A top-down cutting approach for modeling the constrained two- and three-dimensional guillotine cutting problems

Mateus Martin[a], Reinaldo Morabito[a,*], Pedro Munari[a]

[a]*Department of Production Engineering, Federal University of São Carlos*

Abstract

In this paper, we address the Constrained Two-dimensional Guillotine Cutting Problem (C2GCP) and the Constrained Three-dimensional Guillotine Cutting Problem (C3GCP). These problems consist of cutting a rectangular two-/three-dimensional object with orthogonal guillotine cuts to produce ordered rectangular two-/three-dimensional items seeking the most valuable subset of items cut. They often appear in manufacturing settings that cut objects to produce item types of low demand, such as in the cutting of flat glass in the glass industry, rocks in the granite and marble industries and steel blocks in the metallurgical industry. To model and solve these problems, we propose a novel top-down cutting approach that leads to effective mixed integer linear programming models for the C2GCP and the C3GCP. The insight of the proposed approach is to represent the cutting pattern as a binary tree, in which the root node is the object, and branches correspond to guillotine cuts. The results of computational experiments with a general-purpose optimization solver and using three sets of benchmark instances showed that the proposed models are competitive with state-of-the-art formulations of the C2GCP and the C3GCP in quality of solution and processing times, particularly when the number of items in an optimal solution is moderate.

Keywords: Cutting and packing, 2D and 3D guillotine cutting, Non-staged patterns, Mixed-integer linear programming models, Top-down cutting

1. Introduction

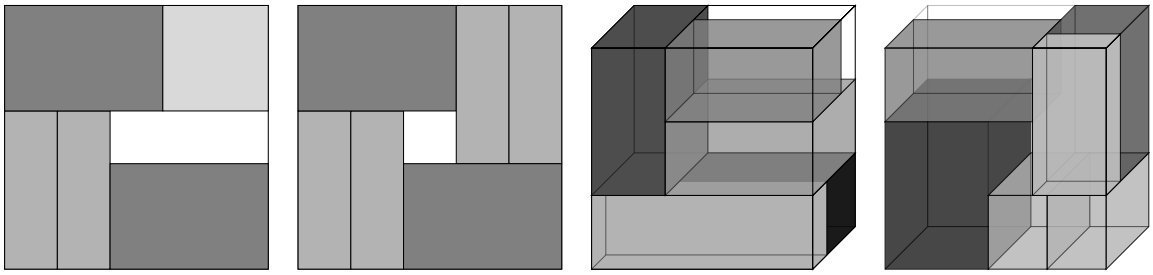
Cutting operations comprise a usual stage of production in several manufacturing industries, in which large stocked objects are cut into small items to produce intermediary pieces or final products. Typical examples include the cutting of flat glass in the glass industry, wooden boards in the furniture industry, rocks in the granite and marble industries, foams in the mattress industry, steel blocks in the metallurgical industry, among others. Such situations require the aid of effective decision making tools that seek to minimize the adverse effects of trim loss on production costs.

The Constrained Two-dimensional Guillotine Cutting Problem (C2GCP) and the Constrained Three-dimensional Guillotine Cutting Problem (C3GCP) are relevant cutting problems that arise

*Corresponding author. Department of Production Engineering, Federal University of São Carlos, Via Washington Luiz km. 235, 13565-905, São Carlos-SP, Brazil. Phone/fax: 55-16-33519516/33518240.

Email addresses: mateus.pmartin@gmail.com (Mateus Martin), morabito@ufscar.br (Reinaldo Morabito), munari@dep.ufscar.com (Pedro Munari)

in the mentioned manufacturing industries. The C2GCP addresses one rectangular object of size $\overline{L} \times \overline{W}$, and a set $I = \{1, \ldots, m\}$ of rectangular item types with sizes $l_i \times w_i$, value v_i and maximum number of copies u_i to be produced. The dimensions $\overline{L}$ and $\overline{W}$ of the object and corresponding dimensions of the item types are along the horizontal and vertical Cartesian axes, respectively. The problem consists of selecting and cutting the most valuable subset of items from the object. Any feasible solution of the problem should satisfy the following requirements: (i) the cut of a rectangular (sub-)object generates two smaller rectangular sub-objects, that is, the cuts are of guillotine-type with unlimited number of guillotine stages (non-staged patterns); and (ii) up to u_i copies of item type $i \in I$ can be produced. The C3GCP is the three-dimensional counterpart of the C2GCP, which considers one rectangular cuboid object of size $\overline{L} \times \overline{W} \times \overline{H}$ and a set of rectangular cuboid item types with sizes $l_i \times w_i \times h_i$. The C2GCP and the C3GCP are related to the Single Large Object Placement Problem (SLOPP) in the improved typology of Cutting and Packing problems (Wäscher et al., 2007). Fig. 1 illustrates two- and three-dimensional cutting patterns highlighting the guillotine requirement, where the grayscale rectangles/rectangular cuboids are items and the blank areas/volumes are waste; the patterns in Figs. 1a and 1c are feasible solutions of the C2GCP and the C3GCP, respectively.



(a) 2D guillotine. (b) 2D non-guillotine. (c) 3D guillotine. (d) 3D non-guillotine.

Figure 1: Examples of two- and three-dimensional cutting patterns.

The solution approaches for the C2GCP include Dynamic Programming (DP) algorithms (Christofides & Hadjiconstantinou, 1995; Morabito & Pureza, 2010; Dolatabadi et al., 2012; Velasco & Uchoa, 2019), tree search algorithms (Oliveira & Ferreira, 1990; Viswanathan & Bagchi, 1993; Cung et al., 2000; Yoon et al., 2013), heuristics/meta-heuristics (Parada et al., 1995; Álvarez-Valdés et al., 2002), and Mixed Integer Linear Programming (MILP) models in the context of a general-purpose optimization solver (Ben Messaoud et al., 2008; Furini et al., 2016; Martin et al., 2020a,b). For a complete revision on the topic, see the recent upper bound review and categorization of Russo et al. (2020) who surveyed more than 90 approaches for the C2GCP.

Regarding the C3GCP and related problems, there are solution approaches based on AND/OR graph algorithms (Morabito & Arenales, 1994), approximation algorithms (Hifi, 2002), DP formulations (Hifi, 2004; De Queiroz et al., 2017), and Constraint Programming (CP) techniques (Amossen & Pisinger, 2010). Recently, Martin et al. (2020c) developed two MILP models and a binary tree search algorithm for non-staged and 3-staged patterns (i.e., solutions limited to 3 guillotine stages). Do Nascimento et al. (2019) addressed the 2- and 3-staged versions of the C2GCP and the 3- and 4-staged versions of the C3GCP by proposing a two-level iterative approach that resorts to a CP-based algorithm. While the C3GCP is a cutting problem in manufacturing settings, the Container Loading Problem (CLP) is a related three-dimensional packing problem that appears

in logistical settings. The main difference is that there is no guillotine requirement in the CLP, because of cargo stability issues and for seeking higher utilization rates of the container. Indeed, the infeasible pattern for the C3GCP in Fig. 1d is feasible for the CLP.

Any solution of a guillotine cutting problem can be represented by a binary tree, in which the nodes are rectangles/rectangular cuboids and the branches are guillotine cuts. This binary tree is then explored either in a top-down cutting approach that cuts the object towards the items; or in a bottom-up packing approach that merges the items towards the dimensions of the object. In this paper, we propose approaches of the first type. Specifically, our main contributions are (*i*) the introduction of a novel non-trivial top-down cutting approach for modeling the C2GCP and the C3GCP; and (*ii*) the proposition of effective MILP models for both problems, based on this top-down modeling approach. The development of these top-down models is motivated for a direct comparison to the recent bottom-up models of Martin et al. (2020b) for the C2GCP and of Martin et al. (2020c) for the C3GCP. We also show how to reformulate these bottom-up models using a straightforward counterpart top-down approach (that differs fundamentally from the top-down approach proposed in this paper) and then compare the obtained top-down models with the proposed top-down models of the present paper.

The remainder of this paper is organized as follows. In Section 2, we present the new non-trivial top-down pseudo-polynomial modeling approach for the C2GCP and the C3GCP. In Section 3, the performance of the proposed models derived from this approach are analyzed through computational experiments using three sets of benchmark instances from the literature. Final remarks and perspectives of future research are discussed in Section 4. In Appendix, we show how to straightforwardly reformulate the two bottom-up models of Martin et al. (2020b) for the C2GCP in a straightforward counterpart top-down approach, which is different from and less effective than the top-down approach proposed in the present paper.

2. A top-down modeling approach for the C2GCP and the C3GCP

Martin et al. (2020a) proposed pseudo-polynomial and compact MILP formulations for the C2GCP based on a bottom-up modeling approach. Although not explored by the authors, their models could be straightforwardly reformulated using a top-down modeling approach, possibly showing computational advantages on certain classes of instances. Based on this observation, we developed such counterpart top-down models in a first stage of the present study, as reported in the appendix, but it turned out that this straightforward approach was not competitive with the original bottom-up approaches of Martin et al. (2020a), based on preliminary computational experiments using a general-purpose optimization solver – see Appendix for details. Therefore, in Sections 2.2 and 2.3, we present a more competitive top-down modeling approach for the C2GCP and the C3GCP, respectively. These novel top-down models explicitly use a binary tree structure as input for indexing the variables and constraints, which is discussed in what follows.

2.1. An explicit binary tree structure

We make use of the same binary tree structure of the pseudo-polynomial formulation in Martin et al. (2020b), which is described in Algorithm 1. The algorithm receives an upper bound $\bar{n}$ on the number of items in any optimal solution of the problem instance. Then, it iterates in a top-down cutting approach for enumerating a binary tree structure represented by the triplets (j, j^-, j^+), where j^- and j^+ are the left and right child nodes of node j, respectively. Fig. 2a illustrates a binary tree structure generated by Algorithm 1 for $\bar{n} = 4$ items, where each node is represented

by a numeric identifier and its capacity (maximum number of items this node is capable of fitting in) in parenthesis. This scheme of enumerating the binary tree is without loss of optimality and has significantly fewer nodes than a full enumeration scheme (Martin et al., 2020b). Each node is classified according to the capacity of its child nodes into the subsets J_A, J_B, J_C and $\overline{J}$. Let $J = \{1, \ldots, \mathcal{N}\}$ be the set of all nodes in the binary tree. Set J_A contains all parent nodes of non-leaf nodes, set J_C contains all parent nodes of leaf nodes, and set $\overline{J}$ contains all leaf nodes. Set J_B is an intermediary case between sets J_A and J_C that contains all nodes with non-leaf left child node j^- and leaf right child node j^+. For instance, we have $J_A = \{1\}$, $J_B = \{2\}$, $J_C = \{3, 4\}$ and $\overline{J} = \{5, 6, 7, 8, 8\}$ for the binary tree in Fig. 2a. The subset of all non-leaf nodes can be represented by $J \backslash \overline{J}$. We note that there is a small typo (node j should be node t) in the description of the corresponding Algorithm 1 in Martin et al. (2020b).

Algorithm 1: Binary tree for the top-down approach.

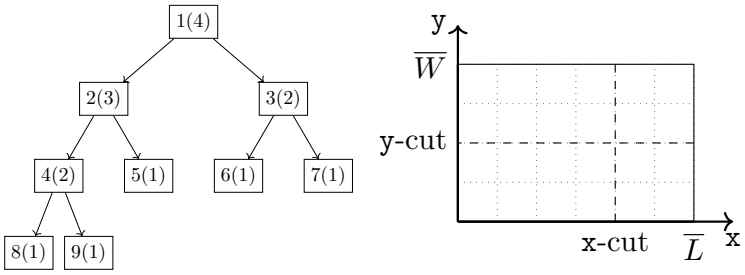
Input: Upper bound $\bar{n}$.

1 $r \leftarrow 1$; $J_A \leftarrow \{\}$; $J_B \leftarrow \{\}$; $J_C \leftarrow \{\}$; $\overline{J} \leftarrow \{\}$;
2 create root node r with capacity $\bar{n}$;
3 initialize queue $T = \{r\}$;
4 **while** *queue T is not empty* **do**
5 select and remove the first node t in queue T;
6 **if** *the capacity of node t is greater than 1* **then**
7 create new node $r + 1$ with a capacity equal to "capacity of node t" - 1;
8 create new node $r + 2$ with a capacity equal to $\lfloor$"capacity of node t"$/2\rfloor$;
9 set node $r + 1$ as the left child (j^-) of node t, and set node $r + 2$ as the right child (j^+) of node t, that is, set triplet $(t, r + 1, r + 2)$ in the binary tree;
10 **if** *both nodes $r + 1$ and $r + 2$ have a capacity greater than 1* **then**
11 $J_A \leftarrow J_A \cup \{t\}$;
12 **if** *capacity of node $r + 1$ is greater than 1 and capacity of node $r + 2$ is 1* **then**
13 $J_B \leftarrow J_B \cup \{t\}$;
14 **if** *both nodes $r + 1$ and $r + 2$ have a capacity equal to 1* **then**
15 $J_C \leftarrow J_C \cup \{t\}$;
16 insert nodes $r + 1$ and $r + 2$ at the end of queue T;
17 $r \leftarrow r + 2$;
18 **else**
19 $\overline{J} \leftarrow \overline{J} \cup \{t\}$;
20 $\mathcal{N} \leftarrow r$;

Output: Triplets (j, j^-, j^+) of the binary tree, parameter $\mathcal{N}$, and sets J_A, J_B, J_C and $\overline{J}$.

We highlight that Algorithm 1 enumerates a binary tree structure for the C2GCP and the C3GCP. In this sense, parameter $\bar{n}$ is an upper bound for the number of items in any optimal solution of the corresponding problem instance. These nodes represent sub-patterns, that is, rectangles for the C2GCP and rectangular cuboids for the C3GCP. As shown next in the models, the difference appears in the branches of the tree as they depend on the number of possible guillotine orientations to cut the nodes.

In the computational experiments of Section 3, we also consider a heuristic enumeration of the binary tree structure to obtain fewer nodes and reduce the computational times. This heuristic is similar to Algorithm 1, except that in line 7 it creates a new node $r + 1$ with capacity equal to $\lceil$"capacity of node t"$/2\rceil$ items, instead of "capacity of node t"-1 items. For instance, using this different enumeration scheme in the binary tree structure of Fig. 2a, nodes 8 and 9 would not exist,



(a) Enumeration of Algorithm 1. (b) Guillotine cuts for the C2GCP.

Figure 2: Illustrations of a binary tree structure with $\bar{n} = 4$ items and of the guillotine cuts for the C2GCP.

as the capacity of nodes 2 and 4 are equal to 2 and 1 items, respectively. It is a heuristic enumeration because it cannot represent all cutting patterns with $\bar{n}$ items. Indeed, it cannot represent e.g. a cutting pattern with $\hat{n} = 4$ items that is cut to produce a sub-pattern with $\hat{n} - 1 = 3$ items and a sub-pattern with only 1 item. In this sense, this heuristic enumeration indirectly supposes that each child node provides half of the items of its parent node.

2.2. An MILP model for the C2GCP

The variables and constraints of the proposed model are related to the nodes of the binary tree structure generated by Algorithm 1. The four sets of variables of the model are presented next. Each node $j \in J$ can contain a copy of an item type $i \in I$, as expressed in Eq. (1). Let $O = \{\mathtt{x}, \mathtt{y}\}$ be the set of possible guillotine orientations to cut a node, as illustrated in Fig. 2b. Each non-leaf node $j \in J \backslash \overline{J}$, instead of containing a copy of an item type, can be cut across one of the Cartesian axes, either x-axis or y-axis, as expressed in Eq. (2). For modeling the problem, when a non-leaf node is cut (i.e., $\sum_{o \in O} x_j^o = 1$), we need to know its cut position and the size of the cut dimension, as expressed in Eqs. (3) and (4), respectively.

$$z_{ji} = \begin{cases} 1, & \text{if node } j \text{ contains a copy of item type } i, \\ 0, & \text{otherwise.} \end{cases} \qquad j \in J, i \in I. \quad (1)$$

$$x_j^o = \begin{cases} 1, & \text{if node } j \text{ is cut with orientation } o, \\ 0, & \text{otherwise.} \end{cases} \qquad j \in J \backslash \overline{J}, o \in O. \quad (2)$$

$$p_j^o\text{: cut position of node } j \text{ with orientation } o, \qquad j \in J \backslash \overline{J}. \quad (3)$$

$$q_j\text{: size of the cut dimension of node } j, \qquad j \in J \backslash \overline{J}. \quad (4)$$

Recall that v_i is the value of item type $i \in I$. The objective function consists of maximizing the total sum of the values of the items cut, given by

$$\textbf{Max} \sum_{j \in J} \sum_{i \in I} v_i z_{ji}. \quad (5)$$

For the sake of clarity, in the following we present the formulation in blocks of constraints with their respective explanation.

Constrained case and node options

Constraints (6) limit the cutting pattern to the constrained case, that is, the number of copies of item type $i \in I$ to its maximum number u_i. Constraints (7) ensure that each non-leaf node $j \in J \backslash \overline{J}$ represents up to one copy of an item type or is cut in one of the orientations. Constraints (8) impose that each leaf node $j \in \overline{J}$ represents up to one copy of an item type.

$$\sum_{j \in J} z_{ji} \leq u_i, \qquad i \in I. \quad (6)$$

$$\sum_{i \in I} z_{ji} + \sum_{o \in O} x_j^o \leq 1, \qquad j \in J \backslash \overline{J}. \quad (7)$$

$$\sum_{i \in I} z_{ji} \leq 1, \qquad j \in \overline{J}. \quad (8)$$

Cutting pattern as a binary tree

Constraints (9), (10) and (11) link the decisions of a parent node j to its child nodes j^- and j^+, according to the triplets (j, j^-, j^+) of the binary tree structure. When a parent node j is cut (i.e., $\sum_{o \in O} x_j^o = 1$), the corresponding left-hand side (LHS) of this constraint becomes 2, which forces the variables of its child nodes j^- and j^+ to assume non-zero values in the right-hand side (RHS) of the constraint, that is, they must be cut or represent copies of an item type. Note that each child node of a parent node $j \in J_A$ can fit in both cases, while each child node of a parent node $j \in J_C$ can only contain a copy of an item type. As mentioned before, child nodes of a parent node $j \in J_B$ fall in an intermediary case, as illustrated by node 2 in Fig. 2a. When a parent node j is not cut (i.e., $\sum_{o \in O} x_j^o = 0$), the corresponding LHS of this constraint becomes zero, which forces the variables related to its child nodes j^- and j^+ to assume zero values, that is, they must not represent copies of an item type neither be cut.

$$2 \sum_{o \in O} x_j^o = \sum_{o \in O} x_{j^-}^o + \sum_{i \in I} z_{j^- i} + \sum_{o \in O} x_{j^+}^o + \sum_{i \in I} z_{j^+ i}, \qquad j \in J_A. \quad (9)$$

$$2 \sum_{o \in O} x_j^o = \sum_{o \in O} x_{j^-}^o + \sum_{i \in I} z_{j^- i} + \sum_{i \in I} z_{j^+ i}, \qquad j \in J_B. \quad (10)$$

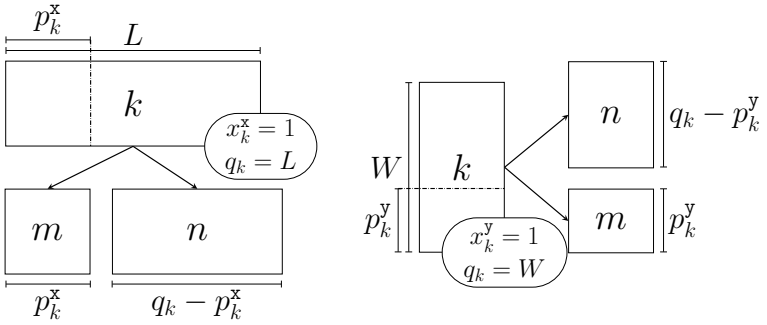
$$2 \sum_{o \in O} x_j^o = \sum_{i \in I} z_{j^- i} + \sum_{i \in I} z_{j^+ i}, \qquad j \in J_C. \quad (11)$$

Top-down geometric constraints

The object $\overline{L} \times \overline{W}$ cut to produce the items is represented by node $k = 1$ in the formulation. Let $L(j)$ and $R(j)$ be subsets containing ancestral nodes of $j \in J$ up to node $k = 1$. For any $j \in J$, set $L(j)$ contains all left ancestral nodes, while set $R(j)$ contains all right ancestral nodes. A node is called left ancestral if it originated j by a left branch in the tree; otherwise, it is called right ancestral. For instance, in the binary tree of Fig. 2a, node $j = 9$ has $L(9) = \{1, 2\}$ and $R(9) = \{4\}$, node $j = 5$ has $L(5) = \{1\}$ and $R(5) = \{2\}$, and node $j = 7$ has $L(7) = \{\}$ and $R(7) = \{1, 3\}$.

In the following top-down geometric constraints, the dimensions of a node take into account the decisions of all its ancestral nodes. The insight of these constraints is illustrated in Fig. 3, which supposes that k, m and n are nodes such that $k \in L(m)$, $k \in R(n)$ and m and n are not necessarily children of the same parent node. We denote as $L \times W$ the size of node k. If node k

is cut with orientation x (i.e., $x_k^{\mathtt{x}} = 1$), variable q_k assumes value L and then we have to set the cut position of node m in the x-axis using variable $p_k^{\mathtt{x}}$. Thus, the cut position of node n in the same axis becomes limited by $q_k - p_k^{\mathtt{x}}$. On the other hand, if node k is cut with orientation y (i.e., $x_k^{\mathtt{y}} = 1$), variable q_k assumes value W and we set the cut position of node m in the y-axis using $p_k^{\mathtt{y}}$, and thus $q_k - p_k^{\mathtt{y}}$ limits the cut position of node n also in the y-axis. Note that if node k is not cut (i.e., $\sum_{o \in O} x_k^o = 0$), the variables associated to nodes m and n assume the value of zero.



(a) x-cut. (b) y-cut.

Figure 3: Illustration of the guillotine geometric constraints in the top-down modeling approach.

Constraints (12) and (13) ensure that the size of the cut dimension of a non-leaf node does not exceed the dimensions of the object. Note that constraints (12) assume $\sum_{o \in O} x_1^o = 1$. Consider the sufficiently large numbers $\overline{M}^{\mathtt{x}} = \overline{L}$ and $\overline{M}^{\mathtt{y}} = \overline{W}$, and the parameters $\overline{d}^{\mathtt{x}} = \min_{i \in I}\{l_i\}$ and $\overline{d}^{\mathtt{y}} = \min_{i \in I}\{w_i\}$. Constraints (14) to (17) model the geometric constraints with disjunctive inequalities of big-M type.

$$q_1 = \overline{L} x_1^{\mathtt{x}} + \overline{W} x_1^{\mathtt{y}}. \tag{12}$$

$$q_j \leq \overline{L} x_j^{\mathtt{x}} + \overline{W} x_j^{\mathtt{y}}, \qquad j \in J \backslash \overline{J}. \tag{13}$$

$$p_j^o \leq p_k^o - \overline{d}^o + \overline{M}^o (2 - x_j^o - x_k^o), \qquad j \in J \backslash \overline{J}, k \in L(j), o \in O. \tag{14}$$

$$p_j^o \leq q_k - p_k^o - \overline{d}^o + \overline{M}^o (2 - x_j^o - x_k^o), \qquad j \in J \backslash \overline{J}, k \in R(j), o \in O. \tag{15}$$

$$q_j \leq p_k^o + \overline{M}^o (2 - x_j^o - x_k^o), \qquad j \in J \backslash \overline{J}, k \in L(j), o \in O. \tag{16}$$

$$q_j \leq q_k - p_k^o + \overline{M}^o (2 - x_j^o - x_k^o), \qquad j \in J \backslash \overline{J}, k \in R(j), o \in O. \tag{17}$$

Constraints (14) and (15) limit the cut positions of a non-leaf node j to the cut positions of its left and right ancestral nodes, respectively. Constraints (14) ensure that if node j and its left ancestral node $k \in L(j)$ are cut with the same orientation o (i.e., $x_j^o + x_k^o = 2$), then the cut position p_j^o of node j is less than or equal to the cut position p_k^o minus $\overline{d}^o$ (i.e., the minimum size of a cut under orientation o). Constraints (15) ensure that if node j and its right ancestral node $k \in R(j)$ are cut with the same orientation o (i.e., $x_j^o + x_k^o = 2$), then the cut position p_j^o of node j is less than or equal to the difference of the size q_k and the cut position p_k^o minus $\overline{d}^o$.

Constraints (16) and (17) limit the size of the cut dimension of a non-leaf node j to the cut positions of its left and right ancestral nodes, respectively. Constraints (16) ensure that if node j and its left ancestral node $k \in L(j)$ are cut with the same orientation o, then the size of the cut

dimension q_j^o of node j is less than or equal to the cut position p_k^o. Constraints (17) ensure that if node j and its right ancestral node $k \in R(j)$ are cut with the same orientation o, then the size of the cut dimension q_j^o of node j is less than or equal to the difference of the size q_k and the cut position p_k^o.

Knapsack constraints

Let $\overline{a}_i^o$ be equal to l_i if $o = \mathtt{x}$, and w_i otherwise. Constraints (18) and (19) ensure that a node $j \in J$ can contain a copy of item type $i \in I$ if l_i and w_i are less than or equal to the cut position in the left and right ancestral nodes with the same orientation (i.e., $x_k^o = 1$), respectively.

$$\sum_{i \in I} \overline{a}_i^o z_{ji} \leq p_k^o + \overline{M}_o(1 - x_k^o), \qquad j \in J, k \in L(j), o \in O. \quad (18)$$

$$\sum_{i \in I} \overline{a}_i^o z_{ji} \leq q_k - p_k^o + \overline{M}_o(1 - x_k^o), \qquad j \in J, k \in R(j), o \in O. \quad (19)$$

Domain of the variables

Constraints (20) to (23) impose the domain of the variables.

$$z_{ji} \in \{0, 1\}, \qquad j \in J, i \in I. \quad (20)$$

$$x_j^o \in \{0, 1\}, \qquad j \in J \backslash \overline{J}, o \in O. \quad (21)$$

$$0 \leq p_j^o \leq \overline{M}^o - \overline{d}^o, \qquad j \in J \backslash \overline{J}, o \in O. \quad (22)$$

$$0 \leq q_j \leq \max\{\overline{L}, \overline{W}\}, \qquad j \in J \backslash \overline{J}. \quad (23)$$

The resulting MILP formulation for the C2GCP based on the described top-down modeling approach is given by

$$\textbf{Max} \quad (5), \quad (24a)$$

$$\textbf{s.t.} \quad (6) - (23). \quad (24b)$$

Note that Model (24) is a pseudo-polynomial formulation, as its variables and constraints are indexed by the nodes of the binary tree generated by Algorithm 1.

Valid inequalities

Valid inequalities (25) were proposed by Martin et al. (2020b) to their pseudo-polynomial formulation for the C2GCP. These inequalities are also valid to Model (24) and hence we consider them in the computational experiments reported in Section 3. The reader is referred to Martin et al. (2020b) for further details on these inequalities; parameters UB_1 and UB_2 are upper bounds to the usable area of the object and to the value of the optimal solution, respectively.

$$\sum_{j \in J} \sum_{i \in I} (l_i w_i) z_{ji} \leq UB_1. \quad (25a)$$

$$\sum_{j \in J} \sum_{i \in I} v_i z_{ji} \leq UB_2. \quad (25b)$$

$$\sum_{j\in J}\sum_{i\in I} z_{ji} \leq \bar{n}. \tag{25c}$$

$$\sum_{j\in J\setminus\overline{J}}\sum_{o\in O} x_{jo} = \sum_{j\in J}\sum_{i\in I} z_{ji} - 1. \tag{25d}$$

$$\sum_{o\in O} x_{j^-o} \geq \sum_{o\in O} x_{j^+o}, \qquad j \in J_A. \tag{25e}$$

$$\sum_{i\in I} i\ z_{j^-i} \leq \sum_{i'\in I} i'\ z_{j^+i'}, \qquad j \in J\setminus\overline{J}. \tag{25f}$$

2.2.1. Illustrative example

Table 1 defines the input data of an illustrative example for the C2GCP. We assume $\bar{n} = 4$ and thus this example has the same binary tree structure as in Fig. 2a. The optimal value is 36 (two copies of $i = 1$ and two copies of $i = 2$) and Fig. 4 shows an optimal solution with the corresponding relevant values of the variables of Model (24). For instance, the length of nodes 4 and 5 are limited by the value of $p^{\mathrm{x}}_1 = 6$, that is, the cut on node 1, while their width are related to the cut on node 2. Similarly, the length of node 3 is limited by the value of $q_1 - p^{\mathrm{x}}_1 = 4$ regarding the cut on node 1. Notice that nodes 8 and 9 are not needed to represent this solution.

Table 1: Illustrative example for the C2GCP with $\overline{L} = 10$ and $\overline{W} = 4$.

Item type	l_i	w_i	v_i	u_i
1	6	2	12	2
2	2	3	6	3

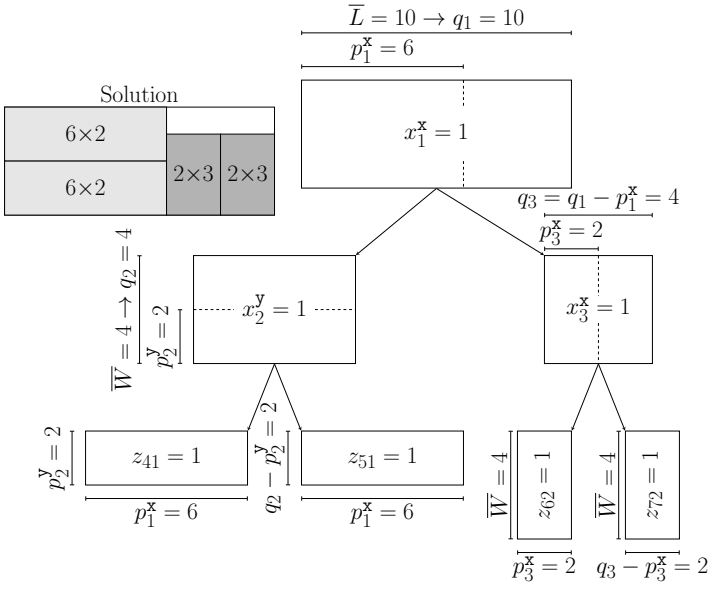


Figure 4: An optimal solution for the illustrative example defined in Table 1 according to Model (24).

2.3. An MILP model for the C3GCP

The proposed top-down modeling approach of the previous section can be extended to the C3GCP. To this purpose, the only set that needs to be redefined is the set of possible orientations to cut a node. Specifically, we redefine this set as $O = \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$, in which orientation $o = \mathtt{z}$ corresponds to heights $\overline{H}$ of the object and h_i of the items, for $i \in I$, in the $\mathtt{z}$-axis. Regarding the variables of the model, z_{ji} and q_j have the same definition as before, while x_j^o and p_j^o are redefined according to the new definition of set O.

The objective function is kept, and constraints (7), (9), (10) and (11) are adapted in order to take into account the new definition of set O. Constraint (12) is redefined to $q_1 = \overline{L}x_1^{\mathtt{x}} + \overline{W}x_1^{\mathtt{y}} + \overline{H}x_1^{\mathtt{z}}$, while constraints (13) are replaced with $q_j \leq \overline{L}x_j^{\mathtt{x}} + \overline{W}x_j^{\mathtt{y}} + \overline{H}x_j^{\mathtt{z}}$ for the corresponding non-leaf nodes. For constraints (14) to (17) we further define the parameters $\overline{M}^{\mathtt{z}} = \overline{H}$ and $\overline{d}^{\mathtt{z}} = \min_{i \in I}\{h_i\}$, and for constraints (18) to (19) we define the parameter $\overline{a}_i^{\mathtt{o}}$ equal to h_i if $o = \mathtt{z}$ for $i \in I$. Valid inequalities (25) are easily extendable for the C3GCP and hence, as for the C2GCP, we consider the resulting inequalities in the computational experiments reported in Section 3.

Similarly to Martin et al. (2020c), we highlight that this top-down modeling approach can be straightforwardly adapted to the n-dimensional case by adding the corresponding dimensions to set O with the previous adaptations. However, we are not aware of any industrial cutting application for a constrained n-dimensional guillotine cutting problem. The approach can also be easily adapted to cope with other cutting problems, like in the case of cutting a few large stocked objects to fulfill the demand of the ordered items, by considering these objects aggregated in a large-sized super-object.

3. Computational experiments

In this section we present the results of computational experiments performed with the proposed top-down approach for the C2GCP and C3GCP. Model (24) proposed for the C2GCP in Section 2.2 is called hereafter `2d-Top`, while the model described for the C3GCP in Section 2.3 is called hereafter `3d-Top`. This section is divided into two parts: we first report the results for the C2GCP in Section 3.1 and then the results for the C3GCP in Section 3.2. At the beginning of these sections, we describe the benchmark instances used in the experiments as well the benchmark models taken from the literature for each case. All the proposed and benchmark models were coded in C++ and all experiments were run on a PC with Intel Core i7-6700 (3.40 GHz) processor and 16 GB of RAM. We used the IBM CPLEX Optimization Studio v.12.8 as the general-purpose MILP solver. Similarly to Martin et al. (2020b), we limited the execution of the solver to 900 seconds, and in the tables we use the letters "tl" to indicate that this time limit was reached for a given instance.

3.1. Results for the C2GCP

For the C2GCP, we used two sets of benchmark instances. Set A consists of the instances cgcut1-3 (Christofides & Whitlock, 1977), OF1-2 (Oliveira & Ferreira, 1990), wang20 (Wang, 1983) and gcut1-12 (Beasley, 1985). The length and width of an item type in the gcut instances were uniformly sampled in the intervals $[\overline{L}/4, 3\overline{L}/4]$ and $[\overline{W}/4, 3\overline{W}/4]$, respectively. For the other instances in set A, the area of an item type was uniformly sampled in the interval $[0, 0.25\overline{LW}]$. Set B consists of instances CU1-11 and CW1-11 (Fayard et al., 1998), which are larger problem instances characterized by items with length and width uniformly sampled in the intervals $[0.1\overline{L}, 0.7\overline{L}]$ and $[0.1\overline{W}, 0.7\overline{W}]$, respectively. Sets A and B include unweighted instances (i.e., instances OF1-2,

wang20, gcut1-12 and CU1-11, where $v_i = l_i w_i$ for $i \in I$) and weighted instances (i.e., instances cgcut1-3 and CW1-11, where $v_i \neq l_i w_i$ for $i \in I$). We assumed that $u_i = 1$, for $i \in I$, in gcut1-12 instances.

For each instance in sets A and B, we report in Tables 2 and 3 the size of the object ($\overline{L} \times \overline{W}$), number of item types (m), total number of items ($n = \sum_{i \in I} u_i$), upper bound on the number of items in any optimal solution of the instance ($\bar{n}$), and the value of the optimal solution (OPT) obtained in Martin et al. (2020b) and Fayard et al. (1998). For the benchmark purposes, we considered the pseudo-polynomial and compact bottom-up models proposed in Martin et al. (2020b), which are called hereafter `2d-Bot-I` and `2d-Bot-II`, respectively. For each of the models, namely `2d-Top`, `2d-Bot-I` and `2d-Bot-II`, we also report in Table 2 the number of variables (var), number of constraints ($cons$), relative optimality gap in percentage ($gap[\%]$) and processing time in seconds ($time[s]$). The relative optimality gap is calculated as $(OPT - OFV)/(OPT + 10^{-10}) * 100$, where OFV is the objective function value of the corresponding model at the end of the execution, which may not be optimal if the time limit was reached. Similar to Martin et al. (2020b), for instances in set A, parameters $\bar{n}$, UB_1, UB_2, required for solving the proposed and benchmark models, were obtained by solving a relaxed version of the "Contiguous relaxations based model of the Two-dimensional Orthogonal Packing Problem" (Scheithauer, 2018). For set B, they were obtained by solving a one-dimensional knapsack problem that considers unitary value and the item types' value as objective function's coefficients, respectively, and also the item types' area and the object's area (the capacity of the knapsack).

The results in Table 2 show that for instances in set A the average optimality gaps of the solver with `2d-Top`, `2d-Bot-I` and `2d-Bot-II` were 0.07%, 0.33% and 0.34%, with the average processing times 159.90 seconds, 128.54 seconds and 259.21 seconds, respectively. The solver was able to find an optimal solution and prove its optimality for 16 (out of 18) instances using `2d-Top` or `2d-Bot-I`, and for 14 using `2d-Bot-II`. An optimal solution for instance OF1 was found by the solver using the three models, but its optimality was not proven within the time limit. Regarding the quality of solutions, the performance of the solver with `2d-Top` is better than with the benchmark models, given that the gap of instance cgcut2 (the only instance with optimality not proven using any of the models) was 1.24% with `2d-Top` and 5.95% and 6.12% with the benchmark models. In contrast, regarding processing times, the solver performed better with `2d-Bot-I` than with `2d-Top` and `2d-Bot-II`, on average. For example, instance gcut4 was solved in 152.39 seconds using `2d-Bot-I`, 509.84 seconds using `2d-Top` and 900.00 seconds using `2d-Bot-II`. There were also cases in which the performance of the solver was superior with `2d-Top`. For instance OF2, the processing time was 5.94 seconds with `2d-Top`, 121.56 seconds with `2d-Bot-I` and 224.07 seconds with `2d-Bot-II`.

Regarding instances in set B, the results in Table 3 show that the average optimality gaps of the solver with `2d-Top`, `2d-Bot-I` and `2d-Bot-II` were 0.21%, 0.60% and 1.07%, while the average processing times were 832.18 seconds, 832.23 seconds and 900.00 seconds, respectively. The solver found optimal solutions with `2d-Top` in 15 (out of 22) instances, in 11 instances with `2d-Bot-I`, and in 10 instances with `2d-Bot-II`. It proved optimality for instances CW2, CW10 and CW11 only with `2d-Top` and `2d-Bot-I`. The difficulty of proving optimality is related to the weak linear programming relaxation bounds (LP-bounds) provided by the proposed and benchmark models. Indeed, the LP-bound of the three models was exactly the value provided by parameter UB_1 for each unweighted instance (i.e., CU1-11) and parameter UB_2 for each weighted instance (i.e., CW1-11) of set B.

The average numbers of variables and constraints of `2d-Top` were 7,9171.36 and 5,338.05, of

2d-Bot-I were 7,721.55 and 1,244.95, and of 2d-Bot-II were 1,240.27 and 2,682.00, respectively, as presented in Table 3. Note that the number of constraints of 2d-Top quickly increases in $\bar{n}$ (and consequently in $\mathcal{N}$), given that the top-down geometric and knapsack constraints of 2d-Top link the decisions of a node to the decisions of all its ancestral nodes. This way, preliminary computational experiments revealed that 2d-Top was not competitive with the benchmark models for the APT instances (Álvarez-Valdés et al., 2002), which are characterized with larger values of $\bar{n}$.

There are other two formulations for the C2GCP (Ben Messaoud et al., 2008; Furini et al., 2016; Martin et al., 2020a) in the literature, but we chose not to include them as benchmark models. Ben Messaoud et al. (2008) argued that their formulation is limited to solve very small problem instances with a general-purpose solver. The bottom-up models of Martin et al. (2020b) outperform the model of Martin et al. (2020a) for non-staged patterns, which is based on object discretization. Additionally, although the formulation of Furini et al. (2016) outperforms the three models discussed in this section for instances of moderate dimensions of the object (e.g. cgcut1-3, OF1-2 and wang20), it is not competitive for the ones with larger dimensions of the object (e.g. gcut1-12, CU1-11 and CW1-11) (Martin et al., 2020b).

Finally, it is worth mentioning that we also performed tests with both sets of instances imposing a time limit of 3600 seconds, but the improvements were marginal. Note that the solver was able to obtain feasible solutions of good quality using the models, although the certificates of optimality were harmed by their weak LP-bounds.

Table 2: Results for the C2GCP with set of instances A.

						2d-Top				2d-Bot-I of Martin et al. (2020b)				2d-Bot-II of Martin et al. (2020b)			
Instance	$\overline{L} \times \overline{W}$	m	n	$\bar{n}$	OPT	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]
cgcut1	15 × 10	7	16	13	244	1,183	2,874	0.00	4.92	1,015	818	0.00	5.67	390	910	0.00	1.37
cgcut2	40 × 70	10	23	18	2,892	3,817	9,595	1.24	tl	3,394	2,044	5.95	tl	765	1,798	6.12	tl
cgcut3	40 × 70	19	62	10	1,860	1,324	1,192	0.00	8.28	1,237	441	0.00	4.14	477	1,039	0.00	30.72
OF1	70 × 40	10	24	10	2,737	982	1,600	0.00	tl	874	532	0.00	tl	345	782	0.00	tl
OF2	70 × 40	10	23	11	2,690	793	1,182	0.00	5.94	706	431	0.00	121.56	297	669	0.00	224.07
wang20	70 × 40	19	42	10	2,721	1,324	1,191	0.00	435.10	1,237	440	0.00	126.08	423	930	0.00	tl
gcut1	250 × 250	10	10	4	48,368	118	83	0.00	0.04	106	71	0.00	0.04	51	117	0.00	0.03
gcut2	250 × 250	20	20	6	59,307	443	252	0.00	4.65	416	153	0.00	2.60	150	337	0.00	5.71
gcut3	250 × 250	30	30	7	60,241	834	386	0.00	13.12	798	206	0.00	10.27	249	554	0.00	138.21
gcut4	250 × 250	50	50	9	60,942	2,404	860	0.00	509.84	2,338	370	0.00	152.39	516	1,132	0.03	tl
gcut5	500 × 500	10	10	5	195,582	172	142	0.00	0.21	154	100	0.00	0.21	74	168	0.00	0.31
gcut6	500 × 500	20	20	5	236,305	302	152	0.00	0.70	284	110	0.00	0.84	114	258	0.00	0.39
gcut7	500 × 500	30	30	6	238,974	633	262	0.00	2.74	606	163	0.00	2.25	200	447	0.00	8.32
gcut8	500 × 500	50	50	8	245,758	1,869	613	0.00	62.44	1,818	298	0.00	43.58	441	969	0.00	8.66
gcut9	1,000× 1,000	10	10	5	919,476	172	142	0.00	0.28	154	100	0.00	0.59	74	168	0.00	0.60
gcut10	1,000× 1,000	20	20	5	903,435	302	152	0.00	0.78	284	110	0.00	1.00	114	258	0.00	0.86
gcut11	1,000× 1,000	30	30	7	955,389	834	386	0.00	14.48	798	206	0.00	25.91	249	554	0.00	155.93
gcut12	1,000× 1,000	50	50	7	970,744	1,334	406	0.00	14.74	1,298	226	0.00	16.52	369	814	0.00	490.65
Averages						1,046.67	1,192.78	0.07	159.90	973.17	378.83	0.33	128.54	294.33	661.33	0.34	259.21

Table 3: Results for the C2GCP with set of instances B.

						2d-Top				2d-Bot-I of Martin et al. (2020b)				2d-Bot-II of Martin et al. (2020b)			
Instance	$\overline{L} \times \overline{W}$	m	n	$\bar{n}$	OPT	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]
CU1	100 × 125	25	82	13	12,330	3,217	2,891	0.15	tl	3,049	835	0.86	tl	762	1,671	0.00	tl
CU2	150 × 175	35	90	11	26,100	2,807	1,625	0.00	tl	2,699	557	2.38	tl	735	1,587	2.13	tl
CU3	134 × 125	45	158	16	16,723	9,745	6,188	0.69	tl	9,445	1,488	0.84	tl	1,575	3,392	2.28	tl
CU4	285 × 354	45	113	15	99,495	7,999	4,827	0.20	tl	7,753	1,229	1.33	tl	1,323	2,861	2.67	tl
CU5	456 × 385	50	120	14	173,364	7,433	3,822	0.53	tl	7,226	1,047	0.00	tl	1,378	2,951	0.24	tl
CU6	356 × 447	45	124	13	158,572	5,477	2,911	0.00	tl	5,309	855	1.19	tl	1,134	2,435	0.51	tl
CU7	563 × 458	25	56	11	247,150	2,077	1,615	0.00	tl	1,969	547	0.00	tl	595	1,297	0.00	tl
CU8	587 × 756	35	78	14	433,331	5,348	3,807	0.14	tl	5,141	1,032	0.14	tl	988	2,156	1.16	tl
CU9	856 × 785	25	76	11	657,055	2,077	1,615	0.00	tl	1,969	547	0.00	tl	665	1,437	0.00	tl
CU10	794 × 985	40	129	21	773,772	19,528	17,428	0.85	tl	18,856	3,268	0.21	tl	2,110	4,622	1.61	tl
CU11	977 × 953	50	134	24	924,696	36,965	30,032	2.02	tl	35,930	5,021	1.98	tl	3,013	6,561	3.89	tl
CW1	125 × 105	25	67	14	6,402	3,958	3,798	0.00	tl	3,751	1,023	0.00	tl	832	1,835	0.00	tl
CW2	145 × 165	35	63	11	5,354	2,807	1,626	0.00	409.66	2,699	558	0.00	589.17	685	1,488	0.00	tl
CW3	267 × 207	40	96	12	5,689	4,042	2,229	0.00	tl	3,904	707	0.00	tl	924	1,990	0.00	tl
CW4	465 × 387	39	86	14	6,175	5,904	3,812	0.00	tl	5,697	1,037	0.08	tl	1,053	2,291	0.08	tl
CW5	524 × 678	35	91	13	11,659	4,347	2,902	0.00	tl	4,179	846	0.00	tl	894	1,946	0.00	tl
CW6	781 × 657	55	149	16	12,923	11,755	6,199	0.00	tl	11,455	1,499	2.10	tl	1,755	3,763	2.60	tl
CW7	276 × 374	45	123	15	9,898	7,999	4,828	0.00	tl	7,753	1,230	0.00	tl	1,351	2,918	4.18	tl
CW8	305 × 287	60	168	16	4,605	12,760	6,204	0.00	tl	12,460	1,504	2.19	tl	1,875	4,008	2.19	tl
CW9	405 × 362	50	131	16	10,748	10,750	6,194	0.00	tl	10,450	1,494	0.00	tl	1,605	3,458	0.00	tl
CW10	992 × 970	60	130	11	6,515	4,632	1,651	0.00	583.77	4,524	583	0.00	344.47	1,125	2,393	0.00	tl
CW11	982 × 967	60	114	10	6,321	3,743	1,233	0.00	214.53	3,656	482	0.00	275.41	909	1,944	0.00	tl
Averages						7,971.36	5,338.05	0.21	832.18	7,721.55	1,244.95	0.60	832.23	1,240.27	2,682.00	1.07	900.00

3.2. Results for the C3GCP

In this section, we evaluate the models `3d-Top` and `3d-Top-HE`. The latter is a modified version of `3d-Top` that uses the heuristic enumeration of the binary tree, as discussed at the end of Section 2.1. The experiments used instances gcut_3d (set C) proposed by De Queiroz et al. (2012), who adapted instances gcut1-12 by adding the third dimension for each item type (randomly chosen from the dimensions already used for the other item types). For these instances, $u_i = \lfloor \overline{L}/l_i \rfloor \lfloor \overline{W}/w_i \rfloor \lfloor \overline{H}/h_i \rfloor$ for $i \in I$. We also adapted the gcut_3d instances by imposing $u_i = 1$ for $i \in I$, corresponding to the most possible constrained case, and refer to them as adapted instances of set C. To differentiate them from the original instances, their names are prefixed with 'c_'. For benchmark purposes, we considered the compact bottom-up model for the C3GCP proposed in Martin et al. (2020c), which is called hereafter `3d-Bot`.

Tables 4 and 5 show the results using the original and adapted instances of set C, respectively. Their headers are similar to those of Tables 2 and 3, but in column OPT we report the values of the optimal solutions as material utilization rates in percentage (i.e., $OFV/\overline{LWH} * 100$), which were obtained in Martin et al. (2020c). For set C and adapted set C, parameters $\bar{n}$ and UB_1 required in the three models were obtained by solving a one-dimensional knapsack problem that considers unitary value and the item types' value as objective function's coefficients, respectively, and also the item types' volume and the object's volume (the capacity of the knapsack).

The results in Table 4 show that for the original instances in set C, models `3d-Top`, `3d-Top-HE` and `3d-Bot` result in average optimality gaps equal to 5.57%, 0.98% and 9.14%, and average processing times equal to 900.00, 743.17 and 900.00 seconds, respectively. The solver was able to find proven optimal solutions for 2 (out of 12) instances using `3d-Top` and 5 instances using `3d-Top-HE`, while no optimal solution was found with `3d-Bot` within the time limit. Regarding the adapted instances of set C, the results in Table 5 show that the average optimality gaps of the solver with `3d-Top`, `3d-Top-HE` and `3d-Bot` were 2.25%, 0.85% and 5.21%, while the average processing times were 578.01 seconds, 450.17 seconds and 845.36 seconds, respectively. For these adapted instances, the solver found optimal solutions in 6 (out of 12) instances with `3d-Top`, 7 instances with `3d-Top-HE`, and 5 instances with `3d-Bot`. Note that the solver could not prove optimality in all the instances with an optimal solution, due to the time limit imposed; as discussed in the previous section, then main reason for this comes from the weak LP-bound provided by these models.

In summary, regarding the quality of solutions and processing times, model `3d-Top-HE` promoted the best overall performance of the solver in the experiments. With this model, for example, the solver found a proven optimal solution for instance c_gcut10_3d in 3.18 seconds, while it took 48.86 seconds when using `3d-Top`, and reached the time limit with `3d-Bot` without proving optimality. Yet, it is worth recalling that `3d-Top-HE` uses a heuristic way of enumerating the binary tree structure, and hence it may not include all possible cutting patterns for the C3GCP. For this reason, in c_gcut10_3d, gcut9_3d and c_gcut2_3d instances, the solver with `3d-Top-HE` finished before the time limit, but the optimality gap is not zero (i.e., with respect to the optimal solution considering all possible cutting patterns for the C3GCP).

Table 4: Results for the C3GCP with set of instances C.

						3d-Top				3d-Top-HE				3d-Bot of Martin et al. (2020c)			
Instance	$\overline{L} \times \overline{W} \times \overline{H}$	m	n	$\bar{n}$	OPT	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]
gcut1_3d	250 × 250 × 250	10	57	18	80.57	4,663	14,004	3.49	tl	571	712	0.00	129.96	952	2,664	16.48	tl
gcut2_3d	250 × 250 × 250	20	113	27	84.88	26,838	71,660	1.70	tl	1,398	1,256	0.62	tl	2,444	7,042	6.66	tl
gcut3_3d	250 × 250 × 250	30	207	27	92.48	36,968	71,670	5.33	tl	1,928	1,266	0.84	tl	3,016	8,768	5.16	tl
gcut4_3d	250 × 250 × 250	50	274	33	95.43	114,632	170,672	20.34	tl	3,666	1,657	4.74	tl	5,184	15,220	5.81	tl
gcut5_3d	500 × 500 × 500	10	60	21	84.34	7,402	25,486	2.21	tl	670	889	0.00	tl	1,280	3,612	4.82	tl
gcut6_3d	500 × 500 × 500	20	82	22	84.84	13,853	30,910	8.51	tl	1,133	958	0.00	tl	1,743	4,999	17.37	tl
gcut7_3d	500 × 500 × 500	30	132	22	88.11	19,083	30,920	1.59	tl	1,563	968	0.08	tl	2,100	6,080	5.20	tl
gcut8_3d	500 × 500 × 500	50	270	22	93.20	29,543	30,940	2.01	tl	2,423	988	0.32	tl	3,003	8,809	3.61	tl
gcut9_3d	1000 × 1000 × 1000	10	82	18	93.16	4,663	14,004	0.00	tl	571	712	4.46	668.45	969	2,715	10.18	tl
gcut10_3d	1000 × 1000 × 1000	20	75	12	85.16	2,458	3,176	0.00	tl	603	412	0.00	19.61	715	2,035	14.88	tl
gcut11_3d	1000 × 1000 × 1000	30	133	18	91.44	10,323	14,024	12.08	tl	1,271	732	0.00	tl	1,598	4,622	12.05	tl
gcut12_3d	1000 × 1000 × 1000	50	251	27	92.65	57,228	71,690	9.63	tl	2,988	1,286	0.73	tl	3,952	11,596	7.49	tl
Averages						27,304.50	45,763.00	5.57	900.00	1,565.42	986.33	0.98	743.17	2,246.33	6,513.50	9.14	900.00

Table 5: Results for the C3GCP with adapted instances of set C.

						3d-Top				3d-Top-HE				3d-Bot of Martin et al. (2020c)			
Instance	$\overline{L} \times \overline{W} \times \overline{H}$	m	n	$\bar{n}$	OPT	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]	var	cons	gap[%]	time[s]
c_gcut1_3d	250 × 250 × 250	10	10	8	62.73	571	807	0.00	12.84	241	214	0.00	0.86	196	516	0.00	291.05
c_gcut2_3d	250 × 250 × 250	20	20	13	73.83	2,988	4,166	1.68	tl	656	460	0.72	458.11	576	1,606	2.74	tl
c_gcut3_3d	250 × 250 × 250	30	30	17	87.09	8,644	11,182	10.37	tl	1,198	673	3.25	tl	1,056	3,008	13.90	tl
c_gcut4_3d	250 × 250 × 250	50	50	18	90.86	15,983	14,044	0.00	tl	1,971	752	1.39	tl	1,496	4,336	9.74	tl
c_gcut5_3d	500 × 500 × 500	10	10	8	56.86	571	807	0.00	32.39	241	214	0.00	1.74	196	516	0.00	853.26
c_gcut6_3d	500 × 500 × 500	20	20	10	76.28	1,557	1,698	0.00	425.76	497	318	0.00	21.11	378	1,048	0.00	tl
c_gcut7_3d	500 × 500 × 500	30	30	13	80.88	4,118	4,176	0.36	tl	906	470	0.00	420.98	696	1,976	15.01	tl
c_gcut8_3d	500 × 500 × 500	50	50	17	89.01	13,384	11,202	8.95	tl	1,858	693	2.83	tl	1,376	3,988	10.39	tl
c_gcut9_3d	1000 × 1000 × 1000	10	10	9	64.14	736	1,164	0.00	116.21	274	261	0.00	2.71	240	636	0.00	tl
c_gcut10_3d	1000 × 1000 × 1000	20	20	9	71.63	1,186	1,174	0.00	48.86	444	271	0.00	3.18	320	886	0.00	tl
c_gcut11_3d	1000 × 1000 × 1000	30	30	14	78.21	5,067	5,500	3.29	tl	979	518	0.00	tl	780	2,216	3.29	tl
c_gcut12_3d	1000 × 1000 × 1000	50	50	16	87.68	11,350	8,993	2.38	tl	1,745	634	2.00	tl	1,260	3,652	7.48	tl
Averages						5,512.92	5,409.42	2.25	578.01	917.50	456.50	0.85	450.72	714.17	2,032.00	5.21	845.36

4. Conclusions

We addressed the Constrained Two-dimensional Guillotine Cutting Problem (C2GCP) and the Constrained Three-dimensional Guillotine Cutting Problem (C3GCP). These problems are relevant in industrial settings that cut a few large stocked objects to produce ordered item types with low demand, seeking minimal trim waste with as many guillotine stages as necessary. For instance, they appear in the cutting of flat glass in the glass industry, wooden boards in the furniture industry, rocks in the granite and marble industries, foams in the mattress industry, steel blocks in the metallurgical industry, among others. These problems have been mainly addressed by Dynamic Programming algorithms, tree search algorithms and heuristics/meta-heuristics (Russo et al., 2020).

The proposition of new and effective formulations to model the C2GCP and C3GCP has increased in the literature in the last years (Ben Messaoud et al., 2008; Furini et al., 2016; Martin et al., 2020a,b,c). In this sense, this work proposes a novel and promising top-down cutting modeling approach to these problems, leading to new pseudopolynomial MILP models. The insight of the approach is to cut the object towards the items in such a way that the sizes of the items are defined according to the decisions taken in the previous residual sub-objects. Computational experiments performed with benchmark instances showed that the proposed MILP models are competitive with state-of-the-art models of the literature in quality of solution and processing times, particularly when the number of items in an optimal solution is moderate.

We highlight that the presented top-down cutting approach could be straightforwardly adapted to the constrained n-dimensional guillotine cutting problem by adding the corresponding dimensions to set O, as discussed in Section 2. Additionally, it could be easily adapted to cope with other cutting problems, like in the case of cutting a few large stocked objects to fulfill the demand of the ordered items, by considering these objects aggregated in a super-object. As future research, the approach could be enhanced with the proposition of new valid inequalities to strengthen the linear relaxation of the MILP models, or even to eliminate possible symmetrical solutions. Another possibility is to develop customized solution methods based on decomposition methods for large scale optimization problems, like Dantzig-Wolfe decomposition, Benders decomposition and Lagrangian relaxation. These tailor-made solution methods could enable the solution of larger problem instances and/or contribute to obtain more certificates of optimality.

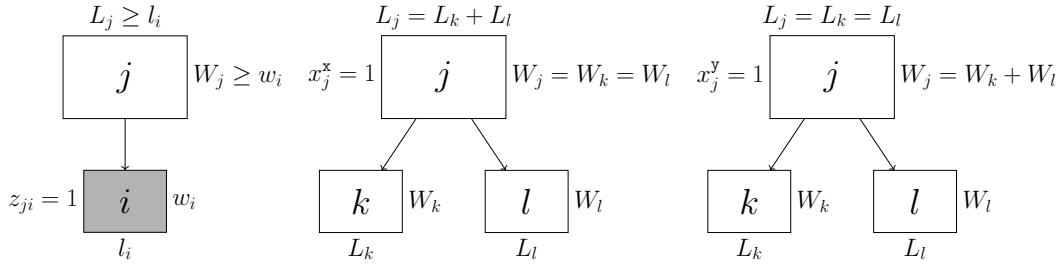
Acknowledgements

The authors would like to thank the National Council for Scientific and Technological Development (CNPq-Brazil) [grant number 200745/2018-2] and the São Paulo Research Foundation (FAPESP-Brazil) [grant numbers 16/08039-1, 16/01860-1] for the financial support. This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP-Brazil [grant number 13/07375-0].

Appendix. Models of Martin et al. (2020b) reformulated in a top-down approach

In what follows we discuss how to straightforwardly reformulate the two bottom-up models proposed in Martin et al. (2020b) for the C2GCP, namely the `2d-Bot-I` and the `2d-Bot-II`, in

a corresponding top-down modeling approach. Notice that in a bottom-up perspective, the waste appears at the beginning of the merging process when copies of item types with different dimensions are combined, while in the corresponding top-down approach the waste appears at the end of the cutting process when a rectangular node is used to contain a copy of an item type smaller than its dimensions. The `2d-Bot-I`, in its original bottom-up form, has four families of variables: z_{ji} for nodes containing copies of item types, x_{jo} for guillotine cuts on the nodes, and L_j and W_j for the dimensions of the nodes. All these variables are kept in its corresponding top-down form, but the domain of variables L_j and W_j is redefined to $j \in J$ (instead of the originally $J \backslash \overline{J}$), because we need to know the dimensions of each node to allow it to contain a copy of an item type. Fig. 5 illustrates the insight of the geometric constraints of `2d-Bot-I` in its counterpart top-down approach, showing that a node may contain a copy of an item type, or be cut in `x`-axis or `y`-axis.



(a) A copy of an item type. (b) `x`-cut. (c) `y`-cut

Figure 5: Illustration of geometric constraints of `2d-Bot-I` in its counterpart top-down modeling approach.

The `2d-Bot-I` in its straightforward counterpart top-down form is given by Model (26). We highlight that `2d-Top` (defined in Section 2.2) and `2d-Bot-I` (in its bottom-up and top-down forms) are based on the same binary tree structure, having the same objective function (5) and sharing constraints (6) to (11). Constraints (26a) to (26f) ensure the top-down geometric constraints, as depicted in Fig. 5. Constraints (26g) to (26i) impose the domain of the variables. Note that constraints (26j) fix variables L_1 and W_1 to the size of the object.

Max (5),

s.t.

(6) – (11),

$$x_j^{\mathtt{x}} = 1 \implies L_j = L_{j^-} + L_{j^+}, \qquad j \in J \backslash \overline{J}, \tag{26a}$$

$$x_j^{\mathtt{x}} = 1 \implies W_j = W_{j^*}, \qquad j \in J \backslash \overline{J}, j^* \in \{j^-, j^+\}, \tag{26b}$$

$$x_j^{\mathtt{y}} = 1 \implies L_j = L_{j^*}, \qquad j \in J \backslash \overline{J}, j^* \in \{j^-, j^+\}, \tag{26c}$$

$$x_j^{\mathtt{y}} = 1 \implies W_j = W_{j^-} + W_{j^+}, \qquad j \in J \backslash \overline{J}, \tag{26d}$$

$$\sum_{i \in I} l_i z_{ji} \leq L_j, \qquad j \in J, \tag{26e}$$

$$\sum_{i \in I} w_i z_{ji} \leq W_j, \qquad j \in J, \tag{26f}$$

$$z_{ji} \in \{0, 1\}, \qquad j \in J, i \in I, \tag{26g}$$

$$x_{jo} \in \{0,1\}, \qquad j \in J \backslash \overline{J}, o \in O, \tag{26h}$$
$$0 \leq L_j \leq L, 0 \leq W_j \leq W, \qquad j \in J, \tag{26i}$$
$$L_1 = \overline{L}, W_1 = \overline{W}. \tag{26j}$$

The `2d-Bot-II`, in its bottom-up form, has five families of variables: z_{jip} for rectangles containing copies of item types, x_{jo} for guillotine cuts on the rectangles, y_{jk} for linking a rectangle j that contain a rectangle k, and L_j and W_j for the dimensions of the rectangles. Any rectangle in its top-down form, if any, represents a sub-pattern that will be further cut or will contain a copy of an item type, so that the set of rectangles J is redefined to $\{1, \ldots, 2\bar{n} - 1\}$, instead of the originally defined set $\{1, \ldots, \bar{n} - 1\}$. From this redefinition of set J, all these variables are kept in its top-down form, but index p is no longer needed in variables z_{jip} (i.e., now there is z_{ji}). The `2d-Bot-II` in its straightforward, counterpart top-down form is given by Model (27).

$$\textbf{Max} \sum_{j \in J} \sum_{i \in I} v_i z_{ji}, \tag{27a}$$

s.t.

$$\sum_{j \in J} z_{ji} \leq u_i, \qquad i \in I, \tag{27b}$$
$$\sum_{i \in I} z_{ji} + \sum_{o \in O} x_{jo} \leq 1, \qquad j \in J, \tag{27c}$$
$$2 \sum_{o \in o} x_{jo} = \sum_{k \in J, j < k} y_{jk}, \qquad j \in J \tag{27d}$$
$$\sum_{j \in J, j < k} y_{jk} = \sum_{o \in O} x_{ko} + \sum_{i \in I} z_{ki}, \qquad k \in \{2, \ldots, \bar{n}\} \tag{27e}$$
$$x_j^{\mathtt{x}} = 1 \implies L_j = \sum_{k \in J, j < k} L_k y_{jk}, \qquad j \in J, \tag{27f}$$
$$x_j^{\mathtt{x}} + y_{jk} = 2 \implies W_j = \sum_{k \in J, j < k} W_k y_{jk}, \qquad j \in J, k \in K, j < k, \tag{27g}$$
$$x_j^{\mathtt{y}} + y_{jk} = 2 \implies L_j = \sum_{k \in J, j < k} L_k y_{jk}, \qquad j \in J, k \in K, j < k, \tag{27h}$$
$$x_j^{\mathtt{y}} = 1 \implies W_j = \sum_{k \in J, j < k} W_k y_{jk}, \qquad j \in J, \tag{27i}$$
$$\sum_{i \in I} l_i z_{ji} \leq L_j, \qquad j \in J, \tag{27j}$$
$$\sum_{i \in I} w_i z_{ji} \leq W_j, \qquad j \in J, \tag{27k}$$
$$z_{ji} \in \{0,1\}, \qquad j \in J, i \in I, \tag{27l}$$
$$y_{jk} \in \{0,1\}, \qquad j \in J, k \in J, j < k, \tag{27m}$$
$$x_{jo} \in \{0,1\}, \qquad j \in J, o \in O, \tag{27n}$$
$$0 \leq L_j \leq L, 0 \leq W_j \leq W, \qquad j \in J, \tag{27o}$$
$$L_1 = \overline{L}, W_1 = \overline{W}. \tag{27p}$$

The objective function (27a) consists of maximizing the total sum of values of the selected copies of item types. Constraints (27b) ensure the constrained case, that is, the production of up to u_i copies for each $i \in I$. Constraints (27c) to (27e) guarantee a cutting pattern represented as a binary tree by using variables y_{jk} to link the rectangles (i.e., sub-patterns) $j \in J$ and $k \in J$, $j < k$. If $x_j^{\mathtt{x}} = 1$, constraints (27f) ensure that a rectangle j has its length L_j equal to the sum of the lengths of the rectangles contained by it, while constraints (27g) ensure that a rectangle j has its width W_j equal to the widths of the rectangles contained by it (i.e., if $y_{jk} = 1$). If $x_j^{\mathtt{y}} = 1$, constraints (27h) ensure that a rectangle j has its length L_j equal to the lengths of the rectangles contained by it (i.e., if $y_{jk} = 1$), while constraints (27i) ensure that a rectangle j has its width W_j equal to the sum of the widths of the rectangles contained by it. Constraints (27j) and (27k) guarantee that a rectangle j contains an item type $i \in I$ only if $l_i \leq L_j$ and $w_i \leq W_j$. Constraints (27l) to (27o) state the domain of the variables. Note that constraints (27p) fixes variables L_1 and W_1 to the size of the object.

As discussed previously, in our preliminary computational experiments using the same general-purpose optimization solver and computational environment described in Section 3, Models (26) and (27) were not competitive with `2d-Bot-I` and `2d-Bot-II`. For instances in set A the average optimality gaps of the solver with Models (26) and (27) were 0.16% and 1.38%, with the average processing times 535.10 seconds and 643.25 seconds, respectively – please compare with the averages in the last row of Table 2. For instances in set B the average optimality gaps of the solver with Models (26) and (27) were 6.37% and 3.64%, and the time limit was reached in all these instances with both models – please compare with the averages in the last row of Table 3. These experiments motivated us to develop a non-trivial way of modeling the top-down geometric constraints in `2d-Top`.

References

Álvarez-Valdés, R., Parajón, A., & Tamarit, J. M. (2002). A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers & Operations Research*, *29*, 925–947. doi:10.1016/S0305-0548(00)00095-2.

Amossen, R. R., & Pisinger, D. (2010). Multi-dimensional bin packing problems with guillotine constraints. *Computers & Operations Research*, *37*, 1999–2006. doi:10.1016/j.cor.2010.01.017.

Beasley, J. E. (1985). Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. *The Journal of the Operational Research Society*, *36*, 297. doi:10.2307/2582416.

Ben Messaoud, S., Chu, C., & Espinouse, M. L. (2008). Characterization and modelling of guillotine constraints. *European Journal of Operational Research*, *191*, 110–124. doi:10.1016/j.ejor.2007.08.029.

Christofides, N., & Hadjiconstantinou, E. (1995). An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. *European Journal of Operational Research*, *83*, 21–38. doi:10.1016/0377-2217(93)E0277-5.

Christofides, N., & Whitlock, C. (1977). An Algorithm for Two-Dimensional Cutting Problems. *Operations Research*, *25*, 30–44. doi:10.1287/opre.25.1.30.

Cung, V.-d., Hifi, M., & Cun, Le, B. (2000). Constrained two-dimensional cutting stock problems a best- first branch-and-bound algorithm. *International Transactions in Operational Research*, *7*, 185–210. doi:10.1111/j.1475-3995.2000.tb00194.x.

De Queiroz, T. A., Hokama, P. H. D. B., Schouery, R. C. S., & Miyazawa, F. K. (2017). Two-dimensional disjunctively constrained knapsack problem: Heuristic and exact approaches. *Computers & Industrial Engineering*, *105*, 313 – 328. doi:10.1016/j.cie.2017.01.015.

De Queiroz, T. A., Miyazawa, F. K., Wakabayashi, Y., & Xavier, E. C. (2012). Algorithms for 3D guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers & Operations Research*, *39*, 200–212. doi:10.1016/j.cor.2011.03.011.

Do Nascimento, O. X., De Queiroz, T. A., & Junqueira, L. (2019). A MIP-CP based approach for two- and three-dimensional cutting problems with staged guillotine cuts. *Annals of Operations Research*, . doi:10.1007/s10479-019-03466-x.

Dolatabadi, M., Lodi, A., & Monaci, M. (2012). Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research*, *39*, 48–53. doi:10.1016/j.cor.2010.12.018.

Fayard, D., Hifi, M., & Zissimopoulos, V. (1998). An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society*, *49*, 1270–1277. doi:10.1057/palgrave.jors.2600638.

Furini, F., Malaguti, E., & Thomopulos, D. (2016). Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. *INFORMS Journal on Computing*, *28*, 736–751. doi:10.1287/ijoc.2016.0710.

Hifi, M. (2002). Approximate algorithms for the container loading problem. *International Transactions in Operational Research*, *9*, 747–774. doi:10.1111/1475-3995.00386.

Hifi, M. (2004). Exact algorithms for unconstrained three-dimensional cutting problems: A comparative study. *Computers & Operations Research*, *31*, 657–674. doi:10.1016/S0305-0548(03)00019-4.

Martin, M., Birgin, E. G., Lobato, R. D., Morabito, R., & Munari, P. (2020a). Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. *International Transactions in Operational Research*, *27*, 767–793. doi:10.1111/itor.12703.

Martin, M., Morabito, R., & Munari, P. (2020b). A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. *Computers & Operations Research*, *115*. doi:10.1016/j.cor.2019.104851.

Martin, M., Oliveira, J. F., Silva, E., Morabito, R., & Munari, P. (2020c). *Three-dimensional guillotine cutting problems with constrained patterns: MILP formulations and a bottom-up algorithm*. Technical Report UFSCAR-DEP-2020-99 Production Engineering Department, Federal University of São Carlos Brasil.

Morabito, R., & Arenales, M. (1994). An and/or-graph approach to the container loading problem. *International Transactions in Operational Research*, *1*, 59–73. doi:10.1111/1475-3995.d01-8.

Morabito, R., & Pureza, V. (2010). A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. *Annals of Operations Research*, *179*, 297–315. doi:10.1007/s10479-008-0457-4.

Oliveira, J., & Ferreira, J. (1990). An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, *44*, 256–266. doi:10.1016/0377-2217(90)90361-E.

Parada, V., Muñoz, R., & de Alvarenga, A. G. (1995). A hybrid genetic algorithm for the two - dimensional guillotine cutting problem. In J. Biethahn, & V. Nissen (Eds.), *Evolutionary Algorithms in Management Applications* (pp. 183–196). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-61217-6_9.

Russo, M., Boccia, M., Sforza, A., & Sterle, C. (2020). Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. *International Transactions in Operational Research*, *27*, 794–834. doi:10.1111/itor.12687.

Scheithauer, G. (2018). *Introduction to cutting and packing optimization: problems, modeling approaches, solution methods*. International Series in Operations Research and Management Science. Springer. doi:10.1007/978-3-319-64143-0.

Velasco, A. S., & Uchoa, E. (2019). Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research*, *272*, 106 – 120. doi:10.1016/j.ejor.2018.06.016.

Viswanathan, K. V., & Bagchi, A. (1993). Best-First Search Methods for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research*, *41*, 768–776. doi:10.1287/opre.41.4.768.

Wang, P. Y. (1983). Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research*, *31*, 573–586. doi:10.1287/opre.31.3.573.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*, 1109–1130. doi:10.1016/j.ejor.2005.12.047.

Yoon, K., Ahn, S., & Kang, M. (2013). An improved best-first branch-and-bound algorithm for constrained two-dimensional guillotine cutting problems. *International Journal of Production Research*, *51*, 1680–1693. doi:10.1080/00207543.2012.693965.