Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Branch-and-Repair Method for Three-Dimensional Bin Selection and Packing in E-Commerce

Pirmin Fontaine, Stefan Minner

To cite this article:
Pirmin Fontaine, Stefan Minner (2023) A Branch-and-Repair Method for Three-Dimensional Bin Selection and Packing in E-Commerce. Operations Research 71(1):273-288. https://doi.org/10.1287/opre.2022.2369

https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 71, No. 1, January–February 2023, pp. 273–288
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Contextual Areas

A Branch-and-Repair Method for Three-Dimensional Bin Selection and Packing in E-Commerce

Pirmin Fontaine,[a,*] **Stefan Minner**[b]

[a] Catholic University of Eichstätt-Ingolstadt, Ingolstadt School of Management, 85049 Ingolstadt, Germany; [b] Technical University of Munich, School of Management and Munich Data Science Institute, 80333 Munich, Germany
*Corresponding author

Contact: pirmin.fontaine@ku.de, https://orcid.org/0000-0002-5893-7010 (PF); stefan.minner@tum.de, https://orcid.org/0000-0001-6127-8223 (SM)

Received: November 15, 2021
Revised: January 31, 2022; July 17, 2022
Accepted: August 24, 2022
Published Online in Articles in Advance: September 28, 2022

Area of Review: Transportation

https://doi.org/10.1287/opre.2022.2369

Abstract. The number of shipped parcels is continuously growing and e-commerce retailers and logistics service providers are seeking to improve logistics, particularly last-mile delivery. Since unused transportation space is a major problem in parcel distribution, one option is to improve the selection of the right parcel size for an order and the optimal packing pattern, which is known as the *three-dimensional bin packing problem* (3D-BPP). Further, the available portfolio of parcel types significantly influences the unused space. Therefore, we introduce the *three-dimensional bin selection problem* (3D-BSP) to find a portfolio of parcel types for a large set of orders. To solve the 3D-BPP with rotation of items, we propose an efficient mixed-integer linear programming formulation and symmetry-breaking constraints that are also used in the 3D-BSP for the subproblem. To solve large instances of the latter, we introduce a branch-and-repair method that improves branch-and-check. We show that our decomposition allows to relax the majority of binary decision variables in the master problem and avoids weak combinatorial cuts without further lifting. Further, we use problem-specific acceleration techniques. The numerical results based on a real-world online retailer data set show that our reformulation reduces the run time compared with existing mixed-integer linear programs for 3D-BPPs by 30% on average. For the 3D-BSP, the branch-and-repair method reduces the run time by more than two orders of magnitude compared with the mixed-integer programming formulation and can even solve instances with millions of binary decision variables and constraints efficiently. We analyze the trade-off between the costs of variety (depending on the number of parcel types) and costs for unused space. Increasing the number of parcel types reduces the unused space significantly.

Supplemental Material: The e-companion is available at https://doi.org/10.1287/opre.2022.2369.

Keywords: three-dimensional bin packing • three-dimensional bin selection • decomposition • reformulation • mixed-integer programming • e-commerce

1. Introduction

In 2018, 87 billion parcels were shipped worldwide, and this number is expected to grow to 200 billion parcels in 2025 (Pitney Bowes 2019). Ralf Kleber—country manager of Amazon in Germany—stated in an interview that for a climate-neutral transportation chain, one essential part is to reduce the unused space in parcels (Westerheide 2019). Esser and Kurte (2020) estimated that this unused space can reach up to 50% and has to be filled with spacing material that causes additional waste. If companies like FedEx use dimensional pricing (FedEx 2019), then unused space will additionally increase shipping costs. To improve efficiency and sustainability in transportation, one approach is to select the parcel type and to optimize the packing pattern of the items such that unused space in a parcel is reduced. UPS is offering a so-called Eco Responsible Packaging Programme, in which they certify customers for sustainable packaging. One part of this program is the cube optimization service that addresses the problems of packing items efficiently and optimizing the packing portfolio (UPS 2021). DHL launched the AI tool OptiCarton for package and pattern recommendation (DHL 2022); Amazon points out the importance of a right packing portfolio since their customers expect right-sized parcels that reduce waste (Amazon 2022); and Walmart puts sustainable packaging as one of their main goals within the Sustainability Hub (Walmart 2022). For the omni-channel fashion industry, Freichel et al. (2020) state that the assortment of secondary package types affects spacing requirements and complexity for dispatch employees and logistics costs at the warehouse.

In the literature, the loading of containers and bins in three-dimensional space is addressed as the three-dimensional container loading problem (CLP) or the three-dimensional bin packing problem (3D-BPP) (see, e.g., Zhao et al. 2016). A given set of three-dimensional boxes has to be placed into a given set of three-dimensional containers. The question is (1) which container to use and (2) how to place the boxes into which of the selected containers to minimize the unused space without violating any constraints such as overlapping of boxes. Additionally, rotation of items is allowed. Since already for the simpler knapsack problem humans can fail to find the optimal solution (Murawski and Bossaerts 2016), decision support tools can help to improve the utilization. Because of the difficulty of solving this problem, the literature focused on heuristics (e.g., Paquay et al. 2018). For solving this problem efficiently, we reformulate the model introduced by Paquay et al. (2016) and strengthen it with new symmetry-breaking constraints.

Having found the optimal packing pattern and parcel type from a given set for one order raises the question of the right selection of parcel types for all orders of a company. There is obviously a trade-off between a large variety of parcel types that increases complexity for packers and logistics (e.g., loss of inventory pooling effects) and a small selection that results in more unused space for many orders. Thus, the tactical planning problem of a good portfolio of parcel types can further help to improve sustainability in the operational problem of packing parcels. From a modeling standpoint, we address this problem by introducing the three-dimensional bin selection problem (3D-BSP) and formulate it as a mixed-integer linear program (MILP). The goal is to minimize the costs of unused space and parcel type variety by selecting a subset of parcel types for a given set of orders. Besides the application in e-commerce, the 3D-BSP can also help to select a set of standardized parcel types in any transportation system. For example, Living Packets (2020) is currently developing standardized environmentally friendly parcels for e-commerce. Moreover, the idea of the physical internet uses so-called π-containers to improve the efficiency of city logistics systems (Crainic and Montreuil 2016). Standardized containers are necessary to improve operational activities. According to Montreuil et al. (2014), open questions are still the sizes and the portfolio of these containers.

From a methodological point, we build on the idea of branch-and-check by Thorsteinsson (2001) and develop an extension called *branch-and-repair* for the $\mathcal{NP}$-hard 3D-BSP. We decompose the problem into a relaxed master problem that is solved with a classical branch-and-cut method. Every time an integer solution is found, a subproblem (the 3D-BPP with the selected parcel type) is solved. We prove that the decomposition allows us to relax many binary variables of the master problem due to unimodularity. The generated combinatorial cuts do not show the typical weak performance, and therefore no lifting is necessary. Further, we use the reformulation of the 3D-BPP and find several acceleration techniques for the branch-and-repair method.

We generate new data sets with up to 10 items and 90 parcel types based on real data to evaluate the performance of the 3D-BPP formulation and the branch-and-repair method. The results show that the reformulation outperforms the run time of existing MILPs on average by 30% for the 3D-BPP. For the 3D-BSP, we compare the branch-and-repair method to using a standard solver. Small instances that cannot be solved to optimality in three hours by the standard solver are solved in a few seconds using branch-and-repair. Larger instances (with about 46 million binary decision variables) are still solved to optimality within less than six hours if orders have to be packed into one parcel. Moreover, we extend the introduced methodology such that orders can be split into two parcels and show that the branch-and-repair method is still able to find good solutions.

From a managerial perspective, we analyze the trade-off between the costs for unused space and costs of parcel variety, the value of rotation, and the possibility of splitting orders into different parcels for a large real data set with almost 100,000 orders.

2. Literature Review

This paper contributes to the streams of packing literature and decomposition methods for large-scale combinatorial optimization problems, which we will briefly review in the following subsections.

2.1. Packing Literature

Bin packing problems have been studied for many decades. Following the typology of Wäscher et al. (2007), the considered problem is an input minimization problem that is characterized by a strongly heterogeneous set of bins and a strongly heterogeneous set of items. Wäscher et al. (2007) define this as the residual bin packing problem and state that it is the least investigated packing problem among all input minimization variants. Another characteristic of the packing problem is the considered dimension, which is three-dimensional in our case. Therefore, we first focus on the three-dimensional setting and then highlight several related works on the two-dimensional setting. For details on one- or two-dimensional problems, we refer to Iori et al. (2021), Delorme et al. (2016), and Castro and Oliveira (2011). The general common assumptions for 3D-BPPs in the literature are that rectangular items have to be placed orthogonally within rectangular bins without overlapping. But, there are

differences with regard to whether items are allowed to rotate and whether the bin sizes are identical or not.

One application of the three-dimensional residual bin packing problem according to Wäscher et al. (2007) is the CLP. Generally, in the CLP, rotation is allowed, and Zhao et al. (2016) give an overview of constraints considered and review existing algorithms for the CLP. The first MILP that minimizes unused space for the general setting with rotation and different bin sizes was proposed by Chen et al. (1995). In their formulation, all items are loaded into several containers. In addition, constraints like the orientation of the packed boxes are considered if an item does not allow specific rotations. For the same problem setting, Paquay et al. (2016) introduce a MILP that is adapted to real-life problems in the field of air cargo. In addition to the typical geometric constraints, they include constraints on stability, fragility, and weight distribution of the small items and the special shape of the unit load devices that are used in air cargo as containers. Wu et al. (2010) introduce a formulation for a three-dimensional single-bin packing problem with flexible height of the bin. Besides an exact formulation, they propose a genetic algorithm.

Without considering rotation, Martello et al. (2000) introduce an exact method for the 3D-BPP with identical bins. For the same problem, Fekete et al. (2007) propose an exact two-level tree search building on previous results for deriving lower bounds (Fekete and Schepers 2004a, b). This method was further improved by Belov and Rohling (2013). What all of these formulations have in common is that they are built for the case that the bins are either homogeneous or evaluated with instances where the number of available bin types is relatively small compared with the number of loaded items. For example, Paquay et al. (2018) use only up to six different bin types. Moreover, the dimensions of the loaded items are mostly the same, or many items have the same dimensions. In e-commerce applications, the items are typically heterogeneous. Additionally, rotation, which significantly increases the complexity (Wu et al. 2010), is often prohibited. Whereas in the two-dimensional problem, rotation results only in two possible orientations, in the three-dimensional problem, there are six possibilities. This is only reduced if a so-called "this-side-up" assumption is implemented, which is not the case in most e-commerce applications.

In the two-dimensional setting, the complexity of rotation is addressed in two variants (Iori et al. 2021). Similar to Chen et al. (1995) and Paquay et al. (2016) in the three-dimensional setting, binary variables indicate the rotation of the items. A second variant uses copies of the items for all possible rotations such that only one of the possibilities is selected and placed into the bin. However, as stated above, in the three-dimensional problem, this results in six, and not just two, copies. The latter variant also uses the space-indexed formulation by Allen et al. (2012) and Castro and Oliveira (2011), where the solution space is discretized into small boxes. If an item is placed into the bin, then it occupies several boxes according to the dimensions of the item and the boxes. In the three-dimensional case with rotation, this results in a very dense grid and therefore in a large number of decision variables.

The selection of a packing portfolio has rarely been addressed in the literature. Dowsland et al. (2007) determine bin sizes (there called *shipper sizes*) for a given set of products that have to be loaded on pallets. They formulate the assignment of pallets to shippers using a *p*-median model and consider a pallet loading problem as a subproblem. A simulated annealing-based heuristic is used to determine the shipper sizes, whereas the two-dimensional subproblem is solved in each iteration. Similarly, Brinker and Gündüz (2016) consider the problem for the three-dimensional space. However, both the 3D-BPP and the problem of assigning items to bins are treated separately. To calculate the unused space of all item-to-bin combinations, a simple wall-building heuristic is used. Alonso et al. (2016) determine different bin sizes for a warehouse. They formulate this as a two-dimensional packing problem, where, additionally, the packing portfolio is selected. Further, Vieira et al. (2021) determine a box portfolio for packing shoes. However, this is a single-item-to-single-bin assignment problem, where only a *p*-median problem has to be solved.

Until now, the problem of identifying a portfolio for the three-dimensional packing problem has not been addressed. Either the packing is reduced to two-dimensional space or the two problems are treated separately. Moreover, none of these papers allows to split orders into several bins.

2.2. Decomposition Methods for Large-Scale Optimization

To solve large combinatorial optimization problems, several decomposition approaches have been proposed. Our approach is inspired by the ideas of logic-based Benders decomposition (e.g., Hooker and Ottosson 2003, Hooker 2007) and branch-and-check methods (Thorsteinsson 2001). Both approaches divide the considered problem into a relaxed master problem and a subproblem. The subproblem is solved to optimality to add the lost information due to the relaxation iteratively to the relaxed master problem. In its original form, Benders decomposition (Benders 1962) is restricted to a continuous subproblem for generating cuts based on the dual variables. Compared with that, logic-based Benders decomposition (Hooker and Ottosson 2003) can be applied to integer subproblems by using combinatorial cuts instead. Further, many acceleration techniques are known for Benders decomposition.

For a review, see Rahmaniani et al. (2017). For very large optimization problems, Fischetti et al. (2017) redesign Benders decomposition for solving the uncapacitated facility location problem, and Baena et al. (2020) present a stabilized version of Benders decomposition. Wang and Jacquillat (2020) propose new dual-integer cuts for two-stage stochastic integer programs with an application in air traffic scheduling. Further, Rahmaniani et al. (2020) introduce a Benders dual decomposition method that combines Benders decomposition and Lagrangian dual decomposition. Zeighami and Soumis (2019) combine Benders decomposition and column generation to solve an integrated crew pairing and assignment problem. Other than Benders decomposition, branch-and-check does not rely on the dual subproblems. Instead, a constraint-programming method is used to validate whether the subproblem is feasible or not. Recently, Emadikhiav et al. (2020) used branch-and-check to solve a consistent vehicle routing problem and evaluated the consistency of the tours in a subroutine.

We build on these decomposition ideas. However, compared with branch-and-check, we not only check the feasibility of a solution in a subproblem, we also repair infeasible solutions where possible by solving an auxiliary subproblem. Additionally, we use problem-specific structures to avoid feasibility checks.

Decomposition techniques are also applied to cutting and packing problems. Côté et al. (2014) use a Benders decomposition for the strip packing problem that is further enhanced using combinatorial cuts and lifting procedures. Delorme et al. (2017) propose logic-based Benders decomposition for the orthogonal stock cutting problem. Côté et al. (2021) present a combinatorial Benders decomposition for the two-dimensional bin packing problem. They improve the standard combinatorial cuts by finding the minimal infeasible subset of items to lift the cuts. Whereas our approach has some similarities, the fundamental difference is that we choose a decomposition where we prove total unimodularity of the assignment variables in the master problem. Additionally, we decompose the subproblem per assignment such that the combinatorial cuts are per assignment. Therefore, the problem of the weak combinatorial cuts is avoided without lifting the cut further.

3. Three-Dimensional Bin Packing Problem

The 3D-BPP consists of a set of three-dimensional rectangular items $i \in I$ that have to be placed into a set of three-dimensional rectangular bins (parcels) $j \in J$. Each bin is defined by its length L_j, its width W_j, and its height H_j. Similarly, each item is defined by its length p_i, its width q_i, and its height r_i.

We consider the so-called standard restrictions that (1) items have to be placed within the dimensions of the selected bins; (2) items that are placed into the same bin are not allowed to overlap each other in any dimension; and (3) items are only allowed to be placed orthogonally into the bin, such that their edges are parallel to the bin's edges. Additional constraints (e.g., items are not allowed to be rotated) can be included, but they are left out initially here, so that we can focus on a stylized version of the problem.

Next, we state the formulation by Paquay et al. (2016) for the 3D-BPP. In Section 3.2, we introduce a reformulation adjusted to the setting of e-commerce where all items are packed into one bin. Section 3.3 introduces symmetry-breaking constraints.

3.1. Formulation by Paquay et al. (2016)

In the formulation proposed by Paquay et al. (2016), n_j is a binary decision that indicates whether a bin $j \in J$ is selected or not. If item $i \in I$ is placed into a bin $j \in J$, then the binary decision variable s_{ij} equals 1, and 0 otherwise. Then, the position of the left-front-bottom corner of the item is indicated through the three continuous decision variables (x_i, y_i, z_i); the position of the right-back-upper corner of the item through the continuous decision variables (x_i^r, y_i^r, z_i^r). In case two items $i,k \in I$ are placed into the same bin, a_{ik}^x (a_{ik}^y, a_{ik}^z) indicates if item i is placed right of (behind, on top of) item k in the x-(y-, z-)dimension. Finally, $t_{i,c,d} \in \{0,1\}$ indicates the orientation of item i in the bin that it is assigned to. For doing so, $C = \{1,2,3\}$ are the three dimensions (x,y,z) of the bin and $D = \{1,2,3\}$ the three dimensions (x, y, z) of the item. For example, $t_{i,1,3} = 1$ indicates that the z-dimension (3) of item i with length r_i is placed along the x-dimension (1) of the chosen bin.

Further, we define $L_0 = \max_{j\in J}\{L_j\}, W_0 = \max_{j\in J}\{W_j\}$, $H_0 = \max_{j\in J}\{H_j\}$. Using this definition, the mathematical model is as follows:

$$\min \quad \sum_{j\in J} L_j W_j H_j n_j - \sum_{i\in I} p_i q_i r_i \tag{1}$$

$$\text{s.t.} \quad x_i^r - x_i = t_{i,1,1}p_i + t_{i,1,2}q_i + t_{i,1,3}r_i \qquad \forall i \in I, \tag{2}$$

$$y_i^r - y_i = t_{i,2,1}p_i + t_{i,2,2}q_i + t_{i,2,3}r_i \qquad \forall i \in I, \tag{3}$$

$$z_i^r - z_i = t_{i,3,1}p_i + t_{i,3,2}q_i + t_{i,3,3}r_i \qquad \forall i \in I, \tag{4}$$

$$x_i^r \le \sum_{j\in J} L_j s_{ij} \qquad \forall i \in I, \tag{5}$$

$$y_i^r \le \sum_{j\in J} W_j s_{ij} \qquad \forall i \in I, \tag{6}$$

$$z_i^r \le \sum_{j\in J} H_j s_{ij} \qquad \forall i \in I, \tag{7}$$

$$a_{ik}^x + a_{ki}^x + a_{i,k}^y + a_{ki}^y + a_{ik}^z + a_{ki}^z \ge s_{ij} + s_{kj} - 1 \qquad \forall i,k \in I, j \in J; i < k, \tag{8}$$

$$\sum_{j\in J} s_{ij} = 1 \qquad \forall i \in I, \tag{9}$$

$$s_{ij} \le n_j \qquad \forall i \in I, j \in J, \tag{10}$$

$$\sum_{c\in C} t_{i,c,d} = 1 \qquad \forall i \in I, d \in D, \tag{11}$$

$$\sum_{d\in D} t_{i,c,d} = 1 \qquad \forall i \in I, c \in C, \tag{12}$$

$$x_k^r \le x_i + (1 - a_{ik}^x)L_0 \qquad \forall i,k \in I, \tag{13}$$

$$x_i + 1 \le x_k^r + a_{ik}^x L_0 \qquad \forall i,k \in I, \tag{14}$$

$$y_k^r \le y_i + (1 - a_{ik}^y)W_0 \qquad \forall i,k \in I, \tag{15}$$

$$y_i + 1 \le y_k^r + a_{ik}^x W_0 \qquad \forall i,k \in I, \tag{16}$$

$$z_k^r \le z_i + (1 - a_{ik}^z)H_0 \qquad \forall i,k \in I, \tag{17}$$

$$s_{ij}, n_j, t_{i,c,d}, a_{ik}^x, a_{ik}^y, a_{ik}^z \in \{0,1\} \qquad \forall i,k \in I, j \in J, c \in C, d \in D, \tag{18}$$

$$x_i, y_i, z_i, x_i^r, y_i^r, z_i^r \ge 0 \qquad \forall i \in I. \tag{19}$$

The objective function minimizes the empty space in the selected bins: The total selected volume is calculated and the constant volume of all items is subtracted. For each item, Constraint (2) links the position variables of the left and right corners based on the chosen orientation of the item for the x-dimension; analogously, Constraints (3) and (4) are for the y- and z-dimensions. Constraints (5)–(7) ensure that the locations of the corner points do not exceed the size of the chosen bin. If two items are placed into the same bin, then Inequality (8) ensures that there is no overlapping. This is already true if at least along one dimension the items are placed next to each other. Then, in (9), each item has to be packed only into a bin that is used (10). Constraints (11) and (12) guarantee that each side of an item has to be chosen exactly once and placed exactly one time into one direction. Further, Constraints (13)–(14) ensure that if item i is placed right of item k, then the right corner of item k has to be smaller than the left corner of item i. Similarly, Constraints (15)–(17) are defined for the y- and z-dimensions. Finally, (18) and (19) define the domains of the decision variables.

Although in e-commerce items can often be rotated in all directions, some applications assume a "this side up" orientation of items (Alonso et al. 2016). To keep the upper side on top, only two instead of six possible rotations exist. This restriction can be included for item i by setting $t_{i,1,3} = 0$, $t_{i,2,3} = 0$, $t_{i,3,1} = 0$, $t_{i,3,2} = 0$, and $t_{i,3,3} = 1$.

3.2. Model Reformulation

Based on the formulation of Section 3.1, we introduce a reformulation of the 3D-BPP. To avoid the Big-Ms L_0,W_0, and H_0 in the logical Constraints (13)–(17), we replace the continuous position variables (x_i, y_i, z_i) with fractional values. To this end, we normalize the x-coordinates with the maximum length L_0, the y-coordinates with maximum width W_0, and the z-coordinates with the maximum height H_0. Thus, the coordinates (x_i, y_i, z_i) and (x_i^r, y_i^r, z_i^r) can only take values between 0 and 1. Further, we need to define the length, width, and height of each item as the relative sizes of the maximum length, width, and height of the bins. Since we do not know which side of the item is placed along with side of the bin, this results in the definition of nine new auxiliary parameters for each item $i \in I$: $p_i^L = \frac{p_i}{L_0}, p_i^W = \frac{p_i}{W_0}, p_i^H = \frac{p_i}{H_0}$ $q_i^L = \frac{q_i}{L_0}, q_i^W = \frac{q_i}{W_0}, q_i^H = \frac{q_i}{H_0}$ $r_i^L = \frac{r_i}{L_0}, r_i^W = \frac{r_i}{W_0}, r_i^H = \frac{r_i}{H_0}$.

Then, we can introduce the reformulation by replacing Constraints (2)–(4) with

$$x_i^r - x_i = t_{i,1,1}p_i^L + t_{i,1,2}q_i^L + t_{i,1,3}r_i^L \qquad \forall i \in I, \tag{20}$$

$$y_i^r - y_i = t_{i,2,1}p_i^W + t_{i,2,2}q_i^W + t_{i,2,3}r_i^W \qquad \forall i \in I, \tag{21}$$

$$z_i^r - z_i = t_{i,3,1}p_i^H + t_{i,3,2}q_i^H + t_{i,3,3}r_i^H \qquad \forall i \in I, \tag{22}$$

by setting L_j, W_j, and H_j to L_j/L_0, W_j/W_0, and H_j/H_0 in Constraints (5)–(6), respectively, and by setting L_0, W_0, and H_0 to 1 in Constraints (13)–(17).

For the e-commerce setting, we further consider that all items have to be packed into one bin. Separate bins/parcels in e-commerce are mainly a result of products being not available at the same time or at the same distribution center (Biggs 2020). This is ensured by adding the following constraint:

$$\sum_{j\in J} n_j \le 1. \tag{23}$$

3.3. Symmetry-Breaking Constraints

Since all items and bins are rectangular, several symmetric solutions exist. One item could be placed in any of the eight corners; the same packing pattern could still be applied. Therefore, the centroid of exactly one item (we pick the first item of the set of items) is forced to be in the left-front-bottom quadrant of the bin:

$$x_1 + \frac{1}{2}\left(t_{1,1,1}p_1^L + t_{1,1,2}q_1^L + t_{1,1,3}r_1^L\right) \le \frac{1}{2}\sum_{j\in J}\frac{L_j}{L_0}s_{1,j}, \tag{24}$$

$$y_1 + \frac{1}{2}\left(t_{1,2,1}p_1^W + t_{1,2,2}q_1^W + t_{1,2,3}r_1^W\right) \le \frac{1}{2}\sum_{j\in J}\frac{W_j}{W_0}s_{1,j}, \tag{25}$$

$$z_1 + \frac{1}{2}\left(t_{1,3,1}p_1^H + t_{1,3,2}q_1^H + t_{1,3,3}r_1^H\right) \le \frac{1}{2}\sum_{j\in J}\frac{H_j}{H_0}s_{1,j}. \tag{26}$$

Each of the three constraints ensures that the centroid of the item, which is calculated by its left position plus half of its length, is smaller than or equal to half of the length (width, height) of the bin.

4. Three-Dimensional Bin Selection Problem

In the 3D-BSP, a set of orders O is given. Each of these orders o requires the solution of a 3D-BPP which consists of a set of items I_o that have to be packed using a set of bin types J. Note that this set does not depend on the order o. The goal is to minimize the total costs

by selecting a set of available bin types. Total costs include shipping depending on the unused space and costs for variety in bin types. Instead of unused space, different objective functions could be considered. For example, if logistics providers start introducing dimensional pricing, then a combination of the size of the bin and the filling material could be used. Furthermore, the maximum number of bin types could be restricted to a maximum of n_C^{max}. To avoid repetitive notation, we define the 3D-BPP of each order as follows.

Definition 1. For each order $o \in O$, $3DBP(o)$ defines the constraints of the reformulation of the 3D-BPP, including Inequalities (23)–(26). All decision variables and parameters of the items are indexed with the respective order o and consider the items of order I_o.

Additionally, c^S defines the costs for unused space per volume unit, and c^V defines the costs of variety per item per order. For the latter, we use a linear function (Benjaafar et al. 2004). For each order-to-bin assignment, the unused space u_{oj} is known before: $u_{oj} = L_j W_j H_j - \sum_{i \in I_o} p_{io} q_{io} r_{io}$. To use different objective functions, u_{oj} can be defined as parcel delivery rates and material costs that consist of the parcel itself and the needed filling material.

The binary decision $\hat{n}_j$ indicates whether bin type j is available for all packing problems (= 1) or not (= 0). Furthermore, the binary decision variable n_{oj} states whether order o is assigned to bin type j (= 1) or not (= 0). Then the problem is formulated as

$$\mathrm{P}_1: \quad \min c^S \sum_{o \in O} \sum_{j \in J} u_{oj} n_{oj} + c^V |O| \sum_{j \in J} \hat{n}_j \tag{27}$$

$$\text{s.t.} \quad 3DBP(o) \qquad \forall o \in O, \tag{28}$$

$$\sum_{j \in J} n_{oj} = 1 \qquad \forall o \in O, \tag{29}$$

$$n_{oj} \le \hat{n}_j \qquad \forall o \in O, j \in J, \tag{30}$$

$$\sum_{j \in J} \hat{n}_j \le n_C^{max}, \tag{31}$$

$$\hat{n}_j, n_{oj} \in \{0,1\} \quad \forall o \in O, j \in J. \tag{32}$$

The objective function minimizes the combined costs of unused space and the costs of variety. Constraint (28) ensures that, for each order, the 3D-BPP constraints are satisfied. Constraint (29) states that each order has to be assigned to exactly one bin type. Constraint (30) states that in each 3D-BPP only bin types are used that are selected, and Inequality (31) limits the number of bin types.

Theorem 1. *The 3D-BSP is $\mathcal{NP}$-hard.*

The 3D-BSP is composed of two hierarchical decisions. The first is the decision on the available bin types. The second is the selection of the bin type and the resulting packing of items for each order. If the available bin types are known, then the problem can be decomposed by orders and solved individually. Further, if the selected bin type j is known for an order o, then the 3D-BPP reduces to a packing problem that only determines the packing pattern for all items I_o.

Let $\bar{J}_o$ be the set of feasible bin types that can be used to pack the items of order o, implying that the $|J||O|$ 3D-BPP must be solved upfront. Then, the 3D-BSPs can be reformulated as an uncapacitated facility location problem that includes the cost of variety in the objective function:

$$\mathrm{P}_2: \quad \min c^S \sum_{o \in O} \sum_{j \in \bar{J}_o} u_{oj} n_{oj} + c^V |O| \sum_{j \in J} \hat{n}_j \tag{33}$$

$$\text{s.t.} \quad \sum_{j \in \bar{J}_o} n_{oj} = 1 \qquad \forall o \in O, \tag{34}$$

$$n_{oj} \le \hat{n}_j \qquad \forall o \in O, j \in \bar{J}_o, \tag{35}$$

$$\sum_{j \in \bar{J}_o} \hat{n}_j \le n_C^{max}, \tag{36}$$

$$\hat{n}_j, n_{oj} \in \{0,1\} \quad \forall o \in O, j \in \bar{J}_o. \tag{37}$$

In this formulation, the 3D-BPP of each order (constraint (28)) is removed by assuming that $\bar{J}_o$ is known. Note that, under this assumption, the problem remains $\mathcal{NP}$-hard (Mirchandani and Francis 1990). But Theorem 2 shows that all binary decision variables n_{oj} can be relaxed to continuous variables and only $\hat{n}_j$ remains binary.

Theorem 2. *The domain of the binary decision variables n_{oj} for all orders o and all bin types j in (37) of P_2 can be relaxed to*

$$0 \le n_{oj} \le 1 \qquad \forall o \in O, j \in J. \tag{38}$$

The relaxation of Theorem 2 only holds for P_2 and is not true for the original formulation P_1 of the 3D-BSP. However, to avoid solving the challenging formulation P_1 and the upfront enumeration of $\bar{J}_o$ in formulation P_2, we propose a new solution method that we call *branch-and-repair* in Section 5.

5. Branch-and-Repair Method

We first introduce the general concept of the branch-and-repair method that uses decomposition to retain the relaxation of Theorem 2. In Section 5.2, we detail several acceleration techniques before introducing a variable removal strategy and giving some remarks on the generalizability and approximation errors of the branch-and-repair method.

5.1. General Procedure

The key idea of the branch-and-repair method is to decompose the problem into a master problem and a subproblem. The subproblem is the packing of all items of all orders into the assigned bin types. The relaxed master problem is defined by P_2 but assumes that all orders fit into all bins (thus, $\bar{J}_o = J$). Note that several assignments of orders are not possible and can

easily be excluded by preprocessing (see Section 5.2). The relaxed master problem is solved using branch-and-cut to obtain bin type selections and the corresponding order assignment to bins. Every time such a selection is found (meaning in every integer node in the branch-and-cut tree), a subroutine (callback) validates if the solution is feasible or not by solving the 3D-BPP of each order individually. If all 3D-BPPs are feasible, then a new feasible (and possibly global best) solution is found. Otherwise, the solution is rejected and repaired if possible. A subroutine solves the 3D-BPPs with the available bin types and returns the smallest feasible bin type if the problem is solvable. In Algorithm 1, we detail the callback that is triggered at each integer node. For this, we define the Boolean matrix θ_{oj} that indicates if it is known that order o fits into bin j or that it is not known yet. For each order o, we first identify bin $\hat{j}$ that is selected based on the current assignment decisions n_{oj} (line 3). If in previous iterations we found already that order o fits into bin $\hat{j}$ (the 3D-BPP was already solved), then no MILP has to be solved again. Otherwise, we solve the 3D-BPP for placing all items of order o into bin $\hat{j}$ using the reformulation of Section 3.2. If all items of order o do not fit into bin $\hat{j}$ (3D-BPP is infeasible), then a combinatorial cut ($n_{o\hat{j}} = 0$, line 8) is added to the relaxed master problem forbidding the assignment of order o to bin $\hat{j}$ and potentially improving the lower bound. In both cases, $\theta_{o\hat{j}}$ is set true and, in the next iterations, the 3D-BPP is not solved anymore for order o and bin $\hat{j}$. Additionally, infeasible solutions can potentially be repaired by finding a larger bin type j that is currently used ($\hat{n}_j = 1$) in line 1. If such a bin type is available in the current portfolio, then the additional unused space can be calculated. This is done for all infeasible orders. If a feasible bin type exists for all orders, then a new solution (with a larger objective value) is returned to the branch-and-cut tree. Otherwise, the Boolean Ψ_{repair} indicates that not all orders can be packed using the selected bin types, and the selected bin types lead to an infeasible solution. Thus, the upper bound is potentially improved.

Algorithm 1 (MIP Node Procedure)

Input: solution $n_{oj}, \hat{n}_j$ of current integer node
1 $\Psi_{repair} = TRUE$
2 **for** $o \in O$ **do**
3 $\quad \hat{j} \leftarrow j$ with $n_{oj} = 1$
4 $\quad$ **if** $\theta_{o\hat{j}} = FALSE$ **then**
5 $\quad\quad$ solve 3D-BPP of order o and bin $\hat{j}$
6 $\quad\quad$ $\theta_{o\hat{j}} = TRUE$
7 $\quad\quad$ **if** *3D-BPP is infeasible* **then**
8 $\quad\quad\quad$ add constraint $n_{o\hat{j}} = 0$
9 $\quad\quad\quad$ **if** *repair order assignment* $= FALSE$ **then**
10 $\quad\quad\quad\quad$ $\Psi_{repair} = FALSE$
11 **end**
12 **if** $\Psi_{repair} = TRUE$ **then** Insert repaired solution;

5.2. Acceleration Techniques for the 3D-BSP

Additionally, we propose five problem-specific acceleration techniques to improve the performance and reduce the search space.

Solution Repairing. Since several assignments of orders to bin types might be infeasible, lines 7–10 in Algorithm 1 try to repair the found solution by assigning these orders to other bin types that are available in the current iteration. Using Lemma 1, we calculate a lower bound on the cost increase of repairing a solution. This lower bound can be used to check if a repaired solution can improve the global best solution and thus generate a new upper bound. If not, then the repair process can be avoided.

Consider a set of bin types $\tilde{J} = \{1, \ldots, m\}$ that is selected, and consider $\tilde{j}_o$ the selected bin type for order o. Without loss of generality, the bin types are sorted according their volume ($V(i) \leq V(j)$ for $I < j$). Moreover, let $\tilde{f}$ be the objective of the current solution.

Lemma 1. *If there exists exactly one order $\tilde{o}$ that does not fit into the selected bin type, then a lower bound of a repaired solution's objective value is given by* $\tilde{f} - V(\tilde{j}_{\tilde{o}}) + \min_{j \in \tilde{J}} \{V(j) \mid j \neq \tilde{j}_{\tilde{o}} \wedge o \text{ can fit into } j\}$.

For more than one order that does not fit into its selected bin type, Lemma 1 is applied repeatedly for each of these orders.

Bin Hierarchy. For each bin, we calculate upfront whether bin type i fits into another bin type j resulting in a bin hierarchy. Then, all orders that fit into bin i also fit into bin j. Therefore, we solve the 3D-BPP for each order o and identify the best bin choice $\hat{j}$ in the preprocessing.

That means that, for all bins j that enclose bin $\hat{j}$ and for bin j itself, u_{oj} is then known and θ_{oj} is set true. Moreover, all bins j with a volume $V(j) < V(\hat{j})$ cannot be used for order o, even if the volume is large enough. Otherwise, the 3D-BPP would have chosen bin j. During the callback, the bin hierarchy is used in the same fashion to update θ_{oj} if a new order to bin assignment is identified.

Order Hierarchy. For each order, we calculate upfront whether order o' fits into another order o'' resulting in an order hierarchy. Let $I_{o'}$ (respectively, $I_{o''}$) be the items of order o' (respectively, o''), and let $M = I_{o'} \times I_{o''}$ be a matching between the two sets.

Lemma 2. *If there exists a matching M of size $|I_{o'}|$, where for all pairs $(i,j) \in M$ it holds that item i fits into item j, then order o'' is in the hierarchy of order o'.*

This information is used during the callback if an order $\hat{o}$ is assigned to a bin $\hat{j}$. Then, all u_{oj} are known and all θ_{oj} are set true for all orders o that are in the

hierarchy of order $\hat{o}$. Note that, particularly for problems with many items, more advanced order hierarchies could be promising, for example, when two items of one order fit into one item of another order.

Infeasible Bin Types. If the volume of all bins of an order o is smaller than the volume of a bin, and thus $u_{oj} < 0$, then $n_{oj} = 0$ holds. Therefore, we can reduce the number of decision variables in the master problem by removing infeasible assignments.

Additionally, a 3D-BPP can be solved to identify the smallest bin type in the preprocessing. All assignments to bins with a lower volume can additionally be removed.

Single-Item Orders. For single-item orders, the 3D-BPP is trivial, and for all bin types we can check by preprocessing whether order o fits into bin type j or not. Thus, θ_{oj} is true for all single-item orders.

5.3. Variable Removal Strategy

If an infeasible assignment of an order $\hat{o}$ to a bin type $\hat{j}$ is found, then the cut $n_{\hat{o}\hat{j}} = 0$ is added and the repair strategy tries to find the smallest feasible bin type $\tilde{j}$ among all available bin types $\tilde{J}$ of the current bin type portfolio. Then, all bin types $j \in \tilde{J}$ with a smaller volume $V(j) < V(\tilde{j})$ are also infeasible. We use the idea of the bin type hierarchy to further leverage the cut generation process. All bin types that are, according the bin type hierarchy, smaller than the infeasible bin types j are infeasible, too, and an additional cut $n_{\hat{o}j} = 0$ can forbid the assignment.

5.4. General Remarks on Branch-and-Repair Decomposition.

The problem structure would allow two additional intuitive decompositions. The first is keeping all integer variables in the master problem and all continuous variables in the subproblem. This would result in the classical Benders decomposition (Benders 1962), and known problem structures are lost. Neither the master problem nor the slave problem would refer to existing problems as, for example, the 3D-BPP. Second, keeping only the strategic decision of the portfolio selection in the master problem and the operational assignment of bin type to each order in the subproblem, would allow for using classical branch-and-check methods (Thorsteinsson 2001). Keeping also the assignment of bin types to orders in the master problem significantly reduces the number of binary variables and therefore the complexity. Additionally, the 3D-BPP of the subproblem is less complex since the bin type for the order is already given. Thus, for each order, at maximum $|J|$ packing problems are solved. Otherwise, for all selected bin portfolios (maximum $2^{|J|}$), the more complex 3D-BPP has to be solved where additionally the bin type is selected. Therefore, it is important to analyze the problem structure when applying branch-and-repair. Problem settings with hierarchical decisions can leverage the potential of this approach.

Cut Strength. Similar to the idea of multicut Benders decomposition, we add one cut for each subproblem of each order, which strengthens the cuts (Birge and Louveaux 1988). Additionally, it is known from the literature that just forbidding patterns results in weak cuts (see, e.g., Côté et al. 2021, for a discussion). In the packing literature, this means that if a subset $\mathcal{S}$ of items is assigned to a bin and results in an infeasible packing problem, then the assignment of items to this bin has to be reduced to at maximum $|\mathcal{S}| - 1$ items out of set $\mathcal{S}$. One strategy to overcome this problem is to identify the minimal infeasible subset and then forbid further assignments. These classical combinatorial cuts, as introduced by Codato and Fischetti (2006), have been successfully applied by Côté et al. (2021) for the two-dimensional bin packing problem. We use this strategy, but our problem structure avoids solving the $\mathcal{NP}$-complete problem of identifying a minimal infeasible subset. The subset $\mathcal{S}$ of the added cuts to forbid infeasible order to bin assignments has cardinality 1 and is therefore the strongest cut that can be generated out of this infeasibility.

Worst-Case Approximation Bound. Branch-and-repair can be applied using the MILP of Section 3 with a time limit, which can result in an optimality gap for each 3D-BPP.

Consider a 3D-BSP bin type portfolio, where each 3D-BPP for order o is solved with a relative optimality gap α_o and unused space u_o^{UB}. Let z^{UB} be the objective function value, let u^{UB} be the unused space of the found solution (and thus an upper bound), and let z^{opt} be the optimal objective value of this bin type portfolio. Then, Theorem 3 gives an upper bound on the approximation error for the considered bin type solution of the 3D-BSP.

Theorem 3. *The approximation error for a bin type portfolio is restricted to*

$$z^{UB} - z^{opt} \leq c^S \sum_{o \in O} \alpha_o u_o^{UB},$$

and the relative error by

$$\frac{z^{UB} - z^{opt}}{z^{UB}} \leq \sum_{o \in O} \alpha_o \frac{u_o^{UB}}{u^{UB}}.$$

Further, let α_{on} be the relative gap, let u_{on}^{UB} be the unused space of order o, and let z_n^{LB} be the lower bound in node n of all evaluated nodes $n \in N$ of the branch-and-repair method. Then, Theorem 4 gives a performance guarantee for branch-and-repair.

Theorem 4. *The relative approximation error of the final solution is*

$$\frac{z^{UB}-z^{opt}}{z^{UB}} \le \frac{z^{UB}-\min_{n\in N} z_n^{LB}}{z^{UB}} \le \max_{n\in N} c^S \sum_{o\in O} \alpha_{on} \frac{u_{on}^{UB}}{z^{UB}}.$$

Using the lower bounds of branch-and-repair can give tighter error estimations. Particularly, if the final node is solved to optimality and the lower bounds of nodes that have been previously evaluated are larger but have a positive gap, then the approximation error of the final solution is still zero.

6. 3D-BSP with Order Splitting

Motivated by many e-commerce applications, we have assumed until now that each order has to be assigned to exactly one bin. Nevertheless, for example, the size of the order might require splitting an order into several bins for at least a subset of orders. Since allowing order splitting always has the potential to reduce unused space, generally, an order can be split such that each item is packed into one bin. This, however, would result in increasing logistics costs and unsatisfied customers. We restrict, therefore, the order splitting to two bins. Generally, the concept could be expanded to more bins. To allow such an order split, both the 3D-BPP as well as the 3D-BSP need to be adjusted.

In the 3D-BPP, the right-hand side of (23), which restricts the number of bins that can be opened to pack all items of an order, needs to be set to 2 for the respective orders. In P 2 of the 3D-BSP, the set of feasible bin types $\bar{J}_o$ of an order o does not only have to include the set of all feasible bin types but also all feasible bin type combinations. Thus, this redefined set consists of both single bins j' and tuples of bins (j'',j'''): $J_o = \{\dots, j', \dots, (j'',j'''), \dots\}$. Then, the assignment of orders to bin types n_{oj} defines if an order is assigned to either a single bin type and packed into a single bin or a combination of bin types and then packed into two bins. Note that this includes the combination of two identical bins, and, because of symmetry, only one of the two combinations (j_1, j_2) and (j_2, j_1) $(j_1, j_2 \in J$ and $j_1 \neq j_2)$ has to be considered. To ensure that only combinations are selected where both bin types are part of the portfolio, the following constraints need to be added to problem P_2:

$$n_{o,(j_1,j_2)} \le \hat{n}_{j_1} \qquad \forall o \in O, (j_1,j_2) \in \bar{J}_o, \tag{39}$$

$$n_{o,(j_1,j_2)} \le \hat{n}_{j_2} \qquad \forall o \in O, (j_1,j_2) \in \bar{J}_o. \tag{40}$$

This extension of the set of bin types maintains the problem structure, allows the application of branch-and-repair as introduced in the previous section, and preserves the cut strength. The bin hierarchy can be further expanded since, for example, an order that fits into bin j_1 also fits into bin combination (j_1, j_2).

7. Numerical Results

We conducted several numerical experiments to analyze the performance of the introduced methodology. All models were implemented in Python 3.7 and solved by Gurobi 9.1. All experiments were conducted on an AMD Ryzen 9 3950X 16-core processor, 3.493 gigahertz with 128 gigabytes RAM. The data and source code are provided as part of the e-companion.

We first introduce the datasets (Section 7.1). Section 7.2 compares the computational performance of the reformulation for the 3D-BPP against the existing formulation in Paquay et al. (2016). For the 3D-BSP, Section 7.3 shows the run time of the branch-and-repair method compared with the MILP formulation solved by Gurobi. Section 7.4 evaluates the performance for the order splitting extension.

7.1. Data Sets

We use 90 different parcel types based on known parcel types of Amazon (Box Dimensions 2019). Note that those parcel types are not necessarily all used at the same time in practice. However, we are interested in selecting a good parcel portfolio from a larger set. The orders are based on a real data set of the Brazilian e-commerce company Olist (Olist 2019). The data set consists of 98,651 valid orders[1] with up to 21 items per order; 90% of the orders have only one item, and 7.6% have two items. According to Hübner et al. (2015), 50% of the orders in online retail have an order size of one to two items, 25% have two to three, 8% three to four, and 16% have more than four items per order. The number of items per order, however, is very much sector related. Whereas in the electronic segment, the average number of items per order is rather small, in the fashion industry, the average is typically higher. The dimensions of a side of the items vary between 2 and 118 centimeters (cm) and for the parcels between 5.715 and 133.35 cm.

For the 3D-BPP, we selected orders with 2 to 10 items. For each number of items, we randomly picked up to 50 orders out of the Olist data set. Since for more than six items less than 50 orders are in the data set, we used all available orders (21 instances with 7 items, 8 with 8, 3 with 9, and 8 with 10) and additionally generated orders by randomly picking items from the Olist data set. Thus, each instance represents one order, where all items have to be packed into the same parcel.

For the 3D-BSP, we generated four different problem categories (O, E, EL, EM) with different shares of order sizes based on the Olist data set. Category O uses the original distribution of the data set. Categories

E, EL, and EM use equally distributed shares, with EL having only orders with a low number of items per order, EM having only a medium number of items per order, and E having all number of items per order. Table 1 gives all details.

For each of these categories, we generated instances with 100, 200, 300, 400, 500, 1,000, 2,000, and 3,000 orders by randomly selecting orders from the original data set. For each of the combinations, five instances were generated. For the ratio between the cost of variety and the costs for unused space in cubic meters (m^3), we assume that $c^V/c^S = 0.0015$. A sensitivity analysis of the trade-off between the two costs factors is shown in Section 8.

7.2. Performance of the Reformulation for the 3D-BPP

We compare the reformulation with the symmetry-breaking constraints (REF + SBC), the reformulation without these constraints (REF), and the formulation by Paquay et al. (2016) (Paquay). The time limit was set to 300 seconds, and all formulations were solved single-threaded.

Table 2 compares the average run time depending on the number of items to be packed. Additionally, the number of instances solved to proven optimality and the average optimality gap for the instances not optimally solved are reported. Further, the fifth column reports the improvement of the reformulation with symmetry-breaking constraints compared with the formulation by Paquay et al. (2016).

The results show that the reformulation with symmetry-breaking constraints reduces the average run time if more than two items have to be placed. For two items, the differences are marginal. This is not surprising since the symmetry problem is less important with only two items. Compared with Paquay et al. (2016), the average improvement over all sizes is 30%. For instances with 10 items, the reformulation with symmetry-breaking constraints could not solve three instances and without symmetry-breaking constraints could not solve four instances. The same holds for Paquay et al. (2016). However, the symmetry-breaking constraints reduce the optimality gap compared with the other formulations. Detailed analyses show the value of allowing rotation since six possibilities are selected with a frequency between 10% and 26% in the final solution, whereas forbidding the three least chosen orientations increases unused space by 5.1% on average.

Table 1. Share of Items per Order for Each Instance Category

Instance category	1	2	3	4	5	6
O	90.12	7.62	1.34	0.51	0.21	0.20
E	16.67	16.67	16.67	16.67	16.67	16.67
EL	25.00	25.00	25.00	25.00	0.00	0.00
EM	0.00	0.00	33.00	33.00	33.00	0.00

7.3. Performance of Branch-and-Repair for the 3D-BSP

First, we compare the performance of the proposed branch-and-repair method with the performance of Gurobi in solving formulation P_1 of the 3D-BSP for the instances with up to 500 orders. Then, we evaluate the performance and different features of the branch-and-repair method for the instances with 1,000 orders and larger. We set the time limit to three hours and execute all methods using four threads in Gurobi.

Table 3 shows the average run times in seconds, including the preprocessing for branch-and-repair for both methods for the five instances of each item size and instance type. Additionally, the optimality gap is reported for the unsolved instances when using Gurobi. For branch-and-repair, we report the time spent for solving 3D-BPPs (BPP (sec)), the number of bin packing problems solved during the subroutines (# BPP), and the number of integer nodes found (# Int). Out of those, we further show the number of repaired solutions (Repair) and the number of solutions that are not repaired and thus are infeasible or have no potential to improve the upper bound (Infeas.). Moreover, for each size, the average results are shown. The results show that branch-and-repair outperforms Gurobi by at least two orders of magnitude in run time for small instances. Most of the instances cannot even be solved by Gurobi within the time limit, whereas branch-and-repair finds and proves the optimal solution within a few seconds, implying an even larger run-time saving. Further, one can see that, for both methods, the O-instances are the easiest to solve. This is intuitive since they include the highest share of orders with a low number of items. EM and E include the most complex packing problems. Thus, the run times for branch-and-repair increase. Similar effects can be seen for the optimality gaps. This gap results from both not finding the optimal solution and not increasing the lower bound fast enough. Only the five O-instances with 100 orders are solved to optimality, and, for 22 out of the 100, Gurobi finds the optimal solution without proving it. The others have an average gap of 1.74% to the optimal solution. The increasing gap does not consistently hold for all categories.

The detailed statistics show that the number of evaluated integer nodes is only slightly increasing. This is not surprising since the number of integer variables only depends on the number of parcel types after applying Theorem 2. Further, there is a large number of solutions that are infeasible or not repaired. This is intuitive, since we are interested in identifying those solutions and removing those from the solution space during the search process. However, 19% of the solutions

Table 2. Run Time Comparison Depending on Number of Items

Number of items	Run time (sec)				Optimal [gap]		
	REF + SBC	REF	Paquay	Avg. impr.	REF + SBC	REF	Paquay
2	0.01	0.01	0.01	−6.04%	50 / 50	50 / 50	50 / 50
3	0.02	0.03	0.03	31.78%	50 / 50	50 / 50	50 / 50
4	0.07	0.09	0.10	32.24%	50 / 50	50 / 50	50 / 50
5	0.26	0.34	0.38	31.29%	50 / 50	50 / 50	50 / 50
6	0.76	1.31	1.12	32.13%	50 / 50	50 / 50	50 / 50
7	0.79	1.51	1.21	34.90%	50 / 50	50 / 50	50 / 50
8	1.14	2.36	1.72	33.69%	50 / 50	50 / 50	50 / 50
9	4.11	7.29	8.89	53.72%	50 / 50	50 / 50	50 / 50
10	28.28	35.89	36.93	23.43%	47 / 50 [10.44%]	46 / 50 [19.04%]	46 / 50 [19.04%]

can be repaired by reassigning orders to other parcel types and improving the upper bound. Thus, good feasible solutions are found efficiently. Finally, it is clear that the number of actually solved 3D-BPPs increases with the order size. But this number stays significantly below the number of 3D-BPPs that have to be evaluated if no parcel hierarchy or previous information is used. Branch-and-repair spends 58% of the total run time for solving 3D-BPPs, that is, 0.02 seconds on average per 3D-BPP since the 3D-BPPs only need to consider the currently available bins of the master problem.

Table 4 shows the performance without preprocessing of branch-and-repair (B&R) for the larger instances. Additionally, branch-and-repair was executed without a variable removal strategy (Section 5.3, column B&R-C), order hierarchy (Section 5.2, column B&R-CO), implemented relaxation (Theorem 2, column B&R-COM), repairing of solutions (Section 5.2, column B&R-COMR), and without considering the bin (parcel) hierarchy (Section 5.2, column B&R-COMRH). All instances are solved to proven optimality.

The results show that the order hierarchy has the largest impact on the run time performance. Additionally, relaxation of the assignment variables and the repairing of solutions have a major impact on the performance of our method. Considering the parcel hierarchy improves for all instances since less bin packing problems have to be solved. Overall, the improvement

Table 3. Performance Branch-and-Repair (Small Instances)

Number of orders	Instance type	Run time (sec)		Gap in % Gurobi	B&R statistics				
		B&R	Gurobi		BPP (sec)	# BPP	# Int	Repair	Infeas.
100	E	24	10,800	3.74	11	472.40	27.60	7.40	16.20
	EL	10	10,800	0.78	6	448.60	30.60	8.20	19.20
	EM	24	10,800	5.34	15	677.20	27.80	5.60	18.60
	O	4	685	0.00	2	181.60	19.00	5.60	9.80
	Avg.	**15**	**8,271**	**2.47**	**9**	**444.95**	**26.25**	**6.70**	**15.95**
200	E	55	10,800	5.78	29	1,137.60	33.20	6.60	23.00
	EL	22	10,800	2.39	14	1,004.60	31.00	5.00	24.60
	EM	54	10,800	9.69	33	1,493.00	30.60	5.00	23.40
	O	9	10,800	0.46	5	361.40	27.00	5.80	18.80
	Avg.	**35**	**10,800**	**4.58**	**20**	**999.15**	**30.45**	**5.60**	**22.45**
300	E	84	10,800	6.83	43	1,637.60	36.80	6.60	27.40
	EL	33	10,800	1.93	22	1,476.80	32.20	7.20	21.40
	EM	82	10,800	11.11	51	2,257.20	31.80	6.40	22.60
	O	15	10,800	0.30	8	606.60	28.60	6.20	18.60
	Avg.	**54**	**10,800**	**5.04**	**31**	**1,494.55**	**32.35**	**6.60**	**22.50**
400	E	110	10,800	8.53	55	2,084.80	36.80	5.20	29.40
	EL	49	10,800	2.53	32	2,203.80	38.00	4.40	31.00
	EM	110	10,800	7.65	68	2,977.00	32.80	7.00	22.40
	O	22	10,800	0.47	11	816.20	28.40	8.00	18.80
	Avg.	**73**	**10,800**	**4.79**	**41**	**2,020.45**	**34.00**	**6.15**	**25.40**
500	E	147	10,800	11.25	76	2,886.40	39.80	6.60	31.00
	EL	56	10,800	3.09	36	2,522.00	34.00	4.80	27.20
	EM	143	10,800	9.75	92	3,899.40	32.80	4.60	24.00
	O	31	10,800	0.49	15	1,000.80	30.00	6.20	21.40
	Avg.	**94**	**10,800**	**6.15**	**55**	**2,577.15**	**34.15**	**5.55**	**25.90**

Table 4. Performance Branch-and-Repair (Large Instances) (in Seconds)

	B&R	B&R-C	B&R-CO	B&R-COM	B&R-COMR	B&R-COMRH	Improvement (in %)
1,000	128	145	156	173	198	238	46.22
2,000	286	321	368	417	457	507	43.61
3,000	525	533	595	745	890	918	42.77

strategies reduce the run time by more than 44% on average. These improvements stay relatively stable with increasing number of orders.

7.4. Performance of Branch-and-Repair with Order Splitting

To evaluate the performance of branch-and-repair with order splitting, we use the large instances of the previous section and allow 5%, 10%, and 20% of the orders to be split into two parcels. Branch-and-repair was executed with a run time limit of three hours. Table 5 shows the run time without preprocessing, the number of instances that were solved to optimality, and the average gap of the instances that were not solved to optimality.

The results show that the number of orders that are allowed to be split increases the complexity. Both increasing the share of split orders and increasing the total number of orders results in less instances solved to optimality and an increasing optimality gap. This is not surprising since the solution space significantly increases. Still, the average optimality gap is only 0.53% and 79 out of 180 instances are solved to optimality. Moreover, the share of run time for solving 3D-BPPs decreases. This results from the larger and more complex master problem since orders cannot only be assigned to bin types, but also to bin type combinations.

8. Case Study

To give insights on how the portfolio of parcel types affects the unused space and also increases costs of variety, we solve the full Olist data set. By identifying orders with identically sized items, this set is reduced to 13,078 unique orders. The unused space of each unique order is then weighted with the number of occurrences; 91% of the single-item orders occur more than once such that this group reduces from 88,849 to 7,972 orders. This potential of reduction reduces with the number of items per order.

The run time of a 3D-BPP subproblem is limited to 200 seconds. In the preprocessing, branch-and-repair identifies the optimal parcel type to reduce the number of possible order-parcel type assignments. If the optimal solution of the 3D-BPP is not found, then all parcel type assignments of this order are possible in the relaxed master problem. Note that we still guarantee finding the optimal solution since later the packing problem is solved for each parcel type assignment that is selected. Only in case this can also not be solved, Theorem 1 is used to calculate the approximation error.

The full model with 13,078 orders consists of 5,068,281 constraints, 1,647,829 continuous, and 45,668,466 binary decision variables. After presolving, Gurobi reduces the problem to 1,263,270 rows, 68,122 continuous, and 670,413 binary variables. Using branch-and-repair, we obtained the optimal solution after 19,523 seconds. Preprocessing took 2,771 seconds (including 11 times when the 200-second time limit was exceeded), and solving the problem using branch-and-repair took 17,347 seconds (including 80 times when the time limit was exceeded). These unsolved 3D-BPPs, however, result only in an approximation error below 0.008%.

In the optimal solution, 12 parcel types out of 90 are selected. Among the parcel types with similar small volumes, one type is selected; for larger volumes,

Table 5. Performance of Branch-and-Repair with Order Splitting

Number of orders	Share split	Run time (sec)	BPP (sec)	Optimal	Gap (%)
1,000	0.05	1,909	234	20 / 20	0.00
	0.1	4,679	296	19 / 20	0.16
	0.2	10,035	509	4 / 20	0.58
	Avg.	**5,541**	**346**	**43 / 60**	**0.25**
2,000	0.05	5,219	536	20 / 20	0.00
	0.1	10,212	691	5 / 20	0.35
	0.2	10,800	1,106	0 / 20	1.03
	Avg.	**8,744**	**778**	**25 / 60**	**0.46**
3,000	0.05	9,156	842	11 / 20	0.25
	0.1	10,800	1,106	0 / 20	0.67
	0.2	10,800	1,852	0 / 20	1.72
	Avg.	**10,252**	**1,267**	**11 / 60**	**0.88**

different dimensions and similar volumes are also selected.

Even though Thonemann and Brandeau (2000) use a linear function as the costs of variety in a real-life application of a production system, they point out that other companies might have different functions. Therefore, the trade-off between costs for unused space and costs of variety is further analyzed by removing the cost of variety term in the objective function and by adding a constraint to fix the number of parcel types to $N = 1,\dots,20$ ($\sum_{j\in J}\hat{n}_j = N$). Thus, we select the optimal portfolio for each portfolio size while minimizing unused space. As an additional benchmark, we first apply a K-means algorithm to cluster the parcel types according to their volume and select the largest parcel within each cluster. Clustering by similar volumes ensures the provision of a sufficient variance of different packages. Second, we use an Add-heuristic that iteratively adds parcel types with the highest savings of unused space to the portfolio. To avoid high computational effort, we use the feasibility matrix θ_{oj} of the preprocessing instead of solving all order parcel type combinations to identify all savings. Simpler strategies, namely, selecting parcel types according to their highest frequency as optimal parcel and dividing the parcel types into quantiles where the highest volume in each quantile is selected, result in gaps larger than 100% to the optimal unused space and are therefore not further discussed.

Figure 1 shows that the unused space, and thus the costs of unused space, significantly decreases when increasing the number of parcel types to four. The unused space is reduced by 86% from one to two parcel types, by 49% from two to three, and by 22% from three to four. Between 4 and 11 parcels, the unused space savings decrease from 9% to 2%. After that, there is still potential to reduce unused space; however, it is only a 2% per parcel type increase (1% from 13 parcel types on). This shows that already a small variety of parcel types could significantly reduce the unused space without leading to large costs of variety. Additionally, Figure 1 depicts the lower bound of unused space costs (assuming a full parcel type portfolio and that each order is assigned to the best parcel type). This shows that, in the optimal solution with 12 parcel types, the unused space has a gap of 18% to a best selection of all parcel types. However, to achieve this lower bound of unused space, the costs of variety would increase significantly since all parcel types are necessary. Moreover, the comparison with the benchmark models shows that a comprehensive model is necessary to capture the relation between parcel types. The portfolios differ in unused space by 14.7% for K-means and 5.7% for the Add-heuristic on average compared with the optimal decision. Further, whereas the K-means performs comparably well for small portfolio sizes (7.3% for size 2 to 6 and 17.4% for size 7 to 20), for the Add-heuristic it is the opposite (9.7% for size 2 to 4 and 5.9% for size 5 to 20).

To compare the unused space objective function with a cost objective, we use parcel delivery rates of UPS and costs for parcels and filling material. Figure 2 shows the parcel portfolios for different cost ratios $\frac{c^V}{c^S}$ and how larger portfolios include smaller portfolios. Additionally, the unused space and the optimal unused space for the same portfolio size are shown.

The portfolio size increases with decreasing costs of variety. Larger portfolios mainly include smaller portfolios, but not completely. This results in relatively stable portfolios that are incrementally augmented. For small portfolio sizes, total cost minimization leads to significantly more unused space. Compared with the optimal unused space solution, 27% more unused space for five parcel types reduces to 6% for 16 parcel types. This results from the fact that the filling material only counts for a small share in the costs of a parcel.

Figure 1. (Color online) Cost of Variety and Cost of Unused-Space for Different Portfolio Sizes

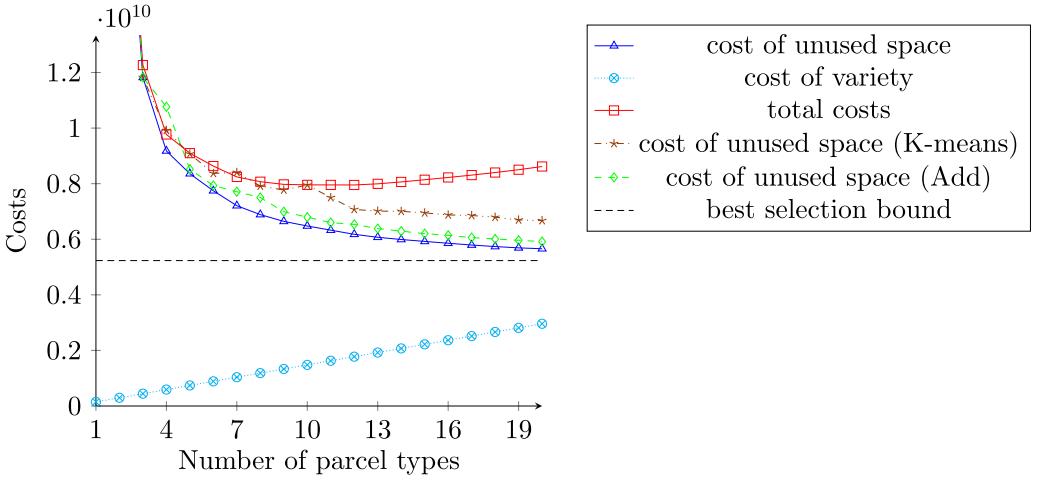
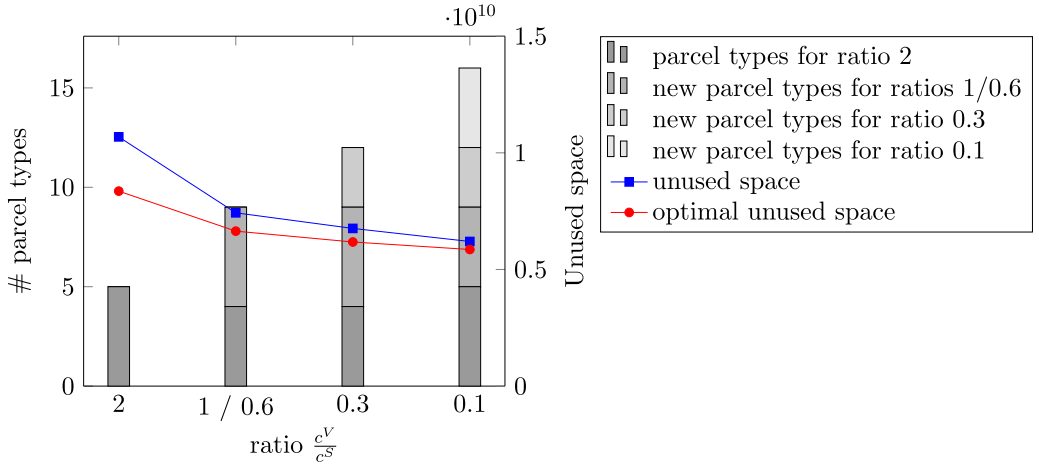


Figure 2. (Color online) Parcel Portfolio and Unused Space for Different Cost Ratios



Additionally, we test the effects of allowing the rotation of items and splitting orders. Due to the size of the problem, the extension with orders splitting (Section 6) was solved on a cluster with 500 gigabytes RAM with a run time limit of 24 hours. We consider three settings where every 20th, 10th, and 5th order can be split into two parcels. The first two settings are solved to optimality, whereas the latter one terminates with an optimality gap of 0.70%.

The results show that the costs of unused space increase by 8.6% without rotation and by 3.5% with a "this-side-up" assumption. Allowing every 20th and 10th order to be split reduces the unused space by 0.26% and 0.37%, respectively, and has a rather low impact. If every fifth order is allowed to be split, then the unused space increases by 1.97%, which is compensated by a smaller parcel portfolio. In the "this-side-up" case, 9 out of the 12 parcel types remained. Out of the three removed parcel types, there is one large parcel type and two medium-sized ones. The two medium-sized parcel types were replaced by three other ones. For the large parcel type, no larger one was selected. This is not so surprising, since restricting rotation results in less efficient packing patterns and, therefore, also requires slightly larger parcel types. Without rotation, the portfolio reduced to 10 parcel types, and again three parcel types were replaced.

9. Conclusion

We introduced an efficient reformulation and valid inequalities for the 3D-BPP. The numerical tests show that existing formulations are outperformed by 30% on average. Further, we defined the 3D-BSP for selecting the parcel type portfolio in e-commerce. A branch-and-repair procedure was developed that shows run time reductions of at least two orders of magnitude compared with solving the MILP using Gurobi. Whereas branch-and-repair solves the test data sets within seconds and minutes, Gurobi could not solve most of the instances within three hours. Further, the decomposition allowed for relaxing a large share of binary decision variables, and further acceleration techniques are introduced.

From a managerial perspective, companies should be aware that (1) increasing the number of parcel types first can significantly reduce the unused space, but the marginal contribution reduces rapidly; (2) minimizing costs or minimizing unused space leads to significantly different results; and (3) allowing order splitting into several parcels is less effective.

For cases including orders with a large number of items per order, the 3D-BPP might become difficult to solve and will limit the performance of the approach in its current form. Therefore, further research on (exact) methods for solving 3D-BPPs with rotation is necessary. However, due to the separation of the integrated problem into two, an assignment problem and the 3D-BPP, branch-and-repair allows for the use of heuristics in the subset of challenging 3D-BPPs while guaranteeing a maximum approximation error. Therefore, the usage of heuristics for the 3D-BPP has potential for future research. Further, companies that use packing algorithms in practice can calculate their optimal portfolio for their packing algorithm.

Acknowledgments

The authors thank the area editor, the associate editor, and the three reviewers for their constructive comments, which helped to improve this paper.

Appendix A. Proofs

A.1. Proof of Theorem 1

Consider a cost of variety of $c^V = 0$ and $n_C^{max} = |J|$. Then, no costs of variety exist, and all parcel types can be selected. Further, consider only one order ($|O| = 1$). Then, the 3D-BSP reduces to the 3D-BPP, which is known to be $\mathcal{NP}$-hard (Paquay et al. 2018). □

A.2. Proof of Theorem 2

Problem P_2 is an uncapacitated facility location problem. Thus, having the selected parcel types leaves an assignment problem. The matrix of the assignment problem is totally unimodular. Then, n_{oj} will always take the value 0 or 1 in an optimal solution, even if continuous values in-between are allowed (Mirchandani and Francis 1990). □

A.3. Proof of Theorem 3

If the optimality gap α is only caused by a weak lower bound, then bin type $\tilde{j}_{\tilde{o}}$ is the optimal decision and the approximation error is 0. We consider one bin type solution $\tilde{J}$ with found unused space objective (upper bound) u_o^{UB} and lower bound u_o^{LB} per order. Then, $\alpha_o = \frac{u_o^{UB} - u_o^{LB}}{u_o^{UB}} \geq 0$ defines the relative gap for each order o. Thus, the 3D-BPP returns a solution for each order that has up to $u_o^{UB} - u_o^{LB} = \alpha_o u^{UB}$ more unused space than the optimal solution. Moreover, let u_o^{opt} be the optimal decision.

Then the objective z^{UB} of solution $\tilde{J}$ for the 3D-BSP has at maximum the following error to the optimal value z^{opt} of solution $\tilde{J}$:

$$z^{UB} - z^{opt} = \left(c^S \sum_{o\in O} u_o^{UB} + c^V |O||\tilde{J}|\right) - \left(c^S \sum_{o\in O} u_o^{opt} + c^V |O||\tilde{J}|\right)$$
$$= c^S \sum_{o\in O} (u_o^{UB} - u_o^{opt}) \leq c^S \sum_{o\in O} (u_o^{UB} - u_o^{LB}) = c^S \sum_{o\in O} \alpha_o u_o^{UB}.$$

Thus, the relative error of a bin type selection solution is

$$\frac{z^{UB} - z^{opt}}{z^{UB}} \leq \frac{c^S \sum_{o\in O} \alpha_o u_o^{UB}}{c^S \sum_{o\in O} u_o^{UB} + c^V |O||\tilde{J}|} \leq \frac{\sum_{o\in O} \alpha_o u_o^{UB}}{\sum_{o\in O} u_o^{UB}}$$
$$\leq \sum_{o\in O} \alpha_o \frac{u_o^{UB}}{u^{UB}}. \quad \square$$

A.4. Proof of Theorem 4

Let z_n^{LB} and z_n^{UB} be the lower and upper bound of solution n and α_{on}, and let u_{on}^{UB} be the optimality gap and the unused space objective (upper bound) of order o in solution n. From Theorem 3, it follows that $z_n^{LB} \geq z_n^{UB} - c^S \sum_{o\in O} \alpha_{on} u_{on}^{UB}$ (+). Then, the relative error is

$$\frac{z^{UB} - z^{opt}}{z^{UB}} \leq \frac{z^{UB} - \min_{n\in N} z_n^{LB}}{z^{UB}}$$
$$\leq^{(+)} \frac{z^{UB} - \min_{n\in N} z_n^{UB} + \max_{n\in N} c^S \sum_{o\in O} \alpha_{on} u_{on}^{UB}}{z^{UB}}$$
$$= \max_{n\in N} c^S \sum_{o\in O} \alpha_{on} \frac{u_{on}^{UB}}{z^{UB}}. \quad \square$$

Endnote

[1] Five orders were removed since the items did not have dimensions or the order was too large to be placed into any parcel type.

References

Allen S, Burke E, Mareček J (2012) A space-indexed formulation of packing boxes into a larger box. *Oper. Res. Lett.* 40(1):20–24.

Alonso M, Alvarez-Valdes R, Parreño F, Tamarit J (2016) Determining the best shipper sizes for sending products to customers. *Internat. Trans. Oper. Res.* 23(1–2):265–285.

Amazon (2022) Improving packaging. Accessed March 29, 2022, https://www.aboutamazon.com/planet/improving-packaging.

Baena D, Castro J, Frangioni A (2020) Stabilized Benders methods for large-scale combinatorial optimization, with application to data privacy. *Management Sci.* 66(7):3051–3068.

Belov G, Rohling H (2013) LP bounds in an interval-graph algorithm for orthogonal-packing feasibility. *Oper. Res.* 61(2):483–497.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1): 238–252.

Benjaafar S, Kim JS, Vishwanadham N (2004) On the effect of product variety in production-inventory systems. *Ann. Oper. Res.* 126(1):71–101.

Biggs K (2020) Sustainability in e-commerce: How can online retailers make their business greener? parcelLab Insights Blog (March 25), https://parcellab.com/e-commerce/sustainability-online-retail/.

Birge JR, Louveaux FV (1988) A multicut algorithm for two-stage stochastic linear programs. *Eur. J. Oper. Res.* 34(3):384–392.

Box Dimensions (2019) Amazon box sizes. Accessed September 27, 2019, https://www.boxdimensions.com/.

Brinker J, Gündüz HI (2016) Optimization of demand-related packaging sizes using a p-median approach. *Internat. J. Adv. Manufacturing Tech.* 87(5):2259–2268.

Castro PM, Oliveira JF (2011) Scheduling inspired models for two-dimensional packing problems. *Eur. J. Oper. Res.* 215(1):45–56.

Chen C, Lee S, Shen Q (1995) An analytical model for the container loading problem. *Eur. J. Oper. Res.* 80(1):68–76.

Codato G, Fischetti M (2006) Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.* 54(4):756–766.

Crainic TG, Montreuil B (2016) Physical internet enabled hyperconnected city logistics. *Transportation Res. Procedia* 12:383–398.

Côté JF, Dell'Amico M, Iori M (2014) Combinatorial Benders' cuts for the strip packing problem. *Oper. Res.* 62(3):643–661.

Côté JF, Haouari M, Iori M (2021) Combinatorial Benders decomposition for the two-dimensional bin packing problem. *INFORMS J. Comput.* 33(3):963–978.

Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur. J. Oper. Res.* 255(1):1–20.

Delorme M, Iori M, Martello S (2017) Logic based Benders' decomposition for orthogonal stock cutting problems. *Comput. Oper. Res.* 78:290–298.

DHL (2022) Artificial intelligence saves costs and emissions by optimizing packaging of shipments for dhl supply chain customers. Press release, Deutsche Post DHL Group (April 13), https://www.dpdhl.com/en/media-relations/press-releases/2022/artificial-intelligence-optimizing-packaging-for-dhl-supply-chain-customers.html.

Dowsland KA, Soubeiga E, Burke E (2007) A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *Eur. J. Oper. Res.* 179(3):759–774.

Emadikhiav M, Bergman D, Day R (2020) Consistent routing and scheduling with simultaneous pickups and deliveries. *Production Oper. Management* 29(8):1937–1955.

Esser K, Kurte J (2020) KEP-Studie 2019—Analyse des Marktes in Deutschland. Technical report, BIEK, Berlin.

FedEx (2019) Service guide. Technical report, https://www.fedex.com/content/dam/fedex/us-united-states/services/Service_Guide_2019.pdf.

Fekete SP, Schepers J (2004a) A combinatorial characterization of higher-dimensional orthogonal packing. *Math. Oper. Res.* 29(2): 353–368.

Fekete SP, Schepers J (2004b) A general framework for bounds for higher-dimensional orthogonal packing problems. *Math. Methods Oper. Res.* 60(2):311–329.

Fekete SP, Schepers J, van der Veen JC (2007) An exact algorithm for higher-dimensional orthogonal packing. *Oper. Res.* 55(3):569–587.

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning Benders decomposition for large-scale facility location. *Management Sci.* 63(7): 2146–2162.

Freichel SL, Wollenburg J, Wörtge JK (2020) The role of packaging in omni-channel fashion retail supply chains–how can packaging contribute to logistics efficiency? *Logist. Res.* 13(1), https://doi.org/10.23773/2020_01.

Hooker JN (2007) Planning and scheduling by logic-based Benders decomposition. *Oper. Res.* 55(3):588–602.

Hooker JN, Ottosson G (2003) Logic-based Benders decomposition. *Math. Programming* 96(1):33–60.

Hübner A, Holzapfel A, Kuhn H (2015) Operations management in multi-channel retailing: an exploratory study. *Oper. Management Res.* 8(3):84–100.

Iori M, de Lima VL, Martello S, Miyazawa FK, Monaci M (2021) Exact solution techniques for two-dimensional cutting and packing. *Eur. J. Oper. Res.* 289(2):399–415.

Living Packets (2020) THE BOX. https://www.livingpackets.com/.

Martello S, Pisinger D, Vigo D (2000) The three-dimensional bin packing problem. *Oper. Res.* 48(2):256–267.

Mirchandani PB, Francis RL, eds. (1990) *Discrete Location Theory* (Wiley, New York).

Montreuil B, Ballot E, Tremblay W (2014) Modular design of physical internet transport, handling and packaging containers. *Proc. 13th Internat. Material Handling Res. Colloquium*, https://digitalcommons.georgiasouthern.edu/pmhr_2014/1.

Murawski C, Bossaerts P (2016) How humans solve complex problems: The case of the knapsack problem. *Sci. Rep.* 6(1):34851.

Olist (2019) Brazilian e-commerce public data set by Olist. Accessed October 2, 2019, https://www.kaggle.com/olistbr/brazilian-ecommerce.

Paquay C, Limbourg S, Schyns M (2018) A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints. *Eur. J. Oper. Res.* 267(1):52–64.

Paquay C, Schyns M, Limbourg S (2016) A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *Internat. Trans. Oper. Res.* 23(1-2):187–213.

Pitney Bowes (2019) Pitney Bowes parcel shipping index. https://www.pitneybowes.com/us/shipping-index.html.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *Eur. J. Oper. Res.* 259(3):801–817.

Rahmaniani R, Ahmed S, Crainic TG, Gendreau M, Rei W (2020) The Benders dual decomposition method. *Oper. Res.* 68(3):878–895.

Thonemann UW, Brandeau ML (2000) Optimal commonality in component design. *Oper. Res.* 48(1):1–19.

Thorsteinsson E (2001) Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. Walsh T, ed. *Internat. Conf. Principles Practice Constraint Programming* (Springer, Berlin), 16–30.

UPS (2021) Cube optimization. https://www.ups.com/media/en/CubeOptimizationSalesSheet.pdf.

Vieira MVC, Ferreira F, Duque JCM, Almeida RMP (2021) On the packing process in a shoe manufacturer. *J. Oper. Res. Soc.* 72(4):853–864.

Walmart (2022) Walmart Sustainability Hub. https://www.walmartsustainabilityhub.com/.

Wang K, Jacquillat A (2020) A stochastic integer programming approach to air traffic scheduling and operations. *Oper. Res.* 68(5):1375–1402.

Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* 183(3):1109–1130.

Westerheide C (2019) Amazon-Chef Kleber: Die letzte Meile schreit nach Innovation. *Deutsche Verkehrs-Zeitung*, https://www.dvz.de/rubriken/land/kep/detail/news/amazon-chef-kleber-die-letzte-meile-schreit-nach-innovation.html.

Wu Y, Li W, Goh M, de Souza R (2010) Three-dimensional bin packing problem with variable bin height. *Eur. J. Oper. Res.* 202(2):347–355.

Zeighami V, Soumis F (2019) Combining Benders' decomposition and column generation for integrated crew pairing and personalized crew assignment problems. *Transportation Sci.* 53(5):1479–1499.

Zhao X, Bennell JA, Bektaş T, Dowsland K (2016) A comparative review of 3D container loading algorithms. *Internat. Trans. Oper. Res.* 23(1–2):287–320.

Pirmin Fontaine is an assistant professor of operations management at the Catholic University of Eichstätt-Ingolstadt, Ingolstadt School of Management. His main research interests are in large-scale optimization and decomposition techniques with applications in logistics, transportation, mobility systems, and supply chain management. He is a recipient of the German Operations Research (GOR) Doctoral Dissertation Prize.

Stefan Minner is a full professor for logistics and supply chain management at the School of Management, Technical University of Munich (TUM) and a core member of the Munich Data Science Institute (MDSI). His research interests are in global supply chain design, transportation optimization, and inventory management. Minner is a fellow of the International Society for Inventory Research.