WORKING PAPER SERIES

A Branch-and-Cut Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes

Tino Henke/M. Grazia Speranza/Gerhard Wäscher

Working Paper No. 4/2017

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF ECONOMICS
AND MANAGEMENT

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:
Tino Henke, M. Grazia Speranza, Gerhard Wäscher
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

http://www.fww.ovgu.de/femm

Bezug über den Herausgeber
ISSN 1615-4274

A Branch-and-Cut Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes

Tino Henke

Department of Management Science, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany

tino.henke@ovgu.de, +49-391-6751840

M. Grazia Speranza

Department of Quantitative Methods, University of Brescia, 25122 Brescia, Italy

grazia.speranza@unibs.it

Gerhard Wäscher

Department of Management Science, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany

School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, 100044 Beijing, China

gerhard.waescher@ovgu.de

Abstract

Multi-compartment vehicle routing problems arise in a variety of problem settings in which different product types have to be transported separated from each other. In this paper, a problem variant which occurs in the context of glass waste recycling is considered. In this problem, a set of locations exists, each of which offering a number of containers for the collection of different types of glass waste (e.g. colorless, green, brown glass). In order to pick up the contents from the containers, a fleet of homogeneous disposal vehicles is available. Individually for each disposal vehicle, the capacity can be discretely separated into a limited number of compartments to which different glass waste types are assigned. The objective of the problem is to minimize the total distance to be travelled by the disposal vehicles.

For solving this problem to optimality, a branch-and-cut algorithm has been developed and implemented. Extensive numerical experiments have been conducted in order to evaluate the algorithm and to gain insights into the problem structure. The corresponding results show that the algorithm is able to solve instances with up to 50 locations to optimality and that it reduces the computing time by 87% compared to instances from the literature. Additional experiments give managerial insights into the use of different variants of compartments with flexible sizes.

Keywords: vehicle routing, multiple compartments, branch-and-cut algorithm, waste collection

1 Introduction

Vehicle routing problems have been studied for almost 60 years in numerous variants [14, 19, 26]. In recent years, researchers have been especially interested in so-called rich vehicle routing problems which take into account various real-world problem extensions. One group of such rich vehicle

routing problems consists of multi-compartment vehicle routing problems. In a multi-compartment vehicle routing problem (MCVRP), several product types are considered which have to be transported separately. This may be necessary because of different transportation requirements, e.g. different temperatures for food products, or because of the consistence of the product types, e.g. liquid or bulk products. Consequently, the vehicles have not only one but several storage compartments in order to serve more than one product type and, thus, to allow for efficient transportation.

A variety of applications of MCVRPs has been mentioned in the literature (see Section 3). These include, amongst others, the distribution of petrol products, the delivery of food products, or the collection of waste. The MCVRP variant considered in this paper is motivated by a real-world problem which occurs in the context of glass waste collection, in which several types of glass waste, i.e. colorless, green and brown glass, have to be collected from public glass waste containers. This problem has the special characteristics that the number of compartments is variable and that the compartment sizes are flexible within a pre-defined set of potential compartment sizes. In the following, this problem will be referred to as the multi-compartment vehicle routing problem with discretely flexible compartment sizes (MCVRP-DFCS).

Although some exact methods have been proposed in order to solve MCVRPs, so far only a single mathematical programming formulation has been presented to solve the MCVRP-DFCS. This paper introduces a new formulation and a branch-and-cut algorithm which includes subtour-elimination cuts, capacity cuts, and several newly developed or adapted valid inequalities. The algorithm has been tested by means of extensive numerical experiments on instances from the literature and newly generated instances. Compared to results from the literature, the computing times can be decreased by 87.19% on average. Furthermore, the experiments demonstrate a good performance of the algorithm for instances with up to 50 locations and give detailed insights into the problem structure. Moreover, the application of different types of flexible compartments has been investigated.

The remainder of this paper is organized in the following manner. In Section 2, the MCVRP-DFCS is described in detail and the new mathematical programming formulation is presented. In Section 3, a brief overview of the existing literature on MCVRPs is given and the differences between the MCVRP-DFCS and other problem variants are identified. Afterwards, the branch-and-cut algorithm with all its components is explained in detail in Section 4. In Section 5, the design of the numerical experiments and the corresponding results are explained and presented, before some conclusions are given in the last section.

2 Problem Description and Formulation

In the following, the MCVRP-DFCS is formulated on an undirected, complete and weighted graph $G = (V, E)$, which consists of a vertex set V and an edge set E. The vertex set $V = \{0,1, \dots, \mathrm{n}\}$ represents the location of the depot $\{0\}$ and the locations of n customers $\{1,2, \dots, \mathrm{n}\}$, whereas the edge

set $E = \{(i,j): i,j \in V, i < j\}$ includes all edges which can be travelled between locations. A non-negative cost c_{ij}, $(i,j) \in E$, is assigned to each edge.

Furthermore, a set of different product types P is considered. For each of these product types a non-negative supply $s_{ip} (i \in V\backslash\{0\}, p \in P)$ exists at each customer location. Each supply must be completely collected by exactly one vehicle and has to be transported to the depot. However, it is not necessary to collect supplies of different product types at the same location by the same vehicle.

In order to collect and transport the supplies, a set of homogeneous vehicles K is available at the depot. Individually for each vehicle, its total capacity Q can be separated into compartments, where each compartment allows for the transportation of exactly one product type. The maximal number $\hat{m}$ of compartments into which the capacity can be divided is limited and may be smaller than the number of product types. The size of each compartment is flexible and can be selected from a pre-defined set of potential compartment sizes. The smallest feasible compartment size in this set is the basic compartment unit size q^{unit}, and the remaining potential compartments sizes are all integer multiples of q^{unit} which do not exceed the total capacity.

The objective of the problem is to assign all supplies to vehicles and to determine a set of vehicle routes such that all capacities are respected and the total cost is minimized.

In the MCVRP-DFCS several decisions have to be made simultaneously. For each vehicle, it must be decided which product types are to be assigned to this vehicle and which compartment size should be selected for each assigned product type. Furthermore, for each vehicle it must be decided which supplies are to be collected and, thus, which locations are to be visited. Finally, the sequence according to which the assigned locations are to be visited has to be determined for each vehicle. In order to formulate a mathematical model for the MCVRP-DFCS, the following five types of variables are introduced:

$u_{ipk} = \begin{cases} 1, \text{ if supply of product type p at location i is collected by vehicle k,} \\ 0, \text{ otherwise,} \end{cases}$ $i \in V\backslash\{0\}, p \in P, k \in K;$

$x_{ijk} = \begin{cases} 2, \text{ if } i = 0 \text{ and edge } (i,j) \text{ is used twice by vehicle k,} \\ 1, \text{ if edge } (i,j) \text{ is used once by vehicle k,} \\ 0, \text{ otherwise,} \end{cases}$ $i, j \in V, i < j, k \in K;$

$z_{ik} = \begin{cases} 1, \text{ if location i is visited by vehicle k,} \\ 0, \text{ otherwise,} \end{cases}$ $i \in V, k \in K;$

$y^B_{pk} = \begin{cases} 1, \text{ if product type p is assigned to a compartment of vehicle k,} \\ 0, \text{ otherwise,} \end{cases}$ $p \in P, k \in K;$

y^I_{pk} : Integer variable indicating the size of the compartment of vehicle k which is assigned to product type p in number of basic unit sizes q^{unit} $p \in P, k \in K.$

The objective function and the constraints of the model can then be formulated as follows:

$$\min \sum_{i \in V} \sum_{\substack{j \in V \\ i<j}} \sum_{k \in K} c_{ij} \cdot x_{ijk} \tag{1}$$

$$\sum_{k \in K} u_{ipk} = 1 \qquad \forall i \in V \setminus \{0\}, p \in P, s_{ip} > 0 \tag{2}$$

$$u_{ipk} \leq z_{ik} \qquad \forall i \in V \setminus \{0\}, p \in P, k \in K \tag{3}$$

$$z_{ik} \leq z_{0k} \qquad \forall i \in V \setminus \{0\}, k \in K \tag{4}$$

$$\sum_{j \in V \setminus \{0\}} \sum_{k \in K} x_{0jk} \leq 2 \cdot |K| \tag{5}$$

$$\sum_{\substack{j \in V: \\ i<j}} x_{ijk} + \sum_{\substack{j \in V: \\ j<i}} x_{jik} = 2 \cdot z_{ik} \qquad \forall i \in V, k \in K \tag{6}$$

$$\sum_{i \in V \setminus \{0\}} s_{ip} \cdot u_{ipk} \leq q^{unit} \cdot y^{I}_{pk} \qquad \forall p \in P, k \in K \tag{7}$$

$$\sum_{p \in P} q^{unit} \cdot y^{I}_{pk} \leq Q \qquad \forall k \in K \tag{8}$$

$$\sum_{p \in P} y^{B}_{pk} \leq \hat{m} \qquad \forall k \in K \tag{9}$$

$$y^{I}_{pk} \leq \frac{Q}{q^{unit}} \cdot y^{B}_{pk} \qquad \forall p \in P, k \in K \tag{10}$$

$$\sum_{i \in S} \sum_{\substack{j \in S: \\ i<j}} x_{ijk} \leq |S| - 1 \qquad \forall k \in K, S \subseteq V \setminus \{0\}, |S| > 2 \tag{11}$$

$$u_{ipk} \in \{0,1\} \qquad \forall i \in V \setminus \{0\}, p \in P, k \in K \tag{12}$$

$$x_{0jk} \in \{0,1,2\} \qquad \forall j \in V \setminus \{0\}, k \in K \tag{13}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in V \setminus \{0\}, j \in V \setminus \{0\}, i < j, k \in K \tag{14}$$

$$y^{B}_{pk} \in \{0,1\} \qquad \forall p \in P, k \in K \tag{15}$$

$$y^{I}_{pk} \in \left\{0,1,\ldots,\left\lfloor \frac{Q}{q^{unit}} \right\rfloor\right\} \qquad \forall p \in P, k \in K \tag{16}$$

$$z_{ik} \in \{0,1\} \qquad \forall i \in V, k \in K. \tag{17}$$

The objective function (1) describes the total cost of all tours, which has to be minimized. Constraints (2) ensure that each supply is assigned to exactly one vehicle, whereas constraints (3) ensure that if a supply is assigned to a specific vehicle, also the respective location is visited by this vehicle. Constraints (4) guarantee that the depot is included on each tour, while constraint (5) limits the maximal number of vehicles to be used. Constraints (6) represent the node degree constraints. Constraints (7) define the capacity constraints of the compartments, whereas constraints (8) describe the capacity constraints of the vehicles. Constraints (9) limit the number of compartments in a vehicle

to the maximal number of compartments. Constraints (10) define the relation between binary variables y^B_{pk} and integer variables y^I_{pk}. Constraints (11) are the subtour elimination constraints. Finally, constraints (12) to (17) define the variable domains.

The above formulation differs from the one presented by Henke et al. [16] mainly with respect to the integer variables y^I_{pk} and the binary variables y^B_{pk}. Instead of using these variables, in the formulation by Henke et al. [16] only binary variables y_{pkm} have been used. These variables indicate whether product type p is assigned to a compartment of vehicle k with compartment size m, where m indicated the selected size out of a set of potential compartment sizes. Accordingly, constraints (7), (8), (9), and (10) have been either modified or added. The changes resulted in significantly improved computing times, as will be shown in Section 5.5.

3 Literature Review

In recent years, several variants of the MCVRP have been discussed in the literature. Coelho and Laporte [7] present a classification scheme, motivated by the context of fuel distribution, which distinguishes between split and unsplit compartments on the one hand and between split and unsplit tanks on the other hand. Split compartments indicate that the content of one compartment can be distributed to more than one customer location, whereas this is not the case with unsplit compartments. Similarly, split tanks indicate that a demand for the same product type of one customer location can be satisfied by more than one vehicle. In addition to this classification they propose models and a branch-and-cut algorithm which is able to solve all problem variants presented by the authors. Another branch-and-cut algorithm is proposed by Lahyani et al. [18], who describe a problem in the context of olive oil collection. They consider fixed compartments sizes and a heterogeneous vehicle fleet.

El Fallahi et al. [11] introduce a problem which arises in the context of animal food distribution. They consider fixed compartment sizes and an a priori fixed assignment of product types to compartments. In order to solve this problem, they present a memetic algorithm and a tabu search. A similar problem variant is considered by Muyldermans and Pang [22] who introduce a guided local search. An application in the context of convenience store deliveries is described by Chajakis and Guignard [6], for which a heuristic based on Lagrangean relaxation is proposed. In order to solve a similar problem, Reed et al. [25] introduce an ant colony optimization algorithm, and Abdulkader et al. [1] solve a distance-constrained variant of the problem by application of a hybrid ant colony optimization algorithm. Rabbani et al. [23] consider a distance-constrained variant of the MCVRP with heterogeneous vehicles, multiple depots and mixed open and closed tours, for which they develop a genetic algorithm.

Another variant of the MCVRP, in which stochastic instead of deterministic demands are considered, is presented by Mendoza et al. [20, 21]. They present several construction procedures and a memetic

algorithm. Goodson [15] proposes a simulated annealing algorithm for a similar problem. A multi-periodic variant of the stochastic problem with two product types is considered by Elbek and Wøhlk [12]. They introduce a heuristic which consists of a construction phase and a variable neighborhood search, which is implemented in a rolling planning horizon framework.

MCVRPs are frequently encountered in petrol replenishment. In these problems, compartment sizes are usually fixed in advance, but the assignment of product types to specific compartments is not. Since in most examples vehicles are not equipped with load meters, the content of each compartment can only be delivered to one customer. Such problems are described by Brown and Graves [4] and Avella et al. [3]. A multi-periodic extension is considered by Cornillier et al. [8] and Vidović et al. [27].

Derigs et al. [10] describe two variants of the MCVRP, namely problems with fixed compartment sizes on one hand and with flexible compartments sizes on the other. In both variants, the assignment of product types to compartments is a decision variable. They introduce a large neighborhood search which is able to solve all of their considered problem variants. A problem in the context of milk collection with fixed compartment sizes, a heterogeneous vehicle fleet, trailers, and without an a-priori assignment of product types to compartments is described by Caramia and Guerriero [5]. They present a two-part heuristic.

Another frequent application of MCVRPs occurs in maritime transportation [13]. Theoretical comparison between vehicle routing problems with one product type and multiple product types is presented by Archetti et al. [2]. They give theoretical and empirical findings between three variants of vehicle routing problems, namely the split delivery vehicle routing problem, the multi-compartment vehicle routing problem, and a combined split delivery and multi-compartment vehicle routing problem.

Overall, specific problems considered in the MCVRP context are quite heterogeneous. To the best of our knowledge, the only contribution which deals with discretely flexible compartment sizes and no a priori given assignment of product types to vehicles is the paper by Henke et al. [16], who propose a variable neighborhood search to solve the MCVRP-DFCS. A slightly different problem with continuously flexible compartment sizes is considered by Koch et al. [17]. They propose a genetic algorithm to solve this variant. However, no specific exact algorithm has been developed in order to solve the MCVRP-DFCS. This paper aims at closing this gap.

4 Branch-and-Cut Algorithm

4.1 Overview

The formulation described in Section 2 has been solved with a branch-and-cut algorithm. The model formulation has been enhanced by adding several types of valid inequalities, capacity cuts and a specific branching scheme. Moreover, the subtour elimination constraints (11) are not included from

the beginning but are instead added by application of a separation procedure. The components of the algorithm are described in detail in the following subsections.

4.2 Separation of Subtour-Elimination Constraints

As the number of subtour elimination constraints grows exponentially with the problem size, only for small instances all constraints can be generated and added to the model by a solver. A frequently-used option for overcoming this drawback consists of starting the solution process without any subtour-elimination constraints and then adding them dynamically during the solution process by means of a separation procedure. For the algorithm described in this paper, a simple procedure has been implemented: Each time an integer solution is found for a problem at a particular node, the solution is searched for subtours. If the solution possesses at least one subtour, the subtour with the smallest number of vertices and the respective set of vertices S is determined. Subsequently, the subtour-elimination constraint (11) for this set S is added to the pool of constraints and the problem represented by the respective node is solved again.

4.3 Valid Inequalities

To further strengthen the model formulation, several types of valid inequalities have been derived or adapted from the literature. On one hand, logical inequalities in the form of big-M-constraints have been developed which describe valid relations between two types of variables; on the other hand, symmetry breaking constraints have been adapted from the literature. All constraints have been tested with respect to how they contribute to decreasing the computing time. A subset of the most beneficial constraints has been implemented in the branch-and-cut algorithm.

For developing logical valid inequalities, all pairs of decision variables have been considered which have at least two indices in common. For a specific pair of variables, a big-M-constraint was derived if no constraint had already been included in the basic model which describes a relation between these variables. Table 1 indicates for each pair of variables whether a corresponding set of constraints has already been part of the basic model (respective inequalities given in italics), or whether a new set of valid inequality has been derived (respective inequalities underlined).

$\dots \leq \dots$	$\mathbf{u_{ipk}}$	$\mathbf{x_{ijk}}$	$\mathbf{y^B_{pk}}$	$\mathbf{y^I_{pk}}$	$\mathbf{z_{ik}}$
$\mathbf{u_{ipk}}$	-	<u>(18)</u>	<u>(19)</u>	*(7)*	*(3)*
$\mathbf{x_{ijk}}$	<u>(20)</u>	-	-	-	*(6)*
$\mathbf{y^B_{pk}}$	<u>(21)</u>	-	-	<u>(22)</u>	-
$\mathbf{y^I_{pk}}$	<u>(23)</u>	-	*(10)*	-	-
$\mathbf{z_{ik}}$	<u>(24)</u>	*(6)*	-	-	-

Table 1: Overview of relations between variables

The seven types of inequalities which have been newly derived are shown in the following explicitly. A detailed explanation for each type of constraint is not given as these constraints are only describing logical relationships between variables.

$$\sum_{p \in P} u_{ipk} \leq \frac{\hat{m}}{2} \cdot \left(\sum_{\substack{j \in V: \\ i<j}} x_{ijk} + \sum_{\substack{j \in V: \\ j<i}} x_{jik} \right) \qquad \forall\, i \in V \setminus \{0\}, k \in K \quad (18)$$

$$\sum_{i \in V \setminus \{0\}} u_{ipk} \leq |V| \cdot y^{B}_{pk} \qquad \forall\, p \in P, k \in K \quad (19)$$

$$\sum_{\substack{j \in V: \\ i<j}} x_{ijk} + \sum_{\substack{j \in V: \\ j<i}} x_{jik} \leq 2 \cdot \sum_{p \in P} u_{ipk} \qquad \forall\, i \in V \setminus \{0\}, k \in K \quad (20)$$

$$y^{B}_{pk} \leq \sum_{i \in V \setminus \{0\}} u_{ipk} \qquad \forall\, p \in P, k \in K \quad (21)$$

$$y^{B}_{pk} \leq y^{I}_{pk} \qquad \forall\, p \in P, k \in K \quad (22)$$

$$y^{I}_{pk} \leq \frac{Q}{q^{unit}} \cdot \sum_{i \in V \setminus \{0\}} u_{ipk} \qquad \forall\, p \in P, k \in K \quad (23)$$

$$z_{ik} \leq \sum_{p \in P} u_{ipk} \qquad \forall\, i \in V \setminus \{0\}, k \in K. \quad (24)$$

In addition to the logical valid inequalities, also symmetry breaking constraints have been adapted from the literature. With respect to the MCVRP-DFCS, symmetry can occur between two solutions when all tours are identical but the assigned vehicles are different. Symmetry within a tour cannot occur as the undirected formulation has been used. In order to overcome the symmetry between solutions, the following three types of constraints have been adapted and tested:

$$z_{0,k+1} \leq z_{0k} \qquad \forall\, k \in K \setminus \{|K|\} \quad (25)$$

$$\sum_{i \in V} \sum_{\substack{j \in V: \\ i<j}} c_{ij} \cdot x_{ij,k+1} \leq \sum_{i \in V} \sum_{\substack{j \in V: \\ i<j}} c_{ij} \cdot x_{ijk} \qquad \forall\, k \in K \setminus \{|K|\} \quad (26)$$

$$\sum_{i \in V \setminus \{0\}} \sum_{p \in P} s_{ip} \cdot u_{ip,k+1} \leq \sum_{i \in V \setminus \{0\}} \sum_{p \in P} s_{ip} \cdot u_{ipk} \qquad \forall\, k \in K \setminus \{|K|\}. \quad (27)$$

Constraints (25) ensure that a vehicle with index k+1 can only be used if the vehicle with index k is also used. Constraints (26) ensure that tours are ordered in such a way that a tour with a higher index cannot have a total cost higher than a tour with a lower index. Similarly, constraints (27) order the tours with respect to non-decreasing total supplies.

A set of preliminary experiments allowed us to determine that a combination of the logical valid inequalities (19), (22), (23) leads to the highest reduction of the computing time. Regarding the symmetry breaking constraints, constraints (26) have shown to be most beneficial. Therefore, these four types of inequalities have been added to the basic model.

4.4 Capacity Cuts

As additional valid inequalities, also an adapted version of the well-known capacity cuts has been added, which were introduced by Cornuejols and Harche [9] for the CVRP. Equation (28) shows the original form of these cuts. The basic idea is that the number of selected edges between a subset of nodes S which does not include the depot and the set of all remaining nodes $V\backslash S$ must be larger than or equal to a theoretical minimum on the number of vehicles which are necessary for satisfying the demands of all nodes in subset S:

$$\sum_{i\in S}\sum_{\substack{j\in V\backslash S:\\ i<j}}\sum_{k\in K} x_{ijk} + \sum_{i\in S}\sum_{\substack{j\in V\backslash S:\\ j<i}}\sum_{k\in K} x_{jik} \geq 2 \cdot \left\lceil \frac{\sum_{i\in S\backslash\{0\}} d_i}{Q} \right\rceil \qquad \forall\, S \subseteq V \setminus \{0\},\ |S| > 1. \tag{28}$$

The adapted capacity cuts for the MCVRP-DFCS are shown in equations (29). The left-hand side, which is similar to the original capacity cuts for the CVRP, must be greater than or equal to the maximum of two alternative lower bounds, LB_1 and LB_2:

$$\sum_{i\in S}\sum_{\substack{j\in V\backslash S:\\ i<j}}\sum_{k\in K} x_{ijk} + \sum_{i\in S}\sum_{\substack{j\in V\backslash S:\\ j<i}}\sum_{k\in K} x_{jik} \geq \max\{LB_1, LB_2\} \qquad \forall\, S \subseteq V \setminus \{0\},\ |S| > 1. \tag{29}$$

LB_1 (see equation (30)) is based on the original idea of the capacity cuts but, additionally, takes into account that compartment capacities can only be multiples of the basic compartment unit size:

$$LB_1 = 2 \cdot \left\lceil \frac{\sum_{p\in P} \left\lceil \frac{\sum_{i\in S} s_{ip}}{q^{unit}} \right\rceil \cdot q^{unit}}{Q} \right\rceil. \tag{30}$$

LB_2 (see equation (31)), in contrast to LB_1, focuses on the maximum number of compartments. The numerator in the fraction of (31) corresponds to a lower bound on the number of compartments which are necessary in order to collect all supplies, whereas the denominator indicates the maximum number of compartments into which the vehicle capacity can be separated. In the numerator, two different cases are distinguished in which the use of a compartment becomes necessary. Parameter $\grave{s}_p$ indicates whether there is at least one supply for a certain product type in subset S (case 1). Moreover, parameter $\tilde{s}_p$ indicates whether it might be necessary to use more than only one compartment for a certain product type (case 2) and, if so, the parameter takes the value of the additional number of

compartments needed. At least more than one compartment for a single product type is needed, when the sum of supplies for this product type in S is greater than the vehicle capacity:

$$LB_2 = 2 \cdot \left\lceil \frac{\sum_{p \in P} (\grave{s}_p + \tilde{s}_p)}{\hat{m}} \right\rceil \qquad (31)$$

$$\grave{s}_p = \begin{cases} 1, \text{ if } \sum_{i \in S} s_{ip} > 0, \\ 0, \text{ else.} \end{cases}$$

$$\tilde{s}_p = \left\lceil \frac{\sum_{i \in S} s_{ip}}{Q} \right\rceil.$$

Let, for example, a vehicle with two compartments and supplies for four different product types (A, B, C, and D) in subset S be given. Then, according to case 1, at least $\lceil (1 + 1 + 1 + 1)/2 \rceil = 2$ vehicles are necessary to collect all supplies. If, moreover, the total supply of product type A would amount to 1.2Q, then at least two compartments in two different vehicles are necessary in order to collect all supplies of product type A. Thus, at least $\lceil (4 + 1)/2 \rceil = 3$ vehicles would be needed to collect all supplies.

As the number of capacity cuts increases exponentially with the number of nodes in the problem, in order to separate these cuts, we use the shrinking heuristic introduced by Ralphs et al. [24].

4.5 Branching Scheme

Finally, the sequence according to which variables are selected for branching has been further specified. Having selected any node for branching, the y^B_{pk}-variables are considered first. Only if all y^B_{pk}-variables are already integral, branching according to z_{ik}-variables is considered, then according to x_{ijk}-variables and finally according to u_{ipk}- and y^I_{pk}-variables with same priority. This sequence has been determined in preliminary tests; it is comprehensible, too, since the overall impact of a decision variable on the composition of a solution decreases according to this sequence. However, the stated sequence is changed slightly if instances are considered in which the number of compartments is not smaller than the number of product types. In such a case, the highest priority is given to the z_{ik}-variables, then to the x_{ijk}-variables, and finally y^B_{pk}-variables, u_{ipk}- and y^I_{pk}-variables with same priority.

5 Computational Experiments

5.1 Overview

In order to evaluate the performance of the branch-and-cut algorithm proposed for the MCVRP-DFCS, two sets of experiments have been performed. In the first set, related to previously published instances with 10 customer locations, the performance of the branch-and-cut algorithm is compared to the performance of the exact solution approach presented in Henke et al. [16]. In the second set of experiments, the performance of the algorithm on larger problem instances is analyzed in greater

detail. For this purpose, new instances with up to 50 locations have been generated randomly. Furthermore, the contribution of specific algorithmic features to the performance of the branch-and-cut algorithm are studied on a subset of the newly generated instances. A third set of experiments has been introduced in order to gain additional managerial insights. In these experiments the costs have been investigated which result from the use of discretely flexible compartments instead of continuously flexible compartments.

The algorithm has been implemented on a 3.2 gigahertz and 8 gigabytes RAM computer using C++ and the interface with CPLEX 12.6. For the separation procedure, a lazy-cut-callback procedure has been used, whereas the capacity cuts are added within a user-cut-callback procedure.

5.2 Problem Instances

For the first set of experiments, the instances previously introduced by Henke et al. [16] (called "small instances" here) have been used. These instances are characterized by 10 customer locations and differ with respect to three parameters: the number of product types $|P|$, the maximal number of compartments $\hat{m}$, and a supply parameter $\bar{s}$. The latter indicates whether the total number of positive supplies is small (1), medium (2), or large (3). For the number of product types $|P|$, 3, 6, and 9 product types, respectively, have been assumed. For the maximal number of compartments $\hat{m}$, 2 or 3 compartments have been considered for instances with three product types, 2, 4 or 6 compartments for instances with 6 product types; and 2, 4, 7 or 9 compartments for instances with 9 product types. This gives rise to 27 problem classes, for which Henke et al. [16] generated 50 instances each. Thus, in total, this problem set includes 1,350 instances.

For the second set of experiments, 675 larger instances with up to 50 locations were newly generated. The instances have been obtained by means of a slight modification of the instance generator described by Henke et al. [16]. The modification concerned the vehicle capacity and the average number of locations assigned to a vehicle. While Henke et al. [16] generated instances in which all vehicle capacities were fixed to the same value and the supplies were generated in such a way that the average number of locations per vehicle remained constant, we decided to consider the number of locations as a problem parameter to be investigated and, thus, introduced a flexible vehicle capacity dependent on the number of locations. As a result, the number of vehicles in the obtained instances remains relatively constant. Furthermore, the parameter specifications on the number of product types and the maximal number of compartments have been changed. For the newly generated instances either 3 or 4 product types were assumed. In case of instances with three product types, 2 or 3 compartments have been considered, while in case of four product types the number of compartments has been set to 2, 3 or 4. For the experiments with respect to the evaluation of the algorithmic components, a subset of 50 instances was selected randomly from the newly generated instances in such a way that for each considered number of locations, at least two instances were included in the subset. As we expected the computing times to increase substantially for these experiments, only

instances were considered for which the computing time of the branch-and-cut algorithm did not exceed ten minutes. The same set of instances has been used for the third set of experiments.

5.3 Experiments with Instances from the Literature

Regarding the small instances from Henke et al. [16], the branch-and-cut algorithm was able to solve all instances to optimality. Table 2 shows the detailed results clustered according to the number of product types, the maximal number of compartments, and the supply parameter. The table also depicts the average total cost for each cluster (avg.tc), the average computing time for the method of Henke et al. [16] (cpu.old), the average computing time of the proposed branch-and-cut algorithm (cpu.new), and the average improvement in computing time (cpu.impr). It can be observed that the branch-and-cut algorithm provides a significant improvement of the computing times, which, on average, decreased by 87.19% from 431.31 seconds to 43.02 seconds across all instances.

parameters			avg.tc	cpu.old	cpu.new	cpu.impr
$\|P\|$	$\hat{m}$	$\bar{s}$				
3	2	1 (small)	366.90	5.43	0.27	95.10%
		2 (medium)	465.35	11.43	0.32	97.24%
		3 (large)	596.16	101.33	0.35	99.65%
	3	1 (small)	349.08	2.60	0.17	93.58%
		2 (medium)	337.15	2.66	0.19	92.77%
		3 (large)	336.64	5.00	0.19	96.15%
6	2	1 (small)	549.98	5.91	0.45	92.39%
		2 (medium)	767.48	4.83	0.65	86.62%
		3 (large)	919.78	1.81	0.88	51.67%
	4	1 (small)	366.98	5.04	0.72	85.63%
		2 (medium)	492.11	27.79	1.46	94.75%
		3 (large)	583.82	326.15	4.57	98.60%
	6	1 (small)	351.98	4.05	0.32	92.10%
		2 (medium)	357.98	5.94	0.49	91.80%
		3 (large)	357.42	8.96	0.57	93.69%
9	2	1 (small)	724.17	502.44	37.83	92.47%
		2 (medium)	1,114.25	1,921.59	127.39	93.37%
		3 (large)	1,419.09	5,400.85	646.32	88.03%
	4	1 (small)	463.05	12.87	2.44	81.01%
		2 (medium)	647.57	267.69	11.59	95.67%
		3 (large)	815.27	1,324.58	20.15	98.48%
	7	1 (small)	360.16	5.25	5.23	0.41%
		2 (medium)	456.33	93.21	8.57	90.81%
		3 (large)	564.50	1,550.47	287.27	81.47%
	9	1 (small)	354.98	3.94	0.58	85.28%
		2 (medium)	392.48	10.53	1.05	90.08%
		3 (large)	383.36	32.94	1.56	95.27%
minimum			336.64	1.81	0.17	0.41%
average			551.63	431.31	43.02	87.19%
maximum			1,419.09	5,400.85	646.32	99.65%

Table 2: Results for the instances from the literature

5.4 Experiments with New Instances

5.4.1 Evaluation of the Algorithm on Larger Instances

For the experiments with the newly generated instances, a time limit of two hours has been set for each instance. The results for the 675 instances have been clustered according to the four problem parameters, i.e. the number of locations (n), the number of product types ($|P|$), the maximum number of compartments ($\hat{m}$), and the supply parameter ($\bar{s}$), in Tables 3 to 6, respectively. In each table and for each cluster the number of instances in the cluster (#inst), the average total cost (avg.tc), the number of optimally solved instances in the cluster (#opt), the relative number of optimally solved instances per cluster (rel.opt), the average gap (avg.gap), the maximal gap (max.gap), as well as the minimal (min.cpu), average (avg.cpu), and maximal computing times (max.cpu) are given.

Table 3 shows the results clustered according to the number of locations. As expected, the performance of the algorithm decreases with an increasing number of locations. For all instances with up to 25 locations the algorithm was able to determine optimal solutions. For the remaining instances the relative number of optimally solved instances decreases from 97.3% for instances with 30 locations down to 66.7% for instances with 50 locations, i.e. even for the largest instances the algorithm was able to find optimal solutions for two out of three instances. Similar changes can be observed with respect to the gaps and computing times. The average gap ranges from 0.04% for instances with 30 locations to 1.71% for instances with 50 locations, and the average computing times from 0.51 seconds for instances with 10 locations to 2,883.37 seconds for instances with 50 locations.

n	#inst	avg.tc	#opt	rel.opt	avg.gap	max.gap	min.cpu	avg.cpu	max.cpu
10	75	447.62	75	100.0%	0.00%	0.00%	0.06	0.51	5.82
15	75	518.57	75	100.0%	0.00%	0.00%	0.06	1.47	18.25
20	75	566.24	75	100.0%	0.00%	0.00%	0.08	5.82	56.87
25	75	626.30	75	100.0%	0.00%	0.00%	0.17	95.69	4,207.43
30	75	663.04	73	97.3%	0.04%	1.90%	0.23	312.57	7,201.13
35	75	710.30	71	94.7%	0.08%	2.13%	0.30	793.22	7,207.21
40	75	750.23	61	81.3%	0.71%	8.16%	0.62	1,609.24	7,202.49
45	75	781.40	52	69.3%	1.38%	11.20%	1.25	2,625.75	7,202.75
50	75	812.14	50	66.7%	1.71%	16.00%	0.66	2,883.37	7,219.71

Table 3: Results for the generated instances clustered according to the number of locations

Table 4 summarizes the results for all instances clustered according to the number of product types. Although the differences between the two clusters are not very large, it can be observed that the algorithm performs better on instances with a smaller number of product types. The relative number of optimally solved instances decreases from 92.6% to 88.1%, the average gap increases from 0.34% to 0.50%, and the average computing time increases from 694.39 seconds to 1,079.23 seconds.

$\|P\|$	#inst	avg.tc	#opt	rel.opt	avg.gap	max.gap	min.cpu	avg.cpu	max.cpu
3	270	628.24	250	92.6%	0.34%	9.35%	0.06	694.39	7,202.51
4	405	669.29	357	88.1%	0.50%	16.00%	0.09	1,079.23	7,219.71

Table 4: Results for the generated instances clustered according to the number of product types

Table 5 demonstrates the effect of the maximum number of compartments on the performance of the algorithm. Instead of presenting the results clustered according to this parameter only, also the number of product types has been taken into account. We do so because a similar maximum number of compartments might imply different consequences on the problem structure when the number of product types is different. For example, for instances with three product types and instances with four product types, it can be observed that the performance of the algorithm improves with an increasing maximum number of compartments, although the maximum number of compartments does not have an impact on the problem size. An explanation for this is that the decision of assigning product types to vehicles becomes more difficult when fewer compartments are available. Similar findings have also been reported by Henke et al. [16]. For the case with three product types, the relative number of optimally solved instances increases from 82.5% with two compartments to 100.0% with three compartments. For the case with four product types, only 82.2% of the instances with two compartments were solved to optimality, whereas 97.8% of the instances with four compartments were solved to optimality. The findings with respect to gaps and computing times show similar trends. When clusters with the same maximum number of compartments, but different product types are compared, the performance of the algorithm decreases slightly when the number of product types increases, which further underlines the findings from Table 4.

$\|P\|$	$\hat{m}$	#inst	avg.tc	#opt	rel.opt	avg.gap	max.gap	min.cpu	avg.cpu	max.cpu
3	2	135	740.14	115	85.2%	0.68%	9.35%	0.06	1,221.94	7,202.51
	3	135	516.34	135	100.0%	0.00%	0.00%	0.07	166.83	3,496.57
4	2	135	808.37	111	82.2%	0.80%	16.00%	0.22	1,499.65	7,219.71
	3	135	674.38	114	84.4%	0.65%	11.33%	0.10	1,434.20	7,209.30
	4	135	525.12	132	97.8%	0.05%	2.59%	0.09	303.84	7,202.20

Table 5: Results for the generated instances clustered according to the number of product types and the maximal number of compartments

Finally, Table 6 shows the results clustered according to the supply parameter. Here, no clear trend can be observed. The performance of the algorithm seems to be best when the number of supplies is large. However, in general, the number of supplies has little effect on the performance of the algorithm.

$\bar{s}$	#inst	avg.tc	#opt	rel.opt	avg.gap	max.gap	min.cpu	avg.cpu	max.cpu
1	225	561.16	202	89.8%	0.56%	16.00%	0.09	855.36	7,219.71
2	225	654.94	200	88.9%	0.30%	9.35%	0.14	1,085.94	7,202.75
3	225	742.50	205	91.1%	0.45%	9.22%	0.06	834.58	7,202.50

Table 6: Results for the generated instances clustered according to the number of supplies

5.4.2 Evaluation of the Components of the Algorithm

In order to investigate the different components of the algorithm further, we conducted experiments on a subset of 50 of the newly generated instances. In the first part of these experiments, the impact of the change in the model formulation (see Section 2) has been analyzed. For this purpose, the 50 instances have been solved by implementing the formulation of Henke et al. [16] and the modified

formulation proposed in this paper without any added components, i.e. without valid inequalities, capacity cuts, or variable priorities. Table 7 shows the corresponding results, clustered according to the number of locations. For each cluster, the table indicates the number of instances (#inst), the number of instances which have been solved to optimality by the formulation of Henke et al. [16] (#opt.old) and by the formulation introduced in Section 2 (#opt.new), the average gaps achieved by each implemented model (gap.old, gap.new) and the gap improvement (gap.impr), as well as the average computing times for each formulation (cpu.old, cpu.new) and the respective reduction of computing time (cpu.impr). The new formulation outperformed the old formulation in all clusters. Although its implementation was only able so solve three additional instances to optimality, the reduction of the average gaps from 5.68% to 2.79% and the reduction of the average computing times from 4,014.74 seconds to 2,904.60 seconds demonstrate the superiority of the changes in the model formulation.

n	#inst	#opt.old	#opt.new	gap.old	gap.new	gap.impr	cpu.old	cpu.new	cpu.impr
10	2	2	2	0.01%	0.01%	1.35%	2541.77	11.33	99.55%
15	5	5	5	0.01%	0.01%	28.03%	512.92	10.19	98.01%
20	7	6	6	1.93%	0.66%	65.66%	1931.11	1083.13	43.91%
25	5	4	4	2.63%	0.41%	84.39%	3179.51	1519.00	52.23%
30	5	2	3	6.12%	1.09%	82.20%	4452.54	3034.38	31.85%
35	7	0	0	15.75%	8.05%	48.86%	7213.26	7206.70	0.09%
40	8	3	4	8.10%	4.56%	43.74%	5374.59	3847.53	28.41%
45	6	2	2	8.79%	5.02%	42.87%	4923.36	4858.49	1.32%
50	5	1	2	7.75%	5.29%	31.81%	6003.58	4570.65	23.87%
sum		25	28						
average				5.68%	2.79%	47.66%	4014.74	2904.60	42.14%

Table 7: Comparison of model formulations

In each of the experiments summarized in Table 8, exactly one algorithmic component has been eliminated from the algorithm, e.g. valid inequalities were not included or the specified branching scheme has not been defined. The first column of Table 8 shows the component which has been neglected. These components are the symmetry breaking constraints, the three types of logical valid inequalities, the capacity cuts, and the definition of variable priorities. In the second column, the average computing times (avg.cpu) are given. In column 3, the relative number of optimally solved instances (rel.opt) is listed and, in column 4, the average gap (avg.gap) is given. As indicated in the last row, the algorithm with all components was able to solve all 50 instances to optimality and, therefore, the relative number of optimally solved instances amounts to 100% and the corresponding average gap is 0.00%. It can be observed that the elimination of any of these six components leads to an increase in the average computing time and a decrease in the overall algorithmic performance. Interestingly, the contribution to the algorithmic performance is very different for each component. Elimination of the capacity cuts (elimination of the variable priorities) increased the average computing times from 81.48 seconds to 1,851.14 seconds (543.80 seconds), while the effect resulting from an elimination of the remaining components is much smaller. With respect to the number of

optimally solved instances, similar findings can be identified. Only 80% of the instances were solved to optimality when the capacity cuts were eliminated, 96% when the variable priorities were neglected, and 98% when valid inequalities (23) were removed from the formulation. For the remaining three components, no change occurred. Finally, the average gaps show similar results. Without capacity cuts an average gap of 1.45% occurred, 0.12% without variable priorities, 0.02% without valid inequalities (23), and 0.00% for the remaining components. Still these components lead to an increase in the performance of the algorithm and have, therefore, been included in its final version.

Relaxed component	**avg.cpu**	**rel.opt**	**avg.gap**
all components	3,273.07	56.00%	3.23%
Capacity cuts	1,851.14	80.0%	1.45%
Variable priorities	543.80	96.0%	0.12%
Valid inequalities (23)	205.44	98.0%	0.02%
Valid inequalities (19)	125.14	100.0%	0.00%
Symmetry breaking constraints (26)	90.37	100.0%	0.00%
Valid inequalities (22)	68.19	100.0%	0.00%
no relaxation	81.48	100.00%	0.00%

Table 8: Results for the design experiments

5.5 Cost of Discretization

In order to gain additional managerial insights related to the problem specific property of compartment discretization, the branch-and-cut algorithm has also been used for an experiment in which the total cost with continuously flexible compartment sizes were compared to the total cost of discretely flexible compartment sizes. To obtain optimal solutions for the case of continuously flexible compartment sizes, the only necessary change in the branch-and-cut algorithm was to set the basic unit compartment size equal to 1.

Table 9 displays the findings of this experiment clustered according to the number of locations. In the second column, the average total cost for the case of continuously flexible compartments (avg.tc) is indicated, and in the third column the average reduction of the total cost compared to the discretely flexible case (red.tc) is shown.

n	**avg.tc**	**red.tc**
10	458.31	5.89%
15	485.13	4.23%
20	610.41	0.59%
25	628.06	3.41%
30	666.72	1.97%
35	627.07	2.98%
40	672.97	1.10%
45	717.70	0.52%
50	727.08	0.81%

Table 9: Cost of discretization

Overall the total cost can be reduced by 2.39% when discretely flexible compartments are substituted by continuously flexible compartments. It can be observed that the potential for cost savings tends to decrease with the number of locations. Whereas 5.89% of total cost can be saved on instances with 10 locations, only 0.81% can be saved on instances with 50 locations.

Further analyses have shown that the saving potential is higher with an increasing number of product types (1.25% for three product types, 2.42% for four product types) and an increasing maximum number of compartments (0.41% for two compartments, 2.43% for three compartments, and 3.97% for four compartments). Again, for the supply parameter no general trend could be observed.

6 Summary and Outlook

In this paper, a multi-compartment vehicle routing problem with discretely flexible compartment sizes (MCVRP-DFCS) has been investigated and a branch-and-cut algorithm has been proposed. The presented algorithm has been tested in extensive numerical experiments based on 1,350 instances from the literature and 675 newly generated instances. The results show that the proposed method outperforms the existing method by 87.19% on average with respect to the computing time. The algorithm is able to solve all newly generated instances with up to 25 locations and 2/3 of instances with 50 locations to optimality within two hours. Even for instances with 50 locations the average gap amounted to 1.71% only. Additional experiments show that savings of 2.39% can be achieved if vehicles with continuous compartment sizes are used instead of vehicles with discrete compartment sizes.

Further research directions related to the MCVRP-DFCS are, amongst others, to consider a multi-periodic context and stochastic supplies instead of deterministic ones. Both aspects can be found in the real-world application of glass waste collection but have not yet been investigated in this context.

References

[1] Abdulkader MMS, Gajpal Y, ElMekkawy TY (2015): Hybridized ant colony algorithm for the multi compartment vehicle routing problem. In: Appl Soft Comput 37:196-203.

[2] Archetti C, Campbell A, Speranza MG (2016): Multi-commodity vs. single-commodity routing. In: Transp Sci 50:461-472.

[3] Avella P, Boccia M, Sforza A (2004): Solving a fuel delivery problem by heuristic and exact approaches. In: Eur J Oper Res 152:170-179.

[4] Brown GG, Graves GW (1981): Real-time dispatch of petroleum tank trucks. In: Manag Sci 27:19-32.

[5] Caramia M, Guerriero F (2010): A milk collection problem with incompatibility constraints. In: Interfaces 40:130-143.

[6] Chajakis ED, Guignard M (2003): Scheduling deliveries in vehicles with multiple compartments. In: J Glob Optim 26:43-78.

[7] Coelho LC, Laporte G (2015): Classification, models and exact algorithms for multi-compartment delivery problems. In: Eur J Oper Res 242:854-864.

[8] Cornillier F, Boctor FF, Laporte G, Renaud J (2008): A heuristic for the multi-period petrol station replenishment problem. In: Eur J Oper Res 191:295-305.

[9] Cornuejols G, Harche F (1993): Polyhedral study of the capacitated vehicle routing problem. In: Math Program 60:21-52.

[10] Derigs U, Gottlieb J, Kalkoff J, Piesche M, Rothlauf F, Vogel U (2011): Vehicle routing with compartments: applications, modelling and heuristics. In. OR Spectr 33:885-914.

[11] El Fallahi A, Prins C, Wolfer Calvo R (2008): A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. In: Comput Oper Res 35:1725-1741.

[12] Elbek M, Wøhlk S (2016): A variable neighborhood search for the multi-period collection of recyclable materials. In: Eur J Oper Res 249:540-550.

[13] Fagerholt K, Christiansen M (2000): A combined ship scheduling and allocation problem. In: J Oper Res Soc 51:834-842.

[14] Golden BL, Raghavan S, Wasil EA (2008): The vehicle routing problem: latest advances and new challenges. Springer, New York.

[15] Goodson JC (2015): A priori policy evaluation and cyclic-order-based simulated annealing for the multi-compartment vehicle routing problem with stochastic demands. In: Eur J Oper Res 241:361-369.

[16] Henke T, Speranza MG, Wäscher G (2015): The multi-compartment vehicle routing problem with flexible compartment sizes. In: Eur J Oper Res 246:730-746.

[17] Koch H, Henke T, Wäscher G (2016): A genetic algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes. Working Paper No. 04/2016, Fakultät für Wirtschaftswissenschaft, Otto-von-Guericke-Universität Magdeburg.

[18] Lahyani R, Coelho LC, Khemakhem M, Laporte G, Semet F (2015): A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. In: Omega 51:1-10.

[19] Laporte G (2009): Fifty years of vehicle routing. In: Transp Sci 43:408-416.

[20] Mendoza JE, Castanier B, Guéret C, Medaglia AL, Velasco N (2010): A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. In: Comput Oper Res 37:1886-1898.

[21] Mendoza JE, Castanier B, Guéret C, Medaglia AL, Velasco N (2011): Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. In: Transp Sci 45:346-363.

[22] Muyldermans L, Pang G (2010): On the benefits of co-collection: experiments with a multi-compartment vehicle routing problem. In: Eur J Oper Res 206:93-103.

[23] Rabbani M, Farrokhi-asl H, Rafiei H (2016): A hybrid genetic algorithm for waste collection problem by heterogeneous fleet of vehicles with multiple separated compartments. In: J Intell Fuzzy Syst 30:1817-1830.

[24] Ralphs TK, Kopman L, Pulleyblank WR, Trotter LE (2003): On the capacitated vehicle routing problem. In: Math Program, Ser B 94:343-359.

[25] Reed M, Yiannakou A, Evering R (2014): An ant colony algorithm for the multi-compartment vehicle routing problem. In: Appl Soft Comput 15:169-176.

[26] Toth P, Vigo D (2014): Vehicle routing: Problems, methods, and applications. Society for Industrial and Applied Mathematics, Philadelphia.

[27] Vidović M, Popović D, Ratković B (2014): Mixed integer and heuristics model for the inventory routing problem in fuel delivery. In: Int J Prod Econ 147:593-604.

Otto von Guericke University Magdeburg
Faculty of Economics and Management
P.O. Box 4120 | 39016 Magdeburg | Germany

Tel.: +49 (0) 3 91 / 67-1 85 84
Fax: +49 (0) 3 91 / 67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274