484	70	5	509	506	H19	grid	*TL*	*TL*	509	499	*TL*
Golden12	484	81	5	502	498	H19	grid	*TL*	*TL*	502	493	*TL*
Golden12	484	97	5	502	498	H19	grid	*TL*	*TL*	502	493	*TL*
Golden13	253	17	4	**530**	**530**	HI20	HI20	116	2378	**519**	**519**	506
Golden13	253	19	4	**521**	**521**	HI20	HI20	189	983	**516**	**516**	26
Golden13	253	20	4	**521**	**521**	HI20	HI20	192	538	**516**	**516**	13
Golden13	253	22	4	**523**	**523**	HI20	HI20	203	*TL*	**516**	**516**	48
Golden13	253	23	4	**523**	**523**	HI20	HI20	215	3496	**516**	**516**	73
Golden13	253	26	4	**523**	**523**	HI20	HI20	118	*TL*	**516**	**516**	74
Golden13	253	29	4	**522**	**522**	HI20	HI20	1483	*TL*	**516**	**516**	1551
Golden13	253	32	4	**521**	**521**	HI20	HI20	286	*TL*	**516**	**516**	*TL*
Golden13	253	37	4	**521**	**521**	HI20	HI20	2305	*TL*	521	516	*TL*
Golden13	253	43	4	**521**	**521**	HI20	HI20	*TL*	*TL*	521	516	*TL*
Golden13	253	51	4	**521**	**521**	H19	HI20	*TL*	*TL*	521	516	*TL*
Golden14	321	22	4	**665**	**665**	HI20	HI20	1814	*TL*	665	653	*TL*
Golden14	321	23	4	**662**	**662**	HI20	HI20	752	*TL*	655	652	*TL*
Golden14	321	25	4	**660**	**660**	HI20	HI20	637	*TL*	653	652	*TL*
Golden14	321	27	4	**660**	**660**	HI20	HI20	3067	*TL*	**652**	**652**	3067
Golden14	321	30	4	**660**	**660**	HI20	HI20	*TL*	*TL*	**652**	**652**	*TL*
Golden14	321	33	4	**660**	**660**	H19	HI20	*TL*	*TL*	660	652	*TL*
Golden14	321	36	4	**658**	**658**	H19	HI20	*TL*	*TL*	658	652	*TL*
Golden14	321	41	4	**658**	**658**	HI20	HI20	*TL*	*TL*	658	652	*TL*
Golden14	321	46	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden14	321	54	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden14	321	65	4	658	657	H19	grid	*TL*	*TL*	658	652	*TL*
Golden15	397	27	4	**815**	**815**	H19	HI20	*TL*	*TL*	815	813	*TL*
Golden15	397	29	4	**815**	**815**	H19	HI20	*TL*	*TL*	815	813	*TL*
Golden15	397	31	4	**813**	**813**	HI20	HI20	3176	*TL*	**813**	**813**	*TL*
Golden15	397	34	4	**813**	**813**	H19	HI20	*TL*	*TL*	**813**	**813**	*TL*
Golden15	397	37	4	815	813	H19	grid	*TL*	*TL*	815	813	*TL*
Golden15	397	40	4	815	813	H19	grid	*TL*	*TL*	815	813	*TL*

Continued on next page

Instance				Results						SoftCluVRP$^{\leq m}$		
						first found by		time T				
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
Golden15	397	45	5	817	815	H19	relax	*TL*	*TL*	817	808	*TL*
Golden15	397	50	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	57	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	67	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden15	397	80	5	**815**	**815**	H19	relax	*TL*	*TL*	815	808	*TL*
Golden16	481	33	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	35	5	**993**	**993**	H19	HI20	*TL*	*TL*	993	980	*TL*
Golden16	481	37	5	993	992	H19	HI20	*TL*	*TL*	993	980	*TL*
Golden16	481	41	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	44	5	993	990	H19	grid	*TL*	*TL*	993	980	*TL*
Golden16	481	49	5	989	987	H19	grid	*TL*	*TL*	989	979	*TL*
Golden16	481	54	5	985	984	H19	grid	*TL*	*TL*	985	977	*TL*
Golden16	481	61	5	985	984	H19	grid	*TL*	*TL*	985	977	*TL*
Golden16	481	69	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden16	481	81	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden16	481	97	5	**984**	**984**	H19	grid	*TL*	*TL*	984	977	*TL*
Golden17	241	17	3	**386**	**386**	HI20	HI20	132	417	**386**	**386**	269
Golden17	241	18	3	**385**	**385**	HI20	HI20	290	341	**385**	**385**	290
Golden17	241	19	3	**385**	**385**	HI20	HI20	220	342	**385**	**385**	295
Golden17	241	21	3	**385**	**385**	HI20	HI20	457	625	**385**	**385**	316
Golden17	241	22	3	**385**	**385**	HI20	HI20	372	750	**385**	**385**	570
Golden17	241	25	3	**382**	**382**	HI20	HI20	487	1093	**382**	**382**	619
Golden17	241	27	3	**382**	**382**	HI20	HI20	1039	1851	**382**	**382**	1774
Golden17	241	31	4	**390**	**390**	HI20	HI20	661	1707	**382**	**382**	2582
Golden17	241	35	4	390	389	H19	HI20	*TL*	*TL*	381	379	*TL*
Golden17	241	41	4	**388**	**388**	HI20	HI20	*TL*	*TL*	382	378	*TL*
Golden17	241	49	4	387	386	H19	HI20	*TL*	*TL*	387	376	*TL*
Golden18	301	21	4	**558**	**558**	HI20	HI20	694	1571	552	547	*TL*
Golden18	301	22	4	**558**	**558**	HI20	HI20	781	1881	**553**	**553**	*TL*
Golden18	301	24	4	**558**	**558**	HI20	HI20	831	*TL*	553	548	*TL*
Golden18	301	26	4	**562**	**562**	HI20	HI20	974	*TL*	552	544	*TL*
Golden18	301	28	4	**558**	**558**	HI20	HI20	*TL*	*TL*	551	541	*TL*
Golden18	301	31	4	**554**	**554**	HI20	HI20	2450	*TL*	554	541	*TL*
Golden18	301	34	4	**554**	**554**	HI20	HI20	1992	*TL*	554	540	*TL*
Golden18	301	38	4	**555**	**555**	HI20	HI20	*TL*	*TL*	551	540	*TL*
Golden18	301	43	4	558	550	H19	⋄B&C	*TL*	*TL*	558	540	*TL*
Golden18	301	51	4	555	549	H19	†B&C	*TL*	*TL*	555	540	*TL*
Golden18	301	61	4	556	548	H19	•B&C	*TL*	*TL*	556	538	*TL*
Golden19	361	25	10	**886**	**886**	HI20	HI20	538	*TL*	738	730	*TL*
Golden19	361	26	10	**888**	**888**	HI20	HI20	1208	*TL*	763	725	*TL*
Golden19	361	28	4	**741**	**741**	HI20	HI20	1479	*TL*	741	730	*TL*
Golden19	361	31	4	**735**	**735**	HI20	HI20	*TL*	*TL*	735	728	*TL*
Golden19	361	33	4	**727**	**727**	HI20	HI20	2719	*TL*	727	723	*TL*
Golden19	361	37	5	**732**	**732**	HI20	HI20	2612	*TL*	732	716	*TL*
Golden19	361	41	5	**730**	**730**	HI20	HI20	*TL*	*TL*	730	714	*TL*
Golden19	361	46	5	730	721	H19	B&C	*TL*	*TL*	730	713	*TL*
Golden19	361	52	5	**730**	**730**	HI20	HI20	*TL*	*TL*	730	712	*TL*
Golden19	361	61	5	737	721	H19	B&C	*TL*	*TL*	737	713	*TL*
Golden19	361	73	5	736	721	H19	B&C	*TL*	*TL*	736	712	*TL*
Golden20	421	29	11	**1170**	**1170**	HI20	HI20	1099	*TL*	1052	971	*TL*
Golden20	421	31	12	**1183**	**1183**	HI20	HI20	1080	*TL*	1088	966	*TL*
Golden20	421	33	12	**1175**	**1175**	HI20	HI20	2381	*TL*	1162	966	*TL*
Golden20	421	36	5	**1005**	**1005**	H19	HI20	*TL*	*TL*	1005	963	*TL*
Golden20	421	39	5	991	971	H19	B&C	*TL*	*TL*	991	961	*TL*

Continued on next page

Instance				Results								
						first found by		time T		SoftCluVRP$^{\leq m}$		
Group	$n+1$	N	m	UB	LB	UB	LB	HI20	B&C	UB	LB	time T
`Golden20`	421	43	5	990	971	H19	†B&C	*TL*	*TL*	990	962	*TL*
`Golden20`	421	47	5	988	970	H19	⋄B&C	*TL*	*TL*	988	961	*TL*
`Golden20`	421	53	5	988	970	H19	relax	*TL*	*TL*	988	961	*TL*
`Golden20`	421	61	5	987	970	H19	relax	*TL*	*TL*	987	961	*TL*
`Golden20`	421	71	5	986	970	H19	relax	*TL*	*TL*	986	961	*TL*
`Golden20`	421	85	5	980	969	H19	relax	*TL*	*TL*	980	960	*TL*

Table 4.8: Detailed results for square `Grid` instances.

Instance				Results											
				reduced						non-reduced					
No.	n	N	m	UB	LB	time T	#nodes	#cols	#rows	UB	LB	time T	#nodes	#cols	#rows
1	121	6	2	**134.9**	**134.9**	11	55	6.240	4.428	**134.9**	**134.9**	22	89	7.281	5.211
2	121	6	2	**135.5**	**135.5**	10	48	5.383	3.836	**135.5**	**135.5**	18	17	7.281	4.938
3	121	6	2	**130.7**	**130.7**	14	273	5.192	3.967	**130.7**	**130.7**	13	82	7.281	5.666
4	121	6	2	**123.7**	**123.7**	7	104	6.144	4.730	**123.7**	**123.7**	4	102	7.281	5.571
5	121	6	2	**124.9**	**124.9**	3	17	5.846	4.571	**124.9**	**124.9**	6	544	7.281	5.735
6	121	6	2	**128.4**	**128.4**	11	28	6.958	4.949	**128.4**	**128.4**	10	253	7.281	5.191
7	121	8	3	**144.7**	**144.7**	82	1.252	6.657	5.598	**144.7**	**144.7**	69	839	7.296	6.132
8	121	8	3	**151.7**	**151.7**	256	6.573	6.357	5.037	**151.7**	**151.7**	114	904	7.296	5.769
9	121	8	3	**151.7**	**151.7**	30	29	6.069	4.881	**151.7**	**151.7**	58	309	7.296	5.779
10	121	8	3	**128.9**	**128.9**	16	660	6.663	5.587	**128.9**	**128.9**	19	423	7.296	6.122
11	121	8	3	**131.6**	**131.6**	28	29	7.080	5.967	**131.6**	**131.6**	21	60	7.296	6.161
12	121	8	4	**134.0**	**134.0**	28	21	6.263	4.677	**134.0**	**134.0**	47	27	7.296	5.416
13	121	10	3	**147.6**	**147.6**	156	2.461	7.207	6.137	**147.6**	**147.6**	247	4.072	7.315	6.231
14	121	10	3	**148.0**	**148.0**	470	4.045	6.891	5.710	148.0	147.9	413	6.264	7.315	6.052
15	121	10	4	**157.8**	**157.8**	140	2.283	6.783	5.769	**157.8**	**157.8**	240	2.931	7.315	6.243
16	121	10	4	**137.0**	**137.0**	65	414	6.993	5.968	**137.0**	**137.0**	64	393	7.315	6.265
17	121	10	4	**134.4**	**134.4**	34	629	6.991	5.643	**134.4**	**134.4**	54	197	7.315	5.859
18	121	10	3	**127.7**	**127.7**	68	5.830	6.991	6.093	**127.7**	**127.7**	26	633	7.315	6.360
19	121	12	5	181.8	181.7	859	10.619	6.696	5.747	181.8	181.7	758	6.614	7.338	6.278
20	121	12	5	**185.8**	**185.8**	2607	19.868	7.338	6.490	**185.8**	**185.8**	2607	19.868	7.338	6.490
21	121	12	4	168.7	164.6	*TL*	34.449	6.284	5.299	168.7	162.0	*TL*	25.116	7.338	6.089
22	121	12	4	**144.7**	**144.7**	217	3.668	6.588	5.675	**144.7**	**144.7**	341	7.769	7.338	6.326
23	121	12	4	142.8	142.7	529	7.727	6.912	6.015	**142.8**	**142.8**	435	4.093	7.338	6.388
24	121	12	4	**149.3**	**149.3**	554	7.700	7.338	6.265	**149.3**	**149.3**	552	8.939	7.338	6.265
25	121	14	5	**174.1**	**174.1**	2224	26.489	7.046	6.207	174.5	169.3	*TL*	33.923	7.365	6.477
26	121	14	5	**166.2**	**166.2**	1506	21.842	7.365	6.708	**166.2**	**166.2**	1507	21.842	7.365	6.708
27	121	14	4	166.2	150.4	*TL*	18.496	7.365	6.789	161.8	152.2	*TL*	25.457	7.365	6.789
28	121	14	5	**136.6**	**136.6**	96	1.249	7.365	6.590	**136.6**	**136.6**	76	802	7.365	6.590
29	121	14	4	**137.4**	**137.4**	768	14.998	6.934	6.000	**137.4**	**137.4**	292	4.003	7.365	6.378
30	121	14	5	**144.9**	**144.9**	186	4.409	6.828	5.968	**144.9**	**144.9**	483	17.003	7.365	6.398
31	169	6	2	**183.2**	**183.2**	41	116	11.406	8.813	**183.2**	**183.2**	65	262	14.217	11.262
32	169	6	3	**191.7**	**191.7**	49	35	10.575	8.497	**191.7**	**191.7**	71	63	14.217	11.586
33	169	6	2	**189.9**	**189.9**	45	374	10.337	7.782	**189.9**	**189.9**	64	110	14.217	10.614
34	169	6	2	**175.1**	**175.1**	24	53	7.885	5.638	**175.1**	**175.1**	9	9	14.217	8.928
35	169	6	3	**178.0**	**178.0**	21	31	13.006	10.166	**178.0**	**178.0**	23	19	14.217	11.165
36	169	6	2	171.7	171.6	11	73	11.402	9.128	**171.7**	**171.7**	8	21	14.217	11.387
37	169	8	3	217.8	211.3	*TL*	42.500	13.012	11.147	**217.0**	**217.0**	1220	1.848	14.232	12.215
38	169	8	2	**184.2**	**184.2**	79	312	12.701	10.475	**184.2**	**184.2**	58	289	14.232	11.833
39	169	8	3	**194.0**	**194.0**	135	2.243	10.744	8.805	**194.0**	**194.0**	71	235	14.232	11.609
40	169	8	3	**180.2**	**180.2**	31	244	12.416	10.200	**180.2**	**180.2**	73	1.547	14.232	11.756
41	169	8	3	**178.5**	**178.5**	56	670	12.580	10.461	**178.5**	**178.5**	56	800	14.232	11.906
42	169	8	3	**184.3**	**184.3**	73	645	11.871	9.374	**184.3**	**184.3**	96	1.004	14.232	11.224
43	169	10	4	**218.7**	**218.7**	2026	6.464	12.588	11.021	**218.7**	**218.7**	443	2.679	14.251	12.430
44	169	10	4	**221.3**	**221.3**	525	3.100	13.186	11.171	**221.3**	**221.3**	1069	3.781	14.251	12.108
45	169	10	3	209.6	197.8	*TL*	17.723	11.754	9.900	217.3	187.9	*TL*	17.151	14.251	12.080
46	169	10	3	**183.1**	**183.1**	215	3.825	11.040	8.812	**183.1**	**183.1**	310	5.254	14.251	11.033
47	169	10	3	190.8	184.5	*TL*	27.593	13.179	11.516	189.3	181.1	*TL*	34.370	14.251	12.487
48	169	10	3	**178.6**	**178.6**	63	985	12.005	10.323	**178.6**	**178.6**	362	5.104	14.251	12.298
49	169	12	4	**222.4**	**222.4**	3559	18.086	12.463	10.411	222.4	217.6	*TL*	14.217	14.274	11.743
50	169	12	5	**237.8**	**237.8**	923	7.220	11.471	9.600	**237.8**	**237.8**	941	8.284	14.274	11.774
51	169	12	4	**213.4**	**213.4**	1688	16.525	12.310	10.715	214.7	204.2	*TL*	14.793	14.274	12.365
52	169	12	4	**186.0**	**186.0**	420	2.353	11.763	9.977	**186.0**	**186.0**	968	3.848	14.274	12.060
53	169	12	4	**184.0**	**184.0**	94	502	13.963	12.297	**184.0**	**184.0**	133	1.479	14.274	12.589
54	169	12	5	**193.4**	**193.4**	198	1.217	11.730	10.291	**193.4**	**193.4**	202	1.789	14.274	12.416
55	169	14	5	241.8	220.4	*TL*	17.207	12.501	10.997	244.0	218.8	*TL*	10.514	14.301	12.560
56	169	14	5	239.2	220.3	*TL*	14.034	13.846	12.227	243.6	209.2	*TL*	12.467	14.301	12.602
57	169	14	5	254.1	207.1	*TL*	5.475	13.990	12.422	273.3	210.7	*TL*	4.615	14.301	12.706
58	169	14	5	**200.2**	**200.2**	335	3.739	14.143	12.149	**200.2**	**200.2**	331	2.508	14.301	12.304
59	169	14	5	**192.5**	**192.5**	309	3.158	14.301	12.997	**192.5**	**192.5**	309	3.158	14.301	12.997
60	169	14	4	**181.5**	**181.5**	228	2.394	11.347	9.844	**181.5**	**181.5**	444	3.796	14.301	12.044
61	225	6	2	**238.0**	**238.0**	3084	143.818	13.152	9.517	**238.0**	**238.0**	749	16.137	25.221	18.062
62	225	6	3	**268.2**	**268.2**	82	89	15.012	10.724	**268.2**	**268.2**	152	162	25.221	16.435
63	225	6	3	**259.9**	**259.9**	67	98	14.081	10.246	**259.9**	**259.9**	493	3.139	25.221	18.243
64	225	6	3	**238.8**	**238.8**	26	178	12.956	9.056	**238.8**	**238.8**	32	44	25.221	16.670
65	225	6	3	**237.5**	**237.5**	30	25	18.579	14.800	**237.5**	**237.5**	55	355	25.221	20.660
66	225	6	2	**229.3**	**229.3**	22	26	15.123	11.577	**229.3**	**229.3**	33	66	25.221	18.956
67	225	8	2	**245.5**	**245.5**	308	1.372	14.857	11.298	**245.5**	**245.5**	789	1.201	25.236	19.394
68	225	8	3	**255.5**	**255.5**	137	229	16.313	12.624	**255.5**	**255.5**	262	381	25.236	19.495
69	225	8	3	**249.0**	**249.0**	237	946	19.107	16.176	**249.0**	**249.0**	105	122	25.236	21.440
70	225	8	2	**229.9**	**229.9**	64	194	17.211	13.317	**229.9**	**229.9**	116	54	25.236	19.093
71	225	8	3	**237.1**	**237.1**	78	203	20.984	17.498	**237.1**	**237.1**	57	234	25.236	21.324
72	225	8	3	**232.0**	**232.0**	22	115	19.659	16.305	**232.0**	**232.0**	38	204	25.236	20.891
73	225	10	4	**279.0**	**279.0**	860	2.002	20.265	17.105	**279.0**	**279.0**	1852	3.746	25.255	21.286
74	225	10	3	252.9	242.8	*TL*	13.441	22.596	19.747	252.4	245.6	*TL*	10.978	25.255	22.084

Continued on next page

Instance				Results											
				reduced						non-reduced					
No.	$n+1$	N	m	UB	LB	time T	#nodes	#cols	#rows	UB	LB	time T	#nodes	#cols	#rows
75	225	10	3	**267.7**	**267.7**	2462	12.827	19.932	14.895	268.3	258.6	*TL*	10.959	25.255	18.826
76	225	10	4	**236.9**	**236.9**	97	519	22.168	19.294	236.9	236.8	98	508	25.255	22.111
77	225	10	4	**242.7**	**242.7**	1164	66.996	20.651	17.535	**242.7**	**242.7**	121	1.097	25.255	21.480
78	225	10	4	**242.4**	**242.4**	96	326	20.995	18.206	**242.4**	**242.4**	88	268	25.255	21.813
79	225	12	4	307.6	265.9	*TL*	5.107	21.421	17.484	308.5	259.5	*TL*	6.998	25.278	20.293
80	225	12	4	293.1	249.0	*TL*	7.661	21.616	18.943	283.9	254.7	*TL*	8.985	25.278	22.044
81	225	12	4	282.6	256.3	*TL*	6.601	21.804	19.103	285.6	257.6	*TL*	5.383	25.278	22.141
82	225	12	5	254.6	254.5	514	5.185	22.210	19.659	254.6	254.5	511	3.118	25.278	22.412
83	225	12	4	**243.7**	**243.7**	997	8.931	20.650	17.955	**243.7**	**243.7**	614	4.816	25.278	21.736
84	225	12	3	**239.3**	**239.3**	1011	9.041	24.853	22.162	**239.3**	**239.3**	1611	14.411	25.278	22.548
85	225	14	5	371.8	259.1	*TL*	6.599	23.430	20.701	360.9	267.6	*TL*	6.127	25.305	22.391
86	225	14	5	330.5	272.1	*TL*	10.855	21.054	18.595	325.0	257.7	*TL*	4.188	25.305	22.165
87	225	14	5	335.3	261.1	*TL*	4.371	22.429	20.178	335.3	264.6	*TL*	3.415	25.305	22.758
88	225	14	5	252.2	252.1	1242	13.298	21.057	18.070	252.2	252.1	1177	7.772	25.305	21.458
89	225	14	4	285.4	233.6	*TL*	10.966	22.424	20.345	272.6	233.1	*TL*	6.631	25.305	23.019
90	225	14	4	249.5	249.4	2711	18.971	22.008	19.684	249.5	249.4	3262	21.263	25.305	22.616
Total (90)						1056	8291	12832	10747			1072	5470	15611	12909

Chapter 5

Exact Algorithms for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes

Katrin Heßler

Abstract

The multi-compartment vehicle routing problem with flexible compartment sizes is a variant of the classical vehicle routing problem in which customers demand different product types and the vehicle capacity can be separated into different compartments each dedicated to a specific product type. The size of each compartment is not fixed beforehand but the number of compartments is limited. We consider two variants for dividing the vehicle capacity: On the one hand the vehicle capacity can be discretely divided into compartments and on the other hand compartment sizes can be divided continuously. The objective is to minimize the total distance of all vehicle routes such that all customer demands are met and vehicle capacities are respected. Modifying a branch-and-cut algorithm based on a three-index formulation for the discrete problem variant from the literature, we introduce an exact solution approach that is tailored to the continuous problem variant. Moreover, we propose two other exact solution approaches, namely a branch-and-cut algorithm based on a two-index formulation and a branch-price-and-cut algorithm based on a route-indexed formulation, that can tackle both problem variants with small adaptions and can be combined into an effective two-stage approach. Extensive computational tests have been conducted to compare the different algorithms. For the continuous variant, we can solve instances with up to 50 customers to optimality and for the discrete variant, several previously open instances can now be solved to proven optimality. Moreover, we analyze the cost savings of using continuously flexible compartment sizes instead of discretely flexible compartment sizes.

5.1 Introduction

Multi-compartment vehicle routing problems (MCVRP) are variants of the classical *capacitated vehicle routing problem* (CVRP, Toth and Vigo, 2014) in which several product types must be transported separately and the vehicle capacity is split or can be split into several zones. The transportation of products in separated zones is necessary for various real-world problems, e.g., the transportation of dangerous goods, liquid or bulk products, as well as the transportation of food products in different temperature zones. Instead of using one type of vehicle for each product type, it is often beneficial to collect or deliver several product types combined in one vehicle (Muyldermans and Pang, 2010). Various different multi-compartment vehicle configurations can be presumed, e.g., the size of separated zones can be fixed or flexible, the assignment of product types to compartments can be preset or arbitrary, and there can exist different (in)compatibilities between different product types or a compartment and a product type (Pollaris *et al.*, 2014).

The paper at hand considers MCVRPs with flexible compartment sizes in which different product types are incompatible with each other such that they must be transported in separate compartments. The assignment of product types to compartments is arbitrary. Two different problem variants are investigated: On the one hand, we consider the *multi-compartment vehicle routing problem with continuously flexible compartment sizes* (MCVRP-CFCS, Koch *et al.*, 2016) in which compartment sizes can be set continuously within the limits of the vehicle capacity. A practical application is, in particular, the distribution of food (Derigs *et al.*, 2010; Hübner and Ostermeier, 2019). On the other hand, we consider the *multi-compartment vehicle routing problem with discretely flexible compartment sizes* (MCVRP-DFCS, Henke *et al.*, 2015) in which compartment sizes can only be set according to pre-defined, equally spaced positions. Practical applications are amongst others the shipment of bulk products (Fagerholt and Christiansen, 2000) and the collection of glass waste (Henke *et al.*, 2015). In the following, we refer collectively to the MCVRP-CFCS and MCVRP-DFCS as *multi-compartment vehicle routing problems with flexible compartment sizes* (MCVRP-FCS).

The contributions of the paper at hand are the following. We introduce a three-index formulation tailored to solve the MCVRP-CFCS exactly. Moreover, we introduce a two-index formulation and a route-based formulation suited to being solved by column generation for both the MCVRP-CFCS and MCVRP-DFCS. Based on these formulations, we propose a branch-and-cut and branch-price-and-cut algorithm, respectively, that can solve the two problem variants with small adaptions and are combined to an effective two-stage approach. To compare the

algorithms, extensive numerical experiments have been conducted on instances from the literature. For the MCVRP-CFCS, the experiments demonstrate good performance for instances with up to 50 customers. For the MCVRP-DFCS, several new instances can be solved to proven optimality for the first time compared to results from the literature. Moreover, further experiments are conducted to analyze the cost savings of using continuously flexible compartment sizes instead of discretely flexible compartment sizes and to compare the performance of the algorithms according to the number of vehicles.

The remainder of the paper is organized as follows. In the next section, we formally define the MCVRP-CFCS and MCVRP-DFCS. In Section 5.3, we give an overview of the existing literature on MCVRPs. Subsequently, three exact solution approaches are presented. At first, a branch-and-cut algorithm based on a three-index formulation and separation procedures are introduced in Section 5.4. We do the same in Section 5.5 with a branch-and-cut algorithm based on a two-index formulation. Afterwards, a branch-price-and-cut algorithm including details on the generation of route variables, stabilization techniques, valid inequalities, and branching is presented in Section 5.6. In Section 5.7, we conduct numerical experiments to compare the different algorithms and compare total costs of the MCVRP-CFCS and MCVRP-DFCS. Conclusions are drawn in Section 5.8.

5.2 Problem Definition

We formally define the MCVRP-CFCS and MCVRP-DFCS as follows. Let $N = \{1, \dots, n\}$ be the set of *customers* and $P = \{1, \dots, \rho\}$ the set of *product types*. The *demand* of customer $i \in N$ for product type $p \in P$ is denoted by d_{ip}. The set of product types $P_i = \{p \in P : d_{ip} > 0\}$ delivered to customer $i \in N$ may contain any and all product types P, i.e., $P_i \subseteq P$ for all $i \in N$.

A maximum of m homogeneous *vehicles* $F = \{1, \dots, m\}$ is available for delivery. Each vehicle has a *capacity* of Q that can be divided into a limited number of C *compartments*. Note that the number of product types demanded by customer i can exceed the number of compartments, i.e., $|P_i| > C$ is possible such that at least two vehicles are needed to serve customer i. For the MCVRP-CFCS, the compartment sizes can be set arbitrarily. For the MCVRP-DFCS, the vehicle capacity can be separated into compartments such that each compartment size is a multiple of *unit size* q^{unit}.

Let $G = (V, E)$ be a complete undirected graph with node set $V = N \cup \{0\}$ and edge set E with $i < j$ for all $\{i, j\} \in E$. Node 0 represents the *depot* and routing *costs* between two nodes $\{i, j\} \in E$ are given by c_{ij}. A *route* $r = (i_0, \dots, i_s, i_{s+1})$ delivering products $S_{i_k} \subseteq P_{i_k}$, $k \in \{1, \dots, s\}$, is feasible if

(i) it is a cycle passing through the depot, i.e., $i_0 = i_{s+1} = 0$;

(ii) all customers $i_1, \ldots, i_s$ are different;
(iii) the number of compartments is respected, i.e., $\left|\bigcup_{k=1}^{s} S_{i_k}\right| \leq C$; and
(iv) capacity constraints hold, i.e., for continuously flexible compartment sizes

$$\sum_{k=1}^{s} \sum_{p \in S_{i_k}} d_{i_k p} \leq Q, \tag{5.1a}$$

or for discretely flexible compartment sizes

$$\sum_{p \in P} \left\lceil \sum_{\substack{k \in \{1,\ldots,s\}, \\ \{p\} \cap S_{i_k} \neq \emptyset}} \frac{d_{i_k p}}{q^{\text{unit}}} \right\rceil \leq \left\lfloor \frac{Q}{q^{\text{unit}}} \right\rfloor. \tag{5.1b}$$

Regardless of compartment division, the task is to determine a cost-minimal set of at most m feasible routes such that all customer demands are met.

The MCVRP-CFCS and MCVRP-DFCS are both a generalization of the CVRP (Toth and Vigo, 2014) and, therefore, NP-hard. Moreover, the MCVRP-CFCS is a restriction of the *commodity-constrained split delivery vehicle routing problem* (C-SDVRP, Archetti *et al.*, 2016; Gschwind *et al.*, 2019) in which customer demands are composed of different commodities but no product types exist such that all commodities can be transported together in one zone, i.e., a feasible route of the C-SDVRP is defined by conditions (i), (ii), and (5.1a). Therefore, if the limit on the number of compartments in the MCVRP-CFCS is greater or equal to the number of product types ($C \geq \rho$), then all different product types can be transported together on one vehicle, i.e., condition (iii) holds true for all routes, and both the MCVRP-CFCS and the C-SDVRP are equivalent.

The formulations that we introduce in the following rely on different graphs. For the sake of clarity, we already define most of these graphs in this section. A summary of all graphs is depicted in Table 5.1. Graph $\bar{G} = (\bar{V}, \bar{E})$ is derived from graph $G = (V, E)$ by duplicating each customer node $i \in N$ for all product types $p \in P_i$ yielding a new customer set $\bar{N}$. The new graph $\bar{G}$ consists of $|\bar{V}| = 1 + \sum_{i \in N} |P_i|$ nodes. For each node $k \in \bar{N}$, let $f_c(k) \in N$ denote the corresponding customer, $f_p(k) \in P$ the corresponding product type, and $f_d(k) \in P$ the corresponding demand, respectively. Moreover, let $\bar{E}$ be the corresponding edge set such that $\bar{G}(\bar{V}, \bar{E})$ results in a complete undirected graph. Both graphs G and $\bar{G}$ can also be converted into directed graphs G^d and $\bar{G}^d$, respectively, by duplicating each edge between customers into two anti-parallel arcs and adding a second depot node $n + 1$. The start depot 0 is connected to all customer nodes by outgoing arcs and the end depot $n + 1$ is connected to all customer nodes by incoming arcs. Let A and $\bar{A}$ denote the arc sets, respectively. Moreover, for any

subset of customers S, $S \neq \emptyset$, of an undirected graph, let $\delta(S)$ denote the set of edges with exactly one endpoint in S and let $\delta(i) := \delta(\{i\})$ denote the set of all edges incident with node i.

Table 5.1: Overview of graphs.

graph	links	node set	customer set	edge/arc set	depot nodes	number of nodes of customer i
G	undirected	V	N	E	0	1
G^d	directed	$V \cup \{n+1\}$	N	A	$0, n+1$	1
$\bar{G}$	undirected	$\bar{V}$	$\bar{N}$	$\bar{E}$	0	$\lvert P_i \rvert$
$\bar{G}^d$	directed	$\bar{V} \cup \{n+1\}$	$\bar{N}$	$\bar{A}$	$0, n+1$	$\lvert P_i \rvert$

5.3 Literature review

In the literature, several variants of the MCVRP and heuristic and exact solution approaches to solve them have been discussed. Pollaris *et al.* (2014) present an overview of vehicle routing problems with loading constraints including a summary of MCVRP literature. Henke (2017) provides a recent review and an extended classification scheme for the MCVRP. In the following, we first give a short overview of publications on MCVRPs with fixed compartment sizes and focus afterwards on literature about MCVRPs with flexible compartment sizes.

Fixed compartments. Numerous MCVRP publications with fixed compartment sizes deal with the distribution of liquid products. In particular, different petrol replenishment problems are studied in which tanker trucks deliver petroleum products to underground tanks located in gas stations. The specialty of petrol distribution is that typically the content of each compartment can only be delivered to one customer because vehicles are not equipped with debit meters. Brown and Graves (1981) present an automated real-time dispatch system. Avella *et al.* (2004) provide a branch-and-price algorithm and Cornillier *et al.* (2008) formulate a set-partitioning problem that can solve instances with a small set of petrol stations optimally. Recent technology allows us to equip vehicles with debit meters so that the content of a compartment can be split between several deliveries and customers may allow different vehicles to fill the same underground tank. Using this fact, Coelho and Laporte (2015) present a classification scheme that distinguishes between split and unsplit compartments and underground tanks. They propose specialized models for particular versions of the problem and a branch-and-cut algorithm applicable to all variants. A variant of the MCVRP that includes

time windows is solved by Benantar *et al.* (2016) with a tabu search algorithm. Other MCVRP variants with liquid products are the collection of olive oil (Lahyani *et al.*, 2015), solved by a branch-and-cut algorithm, and the collection of raw milk (Caramia and Guerriero, 2010), solved by the combination of two mathematical formulations and a local search algorithm.

Literature on routing other goods than liquid products is also extensive. Muyldermans and Pang (2010) introduce a local search algorithm for a waste collection problem and compare separate collection for each waste type with co-collection of different waste types. An ant colony algorithm is proposed by Reed *et al.* (2014) that solves a waste collection problem in which the location of the deposit site is separated from the vehicle depot. Fallahi *et al.* (2008) suggest a memetic algorithm and a tabu search for an animal food distribution problem with sanitary rules that recommend to always use the same compartment for one species. Similar sanitary rules are defined in the livestock collection problem in which animals from farms are collected for slaughter at a slaughterhouse. Oppen and Løkketangen (2008) present a tabu search approach and Oppen *et al.* (2010) introduce an exact column generation-based solution approach. A grocery distribution problem is presented by Ostermeier and Hübner (2018) that includes the decision of using cost-different single-compartment or multi-compartment vehicles. The problem is solved by a large neighborhood search algorithm. An MCVRP with time windows and three time planning periods arising in a city logistics problem is proposed and solved by an adaptive large neighborhood search algorithm by Eshtehadi *et al.* (2020). Mirzaei and Wøhlk (2017) compare two MCVRP variants that allow either only single or multiple visits to the same customer. Both variants are solved exactly by a branch-and-price algorithm. A variable neighborhood search algorithm for the selective MCVRP with time windows is proposed by Melechovskỳ (2013). In this variant profits are dedicated to customers and product types and the aim is to maximize the total profit. Other MCVRP variants consider stochastic instead of deterministic demands (Mendoza *et al.*, 2010, 2011; Goodson, 2015).

Flexible compartments. Little attention has been paid to MCVRP with flexible compartment sizes. Fagerholt and Christiansen (2000) introduce a bulk ship scheduling problem with a flexible cargo hold that can be partitioned discretely into several smaller holds. The problem is solved by a set partitioning approach consisting of two phases for the scheduling and allocation problem. Chajakis and Guignard (2003) propose a model for the distribution to convenience stores and develop approximation schemes based on Lagrangean relaxation. The packing is constrained by two independent dimensions (weight and volume), and apart from transportation also cooling costs of each compartment for non-ambient temperature items are considered. An MCVRP with loading and unloading costs that oc-

curs in grocery distribution is introduced by Hübner and Ostermeier (2019). In this variant, using multi-compartment vehicles saves transportation costs but increases (un)loading costs because more than one shipping gate has to be approached at the warehouse. They present a large-neighborhood search with specialized removal and reinsert operators. Ostermeier and Hübner (2018) include loading constraints to the problem, develop a branch-and-cut algorithm, and extend the large neighborhood search of Hübner and Ostermeier. Derigs *et al.* (2010) consider the MCVRP with fixed and flexible compartment sizes and introduce a solver suite consisting of construction heuristics, improvement heuristics, and metaheuristics. In both variants, products are not dedicated to compartments but incompatibility relations between products and compartments as well as pairs of products are considered. Compartment sizes can be set arbitrarily in the variant with flexible compartment sizes. They do not consider the discrete version. Pirkwieser *et al.* (2012) extend this problem by using a measure to distinguish packings and aim at solutions with a denser packing. They present a variable neighborhood search algorithm with a new neighborhood structure.

Henke *et al.* (2015) introduce the MCVRP-DFCS that occurs in the context of glass waste collection. A mathematical model-based exact solution approach is proposed that can solve problem instances with up to 10 locations to proven optimality. Moreover, they provide a variable neighborhood search that finds good quality solutions. Later on, Henke *et al.* (2018) suggest a branch-and-cut algorithm for the MCVRP-DFCS. Their algorithm can solve instances with up to 50 locations to proven optimality within two hours. The model formulation is also used for the MCVRP-CFCS variant by setting the unit compartment size to one. We later compare against their results. Koch *et al.* (2016) introduce the MCVRP-CFCS and present a heuristic approach that is based on different genetic algorithms for the CVRP from the literature. The algorithm can find an optimal solution for the majority of instances with up to 50 locations within one second (Henke, 2018). The cost saving of using continuously flexible compartments instead of discrete ones is also investigated.

5.4 Branch-and-cut algorithm for a three-index formulation

Henke *et al.* (2018) develop a branch-and-cut algorithm for the MCVRP-DFCS based on a three-index formulation. Their model handles discrete compartment size by variables y^I_{pf}, with $p \in P$ and $f \in F$, that indicate the size of the compartment of vehicle f for product type p in the number of basic unit sizes q^{unit}. To compare the total cost of both (continuous and discrete) problem variants, they

suggest setting $q^{\text{unit}} = 1$ for the continuous variant. Instead, we propose a model for the MCVRP-CFCS that does not use the basic unit compartment size. Note that in this section we only present the solution approach for the MCVRP-CFCS. For the MCVRP-DFCS, we refer to (Henke *et al.*, 2018).

The new model relies on four types of variables. First of all, the symmetric formulation has non-negative integer routing variables x_{ijf} for all edges $\{i,j\} \in E$ and vehicles $f \in F$. Binary delivery variables u_{ipf} indicate whether the demand of product type $p \in P$ at customer $i \in N$ is served by vehicle $f \in F$. The coupling between routing and delivery variables is ensured by variables z_{if} that specify if node $i \in V$ is visited by vehicle $f \in F$. Additionally, to handle the maximal allowed number of compartments per vehicle, we introduce binary variables y_{pf} indicating whether the vehicle $f \in F$ delivers product type $p \in P$. The new formulation is:

$$\min \quad \sum_{\{i,j\}\in E} \sum_{f\in F} c_{ij} x_{ijf} \tag{5.2a}$$

$$\text{subject to} \quad \sum_{f\in F} u_{ipf} = 1 \qquad \forall i \in N, p \in P, d_{ip} > 0 \tag{5.2b}$$

$$u_{ipf} \le z_{if} \qquad \forall i \in N, p \in P, f \in F \tag{5.2c}$$

$$z_{if} \le z_{0f} \qquad \forall i \in N, f \in F \tag{5.2d}$$

$$\sum_{j\in N} \sum_{f\in F} x_{0jf} \le 2m \tag{5.2e}$$

$$\sum_{\{i,j\}\in\delta(i)} x_{ijf} = 2z_{if} \qquad \forall i \in V, f \in F \tag{5.2f}$$

$$\sum_{i\in N} u_{ipf} \le n y_{pf} \qquad \forall p \in P, f \in F \tag{5.2g}$$

$$\sum_{p\in P} y_{pf} \le C \qquad \forall f \in F \tag{5.2h}$$

$$\sum_{i\in N} \sum_{p\in P} d_{ip} u_{ipf} \le Q \qquad \forall f \in F \tag{5.2i}$$

$$\sum_{\{i,j\}\in\delta(S)} x_{ijf} \ge 2\sigma(S) \qquad \forall f \in F, S \subseteq N, S \neq \emptyset \tag{5.2j}$$

$$x_{ijf} \in \{0,1\} \qquad \forall \{i,j\} \in E \setminus \delta(0), f \in F \tag{5.2k}$$

$$x_{0jf} \in \{0,1,2\} \qquad \forall j \in N, f \in F \tag{5.2l}$$

$$u_{ipf} \in \{0,1\} \qquad \forall i \in N, p \in P, f \in F \tag{5.2m}$$

$$z_{if} \in \{0,1\} \qquad \forall i \in V, f \in F \tag{5.2n}$$

$$y_{pf} \in \{0,1\} \qquad \forall p \in P, f \in F \tag{5.2o}$$

The objective function (5.2a) minimizes routing costs. Equalities (5.2b) ensure that each demand is delivered by exactly one vehicle. The coupling between u- and z-variables is established by constraints (5.2c). Constraints (5.2d) ensure that a vehicle only visits customers if the depot is included in the route and (5.2e) restricts the number of vehicles. The degree constraints are established by (5.2f). The coupling between u- and y-variables is guaranteed by constraints (5.2g). Constraints (5.2h) and (5.2i) limit the number of compartments and the capacity per vehicle, respectively. Constraints (5.2j), known as capacity cuts, ensure both solution connectivity and packing feasibility according to (iii) and (5.1). In these constraints, $\sigma(S)$ denotes the minimum number of vehicles needed to serve S. Finally, variable domains are defined by (5.2k)-(5.2o).

Already for the classical CVRP, it is difficult to calculate $\sigma(S)$ because an (NP-hard) one-dimensional bin packing problem with items $k \in S$, weights $f_d(k)$, and bin capacity Q must be solved. Therefore, it is usual to replace $\sigma(S)$ by a lower bound. For the MCVRP-CFCS, one such bound that calculates the minimum of vehicles needed to serve S according to the number of compartments and the vehicle capacity is

$$\max\left\{\left\lceil \frac{|f_p(S)|}{C} \right\rceil, \left\lceil \frac{f_d(S)}{Q} \right\rceil\right\}, \tag{5.3a}$$

where $f_p(S)$ is the set of product types and $f_d(S)$ the sum of the demands of all nodes in S. For the MCVRP-DFCS, we can bound $\sigma(S)$ from below by

$$\max\left\{\left\lceil \frac{|\{f_p(k) : k \in S\}|}{C} \right\rceil, \left\lceil \frac{q^{\text{unit}}}{Q} \sum_{p \in P} \left\lceil \sum_{k \in S, p = f_p(k)} \frac{f_d(k)}{q^{\text{unit}}} \right\rceil \right\rceil\right\}. \tag{5.3b}$$

Here, the second argument additionally takes into account discrete compartment sizes.

5.4.1 Valid inequalities

Additional symmetry breaking constraints are added to formulation (5.2) to avoid equivalent feasible solutions that can occur if the same route is performed by different vehicles. Preliminary experiments showed that ordering routes in decreasing order of their total cost is most beneficial for the MCVRP-DFCS (Henke *et al.*, 2018). Therefore, we also add the following symmetry breaking constraints to formulation (5.2) for the MCVRP-CFCS:

$$\sum_{\{i,j\} \in E} c_{ij} x_{ij,f+1} \leq \sum_{\{i,j\} \in E} c_{ij} x_{ijf} \qquad \forall f \in F \setminus \{m\} \tag{5.4}$$

5.4.2 Separation procedure

The number of capacity cuts is exponential in $|V|$. In this section we describe how these constraints can be added dynamically utilizing a separation procedure.

For both integer and fractional solutions, we apply two different procedures, namely subtour-elimination constraints and exact capacity cuts.

Subtour-elimination constraints. For each vehicle $f \in F$, we find subtours as follows. Let $\bar{x}_{ijf}$ be the values of the variables x_{ijf} of a solution to the linear relaxation of (5.2) and $G^s = (V, E^s)$ be the support graph. To determine subtours, we first construct a support graph $G^s = (V, E^s)$ with edge set $E^s = \{\{i, j\} \in E : \bar{x}_{ijf} > 0\}$. Next, we call Algorithm 11 with the customer-induced subgraph $G^s[N]$ of G^s. Irrespective of whether or not subset S is connected to the depot, all found violated cuts are added to the model. Note that contrary to (Henke *et al.*, 2018), we allow fractional solutions for this procedure.

Capacity cuts. As proposed by Henke *et al.* (2018), capacity cuts are additionally separated. Again, we first construct a combined support graph $G^s = (V, E^s)$ for all vehicles with edge set $E^s = \{\{i, j\} \in E : \bar{x}_{ij} = \sum_{f \in F} \bar{x}_{ijf} > 0, i \neq 0\}$. Next, we call Algorithm 11 with the customer-induced subgraph $G^s[N]$ of G^s. Connected components S are determined and all violated cuts are added.

Algorithm 11: Violated cut generator

input : support graph $G = (V, E)$ with edge set $E = \{\{i, j\} \in E : \bar{x}_{ij} > 0\}$
output: sets $S \in \mathcal{S}$ violating (5.2j)

```
1 Determine connected components S of G via the efficient union-find
    algorithm of Tarjan (1979);
2 foreach connected component S do
3     Set S ← S \ {0};
4     Calculate the flow f_0S between the depot 0 and S;
5     Calculate σ(S) according to (5.3a) or (5.3b), respectively;
6     if f_0S < 2σ(S) then
7         S = S ∪ {S};
```

5.5 Branch-and-cut algorithm for a two-index formulation

The two-index formulation relies on the undirected graph $\bar{G}(\bar{V}, \bar{E})$ defined in Section 5.2. Travel costs between the same customer are set to 0, i.e., $c_{kl} = 0$ for all $\{k, l\} \in \bar{E}$ with $f_c(k) = f_c(l)$. Our formulation is based on the classical symmetric formulation of Laporte *et al.* (1985) that is already successfully applied to other vehicle routing problems (VRP) with difficult packing restrictions, e.g. the VRP with two-dimensional loading constraints (Iori *et al.*, 2007). We use binary routing variables x_{kl} indicating whether a vehicle traverses edge $\{k, l\} \in \bar{E}$. The two-index formulation is:

$$\min \quad \sum_{\{k,l\} \in \bar{E}} c_{kl} x_{kl} \tag{5.5a}$$

$$\text{subject to} \quad \sum_{\{k,l\} \in \delta(k)} x_{kl} = 2 \qquad \forall k \in \bar{N} \tag{5.5b}$$

$$\sum_{\{0,l\} \in \delta(0)} x_{0l} \leq 2m \tag{5.5c}$$

$$\sum_{\{k,l\} \in \delta(S)} x_{kl} \geq 2\sigma(S) \qquad \forall S \subseteq \bar{N}, S \neq \emptyset \tag{5.5d}$$

$$x_{kl} \in \{0, 1\} \qquad \forall \{k, l\} \in \bar{E} \setminus \delta(0) \tag{5.5e}$$

$$x_{0l} \in \{0, 1, 2\} \qquad \forall \{0, l\} \in \delta(0). \tag{5.5f}$$

The objective (5.5a) minimizes travel costs. Constraints (5.5b) impose that each node is visited exactly once and constraint (5.5c) restricts the number of vehicles leaving from and returning to the depot. Constraints (5.5d), known as capacity cuts, ensure both solution connectivity and packing feasibility according to (iii) and (5.1). Again, $\sigma(S)$ denotes the minimum number of vehicles needed to serve S. We bound $\sigma(S)$ from below by (5.3a) and (5.3b) for the MCVRP-CFCS and MCVRP-DFCS, respectively. The domains of routing variables are given by (5.5e)–(5.5f).

Contrary to formulation (5.2), the two-index formulation (5.5) does not use variables having a vehicle-index. Therefore, considering instances with a large number of vehicles (and a small number of product types), the formulation does not suffer from the inherent symmetry when it comes to branching (see Section 5.7.7). Moreover, it is rather simple to implement using the callable library of modern MIP solvers, because only capacity constraints have to be added dynamically, i.e., in a cutting-plane fashion.

However, formulation (5.5) has two disadvantages. On the one hand symmetry can occur between two solutions when considering instances with two or more product types per customer. For example, consider a route supplying a customer with three product types k, l, and s. Then the solution $x_{kl} = x_{ls} = 1$ is equivalent to $x_{ks} = x_{ls} = 1$.

On the other hand it cannot be solved by directly using a MIP solver because it contains the large-size family of constraints (5.5d). In the following, we introduce symmetry breaking constraints as well as other valid inequalities and describe how constraints (5.5d) can be added dynamically using separation procedures.

5.5.1 Valid inequalities

The linear relaxation of formulation (5.5) can be further strengthened by employing valid inequalities. We introduce one class of symmetry breaking constraints and one class of logical inequalities.

Let a total ordering $<$ of all nodes be given, i.e., either $k < l$ or $l < k$ holds for all $l, k \in \bar{V}, l \neq k$. Consider a customer with (at least) three product types k, l, and s with $k < l < s$. Figure 5.1 shows five partial routes supplying the customer with two or three product types. Three equivalent integer solutions exist: $x_{kl} = x_{ls} = 1$ (see Figure 5.1c), $x_{kl} = x_{ks} = 1$ (see Figure 5.1d), and $x_{ks} = x_{ls} = 1$ (see Figure 5.1e). To reduce symmetry, we only allow the first solution in which the indices kl and ls are sorted in a non-decreasing manner ($k < l < s$) whereas this does not hold for the other solutions. To ensure that products belonging to the same customer are collected in such a non-decreasing manner, we introduce the class of symmetry breaking constraints

$$x_{ks} + x_{ls} \leq 1 \qquad \forall k, l, s \in \bar{V}, k < l < s, f_c(k) = f_c(l) = f_c(s), \tag{5.6a}$$

$$x_{kl} + x_{ks} \leq 1 \qquad \forall k, l, s \in \bar{V}, k < l < s, f_c(k) = f_c(l) = f_c(s). \tag{5.6b}$$

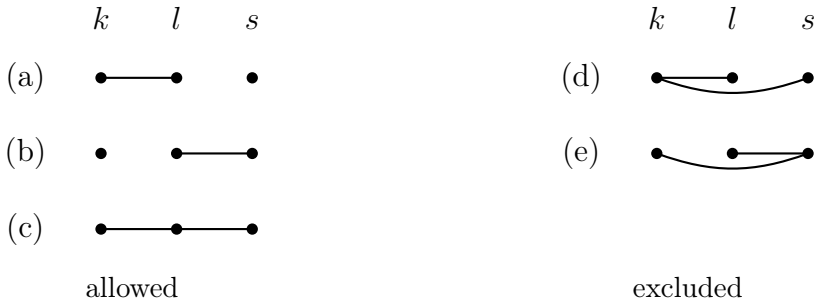


Figure 5.1: Example to illustrate inequalities (5.6a) and (5.6b). All nodes k, l, s belong to one customer and only edges between nodes of this customer are considered.

If the number of compartments is $C = 2$ then the flow from a node of a customer to other nodes of the same customer is at most 1. This is essential for customers with many product types. Therefore, we can employ the second class of valid inequalities

$$\sum_{\substack{\{k,l\}\in\delta(k),\\ f_c(k)=f_c(l)}} x_{kl} \leq 1 \qquad \forall k \in \bar{V}. \tag{5.6c}$$

5.5.2 Separation procedure

Again, formulation (5.5) contains a large-sized family of constraints because the number of capacity cuts is exponential in $|\bar{V}|$. Similar to the separation procedure described in Section 5.4.2, capacity cuts are added dynamically to the model as follows.

Subtour-elimination constraints. Let $\bar{x}_{kl}$ be an integer or fractional solution to the linear relaxation of (5.5) and $\bar{G}^s = (\bar{V}, \bar{E}^s)$ be the support graph with edge set $\bar{E}^s = \{\{k,l\} \in \bar{E} : \bar{x}_{kl} > 0\}$. Subtours are determined by utilizing Algorithm 11. Analogous to Section 5.4.2, irrespective of whether or not subset S is connected to the depot, all found violated cuts are added to the model.

Capacity cuts. First, we apply an exact MIP-based separation algorithm that simultaneously determines the subset $S \subset \bar{V}$, computes the lower bound on $\sigma(S)$, and maximizes the violation (if any) $2\sigma(S) - \bar{x}(\delta(S))$. Our MIP is an adaption of the formulations of (Ahr, 2004; Martinelli *et al.*, 2013) that exactly separate capacity cuts for the capacitated arc-routing problem. The MIP has five types of variables: Binary variables s_k, $k \in \bar{V}$, indicate whether node k belongs to set S. Variables z_{kl}, $\{k,l\} \in \bar{E}$, determine the cut set, i.e., $z_{kl} = 1$ iff $\{k,l\} \in \delta(S)$. Moreover, binary variables y_p, $p \in P$, indicate whether product p belongs to set S, i.e., $y_p = 1$ iff $\sum_{k\in S} f_p(k) \geq 1$. Integer variables r_1 and r_2 describe (the lower bound on) the number of vehicles needed to serve S according to the demand and the number of product types, respectively. The lower bound on $\sigma(S)$ is calculated with variable $r = \max\{r_1, r_2\}$ by means of binary auxiliary variable w. The non-negative continuous variables $f_1, f_2 < 1$ describe the fractional differences $\lceil d(S)/Q \rceil - d(S)/Q$ and $\lceil |\{f_p(k) : k \in S\}| / C \rceil - |\{f_p(k) : k \in S\}| / C$, respectively. The formulation is:

$$\max \quad 2r - \sum_{\{k,l\}\in\bar{E}} \bar{x}_{kl} z_{kl} \tag{5.7a}$$

$$\text{subject to} \quad s_0 = 0 \tag{5.7b}$$

$$z_{kl} - s_k + s_l \geq 0 \qquad \forall \{k,l\} \in \bar{E}, k \neq 0 \qquad (5.7c)$$

$$z_{kl} - s_l + s_k \geq 0 \qquad \forall \{k,l\} \in \bar{E}, l \neq 0 \qquad (5.7d)$$

$$s_k + s_l - z_{kl} \geq 0 \qquad \forall \{k,l\} \in \bar{E} \qquad (5.7e)$$

$$s_k + s_l + z_{kl} \leq 2 \qquad \forall \{k,l\} \in \bar{E} \setminus \delta(0) \qquad (5.7f)$$

$$r_1 = \sum_{k \in \bar{N}} (d_k/Q) s_k + f_1 \qquad (5.7g)$$

$$r_2 = \sum_{p \in P} (1/C) y_p + f_2 \qquad (5.7h)$$

$$\sum_{k \in \bar{N}, f_p(k)=p} s_k \leq n y_p \qquad \forall p \in P \qquad (5.7i)$$

$$\sum_{k \in \bar{N}, f_p(k)=p} s_k \geq y_p \qquad \forall p \in P \qquad (5.7j)$$

$$r \leq r_1 + mw \qquad (5.7k)$$

$$r \leq r_2 + m(1-w) \qquad (5.7l)$$

$$s_k \in \{0,1\} \qquad \forall k \in \bar{N} \qquad (5.7m)$$

$$0 \leq z_{kl} \leq 1 \qquad \forall \{k,l\} \in \bar{E} \qquad (5.7n)$$

$$y_p \in \{0,1\} \qquad \forall p \in P \qquad (5.7o)$$

$$r_1, r_2, r \geq 0, \text{integer} \qquad (5.7p)$$

$$w \in \{0,1\} \qquad (5.7q)$$

$$0 \leq f_1 \leq 1 - 1/Q \qquad (5.7r)$$

$$0 \leq f_2 \leq 1 - 1/C \qquad (5.7s)$$

The objective (5.7a) maximizes the violation of a capacity cut defined by subset $S = \{k \in \bar{V} : s_k = 1\}$. Constraint (5.7b) ensures that $0 \notin S$. Constraints (5.7c) and (5.7d) force $z_{kl} = 1$ if $s_k \neq s_l$, while (5.7e) and (5.7f) force $z_{kl} = 0$ if $s_k = s_l$. Constraints (5.7g) and (5.7h) guarantee correct values of r_1 and r_2. The coupling of the y-variables with the s-variables is established via constraints (5.7i) and (5.7j) to enforce $y_p = 1$ iff $\sum_{k \in S} f_p(k) \geq 1$. Moreover, constraints (5.7k) and (5.7l) are a linearization of $r = \max\{r_1, r_2\}$ and (5.7m)–(5.7s) describe the domains of the variables.

Due to considerable computation time, the MIP-based exact separation procedure is only called for fractional solutions in the root node with a time limit of two seconds and at most 500 times.

The second procedure relies on the support graph $\bar{G}^s$ with edge set $\bar{E}^s$. The algorithm tries to find a subset S of small size that is connected and consists of many different product types. The pseudocode is depicted in Algorithm 12. Starting with a randomly chosen node $k \in \bar{N}$, the set $S = \{k\}$ is enlarged by

adding nodes connected to S with preferably new product types. Set S is further enlarged until either no connected node exists or a violated cut $f_{0S} < 2\sigma(S)$ is found. The algorithm is restarted with a new non-considered randomly chosen node $k \in \bar{N} \setminus U$ (set U contains already considered nodes) until all nodes are processed.

Algorithm 12: Separation of capacity cuts

input : graph $\bar{G}^s = (\bar{V}, \bar{E}^s)$
output: sets to check S

Set $S = U = \varnothing$;
while $U \neq \bar{V} \setminus \{0\}$ **do**
  if $S = \varnothing$ **then**
    Choose randomly a node $k \in \bar{N} \setminus U$ and set $S \leftarrow S \cup \{k\}$ and $U \leftarrow U \cup \{k\}$;
  else if *Nodes connected to S exist* **then**
    Choose randomly a node $k \in \bar{N} \setminus U$ connected to S (if possible with $f_p(k) \notin f_p(S)$) and set $S \leftarrow S \cup \{k\}$ and $U \leftarrow U \cup \{k\}$;
  else
    $S = \varnothing$;
  Compute f_{0S};
  Check S regarding $f_{0S} < 2\sigma(S)$;

5.6 Branch-price-and-cut algorithm

To solve both the MCVRP-CFCS and MCVRP-DFCS with a column-based solution approach, we propose a set-partitioning formulation. Since the MCVRP-CFCS is a restriction of the C-SDVRP, we can adapt the model of Archetti *et al.* (2015). The new formulation is based on the directed graph $G^d(V \cup \{n+1\}, A)$ (cf. Section 5.2). Each vehicle route starts and ends at the depot nodes 0 and $n+1$, respectively. A feasible route is an elementary 0-$(n+1)$-path that respects the number of compartments and capacity constraints (cf. (i)-(5.1) in Section 5.2). Let Ω be the set of feasible routes and $c^r = \sum_{(i,j)\in A(r)} c_{ij}$ the cost of route $r \in \Omega$, where $A(r) \subset A$ is the set of arcs traversed by route r. The formulation uses binary route variables λ^r, $r \in \Omega$, that indicate whether a route is performed. The non-negative integer variables ψ and z_i model the number of used vehicles and the number of times customer $i \in N$ is visited, respectively. The flow over an arc $(i,j) \in A$ is modeled by non-negative integer variables x_{ij}. Moreover, let

$X = \{P' \subseteq P : |P'| = C\}$ be the set of all feasible maximal packing combinations of different product types. For example, an instance with three product types and a maximum of $C = 2$ compartments results in three feasible maximal packing combinations $X = \{\{1,2\},\{2,3\},\{1,3\}\}$. Moreover, let χ_L be the number of routes packed with compartment combination $L \in X$. The formulation is as follows:

$$\min \quad \sum_{r\in\Omega} c^r \lambda^r \tag{5.8a}$$

$$\text{subject to} \quad \sum_{r\in\Omega} a^r_{ip}\lambda^r = 1 \qquad \forall i \in N, p \in P_i \tag{5.8b}$$

$$\sum_{r\in\Omega} \lambda^r - \psi = 0 \tag{5.8c}$$

$$\left\lceil \frac{\sum_{i\in N}\sum_{p\in P_i} d_{ip}}{Q} \right\rceil \le \psi \le m \text{ and integer} \tag{5.8d}$$

$$\lambda^r \in \{0,1\} \qquad \forall r \in \Omega \tag{5.8e}$$

$$\sum_{r\in\Omega} g^r_L \lambda^r - \chi_L = 0 \qquad \forall L \in X \tag{5.8f}$$

$$0 \le \chi_L \le m \text{ and integer} \qquad \forall L \in X \tag{5.8g}$$

$$\sum_{r\in\Omega} e^r_i \lambda^r - z_i = 0 \qquad \forall i \in N \tag{5.8h}$$

$$1 \le z_i \le \min\{|P_i|, m\} \text{ and integer} \qquad \forall i \in N \tag{5.8i}$$

$$\sum_{r\in\Omega} b^r_{ij} \lambda^r - x_{ij} = 0 \qquad \forall (i,j) \in A \tag{5.8j}$$

$$0 \le x_{ij} \le \min\{|P_i|, |P_j|, m\} \text{ and integer} \qquad \forall (i,j) \in A. \tag{5.8k}$$

The objective function (5.8a) minimizes routing costs. Equalities (5.8b) ensure that each supply is delivered by exactly one route. In these constraints, the binary coefficient $a^r_{ip} = 1$ if product $p \in P_i$ is delivered to customer $i \in N$ by route r. Constraint (5.8c) models the number of vehicles and constraints (5.8d) and (5.8e) define the domains for vehicle number variable ψ and route variables λ^r. Constraints (5.8f)–(5.8k) are redundant but might be added for branching and/or to ensure integer solutions. More precisely, constraints (5.8f)–(5.8g) count the number of routes that are packed with compartment combination $L \in X$. Here, the coefficients g^r_L indicate if route r is packed with compartment combination $L \in X$. Moreover, constraints (5.8h)–(5.8k) restrict the number of times customer $i \in N$ is visited and arc $(i,j) \in A$ is traversed. In these constraints, the binary coefficients e^r_i and b^r_{ij} indicate whether customer $i \in N$ is visited and arc $(i,j) \in A$ is traversed by route r, respectively.

Since the set Ω of feasible routes and, accordingly, the number of columns in formulation (5.8) is very big, we perform a branch-price-and-cut (BPC) algorithm to solve the problem. For this purpose, we start with the linear relaxation of (5.8) over a subset $\Omega' \subset \Omega$. This so-called *restricted master problem* (RMP) is solved by column generation (Desaulniers *et al.*, 2005). Similar to the C-SDVRP, the subproblem can be formulated as a variant of the *shortest-path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). To reach integrality this column generation process is embedded in a branch-and-bound algorithm.

In the following, we describe different components of the algorithm, namely how to solve the subproblem, stabilization techniques by the help of dual inequalities, valid inequalities to strengthen the lower bound, the branching procedure, and further acceleration techniques.

5.6.1 Pricing problem

Instead of solving one subproblem at each column generation iteration, we divide the subproblem into several pricing problems and solve each of these pricing problems separately. To reduce the difficulty of packing constraints according to compartments, we consider $|X|$ pricing problems, i.e., one pricing problem for each feasible maximal packing combination $L \in X$, where $L \subseteq P$ denotes the set of considered product types. Recall that for example, an instance with three product types and a maximum of $C = 2$ compartments results in three pricing problems $X = \{\{1,2\},\{2,3\},\{1,3\}\}$.

Let $(\pi_{ip})_{i\in V, p\in P_i}$, σ, $(\nu_L)_{L\in X}$, $(\mu_i)_{i\in V}$, and $(\rho_{ij})_{(i,j)\in A}$ be the dual prices of constraints (5.8b), (5.8c), (5.8f), (5.8h), and (5.8j), respectively. Reconsider the directed graph $\bar{G}^d(\bar{V} \cup \{n+1\}, \bar{A})$ defined in Section 5.2. Let

$$\bar{c}_{kl} = c_{kl} - \frac{1}{2}(\pi_{f_c(k)f_p(k)} + \pi_{f_c(l)f_p(l)}) - \frac{1}{2}(\mu_{f_c(k)} + \mu_{f_c(l)}) - \rho_{f_c(k)f_c(l)}$$

be the modified travel cost over arc $(k,l) \in \bar{A}$. Moreover, let $\bar{A}(L) = \{(k,l) \in \bar{A} : f_p(k), f_p(l) \in L\}$ be the set of arcs associated with pricing problem $L \in X$. Then, the pricing problem for $L \in X$ can be formulated as follows:

$$z(L) = \min \sum_{(k,l)\in\bar{A}(L)} \bar{c}_{kl}x_{kl} - \sigma - \nu_L \tag{5.9a}$$

$$\text{subject to} \sum_{(k,l)\in\bar{A}(L)} x_{kl} - \sum_{(l,s)\in\bar{A}(L)} x_{ls} = 0 \qquad \forall l \in \bar{N} \tag{5.9b}$$

$$\sum_{l\in\bar{V}} x_{0l} = 1 \tag{5.9c}$$

$$\sum_{k \in \bar{V}} x_{k,m+1} = 1 \tag{5.9d}$$

$$\sum_{(k,l) \in \delta(S)} x_{kl} \geq 2\sigma(S) \qquad \forall S \subseteq \bar{N}, S \neq \emptyset \tag{5.9e}$$

$$x_{kl} \in \{0,1\} \qquad \forall (k,l) \in \bar{A}(L). \tag{5.9f}$$

The objective (5.9a) minimizes the reduced cost of the route and float conservation is ensured by constraints (5.9b). Constraints (5.9c) and (5.9d) impose that exactly one vehicle leaves and enters the depot, respectively. Capacity constraints (5.9e) ensure connectivity and packing feasibility according to (5.1). Note that (iii) holds true by construction because the number of used compartments is already limited by $L \in X$. The domain of variables x_{kl} is given by (5.9f).

5.6.2 SPPRC formulation for the pricing problem

We formulate the pricing problem for $L \in X$ as an SPPRC over an undirected multi-graph. For this purpose, the depot node 0 and all customer nodes $i \in N$ are duplicated into two copies $0'$ and $0''$ as well as i' and i'', respectively. Let $V' = \{0'\} \cup \{i' : i \in N\}$ and $V'' = \{0''\} \cup \{i'' : i \in N\}$. Each arc $(i,j) \in A$ results in two *routing edges* $\{i', j''\}$ and $\{i'', j'\}$. To model deliveries to customer i, there are parallel *delivery edges* between i' and i'' for each feasible packing combination $S_i \subseteq P_i$ with $S_i \subseteq L$, denoted as $\{i', i''\}^{S_i}$. Figure 5.2 shows an example of two SPPRC multi-graphs for an instance with three customers.

A route is a $0''$-$0'$-path alternating between edges of the form $\{i'', j'\}$ with $i \neq j$ and edges of the form $\{j', j''\}$. All benchmark instances are symmetric, therefore, the reduced cost can be defined as

$$\tilde{c}_{i',j''} = \tilde{c}_{i'',j'} = c_{ij} - (\mu_i + \mu_j + \rho_{ij} + \rho_{ji})/2 \qquad \forall (i,j) \in A \tag{5.10a}$$

$$\tilde{c}^{S_i}_{i',i''} = -\sum_{p \in S_i} \pi_{ip} \qquad \forall i \in N, S_i \subseteq P_i, S_i \subseteq L \tag{5.10b}$$

with $\mu_0 = \sigma + \nu_L$ yielding a multi-graph with symmetric reduced-cost structure.

The demand is modeled differently for both problem variants. For the MCVRP-CFCS, we set the demand

$$d^{S_i}_{i',i''} = \sum_{p \in S_i} d_{ip} \tag{5.11}$$

for all delivery edges and $d_{i'j''} = d_{i''j'} = 0$ on routing edges $\{i', j''\}$ and $\{i'', j'\}$. A $0''$-$0'$-path represents a feasible route if the accumulated demand does not exceed the vehicle capacity Q.

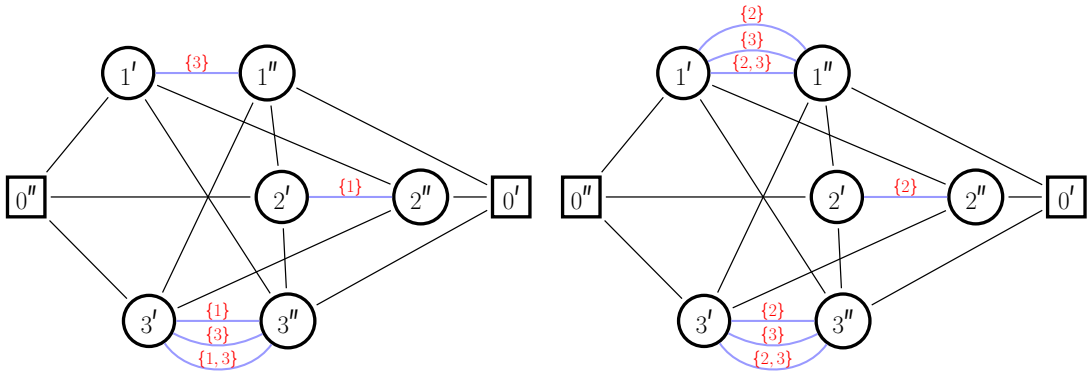


Figure 5.2: Two SPPRC pricing networks with three customers $N = \{1, 2, 3\}$ and product type sets $P_1 = \{2, 3\}, P_2 = \{1, 2\}$ and $P_3 = \{1, 2, 3\}$ for an instance with $C = 2$ and $\rho = 3$ yielding three separate pricing problem $L_1 = \{1, 2\}$, $L_2 = \{1, 3\}$, and $L_3 = \{2, 3\}$. The left picture illustrates the network for L_2 and the right one for L_3. Note that packing combinations including product types $p = 2$ and $p = 1$ are not feasible for L_2 and L_3, respectively.

For the MCVRP-DFCS, we consider instead a demand vector $\boldsymbol{d}$ of dimension $|L|$ as resource with

$$\left(\boldsymbol{d}^{S_i}_{i',i''}\right)_p = \begin{cases} d_{ip} & \text{if } p \in S_i, \\ 0 & \text{otherwise,} \end{cases} \qquad p \in L, \tag{5.12}$$

for delivery edges and $\boldsymbol{d} = \boldsymbol{0}$ for routing edges $\{i', j''\}$ and $\{i'', j'\}$. A $0''$-$0'$-path with accumulated demand vector $\boldsymbol{d}$ represents a feasible route if

$$q^{\text{unit}} \sum_{p \in L} \left\lceil \frac{\boldsymbol{d}_p}{q^{\text{unit}}} \right\rceil \leq Q. \tag{5.13}$$

The solution approach of the pricing problems comprises two phases. First, we pre-compute Pareto-optimal deliveries for each customer $i \in N$. Second, the pricing problem is solved via an SPPRC labeling algorithm on the reduced SPPRC multi-graph only containing Pareto-optimal deliveries.

Pareto-optimal deliveries. Since the number of product types per pricing problem does not exceed ten for all benchmark instances (see Section 5.7.1), the number of Pareto-optimal deliveries can be determined by enumeration. The definition of Pareto-optimality differs for both problem variants. For the MCVRP-CFCS, an

edge $\{i', i''\}^{S_i^1}$ is not Pareto-optimal and can be excluded if an edge $\{i', i''\}^{S_i^2}$ exists with

$$\tilde{c}_{i',i''}^{S_i^2} \leq \tilde{c}_{i',i''}^{S_i^1} \quad \text{and} \quad d_{i',i''}^{S_i^2} < d_{i',i''}^{S_i^1}. \tag{5.14}$$

For the MCVRP-DFCS, additionally $S_i^2 \subseteq S_i^1$ must hold. Note that the Pareto-reduction must be repeated in every column generation iteration because the dual prices change in each iteration.

SPPRC over the reduced multi-graph. To solve the SPPRC on the reduced multi-graph, we use the following resources: (i) accumulated reduced cost according to (5.10); (ii) accumulated demand (vector) according to (5.11) or (5.12), respectively; and (iii) visit indicators for each customer $i \in N$ that are increased when one of the edges $\{i', i''\}^{S_i}$ is traversed. At the beginning, all resources are set to 0 and labels are propagated alternating between node sets V' and V'', i.e., in monodirectional forward labeling, a node i' is only propagated towards the same customer node i'' and nodes $i'' \in V''$ are only propagated towards a different customer node $j' \in V'$ with $i \neq j$. Labels are feasible if the (sum of vector entries of the) demand does not exceed Q and if the visit indicator does not exceed 1. Note that for the MCVRP-DFCS, it does not suffice to compare the accumulated demand for dominance but the demand vector must be taken into account component-by-component.

It is possible to use an implicit bidirectional labeling approach because the SPPRC is completely symmetric such that forward and backward propagation produces identical partial paths. Thereby, the computational effort can be reduced by only propagating in one direction and combining these partial paths in a merge procedure. This technique has already been applied in (Bode and Irnich, 2012; Goeke *et al.*, 2019; Gschwind *et al.*, 2019).

5.6.3 Stabilization and dual inequalities

To stabilize the column generation process, *dual inequalities* (DIs) can be used. Let D^* be the set of optimal solutions to the dual model to the linear relaxation of (5.8). According to (Ben Amor *et al.*, 2006), a *dual-optimal inequality* (DOI) is defined as a DI of the form $\boldsymbol{t}^T\pi \leq t$ with $\boldsymbol{t} \in \mathbb{Z}^m$ and $t \in \mathbb{Z}$ if $D^* \subseteq \{\pi : \boldsymbol{t}^T\pi \leq t\}$. Moreover, a set of DIs $\boldsymbol{Q}^T\pi \leq \boldsymbol{q}$ with $\boldsymbol{Q} \in \mathbb{Z}^{m\times n}$ and $\boldsymbol{q} \in \mathbb{Z}^n$ comprises *deep dual-optimal inequalities* (DDOIs) if $D^* \cap \{\pi : \boldsymbol{Q}^T\pi \leq \boldsymbol{q}\} \neq \emptyset$. A general introduction to the use of DIs for the stabilization of the column generation process can be found in (Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016).

DIs are in general not necessarily DOIs or DDOIs for both the MCVRP-CFCS and MCVRP-DFCS. Nevertheless, it is beneficial to add DI columns a priori to

the RMP to stabilize the column generation process at the risk of a possible over-stabilization. The addition of DIs and possible over-stabilization resolved with a recovery procedure are explained in more detail in the following.

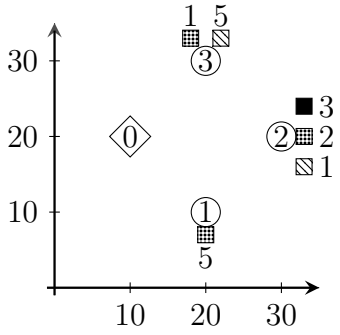
Static addition of dual inequalities. For each customer $i \in N$ and product pair $p, q \in P$ with $d_{ip} \leq d_{iq}$, the DIs columns corresponding to the *pair inequalities* (PI) $\pi_{ip} \leq \pi_{iq}$ are added to the initial RMP. Since the number of product types $|P_i|$ per customer i is small (less than ten for all benchmark instances) and rather many PIs are eliminated because of over-stabilization (see the paragraph below), we decided to add all PIs per customer instead of typically used *ranking inequalities* $\pi_{ip_1} \leq \pi_{ip_2} \leq \cdots \leq \pi_{ip_{|P_i|}}$ with $d_{ip_1} \leq d_{ip_2} \leq \cdots \leq d_{ip_{|P_i|}}$ (Ben Amor *et al.*, 2006). To avoid many iterations of the recovery procedure, we do not add further DIs of the form $\pi_{ip} \leq \sum_{p \in S} \pi_{ip}$ with $S \subseteq P_i$, so-called *subset inequalities*, that strongly influence the compartment composition of the solution. Moreover, we do not dynamically add violated DIs during the pricing approach.

Over-stabilization and recovery procedure. As shown in Figure 5.3, PIs are in general neither DOIs nor DDOIs. In particular, for a customer $i \in N$ and two different product types $p, q \in P_i$ with $d_{iq} \leq d_{ip}$, a route serving customer i and delivering p and not q cannot always be feasibly replaced by an otherwise identical route that delivers q and not p. As a counterexample, consider the first route in Figure 5.3c and the DI column replacing product type ▦ by ▧ of customer 2. The replacement results in a non-feasible route delivering three product types ■, ▦, and ▧. Thus, the constraint matrix of (5.8) does not possess the so-called row replacement property (Gschwind and Irnich, 2016, Definition 2).

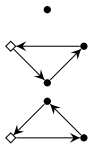
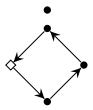
By anyway adding PIs to the initial RMP, all dual-optimal solutions may be cut-off. This possible over-stabilization can be purged by a recovery procedure proposed in (Gschwind and Irnich, 2016). Given the RMP solution, this procedure tries to build a pure route-columns solution. If this is not possible, i.e., a DI column with a positive value corresponding to $\pi_{ip} \leq \pi_{iq}$ remains that is not compatible with any route column, then the RMP is over-stabilized. In this case, the recovery procedure eliminates all PIs $\pi_{i\bar{p}} \leq \pi_{iq}$ with $\bar{p} \in P_i$ from the RMP. Afterwards, the column generation process restarts and iterates until a pure route-columns solution exists. Note that a DI column is classified incompatible with a route column if either the resulting route column exceeds the number of compartments C or a product type is delivered twice.

Since the row replacement property does not hold for the constraint matrix of (5.8), over-stabilization occurs more often compared to other problems such as the bin packing problem or the vertex coloring problem for which the constraint matrix possesses the row replacement property (Gschwind and Irnich, 2016). How-

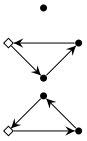
ever, adding PIs to the initial RMP is still beneficial because the recovery procedure is fast such that (over-)stabilization pays off (see Table 5.3).



(a) Instance of the MCVRP-CFCS. The demand of each product type is denoted next to the product type symbols ■, ▦, and ▧.

route	product types	c^r	λ^r
	■ ▦	48.28	0.5
	■ ▧	48.28	0.5
	▦ ▧	28.28	0.5
	▦ ▧	56.57	0.5

(b) Solution of the linear relaxation of (5.8). The optimal objective value is 90.71.

route	product types	c^r	λ^r
	■ ▦	48.28	1.0
	▦ ▧	48.28	0.5
	▦ ▧	28.28	0.5
customer	DI column	c^{DI}	λ^{DI}
2	replace ▦ by ▧,	0.0	0.5

(c) Solution of the linear relaxation of (5.8) with PIs. The optimal objective value is 86.57.

Figure 5.3: Instance and optimal of the linear relaxation of (5.8) solutions of the MCVRP-CFCS with and without PIs with depot node 0, three customer nodes $N = \{1, 2, 3\}$, and $m = 2$ vehicles with capacity $Q = 10$ and $C = 2$ compartments. The optimal objective value of the linear relaxation of (5.8) is 90.71. However, when PIs are added to the model, the optimal objective value is 86.57. Thus, all dual optimal solutions are cut-off.

5.6.4 Valid inequalities and cutting strategy

Three classes of valid inequalities are added to the RMP during the solution process. On the one hand we add two families of non-robust cuts, namely *subset-row inequalities* (SR inequalities) (Jepsen *et al.*, 2008) for subsets of cardinality three and *strong-degree constraints* (SD constraints) (Contardo *et al.*, 2014). Subset-row inequalities for subsets of cardinality three ensure for elementary routes that at most one route that fulfills at least two of three tasks is part of a feasible solution. Strong-degree constraints ensure that a demand d_{ip} with $i \in N$ and $p \in P$ is served by at least one elementary or non-elementary route. The definition of these non-robust cuts including the impact on DIs is the same for both MCVRP-FCS variants as for the C-SDVRP. Therefore, we refer to (Gschwind *et al.*, 2019) for a detailed description. On the other hand, we add the family of robust capacity cuts (Fukasawa *et al.*, 2005) that are described in the following.

Let $S \subseteq N$, $S \neq \emptyset$, be a customer subset and let $\delta^-(S)$ denote the arcs of the digraph $G = (V, A)$ with $i \notin S$ and $j \in S$. Then, we can formulate the *capacity cut* (CC)

$$\sum_{r \in \Omega} \left(\sum_{(i,j) \in \delta^-(S)} b_{ij}^r \right) \lambda^r \geq \max \left\{ \left\lceil \frac{\sum_{i \in V} \sum_{p \in P_i} d_{ip}}{Q} \right\rceil, \left\lceil \frac{|\{p \in P_i : i \in S\}|}{C} \right\rceil \right\}. \quad (5.15)$$

The corresponding dual price is γ_S. The right-hand side does not only consider the vehicle capacity Q but also the available number of compartments C. These cuts are robust because the value $\gamma_S/2$ can be distributed symmetrically on the edges (i', j'') and (i'', j') for all $(i, j) \in \delta^-(S)$ of the undirected SPPRC pricing network.

Overall cutting strategy. The cut-generation strategy depends on the MCVRP-FCS variant and the underlying instance. Since the number of compartments C is typically more restrictive than the vehicle capacity Q, SR inequalities and SD constraints are less effective compared to the C-SDVRP. Moreover, both cut types influence the Pareto-reduction and are therefore not used at all or only separated and added up to level three in the branch-and-bound tree (for details see Section 5.7.3). SD constraints are additionally added deeper in the tree if needed to guarantee elementary routes for the completeness of the branching rule (cf. Section 5.6.5).

In contrast, CCs are very effective for both MCVRP-FCS variants. Therefore, the following CCs are already added at the beginning to the initial RMP. For each customer $i \in N$ with $|P_i| > C$, we add a capacity cut for subset $S = \{i\}$ if the right-hand side of (5.15) is at least 2. Moreover, we want to add efficient CCs by defining customer subsets S that are close together in distance and consist of many

different product types. Therefore, let $r_i(j)$ be a ranking function ordering the neighbors of i by travel cost, i.e. $r_i(j_1) = 1, r_i(j_2) = 2, \dots$ for ordered travel costs $c_{i,j_1} \leq c_{i,j_2} \leq \dots$ for $j_1, j_2, \dots \in N$. For each $i \in N$, we add all CCs for customer subsets $S = \{i, j\} \subseteq N$ with $P_i \cup P_j \neq P_i \cap P_j$, $|P_i \cup P_j| > C$, and minimal ranking function $r_i(j)$. Additionally, for instances with three or more available vehicles, we sort for each customer $i \in N$ the neighbors $j_1, j_2, \dots \in N$ according to the ranking function, i.e. $r_i(j_1) < r_i(j_2) < \dots$, and add a CC for the smallest subset $S = \{i, j_1, j_2, \dots\}$ with the right-hand side of (5.15) equal to 3.

5.6.5 Branching

In the following, we briefly summarize the six-level branching strategy that is similar to the one applied in (Archetti *et al.*, 2015; Gschwind *et al.*, 2019). At the first level, we branch on the number of vehicles and at the second level, we branch on the number of routes that are packed with compartment combination $L \in X$ (see constraints (5.8f)–(5.8g)). At the third level, we branch on the number of visits to each customer. Note that infeasible subsets P_i are eliminated from the customer network. At the fourth level, we branch on the edge flow. Again, edges can be eliminated from the customer network for zero-flow decisions. At level five and six, we use Ryan-Foster branching for supplies at the same customer and at different customers, respectively. Up to level six the branching scheme is not complete because non-elementary routes can still be part of the solution. To guarantee integer route variables, we separate SD constraints if all other values considered at branching levels one to six are integer. For an explanation for the completeness of the branching scheme and the impact of branching on DIs, we refer to (Gschwind *et al.*, 2019, p. 97).

To improve the dual bound as fast as possible, we use a best-bound first tree exploration strategy. In all cases, the branching variable is selected as the one with the fractional part closest to 0.5.

5.6.6 Acceleration techniques

To relax the elementary SPPRC, we use the *ng*-path relaxation proposed by Baldacci *et al.* (2011) that prohibits cycles in a pre-defined neighborhood of node i but allows cycles over node j if j is not in the neighborhood of i. For larger neighborhood sizes, fewer cycles are possible but the computational effort increases on average. In our case, a good tradeoff between neighborhood size and computational effort is obtained with a neighborhood size of ten.

Moreover, the SPPRC is solved first heuristically on several reduced SPPRC network at each iteration to accelerate the column generation process. We consider two types of reduction techniques. The first one reduces the size of the customer

network according to delivery edges. We control the size of the graph with two parameters $D^{del} \in \{1, 3, S^*\}$ and $D^{adj} \in \{2, \text{TSP}, 5, 10, |n|\}$ with the following meaning. For $D^{del} = 1$ ($D^{del} = 3$), we limit the number of delivery edges for each customer $i \in N$ to 1(3). More precisely, if $D^{del} = 1$ then the graph consists of one product combination with the best reduced cost for each customer $i \in N$, i.e. $S_i^* = \arg\min_{S_i \subseteq P_i} \tilde{c}_{i',i''}^{S_i}$. If $D^{del} = 3$ then the graph consists of three Pareto-optimal product combinations with the best reduced cost for each customer $i \in N$, i.e $S_i^* = \{S_{i_1}, S_{i_2}, S_{i_3}\}$ with $\tilde{c}_{i',i''}^{S_{i_1}}, \tilde{c}_{i',i''}^{S_{i_2}}, \tilde{c}_{i',i''}^{S_{i_3}}$ minimal. Moreover, $D^{del} = S^*$ denotes all Pareto-optimal deliveries. The second type of reduction technique eliminates routing edges from the customer network. For $D^{adj} = 2(D^{adj} = 5, D^{adj} = 10)$, we limit the number of edges adjacent to a customer by 2(5,10). Additionally, we only consider edges between customers and the depot as well as edges between customers belonging to the pre-calculated TSP-tour over all nodes. Let $D^{adj} = \text{TSP}$ denote this relaxation. Combining both reduction techniques and considering different pricing problems, the overall pricing strategy is depicted in Algorithm 13.

Algorithm 13: Heuristic pricing strategy

input : dual prices for the SPPRC network
output: negative reduced cost columns or information that no such column exists

```
for D^del ∈ {1, 3, S*} do
    for D^adj ∈ {2, TSP, 5, 10, |n|} do
        for randomly sorted L ∈ X do
            Solve pricing problem L for the reduced SPPRC network with
             delivery edges D^del and routing edges D^adj;
            if at least one negative reduced cost column is found then
                return columns;
return information that no negative reduced cost column exist;
```

5.7 Computational results

In this section, we first give an overview of the benchmark instances and then describe details of the implementation. After presenting an overview of pretests and the computational setup, the section closes with detailed results, a comparison between the algorithms and total costs for both MCVRP-FCS variants, and a comparison of the algorithms according to the number of vehicles.

5.7.1 Benchmark instances

In total, we consider three sets of `small(H15)`, `mid-size(H18)`, and `large(H15)` benchmark instances. All instances are characterized by three parameters: the number of product types ρ, the number of available compartments per vehicle C, and a supply parameter s that denotes if the total number of supplies is small ($s = 1$), medium ($s = 2$), or large ($s = 3$). Note that the classification into `small(H15)`, `mid-size(H18)`, and `large(H15)` only depends on the number of nodes $|V|$ and not on other parameters (especially not on the supply parameter s). The same set of benchmark instances can be used for both problem variants. For the MCVRP-DFCS, the unit compartment size is set to $q^{\text{unit}} = 0.1\,Q$, i.e., the vehicle is divided into 10 basic compartment units. Note that the vehicle capacity Q is divisible by ten for all benchmark instances. If the number of product types equals the number of compartments, i.e., $\rho = C$, then the number of compartments does not restrict the feasible region and the MCVRP-CFCS is actually a C-SDVRP.

The first set of `mid-size(H18)` instances is proposed in (Henke *et al.*, 2018) and consists of 675 instances with 10 to 50 nodes. The number of product types is $\rho \in \{3, 4\}$, the maximal number of compartments is $C \in \{2, 3, 4\}$, and the supply parameter is $s \in \{1, 2, 3\}$. The instances are constructed in such a way that the number of vehicles $m \in \{2, 3\}$ is relatively constant.

The second and third set of benchmark instances are introduced in (Henke *et al.*, 2015). For these instances, the number of product types $\rho \in \{3, 6, 9\}$ and the maximal number of compartments $C \in \{2, 3, 4, 6, 7, 9\}$ are larger compared to the first set of instances while the supply parameter is again $s \in \{1, 2, 3\}$. Moreover, the number of vehicles is not relatively constant but is higher for instances with more customers and total demand. Originally, the second set contained 1350 instances with 10 nodes. Because the instances are small and rather easy to solve, we only use a subset of 135 `small(H15)` instances that consists of 5 (instead of 50) instances for each ρ-C-s-combination. The third set of 27 `large(H15)` instances with 50 nodes contains one instance for each ρ-C-s-combination.

5.7.2 Details of the implementation

The branch-and-cut algorithms are implemented in C++ using CPLEX 12.10.0 with Concert Technology. For the branch-price-and-cut algorithm, the RMP is also solved utilizing CPLEX at each column generation iteration. Moreover, CPLEX is used as a primal MIP-based heuristic solver after the solution of each branch-and-bound node using all generated route columns, i.e., disregarding the DI columns. All algorithms are compiled into 64-bit single-thread code with Microsoft Visual Studio 2015. The computational study is carried out on a 64-bit Microsoft Windows 10 computer with an Intel® Core™ i7-5930k CPU clocked at 3.5 GHz and

64 GB of RAM. For the separation procedure of the branch-and-cut algorithms, generic callbacks are used for both user and lazy cuts. As in (Henke *et al.*, 2018), computation times are limited to a maximum of 7200 seconds (2 hours). Apart from the number of threads and the time limit, CPLEX's default values are kept for all parameters. For all algorithms, we consider an inequality violated if the difference between the left-hand and right-hand side is greater than the threshold $\epsilon = 10^{-4}$.

5.7.3 Pretests and computational setup

In this section, we specify the solution approaches that are compared for both problem variants. Pretests showed that it is beneficial to combine two of the three solution approaches (see details below). Table 5.2 shows an overview of all solution approaches that are explained in detail in the following.

Table 5.2: Overview of solution approaches.

Algorithm\Uses model(s)	`ThreeIndex`	`ThreeIndexDiscrete`	`TwoIndex`	`BaP`	`BaP+ThreeIndex`	`BaP+ThreeIndexDiscrete`	`BaP+TwoIndex`
Branch-price-and-cut (5.8)				×	× max. 60 sec. or ≤ 1000 columns	× max. 60 sec. or ≤ 1000 columns	× max. 60 sec. or ≤ 1000 columns
MIP solver on restricted column set					×	×	×
MIP (5.2)+(5.3a)+(5.4)	×				×		
MIP (5.2)+(5.3b)+(5.4)		×				×	
MIP (5.5)+(5.6)			×				×
MCVRP-CFCS	×	×	×	×	×		×
MCVRP-DFCS		×	×	×		×	×

Note: As proposed by Henke *et al.* (2018), we use `ThreeIndexDiscrete` to solve not only the MCVRP-DFCS but also the MCVRP-CFCS by setting $q^{\text{unit}} = 1$.

The branch-price-and-cut algorithm proposed in Section 5.6 is denoted by `BaP`. Moreover, we refer to the branch-and-cut algorithms based on the three-index and two-index formulation as `ThreeIndex` (for the continuous variant), `ThreeIndexDiscrete` (for the discrete variant), and `TwoIndex` (for both variants),

respectively. Henke *et al.* (2018) propose to solve the MCVRP-CFCS with the three-index formulation for the MCVRP-DFCD and unit size $q^{\text{unit}} = 1$. We also refer to this version as `ThreeIndexDiscrete`.

For the branch-price-and-cut algorithm, pretests showed that the following settings are beneficial. As stated in Section 5.6.4, some CCs are already added at the beginning to the initial RMP and the cut generation of SR inequalities and SD constraints depends on the instance and problem variant. For the MCVRP-CFCS, SR inequalities and SD constraints affect the Pareto-reduction and have a strong impact on computation times unless the supply parameter is $s = 1$. Therefore, we do not use SR inequalities at all and SD constraints within the first levels of the branch-and-bound tree for the MCVRP-CFCS for instances with supply parameter $s = 2$ and $s = 3$. SD constraints are only separated when all values at all branching levels are integer. For the MCVRP-DFCS, the Pareto-reduction is less effective and SR inequalities and SD constraints are used up to level three in the branch-and-bound tree.

For `BaP`, we also conducted pretests to analyze the influence of stabilization (see Section 5.6.3). Results are summarized in Table 5.3. The pretests on `mid-size(H18)` instances with 10 and 15 customers show that adding PIs to the initial RMP is beneficial even though PIs are in general neither DOIs nor DDOIs . Note that the entries of all result tables have the following meaning:

set:	benchmark instance class;
#opt:	number of instances solved to proven optimality within 2 hours (7200 seconds);
time $\bar{T}$:	average computation time in seconds; unsolved instances are taken into account with the time limit TL of 2 hours (7200 seconds);
gap:	$100 \cdot (UB - LB)/LB$, i.e., the gap in percent at termination;
No.:	number of the instance;
#inst:	number of instances;
Total:	best results of all algorithms (using additionally the best found upper bound of the MCVRP-DFCS as upper bound for the MCVRP-CFCS and the best found lower bound of the MCVRP-CFCS as lower bound for the MCVRP-DFCS).

Further pretests showed that the computation time of `ThreeIndex` mainly depends on the number of nodes. However, w.r.t. `TwoIndex` and `BaP`, we did not find any simple correlation pattern to explain the computational time. We can in fact observe instance-specific discrepancy for the latter two in Table 5.4. To take advantage of the obvious discrepancy in the computation times, we combine both algorithms by first solving the problem with `BaP`. If no optimal solution is

Table 5.3: Comparison of stabilized and non-stabilized `BaP` results for the MCVRP-CFCS on `mid-size(H18)` instances with $|V| \in \{10, 15\}$.

Instance			Stabilized			Non-stabilized		
$\|V\|$	s	#inst	#opt	time T	gap	#opt	time T	gap
10	2	25	**25**	**1.7**	**0.0**	**25**	2.3	0.2
	3	25	**25**	**15.3**	**0.0**	24	305.7	4.1
Total ($\|V\| = 10$)		50	**50**	**8.5**	**0.0**	49	154.0	2.1
15	2	25	**25**	**397.5**	**0.1**	24	1202.7	16.2
	3	25	**15**	**3942.0**	**28.2**	11	4411.5	69.5
Total ($\|V\| = 15$)		50	**40**	**2169.7**	**14.2**	35	2807.1	34.4

found after 60 seconds or 1000 generated columns, the MIP solver is called for the restricted model with the generated columns to find a good feasible solution. Afterwards `TwoIndex` is called with the upper bound resulting from the feasible solution. This bound often reduces the size of the branch-and-bound tree and, therefore, yields a better computational performance. We refer to this variant as `BaP`+`TwoIndex`. To test the influence of using upper bounds for the branch-and-cut algorithm of the three-index formulation described in Section 5.4, we also consider the variant `BaP`+`ThreeIndex` in which the `BaP` is analogously interrupted after 60 seconds or 1000 generated columns.

Table 5.4: Comparison of `TwoIndex`, `BaP`, and `BaP`+`TwoIndex` results for the MCVRP-CFCS on `mid-size(H18)` instances with $|V| \in \{10, 15\}$.

Instance					`TwoIndex`			`BaP`			`BaP`+`TwoIndex`		
$\|V\|$	ρ	C	s	*No.*	#opt	time T	gap	#opt	time T	gap	#opt	time T	gap
10	4	2	3	1	**1**	378.3	**0.0**	**1**	30.8	**0.0**	**1**	**24.9**	**0.0**
	4	2	3	3	**1**	2258.2	**0.0**	**1**	**91.0**	**0.0**	**1**	688.3	**0.0**
Total ($\|V\| = 10$)					**75**	99.5	**0.0**	**75**	**7.4**	**0.0**	**75**	17.3	**0.0**
15	3	2	2	3	**1**	**4.3**	**0.0**	**1**	529.6	**0.0**	**1**	66.8	**0.0**
	3	3	2	4	**1**	**0.2**	**0.0**	**1**	1119.9	**0.0**	**1**	63.8	**0.0**
Total ($\|V\| = 15$)					**70**	691.7	0.4	63	1653.8	13.4	69	**677.7**	**0.3**

5.7.4 Results for the MCVRP-CFCS

To compare the MCVRP-CFCS algorithms, we first consider `mid-size(H18)` benchmark instances with a lower number of product types ρ compared to the `small(H15)` and `large(H15)` benchmark instances. As mentioned before, we combine the `BaP` approach with both the `ThreeIndex` and `TwoIndex` approach. The results grouped by the number of supplies are summarized in Table 5.5. Overall, the `ThreeIndex` approach can solve most of the instances to proven optimality and has the lowest average computation time. `ThreeIndexDiscrete` is slightly inferior but can solve one more instance with supply parameter $s = 1$ exactly and is on average slightly faster for instances with supply parameter $s = 2$. Using the `BaP` approach up to 60 seconds does not accelerate the `ThreeIndex` approach on average. The `BaP+TwoIndex` approach is altogether inferior but is almost 50 % faster for instances with supply parameter $s = 1$.

Table 5.5: MCVRP-CFCS results for `mid-size(H18)` instances grouped by the number of supplies.

Instances		`ThreeIndex`			`BaP+ThreeIndex`			`BaP+TwoIndex`			`ThreeIndexDiscrete` (Henke *et al.*, 2018)			Total
s	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
1	225	210	602.4	0.5	210	597.2	0.5	**216**	**393.7**	**0.4**	211	632.6	0.5	220
2	225	**209**	914.9	**0.3**	208	904.4	**0.3**	153	2373.6	16.3	205	**890.8**	**0.3**	211
3	225	**209**	**811.9**	**0.5**	208	857.3	**0.5**	123	3447.0	36.3	207	861.9	**0.5**	211
Total	675	**628**	**776.4**	**0.4**	626	786.3	**0.4**	492	2071.4	17.7	623	795.1	**0.4**	642

Given an instance, it is very simple to compute s and classify the instance. Therefore, we suggest applying the `BaP+TwoIndex` approach for instances with $s = 1$ and the `ThreeIndex` approach for instances with $s = 2, 3$, respectively. In the following, we refer to this combined approach as `ThreeIndex/BaP+TwoIndex`. The results grouped by the number of nodes are summarized in Table 5.6. Excluding results of `ThreeIndex/BaP+TwoIndex`, the `ThreeIndex` approach performs best and is superior in almost all groups. The computation time of `BaP+TwoIndex` is on average considerably slower. Nevertheless, combing both approaches yields six more optimally solved instances and reduces the average computation time by around 70 seconds compared to the `ThreeIndex` approach. In total, the `ThreeIndex/BaP+TwoIndex` approach can solve 634 of 675 instances to proven optimality.

The best-performing algorithms `ThreeIndex`, `BaP+TwoIndex`, and `ThreeIndex/BaP+TwoIndex` are also tested on `small(H15)` and `large(H15)` instances with a higher number of product types compared to the `mid-size(H18)` instances. Additionally, we test the `BaP` approach for these instances because symmetry issues are more relevant for the `TwoIndex` approach for larger ρ.

Table 5.6: MCVRP-CFCS results for `mid-size(H18)` instances grouped by the number of nodes.

Instances		`ThreeIndex`			`BaP+ThreeIndex`			`BaP+TwoIndex`			`ThreeIndexDiscrete` (Henke *et al.*, 2018)			`ThreeIndex/` `BaP+TwoIndex`			Total
$\|V\|$	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
10	75	**75**	0.9	**0.0**	**75**	3.4	**0.0**	**75**	17.3	**0.0**	**75**	1.0	0.0	**75**	**0.8**	**0.0**	75
15	75	**75**	4.0	**0.0**	**75**	32.4	**0.0**	69	677.7	0.3	**75**	5.2	0.0	**75**	**3.8**	**0.0**	75
20	75	**75**	28.8	**0.0**	**75**	57.3	**0.0**	63	1371.3	9.6	**75**	**22.6**	0.0	**75**	27.9	**0.0**	75
25	75	**75**	107.7	**0.0**	**75**	132.1	**0.0**	54	2061.6	13.8	**75**	147.5	0.0	**75**	**84.4**	**0.0**	75
30	75	**75**	214.0	**0.0**	**75**	209.4	**0.0**	51	2440.8	21.9	**75**	280.1	0.0	**75**	**194.9**	**0.0**	75
35	75	**72**	845.4	**0.1**	**72**	757.5	**0.1**	49	2630.3	25.8	**72**	**658.7**	**0.1**	**72**	844.9	**0.1**	74
40	75	**66**	1407.5	0.7	65	1423.8	**0.6**	44	3031.0	28.7	66	**1338.3**	**0.6**	65	1439.9	0.9	68
45	75	58	2205.3	**1.4**	58	2193.5	1.5	44	3119.6	27.9	55	2294.2	**1.4**	**60**	**2017.4**	1.5	62
50	75	57	2174.2	1.8	56	2267.3	1.7	43	3293.2	31.1	55	2408.4	1.6	**62**	**1747.6**	**1.5**	63
Total	675	628	776.4	**0.4**	626	786.3	**0.4**	492	2071.4	17.7	623	795.1	**0.4**	**634**	**706.9**	**0.4**	642

Results grouped by the number of product types and the supply parameter are summarized in Table 5.7. Note that we only report the gap if an upper bound is found.

For the `small(H15)` instances with only 10 nodes, the `ThreeIndex` approach can solve all instances with an average computation time of 37.2 seconds to proven optimality. The other approaches, `BaP` and `BaP+TwoIndex`, only perform appropriately for instances with supply parameter $s = 1$. Also the `ThreeIndex`/`BaP+TwoIndex` approach can solve all instances with an average computation of about 60 seconds.

For `large(H15)` instances, the performance of the algorithms is different. The `ThreeIndex` approach cannot solve any of the instances and cannot even find a feasible solution. The reason for the poor performance is most likely that the number of vehicles is on average three times higher for `large(H15)` instances compared to `small(H15)` and `mid-size(H18)` instances and, therefore, symmetry issues are more relevant. Note that the number of available vehicles for `mid-size(H18)` (`large(H15)`) instances is on average 2.7 (8.4). The `BaP` approach can solve four instances to proven optimality and the `BaP+TwoIndex` approach performs best with eight instances solved. In total, we can find the optimal solution for 8 of 27 `large(H15)` instances.

Summarizing, we advise using the `ThreeIndex` approach for instances with a small number of nodes $|V|$. For `mid-size(H18)` and `large(H15)` instances, we recommend solving the instance with the `ThreeIndex`/`BaP+TwoIndex` approach. The diagram in Figure 5.4 summarizes which algorithm we recommend to apply depending on different instance parameters. Combining all results and, additionally, using the best found upper bound of the MCVRP-DFCS as upper bound, we can solve all 135 `small(H15)` instances, 642 of 675 `mid-size(H18)` instances and 8 of 27 `large(H15)` instances. Instance-by-instance results are shown in the Appendix.

Table 5.7: MCVRP-CFCS results for `small(H15)` and `large(H15)` instances grouped by the number of product types and the supply parameter.

Instance					ThreeIndex			BaP			BaP+TwoIndex			ThreeIndex/ BaP+TwoIndex			Total
class	$\|V\|$	ρ	s	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
small(H15)	10	3	1	10	**10**	0.2	**0.0**	**10**	**0.1**	**0.0**	**10**	**0.1**	**0.0**	**10**	**0.1**	**0.0**	10
			2	10	**10**	**0.4**	**0.0**	**10**	0.9	**0.0**	**10**	0.7	**0.0**	**10**	**0.4**	**0.0**	10
			3	10	**10**	**0.5**	**0.0**	**10**	3.0	**0.0**	**10**	3.1	**0.0**	**10**	**0.5**	**0.0**	10
		6	1	15	**15**	**0.4**	**0.0**	**15**	1.2	**0.0**	**15**	0.8	**0.0**	**15**	0.8	**0.0**	15
			2	15	**15**	**1.1**	**0.0**	**15**	40.5	**0.0**	12	1473.0	0.9	**15**	**1.1**	**0.0**	15
			3	15	**15**	**2.2**	**0.0**	14	531.0	4.3	13	1143.4	8.0	**15**	**2.2**	**0.0**	15
		9	1	20	**20**	**12.3**	**0.0**	**20**	12.9	**0.0**	20	146.9	**0.0**	**20**	146.9	**0.0**	20
			2	20	**20**	**85.6**	**0.0**	18	1287.2	0.3	9	4017.0	15.0	**20**	**85.6**	**0.0**	20
			3	20	**20**	**149.9**	**0.0**	11	4317.9	2.1	5	5407.0	38.2	**20**	**149.9**	**0.0**	20
Total ($\|V\| = 10$)				135	**135**	**37.2**	**0.0**	123	896.2	0.8	104	1709.0	8.9	**135**	57.2	**0.0**	135
large(H15)	50	3	1	2	0	*TL*		1	3615.6		**2**	**425.6**		**2**	**425.6**		2
			2	2	0	*TL*		0	*TL*		**1**	**3723.7**		0	*TL*		1
			3	2	0	*TL*		0	*TL*		**1**	**3813.0**		0	*TL*		1
		6	1	3	0	*TL*		**2**	4203.4		**2**	**2663.2**		**2**	**2663.2**		2
			2	3	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			3	3	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
		9	1	4	0	*TL*		1	6983.1		**2**	**3921.2**		**2**	**3921.2**		2
			2	4	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			3	4	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
Total ($\|V\| = 50$)				27	0	*TL*		4	6569.4		**8**	**5200.2**		**6**	**5708.4**		8

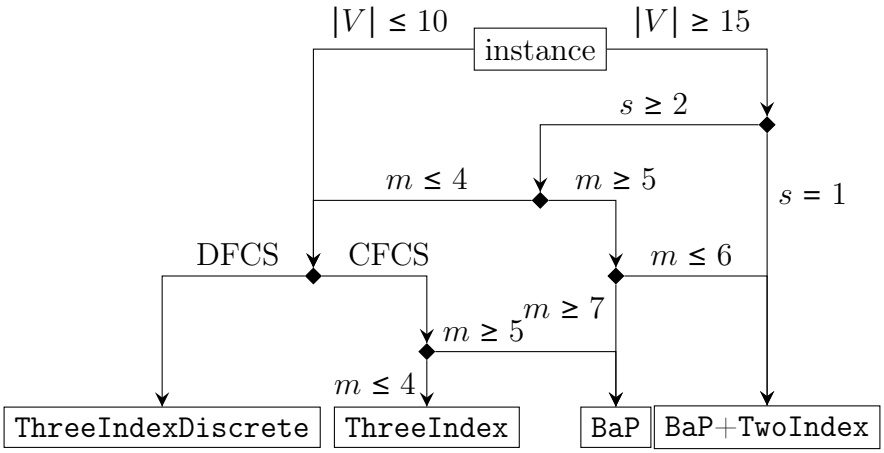


Figure 5.4: Diagram showing which algorithm we recommend to apply depending on the instance parameters $|V|, s, m$, and the problem variant (MCVRP-CFCS or MCVRP-DFCS).

5.7.5 Results for the MCVRP-DFCS

For the MCVRP-DFCS variant, we also consider the `mid-size(H18)` benchmark instances with a lower number of product types ρ first. The results grouped by the number of supplies are summarized in Table 5.8. The performance of the algo-

rithms is similar to the MCVRP-CFCS variant. Again, using the `BaP` approach up to 60 seconds does not accelerate the `ThreeIndexDiscrete` approach on average. Overall, the `ThreeIndexDiscrete` approach performs best but the `BaP+TwoIndex` is superior for instances with supply parameter s = 1. Due to the different performance of the algorithms for different supply parameters s, we also consider the `ThreeIndexDiscrete/BaP+TwoIndex` approach.

Table 5.8: MCVRP-DFCS results for `mid-size(H18)` instances grouped by the number of supplies.

Instances		`ThreeIndexDiscrete` (Henke *et al.*, 2018)			`BaP+ThreeIndexDiscrete`			`BaP+TwoIndex`			Total
s	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
1	225	208	704.9	**0.6**	208	719.7	**0.6**	**215**	**521.6**	0.7	221
2	225	**206**	**1025.1**	**0.3**	**206**	1061.1	0.4	150	2594.5	19.8	207
3	225	**208**	**871.1**	**0.5**	207	941.2	0.6	114	3685.6	37.2	209
Total	675	**622**	**867.0**	**0.5**	621	907.3	**0.5**	479	2267.2	19.2	637

Results grouped by the number of nodes can be found in Table 5.9. Excluding results of `ThreeIndexDiscrete/BaP+TwoIndex`, the `ThreeIndexDiscrete` approach is the best performing single-stage approach with 622 of 675 optimally solved instances and an average computation time of 867.0 seconds. For this problem variant, the `ThreeIndexDiscrete/BaP+TwoIndex` approach can solve 629 of 675 instances to proven optimality. Compared to the MCVRP-CFCS variant, we can solve 5 instances less and the average computation time increases by around 100 seconds.

Table 5.9: MCVRP-DFCS results for `mid-size(H18)` instances grouped by the number of nodes.

Instances		`ThreeIndexDiscrete` (Henke *et al.*, 2018)			`BaP+ThreeIndexDiscrete`			`BaP+TwoIndex`			`ThreeIndexDiscrete/BaP+TwoIndex`			Total
$\|V\|$	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
10	75	**75**	1.2	**0.0**	**75**	19.4	**0.0**	74	175.8	< 0.1	**75**	**1.1**	**0.0**	75
15	75	**75**	**5.5**	**0.0**	**75**	43.3	**0.0**	67	920.9	3.4	**75**	7.0	**0.0**	75
20	75	**75**	**37.7**	**0.0**	**75**	78.3	**0.0**	63	1473.2	8.3	**75**	52.2	**0.0**	75
25	75	**74**	**190.8**	**< 0.1**	**74**	238.4	**< 0.1**	53	2300.9	20.5	**74**	281.4	0.1	75
30	75	**75**	**367.8**	**0.0**	74	379.4	< 0.1	48	2702.3	23.9	**75**	392.2	**0.0**	75
35	75	**72**	**928.2**	**0.2**	72	969.5	**0.2**	48	2838.1	26.2	**72**	959.0	**0.2**	72
40	75	**65**	1365.5	**0.7**	**65**	1428.6	**0.7**	44	3051.2	29.7	**65**	**1331.6**	1.9	67
45	75	57	2336.8	1.3	57	2394.9	1.4	41	3462.2	28.9	**58**	**2212.3**	**1.1**	62
50	75	54	2569.6	2.2	54	2614.0	2.2	41	3480.5	32.2	**60**	**2016.5**	**1.6**	61
Total	675	622	867.0	**0.5**	621	907.3	**0.5**	479	2267.2	19.2	**629**	**805.9**	**0.5**	637

Results for `small(H15)` and `large(H15)` instances are shown in Table 5.10. Again, the performance of the algorithms is similar to the MCVRP-CFCS vari-

ant. For `small(H15)` instances, the `ThreeIndexDiscrete` approach can solve all instances to proven optimality and is very fast with an average computation time of 36.5 seconds compared to the other algorithms. The `large(H15)` instances are very hard to solve for this problem variant. Only one instance can be solved by the `BaP+TwoIndex` approach. Contrary to the MCVRP-CFCS variant, the `BaP` approach performs best here with two instances solved to proven optimality.

Overall, we recommend using the `ThreeIndexDiscrete` approach for `small(H15)` instances with a small number of nodes $|V|$ and the `ThreeIndexDiscrete`/`BaP+TwoIndex` approach for `mid-size(H18)` instances. For `large(H15)` instances, the `BaP` and `BaP+TwoIndex` approach perform best, most likely, because the higher number of available vehicles enforces symmetry issues of the `ThreeIndex` approach. Combining all results and, additionally, using the best found lower bound of the MCVRP-CFCS as lower bound, we can solve all 135 `small(H15)` instances, 637 of 675 `mid-size(H18)` instances and 2 of 27 `large(H15)` instances. Note that instance-by-instance results are shown in the Appendix.

Table 5.10: MCVRP-DFCS results for `small(H15)` and `large(H15)` instances grouped by the number of product types and the supply parameter.

Instance					ThreeIndexDiscrete (Henke *et al.*, 2018)			BaP			BaP+TwoIndex			ThreeIndexDiscrete/ BaP+TwoIndex			Total
class	$\|V\|$	ρ	s	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt
small(H15)	10	3	1	10	**10**	0.2	**0.0**	**10**	**0.1**	**0.0**	**10**	0.1	**0.0**	**10**	**0.1**	**0.0**	10
			2	10	**10**	**0.3**	**0.0**	**10**	1.2	**0.0**	**10**	1.0	**0.0**	**10**	**0.3**	**0.0**	10
			3	10	**10**	**0.5**	**0.0**	**10**	23.7	**0.0**	**10**	10.1	**0.0**	**10**	**0.5**	**0.0**	10
		6	1	15	**15**	**0.3**	**0.0**	**15**	2.3	**0.0**	**15**	1.9	**0.0**	**15**	2.3	**0.0**	15
			2	15	**15**	**1.1**	**0.0**	10	2607.2	1.0	12	1504.4	0.9	**15**	**1.1**	**0.0**	15
			3	15	**15**	**3.1**	**0.0**	6	4965.2	60.1	9	3499.3	21.4	**15**	**3.1**	**0.0**	15
		9	1	20	**20**	**16.6**	**0.0**	19	381.1	**0.0**	**20**	167.3	**0.0**	19	381.1	**0.0**	20
			2	20	**20**	**66.8**	**0.0**	8	4892.3	50.6	7	4886.6	28.8	**20**	**66.8**	**0.0**	20
			3	20	**20**	**159.2**	**0.0**	0	*TL*	71.0	1	6874.7	75.6	**20**	**159.2**	**0.0**	20
Total ($\|V\|$ = 10)				135	**135**	**36.5**	**0.0**	88	2691.4	24.8	94	2324.2	17.9	134	90.7	**0.0**	135
large(H15)	50	3	1	2	0	*TL*		**1**	**3946.0**		**1**	4905.4		**1**	4905.4		1
			2	2	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			3	2	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
		6	1	3	0	*TL*		**1**	**7095.8**		0	*TL*		0	*TL*		1
			2	3	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			3	3	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
		9	1	4	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			2	4	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
			3	4	0	*TL*		0	*TL*		0	*TL*		0	*TL*		0
Total ($\|V\|$ = 50)				27	0	*TL*		**2**	**6947.4**		1	7030.4		1	7030.4		2

5.7.6 Cost comparison between MCVRP-CFCS and MCVRP-DFCS

In this section, we compare the total cost of optimal solutions of continuously flexible compartment sizes with those of discretely flexible compartment sizes. We

consider unit compartment sizes $q^{\text{unit}} \in \{0.05Q, 0.1Q, 0.2Q, 0.5Q\}$. Moreover, the performance of the algorithms is compared for the different unit compartment sizes. For both the total cost comparison and computational study, we consider a subset of 27 `small(H15)` and 135 `mid-size(H18)` instances that consist of one instance for each $|V|$-ρ-C-s-combination.

For the total cost comparison, we only consider instances that are optimally solved for both MCVRP-FCS variants. Note that `large(H15)` instances are not taken into account because only a few instances are solved to proven optimality. Nevertheless, considering the MCVRP-DFCS with $q^{\text{unit}} = 0.1Q$, we observe that the total costs differ for 9 `large(H15)` instances and so far no `large(H15)` instance with identical total costs for both MCVRP-FCS variants is known.

Table 5.11 displays the total cost comparison of `small(H15)` and `mid-size(H18)` instances. The table entries have the following meaning:

#div: number of instances with different total cost, i.e., number of instances with $z_{con} \neq z_{dis}$;

red(%): the average reduction of the total cost in percent, i.e., the average of $100 \cdot (z_{dis} - z_{con})/z_{dis}$.

According to the #inst-columns in Table 5.11, the higher the unit compartment size q^{unit} the lower the number of instances that can be compared. The reason for this is that many instances of the MCVRP-DFCS are infeasible for a big unit compartment size because the number of vehicles is too low. However, the portion of instances with different total cost increases for larger unit compartment sizes. For example, the total cost of 70 of 154 instances differ for $q^{\text{unit}} = 0.05Q$ but the total cost of 55 of 64 instances differ for $q^{\text{unit}} = 0.5Q$. Moreover, the average reduction of the total cost increases significantly from less than one percent for $q^{\text{unit}} = 0.05Q$ to more than 16 percent for $q^{\text{unit}} = 0.5Q$.

We also observe that the number of instances with different total cost does not depend on the number of nodes. However, the supply parameter s impacts the cost savings of continuously flexible compartment sizes compared to discretely flexible compartment sizes. Cost savings are on average higher for instances with $s \in \{2, 3\}$. Moreover, cost savings and the number of instances with different total cost increase for instances with a higher number of product types. Note that this does not hold for the three `small(H15)`instances with $q^{\text{unit}} = 0.5Q$ and $\rho = 9$.

Table 5.12 displays the results of `ThreeIndexDiscrete`, `BaP`+`TwoIndex`, and `BaP` on `small(H15)` and `mid-size(H18)` instances with $q^{\text{unit}} \in \{0.05Q, 0.1Q, 0.2Q, 0.5Q\}$. Independent of the unit compartment size q^{unit}, the `ThreeIndexDiscrete`, `BaP`+`TwoIndex`, and `BaP` can solve about 100%, 65%, and 65% of `small(H15)` instances and 90%, 65%, and 30% of `mid-size(H18)` instances, respectively. Accordingly, we observe that the unit compartment

Table 5.11: Total cost comparison of `small(H15)` and `mid-size(H18)` instances.

Instance				$q^{\text{unit}} = 0.05Q$			$q^{\text{unit}} = 0.1Q$			$q^{\text{unit}} = 0.2Q$			$q^{\text{unit}} = 0.5Q$		
set	$\|V\|$	s	ρ	#inst	#div	red(%)	#inst	#div	red(%)	#inst	#div	red(%)	#inst	#div	red(%)
`small(H15)`	10	1	3–6	9	1	0.3	9	5	1.8	9	5	5.7	6	3	13.5
		2		9	1	0.1	9	3	1.3	9	4	8.0	6	4	21.3
		3		9	1	0.6	9	1	2.1	9	5	11.7	5	2	14.6
		1–3	3	6	0	0.0	6	1	0.5	6	3	1.7	6	4	15.0
			6	9	1	0.3	9	3	1.2	9	5	8.1	8	5	23.9
			9	12	2	0.5	12	5	2.7	12	6	12.1	3	0	0.0
Total (`small(H15)`)				27	3	0.3	27	9	1.7	27	14	8.5	17	9	16.6
`mid-size(H18)`	10	1–3	3–4	15	7	1.7	15	7	3.1	14	9	5.5	7	7	22.5
	15			15	9	1.0	15	11	2.4	13	11	4.9	7	7	10.9
	20			15	5	0.4	15	8	0.9	15	10	4.2	4	4	15.1
	25			15	8	0.5	15	10	1.1	15	12	3.7	8	7	12.6
	30			15	8	0.6	15	11	1.5	13	10	2.9	4	4	7.4
	35			14	8	1.1	13	11	2.0	15	11	3.8	4	4	12.4
	40			13	6	0.7	12	8	1.6	12	10	2.7	5	5	28.9
	45			12	8	1.2	11	9	1.8	9	8	2.8	6	6	16.8
	50			13	8	0.6	12	8	1.1	10	8	4.9	2	2	23.4
	10–50	1	3–4	44	23	0.6	43	30	1.4	39	34	3.3	13	12	9.2
		2		41	24	1.2	39	32	2.4	39	34	4.5	15	15	19.0
		3		42	20	0.8	41	21	1.4	38	21	4.2	19	19	19.0
		1–3	3	52	25	0.8	50	33	1.5	50	40	3.2	24	24	13.9
			4	75	42	0.9	73	50	1.9	66	49	4.6	23	22	18.8
Total (`mid-size(H18)`)				127	67	0.9	123	83	1.7	116	89	4.0	47	46	16.3

size q^{unit} does not impact the performance of the algorithms. Overall, the `ThreeIndexDiscrete` performs best with 510 of 546 solved instances for all unit compartment sizes q^{unit}.

5.7.7 Impact of the vehicle number

In this section, we compare the performance of the algorithms according to the number of vehicles. To this end, we generated new `small(veh)` and `mid-size(veh)` instances by varying the capacity Q and the number of vehicles m for a subset of 27 `small(H15)` and 135 `mid-size(H18)` instances. For each original instance, 7 new instances were generated by decreasing the capacity of the original instance by factors $0.3, 0.4, \ldots, 0.8, 0.9$. To guarantee feasibility, the number of vehicles m was determined by solving a corresponding bin-packing problem. The instances are online available at `https://logistik.bwl.uni-mainz.de/benchmarks/`.

Results are summarized in Table 5.13. For `ThreeIndex` and `ThreeIndexDiscrete`, the performance strongly depends on the number of vehicles: the higher the number of vehicles, the higher the average computation time. Moreover, the number of solved `mid-size(veh)` instances is about 120 for instances with an average number of vehicles $\bar{m} = 2.9$ but only about 20 for instances with $\bar{m} = 7.2$. For `BaP+TwoIndex`, the number of solved instances

Table 5.12: Comparison of `ThreeIndexDiscrete`, `BaP`+`TwoIndex`, and `BaP` results for the MCVRP-DFCS on `small(H15)` and `mid-size(H18)` instances with $q^{\text{unit}} \in \{0.05Q, 0.1Q, 0.2Q, 0.5Q\}$.

Instance			`ThreeIndexDiscrete` (Henke *et al.*, 2018)			`BaP`+`TwoIndex`			`BaP`		
set	q^{unit}	#inst	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap	#opt	time $\bar{T}$	gap
`small(H15)`	$0.05Q$	27	**27**	**32.6**	**0.0**	19	2162.6	19.8	18	2598.9	23.1
	$0.1Q$	27	**27**	**31.7**	**0.0**	18	2424.2	20.1	17	2785.6	22.7
	$0.2Q$	27	**27**	**34.6**	**0.0**	16	2953.4	35.3	17	2794.0	30.1
	$0.5Q$	17	**17**	**86.9**	**0.0**	12	2217.9	18.5	14	1478.6	7.4
Total (`small(H15)`)		98	**98**	**42.3**	**0.0**	65	2439.9	23.5	66	2459.6	21.5
`mid-size(H18)`	$0.05Q$	135	**123**	**908.1**	**0.4**	94	2351.0	19.3	39	5252.9	61.9
	$0.1Q$	135	**125**	**830.5**	**0.7**	94	2330.5	19.6	36	5394.3	62.8
	$0.2Q$	127	**114**	**1188.3**	**0.6**	72	3423.6	31.7	31	5589.5	66.1
	$0.5Q$	51	**50**	**513.4**	**0.0**	22	4259.4	51.1	16	5137.3	61.6
Total (`mid-size(H18)`)		448	**412**	**914.2**	**0.5**	282	2766.9	25.0	122	5365.0	63.2

also decreases for instances with a higher number of vehicles but still 83+47 of `mid-size(veh)` instances with $\bar{m}$ = 7.2 can be solved. For the `BaP`, the opposite is the case: the higher the number of vehicles, the higher the number of solved instances. For `mid-size(veh)`with $\bar{m}$ = 7.2, `BaP` can solve most instances in comparison with `ThreeIndex`/`ThreeIndexDiscrete` and `BaP`+`TwoIndex`. Overall, `ThreeIndex` and `ThreeIndexDiscrete` perform best for `small(veh)` and `BaP`+`TwoIndex` performs best for `mid-size(veh)` instances.

5.8 Conclusion

In this paper, we provided three new exact solution approaches for the multi-compartment vehicle routing problem with continuous flexible compartment sizes and two new exact solution approaches for the multi-compartment vehicle routing problem with discrete flexible compartment sizes. Computational tests have been conducted on benchmark instances from the literature. We identified that the performance of the algorithms strongly depends on the instance parameters.

For the MCVRP-CFCS and MCVRP-DFCS, a branch-and-cut algorithm based on a three-index formulation performs best for `small(H15)` instances with a low number of nodes. A combined algorithm of this branch-and-cut algorithm for instances with high supplies per customer and a two-stage approach consisting of a branch-price-and-cut and a branch-and-cut algorithm of a two-index formulation turned out to perform best for `mid-size(H18)` instances with a low number of vehicles. For the former type (MCVRP-CFCS), the algorithms can solve all

Table 5.13: Comparison of `ThreeIndex`/`ThreeIndexDiscrete`, `BaP`+`TwoIndex`, and `BaP` results for `small(veh)` and `mid-size(veh)` instances with a varying number of vehicles.

Instance				MCVRP-CFCS						MCVRP-DFCS					
				`ThreeIndex`		`BaP`+`TwoIndex`		`BaP`		`ThreeIndexDiscrete`		`BaP`+`TwoIndex`		`BaP`	
set	$\bar{Q}$	$\bar{m}$	#inst	#opt	time $\bar{T}$	#opt	time $\bar{T}$	#opt	time $\bar{T}$	#opt	time $\bar{T}$	#opt	time $\bar{T}$	#opt	time $\bar{T}$
`small(veh)`	300.0	5.5	27	22	1580.8	19	2138.8	**23**	**1211.8**	**15**	3606.7	13	**3246.1**	14	3928.6
	400.0	4.3	27	**26**	**568.6**	20	1933.2	23	1323.2	**22**	**1515.9**	13	3473.0	17	2734.3
	500.0	3.8	27	**27**	**141.9**	19	2137.0	24	999.0	**27**	**493.4**	14	3478.6	13	4009.8
	600.0	3.3	27	**27**	**58.4**	20	1889.6	23	1158.4	**27**	**130.1**	18	2417.2	18	2785.8
	700.0	3.1	27	**27**	**27.0**	21	1628.3	24	959.9	**27**	**50.6**	21	2018.1	17	2864.5
	800.0	2.9	27	**27**	**29.2**	20	2097.1	24	911.6	**27**	**42.4**	16	2979.5	15	3317.8
	900.0	2.6	27	**27**	**27.3**	20	1896.7	25	839.0	**27**	**52.1**	17	2744.2	16	3382.7
Total (`small(veh)`)	600.0	3.6	189	**183**	**347.6**	139	1960.1	166	1057.5	**172**	**841.6**	112	2908.1	110	3289.1
`mid-size(veh)`	674.8	7.2	135	22	6292.5	83	2908.1	**87**	**2842.3**	13	6630.2	47	**4201.5**	**55**	4517.0
	900.0	5.5	135	51	4943.8	91	**2460.5**	**92**	2761.5	28	5978.4	**67**	**3747.4**	56	4377.8
	1124.8	4.6	135	75	3681.2	**92**	**2434.2**	83	3055.4	50	4991.0	**63**	**3571.3**	50	4724.6
	1350.0	3.9	135	**95**	2700.6	89	**2522.2**	72	3657.6	**76**	3723.5	72	**3489.8**	43	5007.3
	1574.8	3.5	135	**110**	**1803.1**	99	2055.4	71	3650.4	**96**	**2597.6**	82	2749.1	47	5003.7
	1800.0	3.0	135	**121**	**1083.3**	95	2269.8	71	3696.9	**116**	**1451.3**	84	2880.1	50	4732.6
	2024.8	2.9	135	**123**	**895.4**	97	2181.6	61	4136.4	**120**	**1124.4**	93	2383.4	38	5296.0
Total (`mid-size(veh)`)	1349.9	4.4	945	597	3057.1	**646**	**2404.5**	537	3400.1	499	3785.2	**508**	**3289.0**	339	4808.4

`small(H15)` instances and `mid-size(H18)` instances with up to 30 nodes and over 80% of the `mid-size(H18)` instances with 50 nodes to optimality within two hours. Moreover, the two-stage approach consisting of the branch-price-and-cut and the branch-and-cut algorithm of the two-index formulation can solve 8 of 27 `large(H15)` instances. For the latter type (MCVRP-DFCS), the algorithms deliver new provably optimal solutions for 16 `mid-size(H18)` instances and 2 `large(H15)` instances.

A comparison of the total costs of both variants shows that the savings potential of using continuously flexible compartment sizes instead of discretely flexible compartment sizes depends on average on the number of supplies and the number of product types. For the MCVRP-DFCS, the performance of the algorithms does not depend on the unit compartments size. However, further experiments showed that the performance of the algorithms strongly depends on the number of vehicles for both problem variants.

Bibliography

Ahr, D. (2004). *Contributions to Multiple Postmen Problems*. Ph.d. dissertation, Department of Computer Science, Heidelberg University, Heidelberg, Germany.

Amor, H. B., Desrosiers, J., and de Carvalho, J. M. V. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Archetti, C., Bianchessi, N., and Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, **64**, 1–10.

Archetti, C., Campbell, A. M., and Speranza, M. G. (2016). Multicommodity vs. single-commodity routing. *Transportation Science*, **50**(2), 461–472.

Avella, P., Boccia, M., and Sforza, A. (2004). Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, **152**(1), 170–179.

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

Benantar, A., Ouafi, R., and Boukachour, J. (2016). A petrol station replenishment problem: new variant and formulation. *Logistics Research*, **9**(1).

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Brown, G. G. and Graves, G. W. (1981). Real-time dispatch of petroleum tank trucks. *Management Science*, **27**(1), 19–32.

Caramia, M. and Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, **40**(2), 130–143.

Chajakis, E. D. and Guignard, M. (2003). Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, **26**(1), 43–78.

Coelho, L. C. and Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research*, **242**(3), 854–864.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, **26**(1), 88–102.

Cornillier, F., Boctor, F. F., Laporte, G., and Renaud, J. (2008). An exact algorithm for the petrol station replenishment problem. *Journal of the Operational Research Society*, **59**(5), 607–615.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., and Vogel, U. (2010). Vehicle routing with compartments: applications, modelling and heuristics. *OR Spectrum*, **33**(4), 885–914.

Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2005). *Column Generation*. GERAD 25th anniversary series. Springer Science+Business Media Inc, Boston, MA.

Eshtehadi, R., Demir, E., and Huang, Y. (2020). Solving the vehicle routing problem with multi-compartment vehicles for city logistics. *Computers & Operations Research*, **115**, 104859.

Fagerholt, K. and Christiansen, M. (2000). A combined ship scheduling and allocation problem. *Journal of the Operational Research Society*, **51**(7), 834–842.

Fallahi, A. E., Prins, C., and Calvo, R. W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, **35**(5), 1725–1741.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2005). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, **106**(3), 491–511.

Goeke, D., Gschwind, T., and Schneider, M. (2019). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. *Discrete Applied Mathematics*, **264**, 43–61.

Goodson, J. C. (2015). A priori policy evaluation and cyclic-order-based simulated annealing for the multi-compartment vehicle routing problem with stochastic demands. *European Journal of Operational Research*, **241**(2), 361–369.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Henke, T. (2017). Multi-compartment vehicle routing problems a review and an extended classification. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, Magdeburg, Germany.

Henke, T. (2018). *Multi-compartment vehicle routing problems in the context of glass waste collection*. Ph.D. thesis, Otto-von-Guericke University Magdeburg, Magdeburg, Germany.

Henke, T., Speranza, M. G., and Wäscher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, **246**(3), 730–743.

Henke, T., Speranza, M. G., and Wäscher, G. (2018). A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. *Annals of Operations Research*, **275**(2), 321–338.

Hübner, A. and Ostermeier, M. (2019). A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*, **53**(1), 282–300.

Iori, M., Salazar-González, J.-J., and Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, **41**(2), 253–264.

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

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

Koch, H., Henke, T., and Wäscher, G. (2016). A Genetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes. FEMM Working Papers 160004, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management.

Lahyani, R., Coelho, L. C., Khemakhem, M., Laporte, G., and Semet, F. (2015). A multi-compartment vehicle routing problem arising in the collection of olive oil in tunisia. *Omega*, **51**, 1–10.

Laporte, G., Nobert, Y., and Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, **33**(5), 1050–1073.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Melechovskỳ, J. (2013). A variable neighborhood search for the selective multi-compartment vehicle routing problem with time windows. *Lecture notes in management science*, **5**, 159–166.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, **37**(11), 1886–1898.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2011). Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science*, **45**(3), 346–363.

Mirzaei, S. and Wøhlk, S. (2017). Erratum to: A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, **6**(2), 185–218.

Muyldermans, L. and Pang, G. (2010). On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research*, **206**(1), 93–103.

Oppen, J. and Løkketangen, A. (2008). A tabu search approach for the livestock collection problem. *Computers & Operations Research*, **35**(10), 3213–3229.

Oppen, J., Løkketangen, A., and Desrosiers, J. (2010). Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, **37**(7), 1308–1317.

Ostermeier, M., Martins, S., Amorim, P., and Hübner, A. (2018). Loading constraints for a multi-compartment vehicle routing problem. *OR Spectrum*, **40**(4), 997–1027.

Pirkwieser, S., Raidl, G. R., and Gottlieb, J. (2012). Improved packing and routing of vehicles with compartments. In *Computer Aided Systems Theory – EUROCAST 2011*, pages 392–399. Springer Berlin Heidelberg.

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., and Limbourg, S. (2014). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, **37**(2), 297–330.

Reed, M., Yiannakou, A., and Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, **15**, 169–176.

Tarjan, R. E. (1979). A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, **18**(2), 110–127.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Appendix

In this Appendix, we present instance-by-instance results. The entries in the Tables 5.14–5.16 have the following meaning:

$\|V\|$:	number of nodes;
ρ:	number of product types;
C:	number of compartments;
s:	supply parameter;
No.:	instance number;
UB:	upper bound; bold if $LB = UB$, i.e., optimality is proven;
LB:	lower bound; bold ditto;
div:	marked if divergent optimal objective values of MCVRP-CFCS and MCVRP-DFCS.

Table 5.14 displays the results for the `small(H15)` instances, Table 5.15 for the `mid-size(H18)` instances, and Table 5.16 for the `large(H15)` instances.

Table 5.14: Detailed results for the `small(H15)` instances.

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
10	3	2	1	1	**339.4**	**339.4**	**339.4**	**339.4**	
10	3	2	1	2	**370.8**	**370.8**	**370.8**	**370.8**	
10	3	2	1	3	**354.9**	**354.9**	**354.9**	**354.9**	
10	3	2	1	4	**347.7**	**347.7**	**347.7**	**347.7**	
10	3	2	1	5	**373.5**	**373.5**	**375.6**	**375.6**	×
10	3	2	2	1	**375.0**	**375.0**	**375.0**	**375.0**	
10	3	2	2	2	**517.3**	**517.3**	**517.3**	**517.3**	
10	3	2	2	3	**477.3**	**477.3**	**478.5**	**478.5**	×
10	3	2	2	4	**478.0**	**478.0**	**478.0**	**478.0**	
10	3	2	2	5	**496.1**	**496.1**	**496.1**	**496.1**	
10	3	2	3	1	**609.4**	**609.4**	**609.4**	**609.4**	
10	3	2	3	2	**622.7**	**622.7**	**622.7**	**622.7**	
10	3	2	3	3	**613.9**	**613.9**	**613.9**	**613.9**	
10	3	2	3	4	**630.3**	**630.3**	**630.3**	**630.3**	
10	3	2	3	5	**578.9**	**578.9**	**578.9**	**578.9**	
10	3	3	1	1	**341.6**	**341.6**	**352.6**	**352.6**	×
10	3	3	1	2	**337.6**	**337.6**	**350.1**	**350.1**	×
10	3	3	1	3	**273.3**	**273.3**	**297.3**	**297.3**	×
10	3	3	1	4	**355.2**	**355.2**	**355.2**	**355.2**	
10	3	3	1	5	**329.1**	**329.1**	**329.1**	**329.1**	
10	3	3	2	1	**358.2**	**358.2**	**358.2**	**358.2**	
10	3	3	2	2	**408.5**	**408.5**	**408.5**	**408.5**	
10	3	3	2	3	**333.1**	**333.1**	**338.7**	**338.7**	×
10	3	3	2	4	**337.5**	**337.5**	**337.5**	**337.5**	
10	3	3	2	5	**352.9**	**352.9**	**367.2**	**367.2**	×
10	3	3	3	1	**412.9**	**412.9**	**412.9**	**412.9**	
10	3	3	3	2	**305.7**	**305.7**	**305.7**	**305.7**	
10	3	3	3	3	**401.4**	**401.4**	**402.6**	**402.6**	×
10	3	3	3	4	**294.7**	**294.7**	**294.7**	**294.7**	
10	3	3	3	5	**339.8**	**339.8**	**339.8**	**339.8**	
10	6	2	1	1	**484.7**	**484.7**	**484.7**	**484.7**	
10	6	2	1	2	**560.1**	**560.1**	**560.1**	**560.1**	
10	6	2	1	3	**629.4**	**629.4**	**629.4**	**629.4**	
10	6	2	1	4	**508.9**	**508.9**	**508.9**	**508.9**	
10	6	2	1	5	**578.6**	**578.6**	**578.6**	**578.6**	
10	6	2	2	1	**753.7**	**753.7**	**753.7**	**753.7**	
10	6	2	2	2	**813.2**	**813.2**	**813.2**	**813.2**	
10	6	2	2	3	**911.7**	**911.7**	**911.7**	**911.7**	
10	6	2	2	4	**880.3**	**880.3**	**880.3**	**880.3**	
10	6	2	2	5	**611.0**	**611.0**	**611.0**	**611.0**	
10	6	2	3	1	**1062.5**	**1062.5**	**1062.5**	**1062.5**	
10	6	2	3	2	**911.7**	**911.7**	**911.7**	**911.7**	
10	6	2	3	3	**1045.3**	**1045.3**	**1045.3**	**1045.3**	
10	6	2	3	4	**1053.3**	**1053.3**	**1053.3**	**1053.3**	
10	6	2	3	5	**834.6**	**834.6**	**834.6**	**834.6**	
10	6	4	1	1	**319.9**	**319.9**	**335.0**	**335.0**	×
10	6	4	1	2	**391.1**	**391.1**	**391.1**	**391.1**	
10	6	4	1	3	**285.2**	**285.2**	**285.2**	**285.2**	
10	6	4	1	4	**389.5**	**389.5**	**391.4**	**391.4**	×

Continued on next column

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
10	6	4	1	5	**369.7**	**369.7**	**392.5**	**392.5**	×
10	6	4	2	1	**428.8**	**428.8**	**428.8**	**428.8**	
10	6	4	2	2	**531.5**	**531.5**	**531.5**	**531.5**	
10	6	4	2	3	**454.8**	**454.8**	**454.8**	**454.7**	
10	6	4	2	4	**499.3**	**499.3**	**501.1**	**501.1**	×
10	6	4	2	5	**381.1**	**381.1**	**381.1**	**381.1**	
10	6	4	3	1	**592.7**	**592.7**	**592.7**	**592.7**	
10	6	4	3	2	**697.0**	**697.0**	**697.0**	**697.0**	
10	6	4	3	3	**565.0**	**565.0**	**565.0**	**565.0**	
10	6	4	3	4	**489.3**	**489.3**	**489.3**	**489.3**	
10	6	4	3	5	**542.6**	**542.6**	**542.6**	**542.6**	
10	6	6	1	1	**395.6**	**395.6**	**405.7**	**405.7**	×
10	6	6	1	2	**318.2**	**318.2**	**318.2**	**318.2**	
10	6	6	1	3	**304.9**	**304.9**	**309.1**	**309.1**	×
10	6	6	1	4	**304.7**	**304.7**	**320.7**	**320.7**	×
10	6	6	1	5	**384.4**	**384.4**	**415.3**	**415.3**	×
10	6	6	2	1	**329.3**	**329.3**	**342.3**	**342.3**	×
10	6	6	2	2	**297.2**	**297.2**	**335.4**	**335.4**	×
10	6	6	2	3	**345.7**	**345.7**	**361.8**	**361.8**	×
10	6	6	2	4	**331.5**	**331.5**	**331.5**	**331.5**	
10	6	6	2	5	**333.2**	**333.2**	**358.4**	**358.4**	×
10	6	6	3	1	**304.3**	**304.3**	**304.3**	**304.3**	
10	6	6	3	2	**381.3**	**381.3**	**381.3**	**381.3**	
10	6	6	3	3	**338.3**	**338.3**	**338.3**	**338.3**	
10	6	6	3	4	**313.1**	**313.1**	**328.4**	**328.4**	×
10	6	6	3	5	**304.2**	**304.2**	**311.4**	**311.4**	×
10	9	2	1	1	**675.0**	**675.0**	**675.0**	**675.0**	
10	9	2	1	2	**566.8**	**566.8**	**566.8**	**566.8**	
10	9	2	1	3	**887.6**	**887.6**	**887.6**	**887.6**	
10	9	2	1	4	**781.3**	**781.3**	**781.3**	**781.3**	
10	9	2	1	5	**822.0**	**822.0**	**822.0**	**822.0**	
10	9	2	2	1	**1110.7**	**1110.7**	**1110.7**	**1110.7**	
10	9	2	2	2	**939.8**	**939.8**	**939.8**	**939.8**	
10	9	2	2	3	**1177.5**	**1177.5**	**1177.5**	**1177.5**	
10	9	2	2	4	**1134.2**	**1134.2**	**1134.2**	**1134.2**	
10	9	2	2	5	**1002.9**	**1002.9**	**1002.9**	**1002.9**	
10	9	2	3	1	**1145.5**	**1145.5**	**1145.5**	**1145.4**	
10	9	2	3	2	**1398.7**	**1398.7**	**1398.7**	**1398.6**	
10	9	2	3	3	**1417.8**	**1417.8**	**1417.8**	**1417.7**	
10	9	2	3	4	**1442.0**	**1442.0**	**1442.0**	**1441.8**	
10	9	2	3	5	**1395.3**	**1395.1**	**1395.3**	**1395.1**	
10	9	4	1	1	**430.6**	**430.6**	**430.6**	**430.6**	
10	9	4	1	2	**272.0**	**272.0**	**272.0**	**272.0**	
10	9	4	1	3	**414.3**	**414.3**	**425.3**	**425.3**	×
10	9	4	1	4	**395.1**	**395.1**	**395.1**	**395.1**	
10	9	4	1	5	**619.5**	**619.5**	**619.5**	**619.5**	
10	9	4	2	1	**705.2**	**705.2**	**705.2**	**705.1**	
10	9	4	2	2	**541.5**	**541.5**	**541.5**	**541.5**	
10	9	4	2	3	**688.2**	**688.2**	**688.2**	**688.2**	
10	9	4	2	4	**651.0**	**651.0**	**651.0**	**651.0**	
10	9	4	2	5	**586.4**	**586.4**	**588.0**	**587.9**	×
10	9	4	3	1	**893.1**	**893.0**	**893.1**	**893.0**	
10	9	4	3	2	**643.1**	**643.0**	**643.1**	**643.0**	

Continued on next column

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
10	9	4	3	3	**830.8**	**830.7**	**830.8**	**830.8**	
10	9	4	3	4	**841.6**	**841.5**	**841.6**	**841.5**	
10	9	4	3	5	**856.5**	**856.4**	**856.5**	**856.4**	
10	9	7	1	1	**307.1**	**307.1**	**318.6**	**318.6**	×
10	9	7	1	2	**376.6**	**376.6**	**382.1**	**382.1**	×
10	9	7	1	3	**384.7**	**384.7**	**400.2**	**400.2**	×
10	9	7	1	4	**388.9**	**388.9**	**388.9**	**388.9**	
10	9	7	1	5	**310.3**	**310.3**	**345.6**	**345.6**	×
10	9	7	2	1	**564.4**	**564.4**	**569.5**	**569.4**	×
10	9	7	2	2	**389.6**	**389.6**	**398.5**	**398.5**	×
10	9	7	2	3	**474.9**	**474.9**	**486.4**	**486.4**	×
10	9	7	2	4	**470.3**	**470.3**	**470.3**	**470.3**	
10	9	7	2	5	**488.2**	**488.2**	**495.5**	**495.5**	×
10	9	7	3	1	**543.3**	**543.3**	**543.3**	**543.3**	
10	9	7	3	2	**493.2**	**493.2**	**493.2**	**493.2**	
10	9	7	3	3	**593.4**	**593.4**	**593.4**	**593.4**	
10	9	7	3	4	**575.1**	**575.1**	**577.3**	**577.2**	×
10	9	7	3	5	**490.0**	**490.0**	**496.7**	**496.6**	×
10	9	9	1	1	**351.2**	**351.2**	**359.4**	**359.4**	×
10	9	9	1	2	**351.0**	**351.0**	**393.9**	**393.9**	×
10	9	9	1	3	**388.1**	**388.1**	**388.9**	**388.9**	×
10	9	9	1	4	**349.0**	**349.0**	**413.5**	**413.5**	×
10	9	9	1	5	**355.4**	**355.4**	**392.5**	**392.5**	×
10	9	9	2	1	**371.4**	**371.4**	**397.7**	**397.7**	×
10	9	9	2	2	**357.8**	**357.8**	**434.7**	**434.6**	×
10	9	9	2	3	**442.5**	**442.5**	**487.7**	**487.7**	×
10	9	9	2	4	**389.2**	**389.2**	**405.7**	**405.7**	×
10	9	9	2	5	**380.7**	**380.7**	**474.1**	**474.1**	×
10	9	9	3	1	**351.9**	**351.9**	**433.9**	**433.9**	×
10	9	9	3	2	**325.2**	**325.2**	**382.6**	**382.6**	×
10	9	9	3	3	**360.3**	**360.3**	**442.9**	**442.9**	×
10	9	9	3	4	**395.9**	**395.9**	**448.2**	**448.2**	×
10	9	9	3	5	**384.4**	**384.4**	**429.5**	**429.5**	×

Table 5.15: Detailed results for the `mid-size(H18)` instances.

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
10	3	2	1	1	**356.6**	**356.6**	**356.6**	**356.6**	
10	3	2	1	2	**356.0**	**356.0**	**408.4**	**408.4**	×
10	3	2	1	3	**391.7**	**391.7**	**391.7**	**391.7**	
10	3	2	1	4	**380.9**	**380.9**	**387.8**	**387.8**	×
10	3	2	1	5	**332.3**	**332.3**	**333.8**	**333.8**	×
10	3	2	2	1	**516.9**	**516.9**	**516.9**	**516.9**	
10	3	2	2	2	**625.0**	**625.0**	**634.2**	**634.2**	×
10	3	2	2	3	**509.5**	**509.5**	**531.8**	**531.8**	×
10	3	2	2	4	**427.1**	**427.1**	**432.4**	**432.4**	×
10	3	2	2	5	**444.1**	**444.1**	**452.6**	**452.6**	×
10	3	2	3	1	**723.6**	**723.6**	**723.6**	**723.6**	
10	3	2	3	2	**630.6**	**630.6**	**630.6**	**630.6**	
10	3	2	3	3	**674.4**	**674.4**	**674.4**	**674.4**	
10	3	2	3	4	**541.8**	**541.8**	**541.8**	**541.8**	
10	3	2	3	5	**546.5**	**546.5**	**566.3**	**566.3**	×
10	3	3	1	1	**348.8**	**348.8**	**351.0**	**351.0**	×
10	3	3	1	2	**297.5**	**297.5**	**297.5**	**297.5**	
10	3	3	1	3	**444.3**	**444.3**	**444.3**	**444.3**	
10	3	3	1	4	**330.6**	**330.6**	**340.3**	**340.3**	×
10	3	3	1	5	**347.0**	**347.0**	**448.4**	**448.4**	×
10	3	3	2	1	**346.6**	**346.6**	**440.8**	**440.8**	×
10	3	3	2	2	**408.9**	**408.9**	**430.3**	**430.3**	×
10	3	3	2	3	**379.4**	**379.4**	**387.3**	**387.3**	×
10	3	3	2	4	**301.2**	**301.2**	**313.5**	**313.5**	×
10	3	3	2	5	**456.2**	**456.2**	**466.1**	**466.1**	×
10	3	3	3	1	**349.2**	**349.2**	**354.5**	**354.5**	×
10	3	3	3	2	**355.6**	**355.6**	**379.1**	**379.1**	×
10	3	3	3	3	**384.0**	**384.0**	**393.6**	**393.6**	×
10	3	3	3	4	**269.3**	**269.3**	**269.3**	**269.3**	
10	3	3	3	5	**318.2**	**318.2**	**338.2**	**338.2**	×
10	4	2	1	1	**474.5**	**474.5**	**474.5**	**474.5**	
10	4	2	1	2	**411.6**	**411.6**	**411.6**	**411.6**	
10	4	2	1	3	**451.0**	**451.0**	**464.5**	**464.5**	×
10	4	2	1	4	**410.6**	**410.6**	**410.6**	**410.6**	
10	4	2	1	5	**444.1**	**444.1**	**444.1**	**444.1**	
10	4	2	2	1	**411.3**	**411.3**	**411.3**	**411.3**	
10	4	2	2	2	**640.3**	**640.3**	**640.3**	**640.3**	
10	4	2	2	3	**476.6**	**476.6**	**476.6**	**476.6**	
10	4	2	2	4	**516.3**	**516.3**	**517.6**	**517.6**	×
10	4	2	2	5	**602.3**	**602.3**	**619.5**	**619.5**	×
10	4	2	3	1	**728.1**	**728.1**	**728.1**	**728.1**	
10	4	2	3	2	**545.3**	**545.3**	**545.3**	**545.3**	
10	4	2	3	3	**591.4**	**591.4**	**591.4**	**591.4**	
10	4	2	3	4	**639.9**	**639.9**	**639.9**	**639.9**	
10	4	2	3	5	**638.7**	**638.7**	**638.7**	**638.7**	
10	4	3	1	1	**306.0**	**306.0**	**306.0**	**306.0**	
10	4	3	1	2	**370.1**	**370.1**	**403.5**	**403.5**	×
10	4	3	1	3	**390.8**	**390.8**	**390.8**	**390.8**	
10	4	3	1	4	**418.3**	**418.3**	**418.3**	**418.3**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
10	4	3	1	5	**410.6**	**410.6**	**426.4**	**426.4**	×
10	4	3	2	1	**452.0**	**452.0**	**488.3**	**488.3**	×
10	4	3	2	2	**327.5**	**327.5**	**327.5**	**327.5**	
10	4	3	2	3	**381.4**	**381.4**	**381.5**	**381.5**	×
10	4	3	2	4	**482.5**	**482.5**	**487.7**	**487.7**	×
10	4	3	2	5	**359.2**	**359.2**	**410.9**	**410.9**	×
10	4	3	3	1	**438.5**	**438.5**	**440.7**	**440.7**	×
10	4	3	3	2	**496.4**	**496.4**	**496.4**	**496.4**	
10	4	3	3	3	**534.3**	**534.3**	**535.6**	**535.6**	×
10	4	3	3	4	**574.8**	**574.8**	**574.8**	**574.8**	
10	4	3	3	5	**617.9**	**617.9**	**617.9**	**617.9**	
10	4	4	1	1	**328.3**	**328.3**	**328.3**	**328.3**	
10	4	4	1	2	**320.7**	**320.7**	**349.0**	**349.0**	×
10	4	4	1	3	**389.5**	**389.5**	**411.9**	**411.9**	×
10	4	4	1	4	**392.3**	**392.3**	**405.6**	**405.6**	×
10	4	4	1	5	**437.5**	**437.5**	**437.5**	**437.5**	
10	4	4	2	1	**351.2**	**351.2**	**401.6**	**401.6**	×
10	4	4	2	2	**294.0**	**294.0**	**374.9**	**374.9**	×
10	4	4	2	3	**376.7**	**376.7**	**392.9**	**392.9**	×
10	4	4	2	4	**265.3**	**265.3**	**303.2**	**303.2**	×
10	4	4	2	5	**321.9**	**321.9**	**393.4**	**393.4**	×
10	4	4	3	1	**367.2**	**367.2**	**379.6**	**379.6**	×
10	4	4	3	2	**379.7**	**379.7**	**423.7**	**423.7**	×
10	4	4	3	3	**330.1**	**330.1**	**341.9**	**341.9**	×
10	4	4	3	4	**378.1**	**378.1**	**378.1**	**378.1**	
10	4	4	3	5	**269.5**	**269.5**	**292.2**	**292.2**	×
15	3	2	1	1	**483.8**	**483.8**	**486.2**	**486.2**	×
15	3	2	1	2	**359.5**	**359.5**	**365.6**	**365.6**	×
15	3	2	1	3	**450.4**	**450.4**	**450.4**	**450.4**	
15	3	2	1	4	**543.0**	**543.0**	**561.7**	**561.7**	×
15	3	2	1	5	**547.8**	**547.8**	**548.3**	**548.3**	×
15	3	2	2	1	**494.6**	**494.6**	**496.4**	**496.4**	×
15	3	2	2	2	**630.3**	**630.3**	**630.3**	**630.3**	
15	3	2	2	3	**569.5**	**569.5**	**574.2**	**574.2**	×
15	3	2	2	4	**596.9**	**596.9**	**608.2**	**608.2**	×
15	3	2	2	5	**606.6**	**606.6**	**611.3**	**611.3**	×
15	3	2	3	1	**689.4**	**689.4**	**706.4**	**706.4**	×
15	3	2	3	2	**742.2**	**742.2**	**742.2**	**742.2**	
15	3	2	3	3	**607.0**	**607.0**	**607.0**	**607.0**	
15	3	2	3	4	**668.0**	**668.0**	**672.7**	**672.7**	×
15	3	2	3	5	**618.8**	**618.8**	**618.8**	**618.8**	
15	3	3	1	1	**467.4**	**467.4**	**474.4**	**474.4**	×
15	3	3	1	2	**448.5**	**448.5**	**454.9**	**454.9**	×
15	3	3	1	3	**361.5**	**361.5**	**361.5**	**361.5**	
15	3	3	1	4	**340.8**	**340.8**	**341.4**	**341.4**	×
15	3	3	1	5	**437.8**	**437.8**	**437.8**	**437.8**	
15	3	3	2	1	**451.3**	**451.3**	**483.2**	**483.2**	×
15	3	3	2	2	**392.6**	**392.6**	**398.5**	**398.5**	×
15	3	3	2	3	**371.7**	**371.7**	**419.3**	**419.3**	×
15	3	3	2	4	**401.5**	**401.5**	**408.3**	**408.3**	×
15	3	3	2	5	**408.4**	**408.4**	**411.7**	**411.7**	×
15	3	3	3	1	**356.1**	**356.1**	**370.2**	**370.2**	×
15	3	3	3	2	**457.7**	**457.7**	**457.7**	**457.7**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
15	3	3	3	3	**450.2**	**450.2**	**453.0**	**453.0**	×
15	3	3	3	4	**425.9**	**425.9**	**428.6**	**428.6**	×
15	3	3	3	5	**423.4**	**423.4**	**459.2**	**459.2**	×
15	4	2	1	1	**526.4**	**526.4**	**526.4**	**526.4**	
15	4	2	1	2	**413.2**	**413.2**	**413.2**	**413.2**	
15	4	2	1	3	**476.7**	**476.7**	**476.7**	**476.7**	
15	4	2	1	4	**586.9**	**586.9**	**615.9**	**615.9**	×
15	4	2	1	5	**570.8**	**570.8**	**570.8**	**570.8**	
15	4	2	2	1	**614.4**	**614.4**	**614.4**	**614.4**	
15	4	2	2	2	**562.0**	**562.0**	**566.4**	**566.4**	×
15	4	2	2	3	**661.3**	**661.3**	**661.3**	**661.3**	
15	4	2	2	4	**614.6**	**614.6**	**614.6**	**614.6**	
15	4	2	2	5	**725.8**	**725.8**	**725.8**	**725.8**	
15	4	2	3	1	**806.2**	**806.2**	**806.2**	**806.2**	
15	4	2	3	2	**641.1**	**641.1**	**643.7**	**643.7**	×
15	4	2	3	3	**694.6**	**694.6**	**696.5**	**696.5**	×
15	4	2	3	4	**820.9**	**820.9**	**820.9**	**820.9**	
15	4	2	3	5	**650.2**	**650.2**	**650.2**	**650.2**	
15	4	3	1	1	**490.4**	**490.4**	**502.4**	**502.4**	×
15	4	3	1	2	**516.1**	**516.1**	**516.1**	**516.1**	
15	4	3	1	3	**394.2**	**394.2**	**394.9**	**394.9**	×
15	4	3	1	4	**387.5**	**387.5**	**400.4**	**400.4**	×
15	4	3	1	5	**420.5**	**420.5**	**420.5**	**420.5**	
15	4	3	2	1	**509.8**	**509.8**	**516.7**	**516.7**	×
15	4	3	2	2	**476.6**	**476.6**	**490.0**	**490.0**	×
15	4	3	2	3	**488.6**	**488.6**	**496.6**	**496.6**	×
15	4	3	2	4	**433.1**	**433.1**	**502.4**	**502.4**	×
15	4	3	2	5	**457.6**	**457.6**	**462.0**	**462.0**	×
15	4	3	3	1	**740.9**	**740.9**	**740.9**	**740.9**	
15	4	3	3	2	**530.2**	**530.2**	**545.4**	**545.4**	×
15	4	3	3	3	**641.9**	**641.9**	**641.9**	**641.9**	
15	4	3	3	4	**430.3**	**430.3**	**430.3**	**430.3**	
15	4	3	3	5	**646.8**	**646.8**	**646.8**	**646.8**	
15	4	4	1	1	**360.4**	**360.4**	**380.6**	**380.6**	×
15	4	4	1	2	**435.3**	**435.3**	**508.5**	**508.5**	×
15	4	4	1	3	**445.5**	**445.5**	**481.0**	**481.0**	×
15	4	4	1	4	**448.5**	**448.5**	**448.5**	**448.5**	
15	4	4	1	5	**373.0**	**373.0**	**374.8**	**374.8**	×
15	4	4	2	1	**440.3**	**440.3**	**462.5**	**462.5**	×
15	4	4	2	2	**432.5**	**432.5**	**476.2**	**476.2**	×
15	4	4	2	3	**410.8**	**410.8**	**410.8**	**410.8**	
15	4	4	2	4	**409.5**	**409.5**	**437.0**	**437.0**	×
15	4	4	2	5	**412.0**	**412.0**	**437.0**	**437.0**	×
15	4	4	3	1	**415.2**	**415.2**	**444.7**	**444.7**	×
15	4	4	3	2	**432.2**	**432.2**	**432.2**	**432.2**	
15	4	4	3	3	**390.9**	**390.9**	**404.1**	**404.1**	×
15	4	4	3	4	**484.6**	**484.6**	**538.9**	**538.9**	×
15	4	4	3	5	**493.1**	**493.1**	**506.5**	**506.5**	×
20	3	2	1	1	**493.6**	**493.6**	**493.6**	**493.6**	
20	3	2	1	2	**448.0**	**448.0**	**448.0**	**448.0**	
20	3	2	1	3	**576.2**	**576.2**	**576.2**	**576.2**	
20	3	2	1	4	**474.8**	**474.8**	**476.3**	**476.3**	×
20	3	2	1	5	**522.7**	**522.7**	**530.3**	**530.3**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
20	3	2	2	1	**632.9**	**632.9**	**633.7**	**633.7**	×
20	3	2	2	2	**593.2**	**593.2**	**593.2**	**593.2**	
20	3	2	2	3	**677.5**	**677.5**	**681.7**	**681.7**	×
20	3	2	2	4	**532.5**	**532.5**	**532.5**	**532.5**	
20	3	2	2	5	**586.1**	**586.1**	**587.5**	**587.5**	×
20	3	2	3	1	**762.9**	**762.9**	**762.9**	**762.9**	
20	3	2	3	2	**640.8**	**640.8**	**640.8**	**640.8**	
20	3	2	3	3	**869.0**	**869.0**	**869.0**	**869.0**	
20	3	2	3	4	**877.9**	**877.9**	**877.9**	**877.9**	
20	3	2	3	5	**789.8**	**789.8**	**793.1**	**793.1**	×
20	3	3	1	1	**462.9**	**462.9**	**462.9**	**462.9**	
20	3	3	1	2	**464.4**	**464.4**	**466.0**	**466.0**	×
20	3	3	1	3	**416.4**	**416.4**	**424.8**	**424.8**	×
20	3	3	1	4	**400.5**	**400.5**	**400.5**	**400.5**	
20	3	3	1	5	**427.5**	**427.5**	**446.4**	**446.4**	×
20	3	3	2	1	**434.6**	**434.6**	**443.2**	**443.2**	×
20	3	3	2	2	**424.0**	**424.0**	**429.9**	**429.9**	×
20	3	3	2	3	**472.1**	**472.1**	**493.4**	**493.4**	×
20	3	3	2	4	**400.3**	**400.3**	**400.3**	**400.3**	
20	3	3	2	5	**433.9**	**433.9**	**466.0**	**466.0**	×
20	3	3	3	1	**477.2**	**477.2**	**477.2**	**477.2**	
20	3	3	3	2	**467.6**	**467.6**	**479.1**	**479.1**	×
20	3	3	3	3	**375.8**	**375.8**	**393.8**	**393.8**	×
20	3	3	3	4	**465.8**	**465.8**	**467.1**	**467.1**	×
20	3	3	3	5	**419.2**	**419.2**	**419.2**	**419.2**	
20	4	2	1	1	**565.2**	**565.2**	**565.2**	**565.2**	
20	4	2	1	2	**575.8**	**575.8**	**585.6**	**585.6**	×
20	4	2	1	3	**485.2**	**485.2**	**485.2**	**485.2**	
20	4	2	1	4	**488.3**	**488.3**	**488.3**	**488.3**	
20	4	2	1	5	**639.9**	**639.9**	**639.9**	**639.9**	
20	4	2	2	1	**716.8**	**716.8**	**716.8**	**716.8**	
20	4	2	2	2	**712.0**	**712.0**	**712.0**	**712.0**	
20	4	2	2	3	**626.7**	**626.7**	**627.7**	**627.7**	×
20	4	2	2	4	**613.4**	**613.4**	**619.7**	**619.7**	×
20	4	2	2	5	**688.7**	**688.7**	**688.7**	**688.7**	
20	4	2	3	1	**822.2**	**822.2**	**823.2**	**823.2**	×
20	4	2	3	2	**688.0**	**688.0**	**688.0**	**688.0**	
20	4	2	3	3	**833.4**	**833.4**	**833.4**	**833.4**	
20	4	2	3	4	**895.7**	**895.7**	**895.7**	**895.7**	
20	4	2	3	5	**812.3**	**812.3**	**812.3**	**812.3**	
20	4	3	1	1	**512.7**	**512.7**	**519.0**	**519.0**	×
20	4	3	1	2	**403.1**	**403.1**	**446.9**	**446.9**	×
20	4	3	1	3	**471.9**	**471.9**	**471.9**	**471.9**	
20	4	3	1	4	**480.6**	**480.6**	**480.6**	**480.6**	
20	4	3	1	5	**442.9**	**442.9**	**451.9**	**451.9**	×
20	4	3	2	1	**521.2**	**521.2**	**525.2**	**525.2**	×
20	4	3	2	2	**527.3**	**527.3**	**527.6**	**527.6**	×
20	4	3	2	3	**576.2**	**576.2**	**582.4**	**582.4**	×
20	4	3	2	4	**608.3**	**608.3**	**640.2**	**640.2**	×
20	4	3	2	5	**514.2**	**514.2**	**514.2**	**514.2**	
20	4	3	3	1	**614.4**	**614.4**	**614.4**	**614.4**	
20	4	3	3	2	**758.3**	**758.3**	**771.2**	**771.2**	×
20	4	3	3	3	**745.3**	**745.3**	**758.6**	**758.6**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
20	4	3	3	4	**893.3**	**893.3**	**893.3**	**893.3**	
20	4	3	3	5	**674.7**	**674.7**	**675.0**	**675.0**	×
20	4	4	1	1	**458.0**	**458.0**	**489.9**	**489.9**	×
20	4	4	1	2	**468.1**	**468.1**	**484.1**	**484.1**	×
20	4	4	1	3	**452.7**	**452.7**	**452.7**	**452.7**	
20	4	4	1	4	**508.4**	**508.4**	**515.4**	**515.4**	×
20	4	4	1	5	**429.1**	**429.1**	**548.1**	**548.1**	×
20	4	4	2	1	**448.6**	**448.6**	**449.3**	**449.3**	×
20	4	4	2	2	**436.9**	**436.9**	**487.0**	**487.0**	×
20	4	4	2	3	**523.3**	**523.3**	**542.0**	**542.0**	×
20	4	4	2	4	**429.0**	**429.0**	**438.0**	**438.0**	×
20	4	4	2	5	**441.6**	**441.6**	**444.8**	**444.8**	×
20	4	4	3	1	**444.6**	**444.6**	**459.0**	**459.0**	×
20	4	4	3	2	**467.7**	**467.7**	**486.3**	**486.3**	×
20	4	4	3	3	**482.0**	**482.0**	**502.5**	**502.5**	×
20	4	4	3	4	**442.5**	**442.5**	**462.1**	**462.1**	×
20	4	4	3	5	**480.7**	**480.7**	**485.8**	**485.8**	×
25	3	2	1	1	**592.1**	**592.1**	**592.1**	**592.1**	
25	3	2	1	2	**585.9**	**585.9**	**585.9**	**585.9**	
25	3	2	1	3	**488.3**	**488.3**	**488.3**	**488.3**	
25	3	2	1	4	**593.9**	**593.9**	**596.0**	**596.0**	×
25	3	2	1	5	**511.9**	**511.9**	**517.4**	**517.4**	×
25	3	2	2	1	**637.1**	**637.1**	**637.8**	**637.8**	×
25	3	2	2	2	**733.5**	**733.5**	**734.1**	**734.1**	×
25	3	2	2	3	**732.4**	**732.4**	**732.4**	**732.4**	
25	3	2	2	4	**735.3**	**735.3**	**735.3**	**735.3**	
25	3	2	2	5	**576.4**	**576.4**	**586.4**	**586.4**	×
25	3	2	3	1	**792.8**	**792.8**	**792.8**	**792.8**	
25	3	2	3	2	**816.8**	**816.8**	**817.4**	**817.4**	×
25	3	2	3	3	**885.6**	**885.6**	**885.6**	**885.6**	
25	3	2	3	4	**895.8**	**895.8**	**923.5**	**923.5**	×
25	3	2	3	5	**887.5**	**887.5**	**887.5**	**887.5**	
25	3	3	1	1	**494.1**	**494.1**	**494.8**	**494.8**	×
25	3	3	1	2	**473.5**	**473.5**	**558.1**	**558.1**	×
25	3	3	1	3	**474.7**	**474.7**	**476.3**	**476.3**	×
25	3	3	1	4	**472.1**	**472.1**	**485.8**	**485.8**	×
25	3	3	1	5	**497.0**	**497.0**	**508.4**	**508.4**	×
25	3	3	2	1	**501.9**	**501.9**	**513.7**	**513.7**	×
25	3	3	2	2	**490.3**	**490.3**	**508.3**	**508.3**	×
25	3	3	2	3	**534.2**	**534.2**	**549.4**	**549.4**	×
25	3	3	2	4	**442.5**	**442.5**	**444.7**	**444.7**	×
25	3	3	2	5	**489.3**	**489.3**	**501.8**	**501.8**	×
25	3	3	3	1	**435.5**	**435.5**	**435.5**	**435.5**	
25	3	3	3	2	**506.8**	**506.8**	**521.5**	**521.5**	×
25	3	3	3	3	**473.9**	**473.9**	**474.6**	**474.6**	×
25	3	3	3	4	**454.9**	**454.9**	**470.9**	**470.9**	×
25	3	3	3	5	**515.9**	**515.9**	**531.2**	**531.2**	×
25	4	2	1	1	**616.1**	**616.1**	**616.1**	**616.1**	
25	4	2	1	2	**724.1**	**724.1**	**724.1**	**724.1**	
25	4	2	1	3	**613.5**	**613.5**	**613.5**	**613.5**	
25	4	2	1	4	**555.7**	**555.7**	**555.7**	**555.7**	
25	4	2	1	5	**608.1**	**608.1**	**614.2**	**614.2**	×
25	4	2	2	1	**773.3**	**773.3**	**773.3**	**773.3**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
25	4	2	2	2	**743.8**	**743.8**	**746.1**	**746.1**	×
25	4	2	2	3	**834.6**	**834.6**	**843.9**	**843.9**	×
25	4	2	2	4	**937.5**	**937.5**	**941.5**	**941.5**	×
25	4	2	2	5	**880.6**	**880.6**	**884.3**	**884.3**	×
25	4	2	3	1	**919.0**	**919.0**	**927.2**	**927.2**	×
25	4	2	3	2	**884.9**	**884.9**	**884.9**	**884.9**	
25	4	2	3	3	**860.7**	**860.7**	**860.7**	**860.7**	
25	4	2	3	4	**798.0**	**798.0**	**798.0**	**798.0**	
25	4	2	3	5	**895.7**	**895.7**	**895.7**	**895.7**	
25	4	3	1	1	**544.5**	**544.5**	**545.5**	**545.5**	×
25	4	3	1	2	**492.3**	**492.3**	**522.9**	**522.9**	×
25	4	3	1	3	**545.4**	**545.4**	**545.4**	**545.4**	
25	4	3	1	4	**517.8**	**517.8**	**520.8**	**520.8**	×
25	4	3	1	5	**543.3**	**543.3**	**543.3**	**543.3**	
25	4	3	2	1	**557.7**	**557.7**	**567.6**	**567.6**	×
25	4	3	2	2	**677.2**	**677.2**	**687.5**	**687.5**	×
25	4	3	2	3	**602.5**	**602.5**	**625.2**	**625.2**	×
25	4	3	2	4	**623.2**	**623.2**	**623.2**	**623.2**	
25	4	3	2	5	**464.8**	**464.8**	**504.3**	**504.3**	×
25	4	3	3	1	**811.8**	**811.8**	**825.2**	**825.2**	×
25	4	3	3	2	**826.6**	**826.6**	**826.6**	**826.6**	
25	4	3	3	3	**793.6**	**793.6**	**800.1**	**800.1**	×
25	4	3	3	4	**774.4**	**774.4**	**774.4**	**774.4**	
25	4	3	3	5	**789.3**	**789.3**	**789.3**	**789.3**	
25	4	4	1	1	**483.1**	**483.1**	**495.9**	**495.9**	×
25	4	4	1	2	**460.4**	**460.4**	**472.1**	**472.1**	×
25	4	4	1	3	**509.8**	**509.8**	**534.3**	**534.3**	×
25	4	4	1	4	**434.2**	**434.2**	**499.5**	**499.5**	×
25	4	4	1	5	**495.8**	**495.8**	**510.8**	**510.8**	×
25	4	4	2	1	**513.3**	**513.3**	**519.6**	**519.6**	×
25	4	4	2	2	**477.2**	**477.2**	**483.1**	**483.1**	×
25	4	4	2	3	**521.1**	**521.1**	**521.1**	**521.1**	
25	4	4	2	4	**432.0**	**432.0**	**436.4**	**436.4**	×
25	4	4	2	5	**532.8**	**532.8**	**539.7**	**539.7**	×
25	4	4	3	1	**512.5**	**512.5**	**542.7**	**542.7**	×
25	4	4	3	2	**492.2**	**492.2**	**501.8**	**501.8**	×
25	4	4	3	3	**553.7**	**553.7**	**561.4**	**561.4**	×
25	4	4	3	4	**521.0**	**521.0**	**527.9**	**527.9**	×
25	4	4	3	5	**445.9**	**445.9**	**457.8**	**457.8**	×
30	3	2	1	1	**552.6**	**552.6**	**563.9**	**563.9**	×
30	3	2	1	2	**559.6**	**559.6**	**559.6**	**559.6**	
30	3	2	1	3	**588.5**	**588.5**	**604.8**	**604.8**	×
30	3	2	1	4	**534.2**	**534.2**	**534.2**	**534.2**	
30	3	2	1	5	**615.6**	**615.6**	**620.5**	**620.5**	×
30	3	2	2	1	**716.7**	**716.7**	**717.1**	**717.1**	×
30	3	2	2	2	**774.3**	**774.3**	**778.4**	**778.4**	×
30	3	2	2	3	**749.7**	**749.7**	**749.7**	**749.7**	
30	3	2	2	4	**755.0**	**755.0**	**758.0**	**758.0**	×
30	3	2	2	5	**758.1**	**758.1**	**758.1**	**758.1**	
30	3	2	3	1	**938.7**	**938.7**	**938.7**	**938.7**	
30	3	2	3	2	**935.1**	**935.1**	**935.1**	**935.1**	
30	3	2	3	3	**897.3**	**897.3**	**897.8**	**897.8**	×
30	3	2	3	4	**908.9**	**908.9**	**909.7**	**909.7**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
30	3	2	3	5	**955.2**	**955.2**	**955.2**	**955.2**	
30	3	3	1	1	**538.0**	**538.0**	**548.7**	**548.7**	×
30	3	3	1	2	**499.7**	**499.7**	**522.3**	**522.3**	×
30	3	3	1	3	**528.2**	**528.2**	**528.2**	**528.2**	
30	3	3	1	4	**531.9**	**531.9**	**540.1**	**540.1**	×
30	3	3	1	5	**439.9**	**439.9**	**457.1**	**457.1**	×
30	3	3	2	1	**482.0**	**482.0**	**482.3**	**482.3**	×
30	3	3	2	2	**596.8**	**596.8**	**608.0**	**608.0**	×
30	3	3	2	3	**540.9**	**540.9**	**570.6**	**570.6**	×
30	3	3	2	4	**472.8**	**472.8**	**486.5**	**486.5**	×
30	3	3	2	5	**512.5**	**512.5**	**522.5**	**522.5**	×
30	3	3	3	1	**445.4**	**445.4**	**445.4**	**445.4**	
30	3	3	3	2	**554.2**	**554.2**	**562.0**	**562.0**	×
30	3	3	3	3	**494.2**	**494.2**	**518.8**	**518.8**	×
30	3	3	3	4	**484.4**	**484.4**	**499.1**	**499.1**	×
30	3	3	3	5	**565.5**	**565.5**	**565.5**	**565.5**	
30	4	2	1	1	**535.2**	**535.2**	**555.3**	**555.3**	×
30	4	2	1	2	**557.0**	**557.0**	**557.8**	**557.8**	×
30	4	2	1	3	**703.2**	**703.2**	**703.2**	**703.2**	
30	4	2	1	4	**639.6**	**639.6**	**639.6**	**639.6**	
30	4	2	1	5	**692.6**	**692.6**	**692.6**	**692.6**	
30	4	2	2	1	**889.9**	**889.9**	**892.5**	**892.5**	×
30	4	2	2	2	**904.5**	**904.5**	**904.5**	**904.5**	
30	4	2	2	3	**847.3**	**847.3**	**852.5**	**852.5**	×
30	4	2	2	4	**869.9**	**869.9**	**869.9**	**869.9**	
30	4	2	2	5	**798.5**	**798.5**	**798.5**	**798.5**	
30	4	2	3	1	**966.8**	**966.8**	**966.8**	**966.8**	
30	4	2	3	2	**898.5**	**898.5**	**898.5**	**898.5**	
30	4	2	3	3	**1032.2**	**1032.2**	**1032.2**	**1032.2**	
30	4	2	3	4	**1017.4**	**1017.4**	**1017.4**	**1017.4**	
30	4	2	3	5	**954.3**	**954.3**	**954.3**	**954.3**	
30	4	3	1	1	**584.5**	**584.5**	**592.9**	**592.9**	×
30	4	3	1	2	**512.4**	**512.4**	**521.8**	**521.8**	×
30	4	3	1	3	**564.3**	**564.3**	**590.1**	**590.1**	×
30	4	3	1	4	**573.2**	**573.2**	**598.8**	**598.8**	×
30	4	3	1	5	**549.1**	**549.1**	**552.0**	**552.0**	×
30	4	3	2	1	**755.7**	**755.7**	**757.7**	**757.7**	×
30	4	3	2	2	**698.6**	**698.6**	**724.4**	**724.4**	×
30	4	3	2	3	**553.9**	**553.9**	**572.6**	**572.6**	×
30	4	3	2	4	**634.1**	**634.1**	**650.6**	**650.6**	×
30	4	3	2	5	**663.1**	**663.1**	**669.7**	**669.7**	×
30	4	3	3	1	**768.5**	**768.5**	**768.5**	**768.5**	
30	4	3	3	2	**829.8**	**829.8**	**829.8**	**829.8**	
30	4	3	3	3	**761.1**	**761.1**	**761.1**	**761.1**	
30	4	3	3	4	**877.8**	**877.8**	**877.8**	**877.8**	
30	4	3	3	5	**840.7**	**840.7**	**840.7**	**840.7**	
30	4	4	1	1	**500.8**	**500.8**	**517.0**	**517.0**	×
30	4	4	1	2	**501.9**	**501.9**	**539.2**	**539.2**	×
30	4	4	1	3	**543.0**	**543.0**	**567.4**	**567.4**	×
30	4	4	1	4	**524.2**	**524.2**	**539.3**	**539.3**	×
30	4	4	1	5	**534.4**	**534.4**	**552.0**	**552.0**	×
30	4	4	2	1	**509.3**	**509.3**	**531.5**	**531.5**	×
30	4	4	2	2	**490.2**	**490.2**	**500.0**	**500.0**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
30	4	4	2	3	**489.1**	**489.1**	**491.7**	**491.7**	×
30	4	4	2	4	**539.6**	**539.6**	**539.6**	**539.6**	
30	4	4	2	5	**557.0**	**557.0**	**567.4**	**567.4**	×
30	4	4	3	1	**549.3**	**549.3**	**577.4**	**577.4**	×
30	4	4	3	2	**481.8**	**481.8**	**508.8**	**508.8**	×
30	4	4	3	3	**529.0**	**529.0**	**539.0**	**539.0**	×
30	4	4	3	4	**451.1**	**451.1**	**473.8**	**473.8**	×
30	4	4	3	5	**496.5**	**496.5**	**501.1**	**501.1**	×
35	3	2	1	1	**622.2**	**622.2**	**628.2**	**628.2**	×
35	3	2	1	2	**640.4**	**640.4**	**647.2**	**647.2**	×
35	3	2	1	3	**671.2**	**671.2**	**674.2**	**674.2**	×
35	3	2	1	4	**560.4**	**560.4**	**560.4**	**560.4**	
35	3	2	1	5	**552.4**	**552.4**	**554.6**	**554.6**	×
35	3	2	2	1	**742.1**	**742.1**	**742.1**	**742.1**	
35	3	2	2	2	**783.0**	**783.0**	**784.4**	**784.4**	×
35	3	2	2	3	**831.1**	**831.1**	**836.2**	**836.2**	×
35	3	2	2	4	**833.1**	**833.1**	**854.2**	**854.2**	×
35	3	2	2	5	**792.7**	**792.7**	**793.6**	**793.6**	×
35	3	2	3	1	**1072.6**	**1072.6**	1082.7	1063.6	
35	3	2	3	2	**963.1**	**963.1**	**963.1**	**963.1**	
35	3	2	3	3	**1030.3**	**1030.3**	**1035.0**	**1035.0**	×
35	3	2	3	4	**1055.0**	**1055.0**	**1055.0**	**1055.0**	
35	3	2	3	5	**938.1**	**938.1**	**938.1**	**938.1**	
35	3	3	1	1	**555.1**	**555.1**	**565.4**	**565.4**	×
35	3	3	1	2	**539.6**	**539.6**	**547.9**	**547.9**	×
35	3	3	1	3	**556.7**	**556.7**	**563.7**	**563.7**	×
35	3	3	1	4	**527.8**	**527.8**	**557.1**	**557.1**	×
35	3	3	1	5	**528.5**	**528.5**	**549.6**	**549.6**	×
35	3	3	2	1	**533.4**	**533.4**	**543.4**	**543.4**	×
35	3	3	2	2	**544.8**	**544.8**	**548.1**	**548.1**	×
35	3	3	2	3	**572.4**	**572.4**	**572.4**	**572.4**	
35	3	3	2	4	**494.4**	**494.4**	**506.1**	**506.1**	×
35	3	3	2	5	**599.9**	**599.9**	**602.4**	**602.4**	×
35	3	3	3	1	**531.8**	**531.8**	**574.4**	**574.4**	×
35	3	3	3	2	**501.7**	**501.7**	**501.7**	**501.7**	
35	3	3	3	3	**519.7**	**519.7**	**521.4**	**521.4**	×
35	3	3	3	4	**542.0**	**542.0**	**554.7**	**554.7**	×
35	3	3	3	5	**530.5**	**530.5**	**532.8**	**532.8**	×
35	4	2	1	1	**665.1**	**665.1**	**669.1**	**669.1**	×
35	4	2	1	2	**686.2**	**686.2**	**686.2**	**686.2**	
35	4	2	1	3	**754.7**	**754.7**	**754.7**	**754.7**	
35	4	2	1	4	**687.8**	**687.8**	**687.8**	**687.8**	
35	4	2	1	5	**687.3**	**687.3**	**690.6**	**690.6**	×
35	4	2	2	1	**1057.9**	**1057.9**	**1057.9**	**1057.9**	
35	4	2	2	2	**856.5**	**856.5**	**856.5**	**856.5**	
35	4	2	2	3	**906.7**	**906.7**	**910.1**	**910.1**	×
35	4	2	2	4	**986.7**	**986.7**	**995.5**	**995.5**	×
35	4	2	2	5	1054.0	1036.7	1056.4	1025.2	
35	4	2	3	1	**1048.6**	**1048.6**	**1048.6**	**1048.6**	
35	4	2	3	2	**954.5**	**954.5**	**954.5**	**954.5**	
35	4	2	3	3	**1049.3**	**1049.3**	**1056.6**	**1056.6**	×
35	4	2	3	4	**1088.6**	**1088.6**	**1088.6**	**1088.6**	
35	4	2	3	5	**990.8**	**990.8**	**999.9**	**999.9**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
35	4	3	1	1	**583.0**	**583.0**	**601.2**	**601.2**	×
35	4	3	1	2	**624.3**	**624.3**	**636.0**	**636.0**	×
35	4	3	1	3	**549.3**	**549.3**	**570.6**	**570.6**	×
35	4	3	1	4	**556.1**	**556.1**	**559.6**	**559.6**	×
35	4	3	1	5	**595.3**	**595.3**	**595.3**	**595.3**	
35	4	3	2	1	**733.1**	**733.1**	**734.1**	**734.1**	×
35	4	3	2	2	**694.1**	**694.1**	**731.1**	**731.1**	×
35	4	3	2	3	**757.0**	**757.0**	**759.6**	**759.6**	×
35	4	3	2	4	**732.5**	**732.5**	**737.2**	**737.2**	×
35	4	3	2	5	**702.7**	**702.7**	**720.5**	**720.5**	×
35	4	3	3	1	**914.8**	**914.8**	923.8	894.7	
35	4	3	3	2	**929.9**	**929.9**	**929.9**	**929.9**	
35	4	3	3	3	**924.9**	**924.9**	**924.9**	**924.9**	
35	4	3	3	4	**904.2**	**904.2**	**939.4**	**939.4**	×
35	4	3	3	5	**860.7**	**860.7**	**860.7**	**860.7**	
35	4	4	1	1	**533.2**	**533.2**	**549.1**	**549.1**	×
35	4	4	1	2	**515.0**	**515.0**	**523.4**	**523.4**	×
35	4	4	1	3	**498.7**	**498.7**	**503.0**	**503.0**	×
35	4	4	1	4	**525.9**	**525.9**	**563.9**	**563.9**	×
35	4	4	1	5	**474.9**	**474.9**	**505.2**	**505.2**	×
35	4	4	2	1	**605.3**	**605.3**	**606.9**	**606.9**	×
35	4	4	2	2	**577.5**	**577.5**	**589.5**	**589.5**	×
35	4	4	2	3	**535.2**	**535.2**	**556.8**	**556.8**	×
35	4	4	2	4	**537.9**	**537.9**	**550.1**	**550.1**	×
35	4	4	2	5	**578.2**	**578.2**	**584.1**	**584.1**	×
35	4	4	3	1	**536.2**	**536.2**	**574.4**	**574.4**	×
35	4	4	3	2	**515.9**	**515.9**	**518.1**	**518.1**	×
35	4	4	3	3	**565.7**	**565.7**	**569.6**	**569.6**	×
35	4	4	3	4	**470.3**	**470.3**	**470.9**	**470.9**	×
35	4	4	3	5	**466.3**	**466.3**	**499.7**	**499.7**	×
40	3	2	1	1	**726.1**	**726.1**	**729.0**	**729.0**	×
40	3	2	1	2	**662.2**	**662.2**	**669.9**	**669.9**	×
40	3	2	1	3	**598.0**	**598.0**	**601.0**	**601.0**	×
40	3	2	1	4	**629.7**	**629.7**	**631.9**	**631.9**	×
40	3	2	1	5	**597.8**	**597.8**	**600.6**	**600.6**	×
40	3	2	2	1	**853.2**	**853.2**	859.5	838.4	
40	3	2	2	2	**842.0**	**842.0**	**842.0**	**842.0**	
40	3	2	2	3	**829.2**	**829.2**	**831.7**	**831.7**	×
40	3	2	2	4	**836.0**	**836.0**	**837.4**	**837.4**	×
40	3	2	2	5	**819.1**	**819.1**	**833.2**	**833.2**	×
40	3	2	3	1	**1053.9**	**1053.9**	**1053.9**	**1053.9**	
40	3	2	3	2	**995.8**	**995.8**	**995.8**	**995.8**	
40	3	2	3	3	**1048.7**	**1048.7**	**1048.7**	**1048.7**	
40	3	2	3	4	**1078.3**	**1078.3**	**1078.3**	**1078.3**	
40	3	2	3	5	**1074.0**	**1074.0**	**1074.0**	**1074.0**	
40	3	3	1	1	**539.9**	**539.9**	**543.7**	**543.7**	×
40	3	3	1	2	**606.5**	**606.5**	**621.3**	**621.3**	×
40	3	3	1	3	**614.8**	**614.8**	**614.8**	**614.8**	
40	3	3	1	4	**554.6**	**554.6**	**554.6**	**554.6**	
40	3	3	1	5	**504.4**	**504.4**	**516.2**	**516.2**	×
40	3	3	2	1	**619.6**	**619.6**	**624.9**	**624.9**	×
40	3	3	2	2	**629.4**	**629.4**	**633.4**	**633.4**	×
40	3	3	2	3	**594.1**	**594.1**	**602.4**	**602.4**	×

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
40	3	3	2	4	**590.2**	**590.2**	**590.6**	**590.6**	×
40	3	3	2	5	**585.2**	**585.2**	**585.6**	**585.6**	×
40	3	3	3	1	**557.6**	**557.6**	**560.1**	**560.1**	×
40	3	3	3	2	**603.9**	**603.9**	**607.8**	**607.8**	×
40	3	3	3	3	**598.7**	**598.7**	**607.0**	**607.0**	×
40	3	3	3	4	**624.5**	**624.5**	**624.5**	**624.5**	
40	3	3	3	5	**564.6**	**564.6**	**564.6**	**564.6**	
40	4	2	1	1	798.0	743.4	813.0	744.6	
40	4	2	1	2	**728.8**	**728.8**	**732.3**	**732.3**	×
40	4	2	1	3	**812.7**	**812.7**	**812.7**	**812.7**	
40	4	2	1	4	**757.4**	**757.4**	**774.2**	**774.2**	×
40	4	2	1	5	**729.4**	**729.4**	**731.9**	**731.9**	×
40	4	2	2	1	**944.1**	**944.1**	**944.1**	**944.1**	
40	4	2	2	2	**931.4**	**931.4**	**938.0**	**938.0**	×
40	4	2	2	3	**1038.6**	**1038.6**	**1043.1**	**1043.1**	×
40	4	2	2	4	990.3	980.3	1015.2	958.3	
40	4	2	2	5	**964.8**	**964.8**	**964.8**	**964.8**	
40	4	2	3	1	**1095.3**	**1095.3**	**1095.3**	**1095.3**	
40	4	2	3	2	**1090.0**	**1090.0**	**1111.2**	**1111.2**	×
40	4	2	3	3	**1079.2**	**1079.2**	**1085.8**	**1085.8**	×
40	4	2	3	4	**1053.5**	**1053.5**	**1053.5**	**1053.5**	
40	4	2	3	5	**1169.1**	**1169.1**	**1169.1**	**1169.1**	
40	4	3	1	1	**575.9**	**575.9**	**575.9**	**575.9**	
40	4	3	1	2	**688.3**	**688.3**	**688.3**	**688.3**	
40	4	3	1	3	**559.3**	**559.3**	**606.1**	**606.1**	×
40	4	3	1	4	**598.2**	**598.2**	**598.3**	**598.3**	×
40	4	3	1	5	**609.0**	**609.0**	**616.6**	**616.6**	×
40	4	3	2	1	**695.0**	**695.0**	**695.1**	**695.1**	×
40	4	3	2	2	**684.5**	**684.5**	**710.7**	**710.7**	×
40	4	3	2	3	**772.5**	**772.5**	**774.4**	**774.4**	×
40	4	3	2	4	769.9	757.2	**774.1**	**774.1**	×
40	4	3	2	5	825.2	796.7	848.1	809.5	
40	4	3	3	1	**952.3**	**952.3**	970.0	926.0	
40	4	3	3	2	926.9	912.4	930.0	911.1	
40	4	3	3	3	**945.8**	**945.8**	**945.8**	**945.8**	
40	4	3	3	4	880.4	852.8	874.2	851.2	
40	4	3	3	5	968.6	950.5	991.8	955.7	
40	4	4	1	1	**568.1**	**568.1**	**583.7**	**583.7**	×
40	4	4	1	2	**625.7**	**625.7**	**640.9**	**640.9**	×
40	4	4	1	3	**525.7**	**525.7**	**550.6**	**550.6**	×
40	4	4	1	4	**602.8**	**602.8**	**606.8**	**606.8**	×
40	4	4	1	5	**596.6**	**596.6**	**622.5**	**622.5**	×
40	4	4	2	1	**561.2**	**561.2**	**633.5**	**633.5**	×
40	4	4	2	2	**542.4**	**542.4**	**580.2**	**580.2**	×
40	4	4	2	3	**569.4**	**569.4**	**577.4**	**577.4**	×
40	4	4	2	4	**575.5**	**575.5**	**585.4**	**585.4**	×
40	4	4	2	5	**604.2**	**604.2**	**609.6**	**609.6**	×
40	4	4	3	1	**541.9**	**541.9**	**558.3**	**558.3**	×
40	4	4	3	2	**593.8**	**593.8**	**618.1**	**618.1**	×
40	4	4	3	3	**546.8**	**546.8**	**546.8**	**546.8**	
40	4	4	3	4	**581.1**	**581.1**	**595.0**	**595.0**	×
40	4	4	3	5	**553.8**	**553.8**	**560.8**	**560.8**	×
45	3	2	1	1	**749.2**	**749.2**	**749.2**	**749.2**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
45	3	2	1	2	**725.1**	**725.1**	**725.1**	**725.1**	
45	3	2	1	3	**718.5**	**718.5**	**721.7**	**721.7**	×
45	3	2	1	4	**632.3**	**632.3**	**634.1**	**634.1**	×
45	3	2	1	5	**722.1**	**722.1**	**722.5**	**722.5**	×
45	3	2	2	1	936.2	919.3	957.8	922.3	
45	3	2	2	2	907.3	889.1	919.9	884.5	
45	3	2	2	3	**803.4**	**803.4**	**806.3**	**806.3**	×
45	3	2	2	4	**860.4**	**860.4**	860.5	843.3	
45	3	2	2	5	**836.3**	**836.3**	**846.4**	**846.4**	×
45	3	2	3	1	**1113.2**	**1113.2**	**1113.2**	**1113.2**	
45	3	2	3	2	**1062.7**	**1062.7**	**1062.7**	**1062.7**	
45	3	2	3	3	**1129.7**	**1129.7**	1129.7	1129.2	
45	3	2	3	4	**982.3**	**982.3**	**982.3**	**982.3**	
45	3	2	3	5	1246.0	1233.8	1240.3	1237.6	
45	3	3	1	1	**577.3**	**577.3**	**580.6**	**580.6**	×
45	3	3	1	2	**573.9**	**573.9**	**606.4**	**606.4**	×
45	3	3	1	3	**578.2**	**578.2**	**578.2**	**578.2**	
45	3	3	1	4	**552.0**	**552.0**	**556.1**	**556.1**	×
45	3	3	1	5	**570.5**	**570.5**	**578.2**	**578.2**	×
45	3	3	2	1	**598.8**	**598.8**	**604.4**	**604.4**	×
45	3	3	2	2	**609.5**	**609.5**	**609.5**	**609.5**	
45	3	3	2	3	**577.4**	**577.4**	**577.4**	**577.4**	
45	3	3	2	4	**638.9**	**638.9**	**638.9**	**638.9**	
45	3	3	2	5	**574.7**	**574.7**	**581.0**	**581.0**	×
45	3	3	3	1	**602.9**	**602.9**	**613.6**	**613.6**	×
45	3	3	3	2	**558.7**	**558.7**	**560.7**	**560.7**	×
45	3	3	3	3	**605.8**	**605.8**	**614.9**	**614.9**	×
45	3	3	3	4	**604.7**	**604.7**	**618.3**	**618.3**	×
45	3	3	3	5	**585.8**	**585.8**	**603.8**	**603.8**	×
45	4	2	1	1	**809.0**	**809.0**	823.7	799.9	
45	4	2	1	2	**758.6**	**758.6**	**765.9**	**765.9**	×
45	4	2	1	3	**637.1**	**637.1**	**637.1**	**637.1**	
45	4	2	1	4	871.3	780.2	835.3	787.6	
45	4	2	1	5	731.9	725.1	**731.9**	**731.9**	
45	4	2	2	1	1087.6	1054.4	**1074.7**	**1074.7**	
45	4	2	2	2	989.2	980.0	989.2	977.2	
45	4	2	2	3	992.7	960.7	995.7	974.9	
45	4	2	2	4	1044.0	991.0	**1029.9**	**1029.9**	
45	4	2	2	5	**1039.8**	**1039.8**	**1040.7**	**1040.7**	×
45	4	2	3	1	**1104.8**	**1104.8**	**1113.3**	**1113.3**	×
45	4	2	3	2	**1169.1**	**1169.1**	**1169.1**	**1169.1**	
45	4	2	3	3	**1089.0**	**1089.0**	**1089.0**	**1089.0**	
45	4	2	3	4	**1198.8**	**1198.8**	**1214.4**	**1214.4**	×
45	4	2	3	5	**1107.7**	**1107.7**	**1107.7**	**1107.7**	
45	4	3	1	1	**609.2**	**609.2**	**616.4**	**616.4**	×
45	4	3	1	2	**676.1**	**676.1**	**676.1**	**676.1**	
45	4	3	1	3	**596.0**	**596.0**	**607.1**	**607.1**	×
45	4	3	1	4	**634.8**	**634.8**	**664.3**	**664.3**	×
45	4	3	1	5	**613.8**	**613.8**	**624.1**	**624.1**	×
45	4	3	2	1	**777.4**	**777.4**	**799.9**	**799.9**	×
45	4	3	2	2	**777.4**	**777.4**	**778.8**	**778.8**	×
45	4	3	2	3	**777.2**	**777.2**	800.3	785.9	×
45	4	3	2	4	**805.1**	**805.1**	**805.1**	**805.1**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
45	4	3	2	5	**802.0**	**802.0**	844.8	797.5	
45	4	3	3	1	1034.2	1010.2	1056.5	1028.4	
45	4	3	3	2	**918.2**	**918.2**	**930.7**	**930.7**	×
45	4	3	3	3	1070.6	1016.8	1100.5	1015.3	
45	4	3	3	4	1033.4	978.2	1030.9	979.7	
45	4	3	3	5	1021.2	955.1	1033.3	940.4	
45	4	4	1	1	**567.4**	**567.4**	**587.5**	**587.5**	×
45	4	4	1	2	**598.8**	**598.8**	**627.8**	**627.8**	×
45	4	4	1	3	**622.9**	**622.9**	**628.8**	**628.8**	×
45	4	4	1	4	**593.8**	**593.8**	**593.8**	**593.8**	
45	4	4	1	5	**620.8**	**620.8**	**620.8**	**620.8**	
45	4	4	2	1	**606.7**	**606.7**	**652.4**	**652.4**	×
45	4	4	2	2	**585.1**	**585.1**	**592.2**	**592.2**	×
45	4	4	2	3	**605.0**	**605.0**	**626.1**	**626.1**	×
45	4	4	2	4	**670.5**	**670.5**	**676.0**	**676.0**	×
45	4	4	2	5	**598.2**	**598.2**	**606.3**	**606.3**	×
45	4	4	3	1	**627.8**	**627.8**	**639.6**	**639.6**	×
45	4	4	3	2	**589.9**	**589.9**	**625.5**	**625.5**	×
45	4	4	3	3	**610.4**	**610.4**	**627.0**	**627.0**	×
45	4	4	3	4	**633.1**	**633.1**	**637.2**	**637.2**	×
45	4	4	3	5	**574.2**	**574.2**	**603.0**	**603.0**	×
50	3	2	1	1	**686.6**	**686.6**	**692.8**	**692.8**	×
50	3	2	1	2	**684.6**	**684.6**	**684.6**	**684.6**	
50	3	2	1	3	**792.7**	**792.7**	**792.7**	**792.7**	
50	3	2	1	4	**682.4**	**682.4**	**694.4**	**694.4**	×
50	3	2	1	5	**770.4**	**770.4**	**770.4**	**770.4**	
50	3	2	2	1	940.0	910.0	984.8	904.9	
50	3	2	2	2	**900.1**	**900.1**	916.1	886.8	
50	3	2	2	3	**899.2**	**899.2**	**899.3**	**899.3**	×
50	3	2	2	4	**928.9**	**928.9**	**930.8**	**930.8**	×
50	3	2	2	5	**872.5**	**872.5**	**873.5**	**873.5**	×
50	3	2	3	1	**1258.3**	**1258.3**	**1258.3**	**1258.3**	
50	3	2	3	2	1215.7	1202.4	1250.2	1203.9	
50	3	2	3	3	**1078.4**	**1078.4**	**1078.4**	**1078.4**	
50	3	2	3	4	1206.2	1163.6	1200.9	1164.7	
50	3	2	3	5	**1093.8**	**1093.8**	**1093.8**	**1093.8**	
50	3	3	1	1	**613.2**	**613.2**	**643.3**	**643.3**	×
50	3	3	1	2	**601.1**	**601.1**	**604.7**	**604.7**	×
50	3	3	1	3	**609.0**	**609.0**	**621.2**	**621.2**	×
50	3	3	1	4	**627.2**	**627.2**	**627.2**	**627.2**	
50	3	3	1	5	**664.1**	**664.1**	**667.3**	**667.3**	×
50	3	3	2	1	**618.9**	**618.9**	**633.0**	**633.0**	×
50	3	3	2	2	**625.9**	**625.9**	**640.7**	**640.7**	×
50	3	3	2	3	**631.7**	**631.7**	**631.7**	**631.7**	
50	3	3	2	4	**698.8**	**698.8**	**707.1**	**707.1**	×
50	3	3	2	5	**641.3**	**641.3**	**641.3**	**641.3**	
50	3	3	3	1	**650.5**	**650.5**	**650.5**	**650.5**	
50	3	3	3	2	**665.3**	**665.3**	**665.3**	**665.3**	
50	3	3	3	3	**679.9**	**679.9**	**679.9**	**679.9**	
50	3	3	3	4	**608.2**	**608.2**	**608.2**	**608.2**	
50	3	3	3	5	**651.2**	**651.2**	**653.8**	**653.8**	×
50	4	2	1	1	**830.3**	**830.3**	**830.3**	**830.3**	
50	4	2	1	2	**861.6**	**861.6**	**861.6**	**861.6**	

Continued on next column/page

Instance					MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	No.	UB	LB	UB	LB	div
50	4	2	1	3	791.3	734.2	827.8	734.6	
50	4	2	1	4	**726.7**	**726.7**	**726.7**	**726.7**	
50	4	2	1	5	771.4	750.5	**758.2**	**758.2**	
50	4	2	2	1	975.0	956.1	979.5	943.7	
50	4	2	2	2	**1079.1**	**1079.1**	1088.2	1068.1	
50	4	2	2	3	**1034.0**	**1034.0**	**1034.0**	**1034.0**	
50	4	2	2	4	1076.3	1064.8	1102.8	1061.3	
50	4	2	2	5	1062.3	1041.1	1062.0	1046.0	
50	4	2	3	1	**1289.9**	**1289.8**	**1293.5**	**1293.4**	×
50	4	2	3	2	**1182.5**	**1182.3**	**1182.5**	**1182.3**	
50	4	2	3	3	1070.1	1062.5	**1072.3**	**1072.2**	×
50	4	2	3	4	**1145.2**	**1145.2**	**1145.2**	**1145.2**	
50	4	2	3	5	**1154.2**	**1154.1**	**1156.4**	**1156.3**	×
50	4	3	1	1	**651.6**	**651.5**	**664.7**	**664.7**	×
50	4	3	1	2	**617.2**	**617.2**	**617.2**	**617.2**	
50	4	3	1	3	**725.1**	**725.0**	**728.5**	**728.4**	×
50	4	3	1	4	**613.6**	**613.6**	**618.2**	**618.2**	×
50	4	3	1	5	**694.6**	**694.6**	**700.1**	**700.1**	×
50	4	3	2	1	**804.3**	**804.3**	805.2	799.1	
50	4	3	2	2	**830.2**	**830.1**	884.1	834.5	×
50	4	3	2	3	**840.5**	**840.4**	**840.7**	**840.7**	×
50	4	3	2	4	**762.6**	**762.6**	**769.7**	**769.6**	×
50	4	3	2	5	**768.0**	**767.9**	**778.5**	**778.5**	×
50	4	3	3	1	**1030.4**	**1030.4**	**1030.4**	**1030.4**	
50	4	3	3	2	**1098.4**	**1098.3**	**1098.4**	**1098.3**	
50	4	3	3	3	1069.2	1020.2	1147.0	1038.0	
50	4	3	3	4	1018.2	967.6	1062.0	989.9	
50	4	3	3	5	1081.6	996.7	1077.3	998.2	
50	4	4	1	1	**668.3**	**668.3**	**672.7**	**672.7**	×
50	4	4	1	2	**601.7**	**601.7**	**612.7**	**612.7**	×
50	4	4	1	3	**658.9**	**658.9**	**659.3**	**659.3**	×
50	4	4	1	4	**570.1**	**570.1**	**581.2**	**581.2**	×
50	4	4	1	5	**624.9**	**624.9**	**626.3**	**626.3**	×
50	4	4	2	1	**625.3**	**625.3**	**627.4**	**627.3**	×
50	4	4	2	2	**585.3**	**585.3**	**605.2**	**605.2**	×
50	4	4	2	3	**592.7**	**592.7**	**620.6**	**620.5**	×
50	4	4	2	4	**584.2**	**584.2**	**598.9**	**598.8**	×
50	4	4	2	5	**608.6**	**608.6**	**641.5**	**641.5**	×
50	4	4	3	1	**614.7**	**614.6**	**629.5**	**629.5**	×
50	4	4	3	2	**614.6**	**614.6**	**629.8**	**629.7**	×
50	4	4	3	3	**606.6**	**606.6**	**616.8**	**616.7**	×
50	4	4	3	4	**604.4**	**604.4**	**618.6**	**618.6**	×
50	4	4	3	5	**591.7**	**591.6**	**615.3**	**615.2**	×

Table 5.16: Detailed results for the `large(H15)` instances.

Instance				MCVRP-CFCS		MCVRP-DFCS		
$\|V\|$	ρ	C	s	UB	LB	UB	LB	div
50	3	2	1	**1000.0**	**1000.0**	**1022.3**	**1022.3**	×
50	3	2	2	1486.5	1037.4	2537.4	1045.0	
50	3	2	3		1342.0		1342.0	
50	3	3	1	**1027.8**	**1027.8**	2227.3	1037.2	×
50	3	3	2	**1017.2**	**1017.2**		1039.1	×
50	3	3	3	**952.4**	**952.4**		975.1	×
50	6	2	1	1348.5	1310.0	**1323.7**	**1323.7**	
50	6	2	2	6371.4	1526.9		1542.7	
50	6	2	3		1819.1		1820.0	
50	6	4	1	**917.4**	**917.4**	2369.5	945.1	×
50	6	4	2		1171.3		1188.5	
50	6	4	3		1302.6		1341.9	
50	6	6	1	**971.9**	**971.9**		1011.1	×
50	6	6	2	961.9	953.9		993.5	×
50	6	6	3		917.6		956.3	
50	9	2	1	2492.6	1513.0	2777.3	1332.2	
50	9	2	2		1888.2		1885.9	
50	9	2	3		2435.2		2434.7	
50	9	4	1	3352.5	1103.0		1128.0	
50	9	4	2		1166.2		1188.5	
50	9	4	3		1401.9		1404.0	
50	9	7	1	**951.1**	**951.0**		1009.7	×
50	9	7	2		1051.0		1059.6	
50	9	7	3		1252.3		1254.2	
50	9	9	1	**963.3**	**963.3**		1008.3	×
50	9	9	2		973.1		1076.9	
50	9	9	3		982.6		1045.2	

Chapter 6

Conclusion

The main goal of this thesis was to contribute to the development of exact solution approaches to packing and routing problems. In this chapter, main results are summarized and concluding remarks are made.

To the best of our knowledge, we proposed in Chapter 2 the first BP approach for the VPP that also tackles non-binary formulations. The CG subproblems are variants of the multidimensional knapsack problem that were formulated as SPPRCs to obtain a generic solution approach. We derived monodirectional and bidirectional labeling algorithms for their solution. For the first time, dual inequalities have been used in a BP for the VPP to stabilize the CG process. Moreover, our solution approach for the subproblem is compatible with Ryan and Foster (1981) and Vanderbeck (1999) branching. An extensive computational study showed that our algorithms are competitive with respect to the state of the art. Several previously open 2- and 20-dimensional instances can be solved to proven optimality.

In Chapter 3, we proposed exact and heuristic algorithms for a real-world packing problem where products should be packed into trucks such that a five-level lexicographic objective function is minimized. The first-level objective minimizes the overall number of trucks and secondary objectives concern operational costs of cooling and freezing devices of a truck and splitting policies. To compute exact solutions, we proposed a BP algorithm in which the subproblems of all levels are formulated as an SPPRC. This yields a uniform solution approach for the five different problems arising in the different levels. Moreover, advanced concepts of stabilization and acceleration techniques are employed. To compute heuristic solutions for a single level of the packing problem, we developed a number of constructive heuristics that try to generate solutions with a favorable mix of weight and volume. Thereafter, these are integrated into a more complex heuristic framework for building high quality solutions. Computational experiments for real-world and generated benchmark instances show that the BP algorithm is highly effective in computing optimal solutions. Furthermore, the heuristic framework matches the optimal solutions in 86 % of all subproblems.

In Chapter 4, we provided the first two-index formulation with symmetric rout-

ing part and asymmetric soft-cluster part for the SoftCluVRP. Since only capacity constraints have to be added dynamically (all other presented valid inequalities are not mandatory), the formulation is rather simple to use with modern MIP solvers. On the theoretical side, the impact of the fleet-size constraint is analyzed and new constraints are derived that allow coping with non-minimal fleets. We proved that one third of the transitivity constraints are redundant and presented new strong capacity cuts. Moreover, we proved a deep result that SoftCluVRPs defined on a square grid are reducible. Note that this reduction scheme is also applicable to the respective traveling salesman problem. On the algorithmic side, the new BC algorithm is complementary to the only other exact solution approach from the literature and performs well for truly clustered instances. Overall, our algorithm and lower-bounding strategies for square grid instances deliver new provable optimal solutions to five `GVRP` and nine `Golden` benchmark instances.

In Chapter 5, we proposed three new exact solution approaches for the MCVRP-CFCS and two new exact solution approaches for the MCVRP-DFCS. Computational tests identified that the performance of the algorithms depends on the number of vertices, the supply parameter, and the number of vehicles. The algorithms can solve all instances with up to 30 vertices and over 80 % of the `mid-size(H18)` instances with 50 vertices to optimality within two hours of computation time. Moreover, 6 (2) `large(H15)` instances of the MCVRP-CFCS (MCVRP-DFCS) are solved for the first time. Finally, we showed that the savings potential of using continuously flexible compartment sizes instead of discretely flexible compartment sizes depends on average on the number of vertices, the number of supplies, and the number of product types.

Bibliography

Adulyasak, Y., Cordeau, J.-F., and Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, **26**(1), 103–120.

Ahr, D. (2004). *Contributions to Multiple Postmen Problems*. Ph.d. dissertation, Department of Computer Science, Heidelberg University, Heidelberg, Germany.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs, NJ.

Alves, C. and Valério de Carvalho, J. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, **35**(4), 1315–1328.

Alves, C., Valério de Carvalho, J., Clautiaux, F., and Rietz, J. (2014). Multidimensional dual-feasible functions and fast lower bounds for the vector packing problem. *European Journal of Operational Research*, **233**(1), 43–63.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2011). *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton.

Archetti, C., Bianchessi, N., and Speranza, M. G. (2015). A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem. *Computers & Operations Research*, **64**, 1–10.

Archetti, C., Campbell, A. M., and Speranza, M. G. (2016). Multicommodity vs. single-commodity routing. *Transportation Science*, **50**(2), 461–472.

Aringhieri, R., Duma, D., Grosso, A., and Hosteins, P. (2018). Simple but effective heuristics for the 2-constraint bin packing problem. *Journal of Heuristics*, **24**(3), 345–357.

Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. Ph.D. thesis, Institut National Polytechnique de Grenoble, Grenoble, France.

Avella, P., Boccia, M., and Sforza, A. (2004). Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, **152**(1), 170–179.

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

Bansal, N., Eliáš, M., and Khan, A. (2016). Improved approximation for vector bin packing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 3, pages 1561–1579. SIAM.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Battarra, M., Erdoğan, G., and Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Operations Research*, **62**(1), 58–71.

Bektaş, T., Erdoğan, G., and Røpke, S. (2011). Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, **45**(3), 299–316.

Ben Amor, H. and Valério de Carvalho, J. (2005). Cutting stock problems. In Desaulniers *et al.* (2005), chapter 5, pages 131–161.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Benantar, A., Ouafi, R., and Boukachour, J. (2016). A petrol station replenishment problem: new variant and formulation. *Logistics Research*, **9**(1).

Bentley, J. (1980). Multidimensional divide-and-conquer. *Communications of the ACM*, **23**(4), 214–229.

Bertazzi, L., Golden, B., and Wang, X. (2019). The bin packing problem with item fragmentation: A worst-case analysis. *Discrete Applied Mathematics*, **261**, 63–77.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Brandão, F. and Pedroso, J. P. (2016). Bin packing and related problems: General arc-flow formulation with graph compression. *Computers & Operations Research*, **69**, 56–67.

Brown, G. G. and Graves, G. W. (1981). Real-time dispatch of petroleum tank trucks. *Management Science*, **27**(1), 19–32.

Buljubašić, M. and Vasquez, M. (2016). Consistent neighborhood search for one-dimensional bin packing and two-dimensional vector packing. *Computers & Operations Research*, **76**, 12–21.

Caprara, A. (1998). Properties of some ILP formulations of a class of partitioning problems. *Discrete Applied Mathematics*, **87**(1-3), 11–23.

Caprara, A. and Toth, P. (2001). Lower bounds and algorithms for the 2-dimensional vector packing problem. *Discrete Applied Mathematics*, **111**(3), 231–262.

Caprara, A., Kellerer, H., and Pferschy, U. (2003). Approximation schemes for ordered vector packing problems. *Naval Research Logistics*, **50**(1), 58–69.

Caprara, A., Dell'Amico, M., Díaz-Díaz, J. C., Iori, M., and Rizzi, R. (2014). Friendly bin packing instances without integer round-up property. *Mathematical Programming*, **150**(1), 5–17.

Caramia, M. and Guerriero, F. (2010). A milk collection problem with incompatibility constraints. *Interfaces*, **40**(2), 130–143.

Casazza, M. (2019). New formulations for variable cost and size bin packing problems with item fragmentation. *Optimization Letters*, **13**(2), 379–398.

Casazza, M. and Ceselli, A. (2016). Exactly solving packing problems with fragmentation. *Computers & Operations Research*, **75**, 202–213.

Chajakis, E. D. and Guignard, M. (2003). Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, **26**(1), 43–78.

Christensen, H. I., Khan, A., Pokutta, S., and Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, **24**, 63–79.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**(4), 305–337.

Coelho, L. C. and Laporte, G. (2015). Classification, models and exact algorithms for multi-compartment delivery problems. *European Journal of Operational Research*, **242**(3), 854–864.

Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., and Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. *Handbook of combinatorial optimization*, pages 455–531.

Contardo, C., Cordeau, J.-F., and Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *INFORMS Journal on Computing*, **26**(1), 88–102.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

Cornillier, F., Boctor, F. F., Laporte, G., and Renaud, J. (2008). An exact algorithm for the petrol station replenishment problem. *Journal of the Operational Research Society*, **59**(5), 607–615.

Côté, J.-F. and Iori, M. (2018). The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, **30**(4), 646–661.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, **8**(3), 250–255.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, **6**(1), 80–91.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**(1), 101–111.

Defryn, C. and Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, **83**, 78–94.

Delorme, M. and Iori, M. (2020). Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, **32**(1), 101–119.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., and Vogel, U. (2010). Vehicle routing with compartments: applications, modelling and heuristics. *OR Spectrum*, **33**(4), 885–914.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**(1), 269–271.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, **44**(2), 145–159.

Eshtehadi, R., Demir, E., and Huang, Y. (2020). Solving the vehicle routing problem with multi-compartment vehicles for city logistics. *Computers & Operations Research*, **115**, 104859.

Fagerholt, K. and Christiansen, M. (2000). A combined ship scheduling and allocation problem. *Journal of the Operational Research Society*, **51**(7), 834–842.

Fallahi, A. E., Prins, C., and Calvo, R. W. (2008). A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, **35**(5), 1725–1741.

Fischetti, M., Salazar González, J. J., and Toth, P. (1995). Experiments with a multi-commodity formulation for the symmetric capacitated vehicle routing problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*.

Fischetti, M., Salazar Gonzáles, J. J., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, **45**(3), 378–394.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, **4**(1), 61–75.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2005). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, **106**(3), 491–511.

Garey, M. R., Graham, R. L., and Ullman, J. D. (1972). Worst-case analysis of memory allocation algorithms. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 143–150. ACM.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Goeke, D., Gschwind, T., and Schneider, M. (2019). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. *Discrete Applied Mathematics*, **264**, 43–61.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. In *Fleet Management and Logistics*, pages 33–56. Springer US.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill Book Company.

Goodson, J. C. (2015). A priori policy evaluation and cyclic-order-based simulated annealing for the multi-compartment vehicle routing problem with stochastic demands. *European Journal of Operational Research*, **241**(2), 361–369.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Henke, T. (2017). Multi-compartment vehicle routing problems – a review and an extended classification. Technical report, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management, Magdeburg, Germany.

Henke, T. (2018). *Multi-compartment vehicle routing problems in the context of glass waste collection*. Ph.D. thesis, Otto-von-Guericke University Magdeburg, Magdeburg, Germany.

Henke, T., Speranza, M. G., and Wäscher, G. (2015). The multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research*, **246**(3), 730–743.

Henke, T., Speranza, M. G., and Wäscher, G. (2018). A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. *Annals of Operations Research*, **275**(2), 321–338.

Heßler, K., Gschwind, T., and Irnich, S. (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, **271**(2), 401–419.

Hintsch, T. (2019). Large multiple neighborhood search for the soft-clustered vehicle-routing problem. Technical Report LM-2019-01, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hintsch, T. and Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, **270**(1), 118–131.

Hintsch, T. and Irnich, S. (2020). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, **280**(1), 164–178.

Hintsch, T., Irnich, S., and Kiilerich, L. (2019). Branch-price-and-cut for the soft-clustered capacitated arc-routing problem. Technical Report LM-2019-02, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hu, Q., Zhu, W., Qin, H., and Lim, A. (2017). A branch-and-price algorithm for the two-dimensional vector packing problem with piecewise linear cost function. *European Journal of Operational Research*, **260**(1), 70–80.

Huang, E. and Korf, R. E. (2012). Optimal rectangle packing: An absolute placement approach. *Journal of Artificial Intelligence Research*, **46**, 47–87.

Hübner, A. and Ostermeier, M. (2019). A multi-compartment vehicle routing problem with loading and unloading costs. *Transportation Science*, **53**(1), 282–300.

Iori, M., Salazar-González, J.-J., and Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, **41**(2), 253–264.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In Toth and Vigo (2014), chapter 1, pages 1–33.

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

Johnson, D. S. (1973). *Near-Optimal Bin Packing Algorithms*. Ph.D. thesis, Massachusetts Institute of Technology.

Karger, D. R. (1993). Global min-cuts in RNC, and other ramifications of a simple min-out algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

Koch, H., Henke, T., and Wäscher, G. (2016). A Genetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes. FEMM Working Papers 160004, Otto-von-Guericke University Magdeburg, Faculty of Economics and Management.

Korte, B. and Vygen, J. (2006). *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics 21*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, third edition edition.

Lahyani, R., Coelho, L. C., Khemakhem, M., Laporte, G., and Semet, F. (2015). A multi-compartment vehicle routing problem arising in the collection of olive oil in tunisia. *Omega*, **51**, 1–10.

Laporte, G., Nobert, Y., and Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, **33**(5), 1050–1073.

LeCun, B., Mautor, T., Quessette, F., and Weisser, M.-A. (2015). Bin packing with fragmentable items: Presentation and approximations. *Theoretical Computer Science*, **602**, 50–59.

Lodi, A., Martello, S., and Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, **123**(1-3), 379–396.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, **100**(2), 423–445.

Malaguti, E., Monaci, M., Paronuzzi, P., and Pferschy, U. (2019). Integer optimization with penalized fractional values: The Knapsack case. *European Journal of Operational Research*, **273**(3), 874–888.

Mandal, C. A., Chakrabarti, P. P., and Ghose, S. (1998). Complexity of fragmentable object bin packing and an application. *Computers & Mathematics with Applications*, **35**(11), 91–97.

Martello, S. and Toth, P. (1990). Bin-packing problem. In *Knapsack problems: Algorithms and Computer Implementations*, Wiley Series in Discrete Mathematics and Optimization, pages 221–245. Wiley.

Martello, S. and Toth, P. (2003). An exact algorithm for the two-constraint 0–1 knapsack problem. *Operations Research*, **51**(5), 826–835.

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, **48**(2), 256–267.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Melechovskỳ, J. (2013). A variable neighborhood search for the selective multicompartment vehicle routing problem with time windows. *Lecture notes in management science*, **5**, 159–166.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, **37**(11), 1886–1898.

Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2011). Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transportation Science*, **45**(3), 346–363.

Miller, C., Tucker, A., and Zemlin, R. (1960). Integer programming formulations of traveling salesman problems. *Journal of the Association for Computing Machinery (JACM)*, **7**, 326–329.

Mirzaei, S. and Wøhlk, S. (2017). Erratum to: A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, **6**(2), 185–218.

Monaci, M. and Toth, P. (2006). A Set-Covering-Based heuristic approach for Bin-Packing problems. *INFORMS Journal on Computing*, **18**(1), 71–85.

Muyldermans, L. and Pang, G. (2010). On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research*, **206**(1), 93–103.

Oppen, J. and Løkketangen, A. (2008). A tabu search approach for the livestock collection problem. *Computers & Operations Research*, **35**(10), 3213–3229.

Oppen, J., Løkketangen, A., and Desrosiers, J. (2010). Solving a rich vehicle routing and inventory problem using column generation. *Computers & Operations Research*, **37**(7), 1308–1317.

Ostermeier, M. and Hübner, A. (2018). Vehicle selection for a multi-compartment vehicle routing problem. *European Journal of Operational Research*, **269**(2), 682–694.

Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. (2011). Heuristics for vector bin packing. Technical report, Microsoft Research. `https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/`.

Pirkwieser, S., Raidl, G. R., and Gottlieb, J. (2012). Improved packing and routing of vehicles with compartments. In *Computer Aided Systems Theory – EUROCAST 2011*, pages 392–399. Springer Berlin Heidelberg.

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., and Limbourg, S. (2014). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, **37**(2), 297–330.

Reed, M., Yiannakou, A., and Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, **15**, 169–176.

Rietz, J. (2003). *Untersuchungen zu MIRUP für Vektorpackprobleme*. Dissertation, Faculty of Mathematics und Computer Science, Technischen Universität Bergakademie Freiberg, Germany.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, chapter 17, pages 269–280. Elsevier, North-Holland.

Scheithauer, G. and Terno, J. (1995). The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research*, **84**, 562–571.

Schrijver, A. (2003). *Combinatorial optimization: Polyhedra and efficiency*, volume 24 of *Algorithms and combinatorics*. Springer, Berlin and New York.

Sevaux, M. and Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. In *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME*, volume 8, pages 411–417.

Spieksma, F. C. (1994). A branch-and-bound algorithm for the two-dimensional vector packing problem. *Computers & Operations Research*, **21**(1), 19–25.

Tarjan, R. E. (1979). A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, **18**(2), 110–127.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Toth, P. and Vigo, D., editors (2014). *Vehicle Routing*, volume 18 of *MOS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, **86**(3), 565–594.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In Desaulniers *et al.* (2005), chapter 12, pages 331–358.

Vasquez and Buljubašić (2018). Consistent neighbourhood search for two-dimensional vector packing. Presentation at ROADEF conference of the French Society for Operational Research and Decision Aiding, February 21-23.

Vidal, T., Battarra, M., Subramanian, A., and Erdoğan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, **58**, 87–99.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2019). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*.

Wei, L., Lai, M., Lim, A., and Hu, Q. (2020). A branch-and-price algorithm for the two-dimensional vector packing problem. *European Journal of Operational Research*, **281**(1), 25–35.