**Noname manuscript No.**
(will be inserted by the editor)

# Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern

**Mateus P. Martin · Ernesto G. Birgin · Rafael D. Lobato · Reinaldo Morabito* · Pedro Munari**

Received: date / Accepted: date

**Abstract** In this paper, we address the Constrained Two-dimensional Rectangular Guillotine Single Large Placement Problem (2D_R_CG_SLOPP). This problem involves cutting a rectangular object to produce smaller rectangular items from orthogonal guillotine cuts.In addition, there is an upper limit on the number of copies that can be produced of each item type. To model this problem, we propose a new compact integer non-linear programming (INLP) formulation and obtain an equivalent integer linear programming (ILP) formulation from it. Additionally, we developed a procedure to reduce the numbers of variables and constraints of the ILP formulation, without loss of optimality. From the ILP formulation, we derive two new compact models for particular cases of the 2D_R_CG_SLOPP, which consider only 2-staged or 1-group patterns. Finally, as a specific solution method for the 2D_R_CG_SLOPP, we apply Benders decomposition to the proposed ILP formulation and develop a branch-and-Benders-cut algorithm. All proposed approaches are evaluated through computational experiments using benchmark instances and compared with other formulations available in the literature. The results show that the new formulations are appropriate in scenarios characterized by few item types that are large with respect to the object's dimensions.

**Keywords** Cutting & packing problems · Constrained two-dimensional guillotine cuts · Integer programming models · branch-and-Benders-cut algorithm.

## 1 Introduction

Cutting operations in manufacturing industries can be seen as a policy for economies of scale in the purchase and storage of raw material (objects), or even to avoid risks of object unavailabilities in future purchases. In fact, cutting a large stocked object to produce the required items of demands known *a*

Mateus P. Martin
Department of Production Engineering, Federal University of São Carlos, Brazil
E-mail: mateus.pmartin@gmail.com

Ernesto G. Birgin
Department of Computer Science, University of São Paulo, Brazil
E-mail: egbirgin@ime.usp.br

Rafael D. Lobato
Department of Computer Science, University of São Paulo, Brazil
E-mail: rafael.lobato@gmail.com

Reinaldo Morabito
*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
E-mail: morabito@ufscar.br

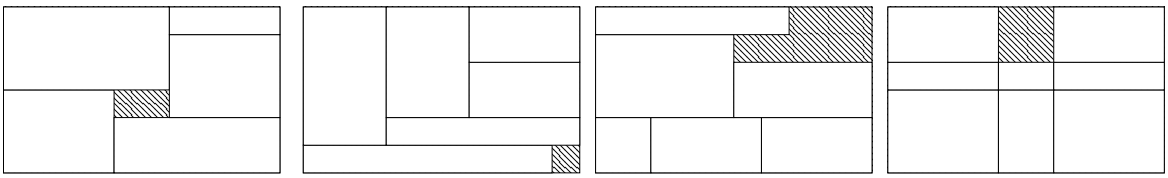
Pedro Munari
Department of Production Engineering, Federal University of São Carlos, Brazil
E-mail: munari@dep.ufscar.br

*posteriori* is often a safer economical policy than storing several required items of demand unknown *a priori*. These decisions are known as *Cutting and Packing* (C&P) problems in the field of Operations Research – see Wäscher et al (2007) for a survey and typology. They may represent the cutting of steel bars, paper reels, wooden boards, flat glass, stones, among other types of objects and items from different productive systems. C&P problems can also be analyzed in a wider framework, for instance, in logistical scenarios (de Queiroz et al, 2017; Côté et al, 2017) or in combination with other manufacturing decisions, such as layout, setups and lot-sizing issues (Linhares and Yanasse, 2002; Henn and Wäscher, 2013; Melega et al, 2018).

This paper addresses the Constrained Two-dimensional Rectangular Guillotine Single Large Placement Problem (2D_R_CG_SLOPP). This problem considers cutting a rectangular stocked object of dimensions $L \times W$ to produce an assortment of rectangular small item types, while maximizing the total (usable) area or the sum of the values of the cut items. Each item type $k \in \{1, \ldots, m\}$ is characterized by its dimensions $l_k \times w_k$, value $v_k$, and maximum number of copies to be cut $r_k$. The cutting must satisfy the following requirements:

i. Cuts are orthogonal to the object's edges and of guillotine-type, i.e., any small item of the cutting pattern is obtained by a sequence of edge-to-edge cuts;
ii. The number of guillotine stages is unlimited;
iii. No rotations of items is allowed (fixed orientation);
iv. The items must not overlap each other and must comply with the object dimensions;
v. For each item type $k$ up to $r_k$ copies can be produced.

According to Wäscher et al (2007), the 2D_R_CG_SLOPP is a variant of the standard Two-dimensional Rectangular Single Large Placement Problem (2D_R_SLOPP) in which the cuts are of the guillotine-type and there is a limit on the number of copies that can be produced of each item type. Additionally, we address two special types of guillotine patterns, namely 2-staged and 1-group patterns, which arise in productive systems in which shorter cycle times are sought even if detrimental to exploitation rates. In such contexts, cutting patterns with a few guillotine stages are likely to be more relevant than highly-staged patterns – the number of stages refers to the quantity of $90^\circ$ rotations of the cutting saw. In 2-staged patterns, items are obtained by first cutting horizontal (resp. vertical) shelves that are then followed by vertical (resp. horizontal) cuts – a shelf is any edge-to-edge cut on the large object. In 1-group patterns, there are horizontal and vertical shelves, and shelves which are cut in the first stage are stacked and cut together in the second stage. Fig. 1 illustrates different types of patterns – blank rectangles represent the items and the hatched areas are waste. The term *non-exact* refers to the possibility of an additional cut for splitting an item and waste for guillotine patterns, whereas in the *exact* case this step is not allowed. The approaches proposed in this paper can also be used to address the non-fixed orientation case, if we add a new item type $k'$ with length $w_k$, width $l_k$, and value $v_k$, for each $k \in K$.



(a) Non-guillotine. (b) Exact 5-staged. (c) Non-exact 2-staged. (d) Exact 1-group.

Fig. 1: Different types of two-dimensional cutting patterns.

### 1.1 Related work

The 2D_R_CG_SLOPP is generally addressed by search tree based strategies due to an intrinsic characteristic of the problem: each cut on a rectangle always generates two smaller rectangles. The literature presents two main approaches for search trees: *top-down* and *bottom-up* approaches. The former considers

cut operations over the original object (and the residual sheets) to obtain the most valuable subset of demanded pieces. The latter takes the small pieces (and the built sub-patterns) belonging to the most valuable subset of demanded pieces and combines them, horizontally and vertically, until it obtains a large rectangle that fits into the original object. Christofides and Whitlock (1977) developed the first solution method for the 2D_R_CG_SLOPP, which consists of an exact depth-first search algorithm (top-down approach). The solution space is represented by a tree that enumerates all possible cutting patterns and the branches represent the guillotine cuts and the nodes represent the small items. To restrict the search, they used bounds generated by a transportation problem and a one-dimensional knapsack problem. Christofides and Hadjiconstantinou (1995) improved this method using bounds provided by a relaxed space state formulation based on dynamic programming and a procedure based on the sub-gradient algorithm. Morabito and Arenales (1996) considered this approach in the context of AND/OR graph.

In the context of bottom-up approaches, Wang (1983) proposed two combinatorial algorithms that successively combine small items through horizontal and vertical builds. Additionally, they proposed two indicators to reduce the amount of possible builds and to assess a feasible solution with respect to the optimal value. These algorithms were revisited and improved by Oliveira and Ferreira (1990), Viswanathan and Bagchi (1993) and Parada et al (1995). The 2D_R_CG_SLOPP is also addressed using dynamic programming based techniques, as presented in Hifi (2004), Morabito and Pureza (2010), Dolatabadi et al (2012), and Velasco and Uchoa (2018). Furthermore, there are meta-heuristic approaches based on Simulated Annealing and Tabu Search frameworks (Parada et al, 1998; Alvarez-Valdés et al, 2002).

Although the 2D_R_CG_SLOPP has been addressed by different approaches, it took more than 30 years for a mathematical formulation to appear in the literature that represents the (unconstrained or constrained) guillotine case – and there are only two formulations up to this moment. To the best of our knowledge, the first formulation was proposed by Ben Messaoud et al (2008), who proved a theorem for characterizing a guillotine pattern and, based on this result, proposed an ILP formulation for the Guillotine Strip Packing Problem (GSPP). The authors mentioned that this formulation is restricted to scenarios with just a few small items – they considered computational experiments with 5 items. More recently, Furini et al (2016) proposed a pseudo-polynomial ILP formulation for the 2D_R_CG_SLOPP, which can be seen as a generalization of the one-cut model proposed by Dyckhoff (1981), originally for the one-dimensional case. This model is non-compact and thus requires an enumerative variable procedure in a preprocessing phase, to enumerate all possible (relevant) cut decisions for any rectangle (large object or residual sheets). The authors also proposed a solution method based on this formulation related to a variable pricing technique. Their computational experiments used instances from the literature and showed that the method was capable of solving medium-size instances.

Regarding 2-staged patterns, we mention the mathematical formulations proposed by Lodi and Monaci (2003), Silva et al (2010) and Puchinger and Raidl (2007); and the column generation based approaches developed by Gilmore and Gomory (1965) and Belov and Scheithauer (2006) – see Silva et al (2010) for a detailed description. Silva et al (2010) and Puchinger and Raidl (2007) also proposed models for 3-staged patterns. In addition, there are a few approaches to directly address checkerboard patterns ($p$-groups) – any $p$-group pattern, with $p > 1$, contains $p$ sub-patterns of 1-group type. For 1-group patterns, see the INLP model by Morabito and Arenales (2000) and the ILP models by Yanasse and Morabito (2006); whereas for 2-group and 3-group patterns, see the ILP models of Yanasse and Morabito (2008). Additionally, Yanasse and Katsurayama (2005, 2008) proposed enumerative algorithms for these types of cutting patterns.

### 1.2 Our contributions

The main contributions of this paper are: (i) the proposition of an INLP formulation and one equivalent ILP formulation for the 2D_R_CG_SLOPP, which are extensions of the ILP formulation of Beasley (1985) to the constrained guillotine case; (ii) the development of an enumerative variable procedure (EVP) for the ILP formulation that, without loss of optimality, leads to fewer variables and constraints; (iii) the proposition of new compact models based on the proposed ILP formulation, for the special cases of the 2D_R_CG_SLOPP that consider 2-staged or 1-group patterns; (iv) the development of a branch-and-Benders-cut (BBC) algorithm, based on applying Benders decomposition to the proposed ILP formulation. In addition, we show how to adapt the formulation by Ben Messaoud et al (2008) to model the 2D_R_CG_SLOPP. In advance, we highlight that our approaches stand out in scenarios

characterized by few item types that are large with respect to the object's dimensions. We also note the relevance of developing models for optimization problems. A model contributes to the characterization and understanding of the problem. In addition, it may motivate the development of new solution methods based on decomposition techniques (such as Lagrangean relaxation or Dantzig-Wolfe/Benders decomposition), matheuristic methods and even hybrid methods that can enable the solution of larger instances of the problem.

### 1.3 Organization of the paper

The remainder of this paper is organized as follows. In Section 2 we propose a compact INLP formulation and a related compact ILP formulation for the 2D_R_CG_SLOPP, and develop the EVP that reduces the number of variables of the ILP formulation. In Section 3, we adapt the proposed ILP formulation to strictly generate 2-staged or 1-group patterns, which are special cases of guillotine patterns. Section 4 presents the proposed BBC algorithm to solve the 2D_R_CG_SLOPP. Section 5 discusses the computational experiments carried out to assess the proposed formulations and methods and highlights the most appropriate scenarios for each of them. We conclude with final remarks in Section 6.

## 2 Mathematical formulation for the 2D_R_CG_SLOPP

This section is divided into three parts. In Section 2.1, we propose a compact INLP formulation by extending the ILP model of Beasley (1985) by adding two sets of variables and constraints related to horizontal and vertical cuts, which are used to ensure the generation of guillotine patterns only. Then, in Section 2.2, we develop a compact ILP formulation that is equivalent to the proposed INLP formulation. Section 2.3 states the algorithm of the EVP, which relies on the discretization technique of the normal sets (Herz, 1972) to reduce the numbers of variables and constraints in the ILP model. In these sections, we use the following sets and parameters to represent the 2D_R_CG_SLOPP:

- $L$ and $W$ are, respectively, the length and width of the original large object;
- $K = \{1, \ldots, m\}$ is the set of small item types, where $m$ is the number of item types;
- $l_k$, $w_k$, and $v_k$ are, respectively, the length, width, and value of the item type $k \in K$;
- $r_k$ is the maximum number of copies that can be produced item type $k$.

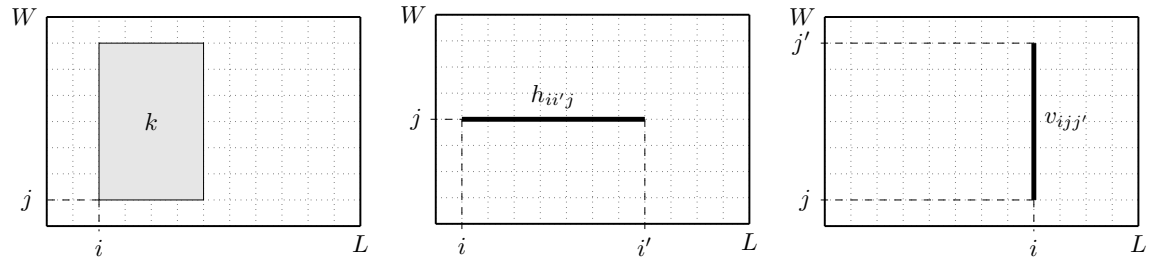
### 2.1 An INLP formulation based on Object's discretization

Beasley (1985) proposed a formulation that allocates small items (according to their left-lower corners) to points of a discretized object, using constraints to avoid the overlap between any pair of allocated items. According to this approach, we assume without loss of generality that the input data consists of positive integers. Let set $R = \{(i,j) \in \mathbb{Z}^2 \mid 0 \leq i \leq L, 0 \leq j \leq W\}$ be the discretization of the large object, assuming a complete discretization, and set $R_k = \{(i,j) \in R \mid 0 \leq i \leq L - l_k, 0 \leq j \leq W - w_k\}$ be the allowed points for allocating the left-lower corner of an item of type $k$ to be allocated. The decision variable related to the allocation of an item of type $k$ at position $(i,j)$ is defined in Equation (1). The formulation has also decision variables related to horizontal ($h_{ii'j}$) and vertical ($v_{ijj'}$) cuts, respectively defined in Equations (2) and (3). Fig. 2 illustrates these variables.

$$x_{kij} = \begin{cases} 1, & \text{if an item type } k \text{ is allocated at position } (i,j), \\ 0, & \text{otherwise,} \end{cases} \qquad k \in K, (i,j) \in R_k. \quad (1)$$

$$h_{ii'j} = \begin{cases} 1, & \text{if a horizontal cut is performed from } (i,j) \text{ to } (i',j), \\ 0, & \text{otherwise,} \end{cases} \qquad (i,j) \in R, (i',j) \in R, i < i'. \quad (2)$$

$$v_{ijj'} = \begin{cases} 1, & \text{if a vertical cut is performed from } (i,j) \text{ to } (i,j'), \\ 0, & \text{otherwise,} \end{cases} \qquad (i,j) \in R, (i,j') \in R, j < j'. \quad (3)$$



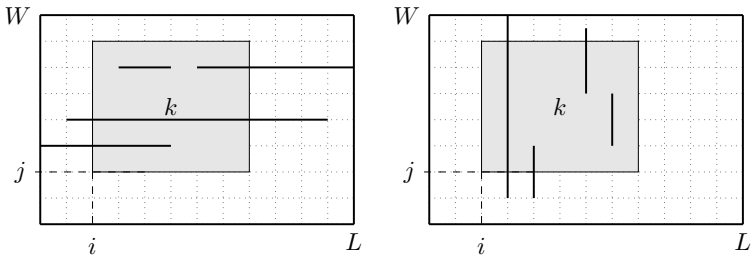

(a) Variable $x_{kij}$. (b) Variable $h_{ii'j}$. (c) Variable $v_{ijj'}$.

Fig. 2: Illustration of variables $x_{kij}, h_{ii'j}$ and $v_{ijj'}$.

Let $S^h_{kij}$ be the set of all horizontal segments $(i', i'', j)$ that cross the allocated item represented by variable $x_{kij}$, as defined in Equation (4) and illustrated in Fig 3a. Similarly, let $S^v_{kij}$ be the corresponding set for vertical segments, which is defined in Equation (5) and illustrated in Fig 3b.

$$S^h_{kij} = \{(i', i'', j') \in \mathbb{Z}^3 \mid j < j' < j + w_k, 0 \leq i' < i + l_k, \max\{i, i'\} < i'' \leq L\}, \quad k \in K, (i,j) \in R_k. \quad (4)$$

$$S^v_{kij} = \{(i', j', j'') \in \mathbb{Z}^3 \mid i < i' < i + l_k, 0 \leq j' < j + w_k, \max\{j, j'\} < j'' \leq W\}, \quad k \in K, (i,j) \in R_k. \quad (5)$$




(a) Set $S^h_{kij}$. (b) Set $S^v_{kij}$.

Fig. 3: Illustrations of a few segments of $S^h_{kij}$ and $S^v_{kij}$.

To avoid the overlapping between any pair of allocated items, we create the following parameter, for each $k \in K$, $(i,j) \in R_k$ and $(i', j') \in R$, with $i' < L$ and $j' < W$:

$$f_{kiji'j'} = \begin{cases} 1, & \text{if } 0 \leq i \leq i' \leq i + l_k - 1 < L \text{ and } 0 \leq j \leq j' \leq j + w_k - 1 < W, \\ 0, & \text{otherwise.} \end{cases}$$

Model (6) defines the compact INLP formulation for the 2D_R_CG_SLOPP. Regarding the decision variables in the model, we highlight that: (i) the objective function and the constraints that ensure the non-overlapping of allocated items and the constrained case are based on the allocation variables $x_{kij}$ only, similarly to Beasley (1985); (ii) the cut variables $h_{ii'j}$ and $v_{ijj'}$ are used to create a guillotine framework on the large object; (iii) the allocation and cut variables are used together to avoid cuts over allocated items, and to limit allocations to cut corners; together with the other constraints, they ensure the guillotine restriction. Variables $v_{00W}$, $v_{L0W}$, $h_{0L0}$ and $h_{0LW}$ are restricted to the unitary value, i.e., the object's borders are considered as cuts in this model.

$$\textbf{Max} \sum_{k \in K} \sum_{(i,j) \in R_k} v_k x_{kij}. \quad (6a)$$

**s.t.**

$$\sum_{k\in K}\sum_{(i,j)\in R_k} f_{kiji'j'}x_{kij} \leq 1, \qquad (i',j') \in R, i' < L, j' < W, \quad (6b)$$

$$\sum_{(i,j)\in R_k} x_{kij} \leq r_k, \qquad k \in K, \quad (6c)$$

$$h_{i_1i_2j} \leq \sum_{\substack{0\leq i'_1,i''_1\leq i_1;\\ i_2\leq i'_2,i''_2\leq L;\\ 0\leq j_1<j<j_2\leq W}} \sum_{\substack{0\leq j'_1,j''_1\leq j_1;\\ j_2\leq j'_2,j''_2\leq W}} h_{i'_1i'_2j_1}h_{i''_1i''_2j_2}v_{i_1j'_1j'_2}v_{i_2j''_1j''_2}, \qquad (i_1,j)\in R, (i_2,j)\in R,$$

$$i_1 < i_2, 0 < j < W, \quad (6d)$$

$$v_{ij_1j_2} \leq \sum_{\substack{0\leq j'_1,j''_1\leq j_1;\\ j_2\leq j'_2,j''_2\leq W;\\ 0\leq i_1<i<i_2\leq L}} \sum_{\substack{0\leq i'_1,i''_1\leq i_1;\\ i_2\leq i'_2,i''_2\leq L}} h_{i'_1i'_2j_1}h_{i''_1i''_2j_2}v_{i_1j'_1j'_2}v_{i_2j''_1j''_2}, \qquad (i,j_1)\in R, (i,j_2)\in R,$$

$$0 < i < L, j_1 < j_2, \quad (6e)$$

$$\sum_{(i,'i'',j')\in S^h_{kij}} h_{i'i''j'} + \sum_{(i',j',j'')\in S^v_{kij}} v_{i'j'j''} \leq (1-x_{kij})M_{kij}, \qquad k\in K, (i,j)\in R_k, \quad (6f)$$

$$x_{kij} \leq \sum_{\substack{0\leq i'\leq i;\\ i+l_k\leq i''\leq L}} h_{i'i''j}, \qquad k\in K, (i,j)\in R_k, \quad (6g)$$

$$x_{kij} \leq \sum_{\substack{0\leq j'\leq j;\\ j+w_k\leq j''\leq W}} v_{ij'j''}, \qquad k\in K, (i,j)\in R_k. \quad (6h)$$

$$x_{kij} \in \{0,1\}, \qquad k\in K, (i,j)\in R_k, \quad (6i)$$

$$h_{ii'j} \in \{0,1\}, \qquad (i,j)\in R, (i',j)\in R, i < i', \quad (6j)$$

$$v_{ijj'} \in \{0,1\}, \qquad (i,j)\in R, (i,j')\in R, j < j'. \quad (6k)$$

The objective function (6a) consists of maximizing the total value of the allocated items. Constraints (6b) avoid overlaps between any pair of allocated items. Constraints (6c) limit the cutting pattern to the constrained case. We note that the model {**Max** (6a), **s.t.** (6b), (6c), (6i)} is exactly ILP formulation proposed by Beasley (1985) for the constrained non-guillotine case. Therefore, the cut variables and related constraints have to be included to generate the guillotine restriction.

Model (6) is non-linear due to constraints (6d) and (6e), which present the sums of products of four binary variables on their right-hand-side (rhs). These constraints create a guillotine framework in the object. A guillotine cut requires the existence of the *enabling rectangle* in the object, over which an edge-to-edge cut is made. Thus, a horizontal or vertical cut may exist only if there is an enabling rectangle that supports this segment. Fig. 4 outlines these constraints using the rectangle $(i_1, i_2, j_1, j_2)$ that is enabling because its edges are cut, therefore allowing guillotine cuts within it. Therefore, if the rhs of a constraint (6d) is at least one, the corresponding variable $h_{ii'j}$ may assume value zero or one; but if the rhs of such a constraint is zero, then $h_{ii'j} = 0$. The same analysis holds for constraints (6e).

The disjunctive constraints (6f) forbid cuts over an allocated item (i.e. when $x_{kij} = 1$), where $M_{kij}$ is a sufficiently large parameter that can be set as the sum of the cardinality of sets $S^h_{kij}$ and $S^v_{kij}$. Fig. 5a illustrates these constraints. Constraints (6g) and (6h) limit the item allocations to corners (vertices) formed from horizontal and vertical cuts, as depicted in Fig. 5b. These constraints enforce the generation of the guillotine framework, and therefore the non-guillotine patterns are eliminated from the solution space. Constraints (6i) to (6k) impose the domain of the decision variables.

In preliminary computational experiments, we considered an efficient constraint programming software to solve Model (6) to optimality, namely the CPOptimizer of the IBM CPLEX Optimization Studio v.12.8. However, it turned out to be very ineffective for benchmark instances from the literature, as the numbers of variables and constraints in the model quickly grow as the object's dimensions increase. As future research, one could explore the use of other INLP software, such as ANTIGONE (Misener and Floudas, 2013, 2014), Baron (Tawarmalani and Sahinidis, 2005) or SCIP (Vigerske et al, 2012), as a general purpose solver for this model. The experience with CPOptimizer motivated us to develop a linear

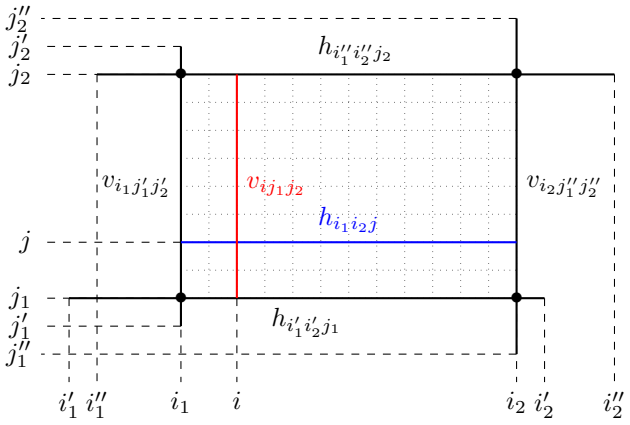
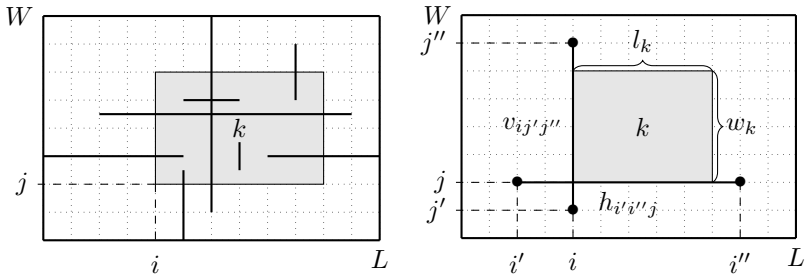



Fig. 4: Illustration of constraints (6d) and (6e).




(a) Constraints (6f) forbid cuts over $x_{kij} = 1$.

(b) Constraints (6g) and (6h).

Fig. 5: Illustrations of constraints (6f), (6g) and (6h).

version of Model (6). In a first attempt, we represented each possible rectangle of the rhs of constraints (6d) and (6e) as a binary variable that was introduced in the model and linked to generated cuts, but the number of variables was too large and undermined this approach. Additionally, we tried different types of relaxations over these constraints, but they were with loss of generality and we could not find an effective way to restore them. Finally, we obtained the ILP formulation described in the next section, which seems to us a more promising alternative.

## 2.2 An equivalent ILP formulation based on the object's discretization

To obtain an equivalent ILP formulation from Model (6), we ensure a new set of variables that represent the enabled and disabled rectangles and, hence, can be used to ensure the guillotine cuts. For each $(i_1, j_1), (i_2, j_2) \in R$, with $i_1 < i_2$ and $j_1 < j_2$, let the binary variable $p_{i_1i_2j_1j_2}$ represent a (sub)area of the original plate, as defined in Equation (7), which may assume unitary value if its borders are cut.

$$p_{i_1i_2j_1j_2} = \begin{cases} 1, & \text{if the rectangle of left-lower corner } (i_1, j_1) \text{ and right-upper corner } (i_2, j_2) \text{ is enabled,} \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

The non-linear constraints (6d) and (6e) of Model (6) can be replaced, without loss of generality, by the linear constraints (8).

$$p_{i_1i_2j_1j_2} \leq \sum_{i_1' \leq i_1, i_2 \leq i_2'} h_{i_1'i_2'j_1}, \qquad (i_1, j_1) \in R, (i_2, j_2) \in R, i_1 < i_2 \text{ and } j_1 < j_2, \tag{8a}$$

$$p_{i_1 i_2 j_1 j_2} \leq \sum_{i'_1 \leq i_1, i_2 \leq i'_2} h_{i'_1 i'_2 j_2}, \qquad (i_1, j_1) \in R, (i_2, j_2) \in R, i_1 < i_2 \text{ and } j_1 < j_2, \quad (8b)$$

$$p_{i_1 i_2 j_1 j_2} \leq \sum_{j'_1 \leq j_1, j_2 \leq j'_2} v_{i_1 j'_1 j'_2}, \qquad (i_1, j_1) \in R, (i_2, j_2) \in R, i_1 < i_2 \text{ and } j_1 < j_2, \quad (8c)$$

$$p_{i_1 i_2 j_1 j_2} \leq \sum_{j'_1 \leq j_1, j_2 \leq j'_2} v_{i_2 j'_1 j'_2}, \qquad (i_1, j_1) \in R, (i_2, j_2) \in R, i_1 < i_2 \text{ and } j_1 < j_2, \quad (8d)$$

$$h_{i_1 i_2 j} \leq \sum_{j_1 < j \leq \lfloor (j_1 + j_2)/2 \rfloor} p_{i_1 i_2 j_1 j_2}, \qquad (i_1, j) \in R, (i_2, j) \in R, i_1 < i_2, \quad (8e)$$

$$v_{i j_1 j_2} \leq \sum_{i_1 < i \leq \lfloor (i_1 + i_2)/2 \rfloor} p_{i_1 i_2 j_1 j_2}, \qquad (i, j_1) \in R, (i, j_2) \in R, j_1 < j_2, \quad (8f)$$

$$p_{i_1 i_2 j_1 j_2} \in \{0, 1\}, \qquad (i_1, j_1) \in R, (i_2, j_2) \in R, i_1 < i_2, j_1 < j_2. \quad (8g)$$

Constraints (8a) to (8d) impose that variable $p_{i_1 i_2 j_1 j_2}$ assumes the value of 1 if the rhs of all the four corresponding constraints are at least 1, i.e., the borders of the rectangle $(i_1, i_2, j_1, j_2)$ are cut; otherwise, $p_{i_1 i_2 j_1 j_2}$ assumes the value of 0. Constraints (8e) and (8f) allow guillotine cuts only over enabled rectangles. They ensure that the variable related to a horizontal or vertical cut assumes the value of 1 only if the corresponding constraint has an rhs greater than or equal to 1. Note that these constraints also impose that the horizontal and vertical cuts are up to the half of an enabled rectangle, where $\lfloor x \rfloor$ is the largest integer smaller than or equal to $x$. This assumption is without loss of optimality as the large object is homogeneous and contributes to avoid some symmetrical cutting patterns in the solution space (Christofides and Whitlock, 1977) – the same assumption can be made in the guillotine cuts of Model (6). Constraints (8g) impose the domain of the decision variables. Using all these constraints, we obtain Model (9), which is a compact ILP formulation for the 2D_R_CG_SLOPP.

$$\textbf{Max} \quad (6a), \qquad (9a)$$

$$\textbf{s.t.} \quad (6b) - (6c), (6f) - (6k), (8a) - (8g) \qquad (9b)$$

Similarly to the non-linear Model (6), we assume that all borders of the large object are cut in Model (9). Thus, the corresponding variables are fixed at unitary value. Also, variable $p_{0L0W}$ may be fixed at 1, because the original rectangle (object) is enabled. We note that valid inequalities could be developed to link variables $x_{kij}$ to $p_{i_1 i_2 j_1 j_2}$ and, hence, enhance the LP relaxation of the model. However, since the number of constraints in the model is already considerably large, we decided to consider this implicitly using the algorithm proposed in the next section.

### 2.3 Enumerative variable procedure

The literature related to C&P problems typically recurs to the discretization of the normal sets instead of the complete discretization – such as in Sections 2.1 and 2.2 – for decisions related to allocations and guillotine cuts (Herz, 1972; Scheithauer, 2018). To avoid symmetrical patterns, this technique considers only the integer positions representing combinations of the small items and defines the cutting patterns using left-bottom justified items only. For the 2D_R_CG_SLOPP, the domain of variable $x_{kij}$ can be modified, without loss of generality, according to the redefinition of set $R$ as in Equation (10).

$$R = \{(i, j) \in \mathbb{Z}^2 \mid i \in \bar{X}, j \in \bar{Y}\}, \qquad (10)$$

where:

$$\bar{X} = \left\{ q_x \mid q_x = \sum_{k \in K} n_k l_k,\ 0 \leq q_x \leq L - \min_{k \in K}\{l_k\}, n_k \in \mathbb{N}, n_k \leq r_k \right\} \cup \{L\} \text{ and}$$

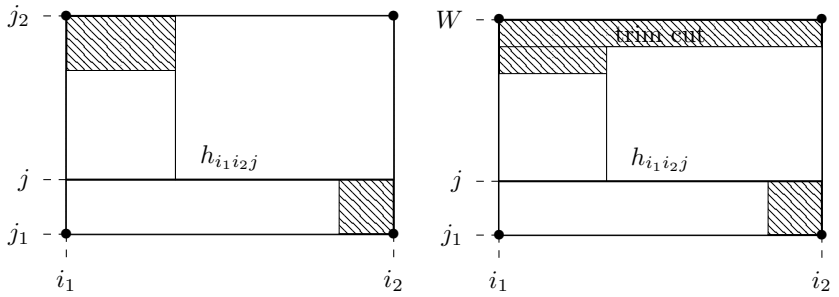
$$\bar{Y} = \left\{ q_y \mid q_y = \sum_{k \in K} n_k w_k,\ 0 \leq q_y \leq W - \min_{k \in K}\{w_k\}, n_k \in \mathbb{N}, n_k \leq r_k \right\} \cup \{W\}$$

represent the discretization of the normal sets for the vertical and horizontal cuts, respectively.

We now propose an enumerative variable procedure (EVP) that focuses on reducing not only the number of variables $x_{kij}$, but also the number of variables $h_{ii'j}$, $v_{ijj'}$, and $p_{i_1i_2j_1j_2}$, in Model (9). To achieve that, without loss of generality, the procedure relies on the discretization of normal sets and some properties of the problem. As outputs, the EVP determines the sets $\mathcal{X}$, $\mathcal{H}$, $\mathcal{V}$ and $\mathcal{P}$ that correspond to the reduced sets of indices of variables $x_{kij}$, $h_{ii'j}$, $v_{ijj'}$, and $p_{i_1i_2j_1j_2}$, respectively. The procedure is defined according to the following assumptions:

1. the horizontal cut $(i_1, i_2, j)$ must belong to $\mathcal{H}$, according to the rectangle $(i_1, i_2, j_1, j_2)$, if each of the following conditions hold:
   - the cut is up to the half of the rectangle ($j_1 < j \leq \lfloor (j_1 + j_2)/2 \rfloor$);
   - the lower sub-rectangle $(i_1, i_2, j_1, j)$ fits at least one small item;
   - the upper sub-rectangle $(i_1, i_2, j, j_2)$ fits at least one small item;
   - the value $j - j_1$ belongs to $\bar{Y}$, which assumes that each horizontal cut will always generate a lower sub-rectangle (or sub-pattern) with a width equal to a linear combination of item types' width; and
   - the value $j_2 - j$ belongs to $\bar{Y}$ or $j_2 = W$, which assumes that each horizontal cut will always generate: (i) an upper sub-rectangle (or sub-pattern) with a width equal to a linear combination of item types' width, or (ii) allows the existence of patterns which lead to trim cuts;
2. the vertical cut $(i, j_1, j_2)$ must belong to $\mathcal{V}$, according to the rectangle $(i_1, i_2, j_1, j_2)$, if each of the following conditions hold:
   - the cut is up to the half of the rectangle ($i_1 < i \leq \lfloor (i_1 + i_2)/2 \rfloor$);
   - the left sub-rectangle $(i_1, i, j_1, j_2)$ fits at least one small item;
   - the right sub-rectangle $(i, i_2, j_1, j_2)$ fits at least one small item;
   - the value $i - i_1$ belongs to $\bar{X}$, which assumes that each vertical cut will always generate a left sub-rectangle (or sub-pattern) with a length equal to a linear combination of item types' length; and
   - the value $i_2 - i$ belongs to $\bar{X}$ or $i_2 = L$, which assumes that each vertical cut will always generate: (i) a right sub-rectangle (or sub-pattern) with a length equal to a linear combination of item types' length, or (ii) allows the existence of patterns which lead to trim cuts;
3. the domain $(i_1, i_2, j_1, j_2)$ must belong to $\mathcal{P}$ if the area $(i_2 - i_1) \times (j_2 - j_1)$ is able to fit at least two small items.

The proposed EVP is described in Algorithm 1. Fig. 6 shows two illustrative examples of possible cutting patterns considered by the EVP that highlight: (*i*) if $i_2 < L$ and $j_2 < W$, the formulation only needs to consider a rectangle $p_{i_1i_2j_1j_2}$ with length $(i_2 - i_1)$ and width $(j_2 - j_1)$ belonging to $\bar{X}$ and $\bar{Y}$, respectively; and (*ii*) if $i_2 = L$ and $j_2 = W$, we allow trim cuts in the upper or right rectangles.




(a) $j - j_1 \in \bar{Y}$ and $j_2 - j \in \bar{Y}$. (b) $j - j_1 \in \bar{Y}$ and $W - j \notin \bar{Y}$.

Fig. 6: Illustrations of possible cutting patterns considered by the EVP.

We emphasize that Model (9) can be stated without loss of generality using domains $\mathcal{X}$, $\mathcal{H}$, $\mathcal{V}$ and $\mathcal{P}$ to define all its variables and constraints. The improvements discussed in this section are used in the computational experiments presented in Section 5.

**Algorithm 1:** Enumerative variable procedure for a large object.

**Data:** $L$, $W$, $m$, $K$, $(l_k, w_k, v_k, r_k)$ for all $k \in K$, and discretization sets $\bar{X}$ and $\bar{Y}$.
$\mathcal{X} = \emptyset$, $\mathcal{H} = \{(0, L, 0), (0, L, W)\}$ , $\mathcal{V} = \{(0, 0, W), (L, 0, W)\}$, $\mathcal{P} = \{(0, L, 0, W)\}$, $auxP = \{(0, L, 0, W)\}$
**for** $k \in K$ **do**
  **for** $i \in \bar{X}$ **do**
    **for** $j \in \bar{Y}$ **do**
      **if** $\{(i + l_k \leq L)$ *and* $(j + w_k \leq W)\}$ **then**
        $\mathcal{X} = \mathcal{X} \cup (k, i, j)$
**while** *auxP is not empty* **do**
  select a rectangle $(i_1, i_2, j_1, j_2) \in$ auxP and remove it from auxP
  `// horizontal cuts`
  **for** $j \in \bar{Y} \mid j_1 < j \leq \lfloor (j_1 + j_2)/2 \rfloor$ **do**
    **if** *each of the rectangles* $(i_1, i_2, j_1, j)$ *and* $(i_1, i_2, j, j_2)$ *fits at least one small item, and* $j - j_1 \in \bar{Y}$ *and* $(j_2 - j \in \bar{Y}$ *or* $j_2 = W)$ **then**
      $\mathcal{H} = \mathcal{H} \cup (i_1, i_2, j)$
      **if** *rectangle* $(i_1, i_2, j_1, j) \notin \mathcal{P}$ *and fits at least two small items* **then**
        $\mathcal{P} = \mathcal{P} \cup (i_1, i_2, j_1, j)$, auxP = auxP $\cup (i_1, i_2, j_1, j)$
      **if** *rectangle* $(i_1, i_2, j, j_2) \notin \mathcal{P}$ *and fits at least two small items* **then**
        $\mathcal{P} = \mathcal{P} \cup (i_1, i_2, j, j_2)$, auxP = auxP $\cup (i_1, i_2, j, j_2)$
  `// vertical cuts`
  **for** $i \in \bar{X} \mid i_1 < i \leq \lfloor (i_1 + i_2)/2 \rfloor$ **do**
    **if** *each of the rectangles* $(i_1, i, j_1, j_2)$ *and* $(i, i_2, j_1, j_2)$ *fits at least one small item, and* $i - i_1 \in \bar{X}$ *and* $(i_2 - i \in \bar{X}$ *or* $i_2 = L)$ **then**
      $\mathcal{V} = \mathcal{V} \cup (i, j_1, j_2)$
      **if** *rectangle* $(i_1, i, j_1, j_2) \notin \mathcal{P}$ *and fits at least two small items* **then**
        $\mathcal{P} = \mathcal{P} \cup (i_1, i, j_1, j_2)$, auxP = auxP $\cup (i_1, i, j_1, j_2)$
      **if** *rectangle* $(i, i_2, j_1, j_2) \notin \mathcal{P}$ *and fits at least two small items* **then**
        $\mathcal{P} = \mathcal{P} \cup (i, i_2, j_1, j_2)$, auxP = auxP $\cup (i, i_2, j_1, j_2)$
**Result:** Domains $\mathcal{X}$, $\mathcal{H}$, $\mathcal{V}$ and $\mathcal{P}$.

## 3 Mathematical formulations for generating 2-staged and 1-group patterns

In this section, we propose ILP formulations for two particular variants of the 2D_R_CG_SLOPP, which strictly consider special cases of guillotine patterns, known as 2-staged and 1-group patterns. These formulations are obtained from Model (9) by removing the set of variables $p_{i_1 i_2 j_1 j_2}$ and modifying the domain of the cut variables, which contributes to simplifying a few sets of constraints.

### 3.1 Formulation for 2-staged patterns

The 2-staged guillotine cutting patterns are formed by horizontal or vertical shelves in the object, from which the items are cut. Considering the case of horizontal shelves, the horizontal cut variable $h_{ijj'}$ can be redefined as in Equation (11), because in this context the horizontal cuts are always edge-to-edge in the original object.

$$h_j = \begin{cases} 1, & \text{if a horizontal cut is performed from } (0, j) \text{ to } (L, j), \\ 0, & \text{otherwise,} \end{cases} \qquad 0 \leq j < W. \quad (11)$$

An allocated small item cannot cross over a horizontal cut to form a horizontal shelf. Hence, as the allocated small items inside a shelf do not overlap each other, the vertical cuts are no longer needed. Model (12) is a compact ILP formulation for the 2D_R_CG_SLOPP with non-exact 2-staged patterns, obtained from Model (9). We eliminate the sets of variables $v_{ijj'}$ and $p_{i_1 i_2 j_1 j_2}$, whereas variables $x_{kij}$ are kept as defined in Section 2.

**Max** (6*a*),

**s.t.**

(6b), (6c), (6i),

$$\sum_{j<j'<j+w_k} h_{j'} \leq (1 - x_{kij})M_{kij}, \qquad k \in K, (i,j) \in R_k, \quad (12a)$$

$$x_{kij} \leq h_j, \qquad k \in K, (i,j) \in R_k, \quad (12b)$$

$$h_j \in \{0,1\}, \qquad 0 \leq j < W. \quad (12c)$$

Constraints (12a) ensure that no horizontal cut crosses over an allocated item, where the parameter $M_{kij}$ can be set as $w_k - 1$. Constraints (12b) limit the allocations over a horizontal cut. Constraints (12c) impose the domain of variable $h_j$. The other constraints are as previously described. Similarly to the models of Section 2, the lower border of the object is considered cut (i.e. $h_0 = 1$).

For a 2-staged pattern formed by vertical shelves, we can carry out a similar analysis, but converting variables $v_{ii'j}$ into $v_i$, whereas variables $h_{ijj'}$ remain as in the original formulation. Another possibility is to interchange the length and the width of each item type, before using Model (13) to solve a given instance.

Model (12) can be reformulated using a reduced number of variables and constraints, which can contribute to its average performance on general purpose solvers. After this reduction, the resulting formulation is given by Model (13).

**Max** (6*a*),

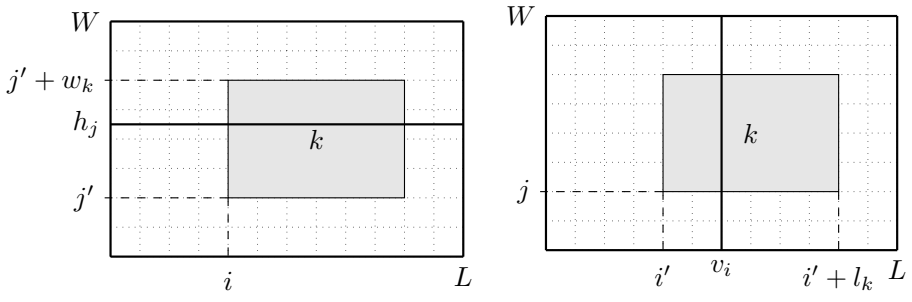
**s.t.**

(6b), (6c), (6i), (12c),

$$\sum_{k \in K} \sum_{0 \leq i \leq L - l_k} \sum_{j'<j<j'+w_k} x_{kij'} \leq (1 - h_j)M_j^1, \qquad 0 < j < W, \quad (13a)$$

$$\sum_{k \in K} \sum_{0 \leq i \leq L - l_k} x_{kij} \leq M_j^2 h_j, \qquad 0 < j < W, \quad (13b)$$

Constraints (13a) replace constraints (12a) of Model (12) and ensure that no small item crosses a horizontal cut. The parameter $M_j^1$ can be set as $\sum_{k \in K}(L - l_k + 1) * (w_k - 1)$, which is an upper bound for the left-hand-side (lhs) of the constraint. Fig. 7a illustrates the cases forbidden by these constraints. Constraints (13b) replace constraints (12b) of Model (12) and limit the allocations that occur over a horizontal cut, where the parameter $M_j^2$ can be set as $\sum_{k \in K}(L - l_k + 1)$.




(a) A horizontal cut over $x_{kij}$. (b) A vertical cut over $x_{kij}$.

Fig. 7: Illustrations of guillotine cuts over an allocated item for 2-staged patterns.

### 3.2 Formulation for 1-group patterns

The 1-group cutting pattern is formed only by horizontal and vertical shelves in the object. Therefore, in addition to horizontal shelves $h_j$, the variable $v_{ii'j}$ can be redefined as in Equation (14), because all vertical cuts also become edge-to-edge in the original object.

$$v_i = \begin{cases} 1, & \text{if a vertical cut is performed from } (i,0) \text{ to } (i,W), \\ 0, & \text{otherwise,} \end{cases} \qquad 0 \leq i < L. \quad (14)$$

Model (15) is a compact ILP formulation for the 2D_R_CG_SLOPP using only non-exact 1-group patterns, obtained from the model defined in Section 2.2. The variables $x_{kij}$ remain as defined in Section 2, while variables $p_{i_1 i_2 j_1 j_2}$ are no longer needed because all 1-group cuts lead to a guillotine pattern.

**Max** (6a),
**s.t.**
(6b), (6c), (6i), (12b), (12c),

$$\sum_{j<j'<j+w_k} h_{j'} + \sum_{i<i'<i+l_k} v_{i'} \leq (1 - x_{kij}) M_{kij}, \qquad k \in K, (i,j) \in R_k, \quad (15a)$$

$$x_{kij} \leq v_i, \qquad k \in K, (i,j) \in R_k. \quad (15b)$$

$$v_i \in \{0,1\}, \qquad 0 \leq i < L. \quad (15c)$$

Constraints (15a) avoid that any horizontal or vertical cut crosses over an allocated item, where $M_{kij}$ is a parameter that can be set as $w_k + l_k - 2$. Constraints (15b) forbid the allocations over a vertical cut. Constraints (15c) impose the domain of variables $v_i$. The other constraints are as previously described. Similarly to the models of Section 2, the object's left border is considered cut ($v_0 = 1$).

As in the previous subsection, we can reformulate Model (15) using fewer variables and constraints, resulting in Model (16), which has a better average performance on general purpose solvers.

**Max** (6a),
**s.t.**
(6b), (6c), (6i), (12c), (13a), (13b), (15c),

$$\sum_{k \in K} \sum_{0 \leq j \leq W - w_k} \sum_{i'<i<i'+l_k} x_{ki'j} \leq (1 - v_i) N_i^1, \qquad 0 < i < L, \quad (16a)$$

$$\sum_{k \in K} \sum_{0 \leq j \leq W - w_k} x_{kij} \leq N_i^2 v_i, \qquad 0 < i < L, \quad (16b)$$

Constraints (16a) replace the constraints (15a) of Model (15) and ensure that no allocated small item crosses over a vertical cut, where $N_i^1$ is a parameter that can be set as $\sum_{k \in K} (W - w_k + 1) * (l_k - 1)$, which is a sufficiently large bound for the lhs of the constraint. Fig. 7b illustrates how these constraints act in the model. Constraints (16b) replace (15b) and impose that allocations have to occur over a vertical cut, where the parameter $N_i^2$ can be set as $\sum_{k \in K} (W - w_k + 1)$.

As the set of variables $p_{i_1 i_2 j_1 j_2}$ are not present in any model of this section, lines 7 to 24 of Algorithm 1 are no longer needed. Therefore, the cut variables $h_j$ and $v_i$ should consider only the positions in the sets $\bar{X}$ and $\bar{Y}$, respectively. It is worth mentioning that we can easily adapt these models to address the exact case. To do that, we just need to further impose that an item type $k$ can be allocated at position $(i,j)$, only if the cuts $h_{j'}$ and $v_{i'}$ are performed, for $j' = j + w_k$ and $i' = i + l_k$.

## 4 A branch-and-Benders-cut algorithm

As a solution method for the 2D_R_CG_SLOPP, we propose a branch-and-Benders-cut (BBC) algorithm based on Model (9). It consists of using the Benders decomposition (Benders, 1962) to reformulate Model (9) and then using a branch-and-cut algorithm to solve the resulting Benders-Master-Problem (BMP). The generation of Benders cuts is embedded in the branch-and-cut algorithm and hence can be carried out at each node of the tree. For detailed descriptions and comprehensive surveys on the Benders decomposition and the BBC algorithm, see e.g. Costa (2005), Rahmaniani et al (2017) and Moreno et al (2018).

In our method, we use the Benders decomposition to reformulate Model (9), as follows:

1. The BMP is defined by the sets of non-overlapping and constrained case constraints, as presented in Model (17). Thus, the BMP searches for promising cutting patterns (trial solutions), represented by the solution vector $\bar{x}_{kij}$. Notice that the BMP is exactly the ILP formulation proposed by Beasley (1985) for the constrained non-guillotine case – see Section 2.1.

   **Max** (6a), **s.t.** (6b), (6c), (6i). (17)

2. Each trial solution obtained from the BMP is checked considering the guillotine restriction using the Benders-Sub-Problem (BSP), which is defined by Model (18). It consists of the variables $h_{ii'j}$, $v_{ijj'}$ and $p_{i_1 i_2 j_1 j_2}$ and constraints of Model (9) that are not included in the BMP.

   **Max** 0, (18a)

   **s.t.** (6j), (6k), (8), (18b)

   $$\sum_{(i'i''j')\in S^h_{kij}} h_{i'i''j'} + \sum_{(i'j'j'')\in S^v_{kij}} v_{i'j'j''} \leq 0, \qquad (k,i,j)\in\bar{\mathcal{X}}, \quad (18c)$$

   $$1 \leq \sum_{i'\leq i; i+l_k\leq i''} h_{i'i''j}, \qquad (k,i,j)\in\bar{\mathcal{X}}, \quad (18d)$$

   $$1 \leq \sum_{j'\leq j; j+w_k\leq j''} v_{ij'j''}, \qquad (k,i,j)\in\bar{\mathcal{X}}. \quad (18e)$$

Constraints (18c)–(18e) are adapted from Model (9) to take into account only the unitary components of the trial solution $\bar{x}_{kij}$. Hence, we define them using $\bar{\mathcal{X}} = \{(k,i,j) \mid \bar{x}_{kij} = 1\}$, which is the set of triples corresponding to the allocated small items in the cutting pattern of this solution. As in the original formulation, these constraints avoid cuts over the allocated small items and ensure that these items are allocated at cut corners.

The BBC algorithm works as follows. In a given node of the branch-and-cut tree, every trial solution $\bar{x}_{kij}$ obtained from the BMP is checked in the BSP. If the BSP is feasible for this solution, then the corresponding cutting pattern is of guillotine-type; thus, the solution is accepted and stored as an incumbent if it improves the lower bound of the tree. Otherwise, the cutting pattern is of non-guillotine type and the solution is used to generate a Benders combinatorial cut, given by inequality (19). This inequality is added to the BMP of the current node, which is then reoptimized.

$$\sum_{(k,i,j)\in\bar{\mathcal{X}}} x_{kij} \leq |\bar{\mathcal{X}}| - 1. \quad (19)$$

As usual in the implementations of the BBC algorithm, we add the Benders cuts as lazy constraints using callback procedures of the general purpose solver. Hence, the BMP is solved at once, inside a branch-and-cut single-tree. We point out that the set of indices of variables $h_{ii'j}$, $v_{ijj'}$, and $p_{i_1 i_2 j_1 j_2}$ can be significantly reduced in Model (18) as the trial solution $\bar{x}_{kij}$ is fixed in the BSP. In such case, the domain of these variables can be restricted, in addition to zero and sheet dimension positions, to: (i) horizontal positions $i$ (resp. vertical positions $j$) from the triples in $\bar{\mathcal{X}}$ (i.e., the allocated items); and (ii) horizontal segments $i' - i$ or $i_2 - i_1$ (resp. vertical segments $j' - j$ or $j_2 - j_1$) that are greater than or equal to the length (resp. width) of the smallest allocated item in $\bar{\mathcal{X}}$. We use this reduction in the computational experiments described in the next section. Therefore, lines 7 to 24 of Algorithm (1) (EVP) are not required in the definition of the BSP used in our BBC algorithm. For future research, this decomposition could be further analyzed in the context of the 2-staged and 1-group models of Section 3.

## 5 Computational results

We carried out computational experiments using benchmark instances from the literature, to assess the performance of the ILP formulations and the BBC algorithm proposed in this paper. The purpose of this section is twofold. First, for the 2D_R_CG_SLOPP, we compared the solution quality and processing times of the proposed ILP formulation – Model (9) – with those obtained by two other formulations available in the literature (Ben Messaoud et al, 2008; Furini et al, 2016), when they are solved by a general purpose solver. Additionally, we verified the performance of the proposed BBC algorithm with respect to these formulations, using the same set of instances. From the results, we identified the most appropriate scenarios for each of these approaches. Second, we analyzed the performances of the ILP formulations proposed for the special cases of the 2D_R_CG_SLOPP that consider only 2-staged or 1-group patterns – Models (13) and (16). Their performances are compared with the results obtained for the models proposed by Lodi et al (2002) and Yanasse and Morabito (2006), for the same variants.

The implementation of Model (9) uses the improvements of Algorithm 1 and is referred to as `Grid`. Models (13) and (16) are implemented using the discretization improvements discussed in Section 3.2 and are referred to as `Grid-2staged` and `Grid-1group`, respectively. All the approaches were coded in C++ using the Concert library which is part of IBM CPLEX Optimization Studio v.12.8. The experiments were carried out on a PC with Intel Core i7-3770 (3.40GHz) processor, 16 GB of RAM, under Ubuntu 18.04 Operating System.

Table 1 shows the characteristics of the benchmark instances taken from the literature. The sets cgcut and gcut were obtained from the online repository OR-Library[1], while the other instances were obtained from Wang (1983) and Oliveira and Ferreira (1990). We report for each instance the name of the instance (column Instance), the length of the large object (column $L$), the width of the large object (column $W$), the number of small item types (column $m$) and the maximum number of small items (column $n$), where $n = \sum_{k \in K} r_k$. We also provide the optimal value of each instance with respect to the non-staged, 2-staged and 1-group versions of the 2D_R_CG_SLOPP. The optimal values in the non-staged and 2-staged columns are well-known results and were taken from Lodi et al (2002) and Furini et al (2016); whereas the values in column 1-group were obtained solving the ILP formulation of Yanasse and Morabito (2006) through CPLEX, without imposing a time limit.

Table 1: Set of benchmark instances for 2D_R_CG_SLOPP.

| Instance | $L$ | $W$ | $m$ | $n$ | OPT | | |
|---|---|---|---|---|---|---|---|
| | | | | | non-staged | 2-staged | 1-group |
| cgcut1 | 15 | 10 | 7 | 16 | 244 | 240 | 240 |
| cgcut2 | 40 | 70 | 10 | 23 | 2,892 | 2,535 | 2,069 |
| cgcut3 | 40 | 70 | 19 | 62 | 1,860 | 1,720 | 1,580 |
| OF1 | 70 | 40 | 10 | 23 | 2,737 | 2,713 | 2,361 |
| OF2 | 70 | 40 | 10 | 24 | 2,690 | 2,515 | 2,342 |
| wang20 | 70 | 40 | 19 | 42 | 2,721 | 2,623 | 2,470 |
| gcut1 | 250 | 250 | 10 | 10 | 48,368 | 43,024 | 43,024 |
| gcut2 | 250 | 250 | 20 | 20 | 59,307 | 57,996 | 55,680 |
| gcut3 | 250 | 250 | 30 | 30 | 60,241 | 59,895 | 59,895 |
| gcut4 | 250 | 250 | 50 | 50 | 60,942 | 60,504 | 60,504 |
| gcut5 | 500 | 500 | 10 | 10 | 195,582 | 193,379 | 192,907 |
| gcut6 | 500 | 500 | 20 | 20 | 236,305 | 224,399 | 224,399 |
| gcut7 | 500 | 500 | 30 | 30 | 238,974 | 238,974 | 231,494 |
| gcut8 | 500 | 500 | 50 | 50 | 245,758 | 245,758 | 242,656 |
| gcut9 | 1,000 | 1,000 | 10 | 10 | 919,476 | 919,476 | 806,912 |
| gcut10 | 1,000 | 1,000 | 20 | 20 | 903,435 | 856,445 | 856,445 |
| gcut11 | 1,000 | 1,000 | 30 | 30 | 955,389 | 942,219 | 915,219 |
| gcut12 | 1,000 | 1,000 | 50 | 50 | 970,744 | 970,744 | 970,744 |

[1] http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/files/

### 5.1 Results for the 2D_R_CG_SLOPP

We compare the performance of `Grid` with the performances of the ILP formulations of Ben Messaoud et al (2008) and Furini et al (2016). As the formulation of Ben Messaoud et al (2008) was originally proposed for the GSPP, we adapted this formulation as follows:

- the indices $i$ and $j$ vary in $\{1, \ldots, \bar{n}\}$ (instead of $n$), where $\bar{n}$ is an upper bound on the number of small items in an optimal solution. We consider $\bar{n} = \max \{\sum_{k \in K} y_k : \sum_{k \in K} (l_k w_k) y_k \leq LW,\ y_k \leq r_k, y_k \in \mathbb{Z}\}$;
- the objective function consists of maximizing the sum of $v_k z_{ijk}$ over all small items $k$;
- the types of their constraints (4) to (6) are redefined to $\leq$ (instead of $=$);
- parameter $\bar{H}$ is replaced by the corresponding sheet dimension and variable $H$ is no longer needed;
- the parameter $n$ in the lhs of their constraints (11) is replaced by $\bar{n}$.

We refer to the resulting model as `BMCE`. We note that, in their notation, the index $k$ varies from 1 to $n$ (instead of $m$ as we consider in our models) as each small item is considered separately in their model.

For the model of Furini et al (2016), we used their enumerative variable procedure and the two reductions proposed by them. Apparently, there is a typo in their definition of parameter $a^o_{qkj}$, as the indices $j$ and $k$ seem to be exchanged. In the following analysis, we refer to this model as `FMT`.

Table 2 shows the results of the three formulations. For each formulation and instance, we report on the number of variables (column var), number of constraints (column cons), the optimality gap (column gap) as a percentage, the value of the LP-relaxation (column solr) and the processing time to solve the instance (t[s]) in seconds. The gap is calculated as $(OPT - sol)/(OPT + 10^{-10}) * 100$, where OPT is the corresponding optimal value taken from Table 1 and *sol* is the value of the best integer solution found by the solver using the formulation. The time limit for integer solutions on CPLEX was set to 1 hour and we denote by "tl" when it was reached. The symbol "*" means that CPLEX ran out of memory during its execution, and a gap of "100.0%" means that no integer solution was found. In all formulations, we provided an optimal 2-staged solution of the instance as an initial solution, which was obtained solving the model of Lodi et al (2002) through CPLEX.

First, we analyze the results of `BMCE` for the 2D_R_CG_SLOPP, as the original model was tested only for the GSPP in Ben Messaoud et al (2008). Using parameter $\bar{n}$ instead of $n$ proved useful as this model has, according to the original authors, around $3n^4/4$ binary variables and $2n^4$ constraints. For example, for the gcut instances, the numbers of variables and constraints in this model are smaller than in the other two models, which present $\bar{n} \leq 9$ (the upper bound from the one-dimensional knapsack. However, its LP-relaxation provides a very weak bound, which contributed to a poor performance of CPLEX. The solver could prove optimality for only 2 of the 18 instances using `BMCE`, reaching the time limit in 15 of them.

Furini et al (2016) compared the performance of their model with respect to the model of Ben Messaoud et al (2008), in the context of the GSPP. The results in Table 2 are in accordance with their findings, as `FMT` also outperforms `BMCE` for the 2D_R_CG_SLOPP. Using `FMT`, CPLEX was able to prove optimality in 12 of the instances and ran out of memory only for those in which the number of variables exceeded two million. The LP-relaxation of this model is the tightest among the three formulations and the performance in the medium-size instances (first six rows of the tables) is outstanding, especially regarding the number of constraints and the solution time. However, it is worth mentioning that `FMT` is a pseudo-polynomial model (non-compact formulation) that requires a preprocessing phase, although it tends not to be time consuming.

The results of `Grid` in Table 2 show that our model has a better average performance than `FMT` in the gcut instances, as the former was able to prove optimality in two additional instances and with significantly less average processing time. In particular, for the instances with $m = 10$ item types (gcut1, gcut5 and gcut9), the performance of `Grid` is significantly better than the performance of `FMT`. For instance gcut9, CPLEX took 1.30 seconds to prove optimality using `Grid`, while it took 1,243.86 seconds for `FMT`. For some instances with $m = 20$ item types, e.g. gcut2 and gcut7, it took less than 90 seconds to prove optimality using `Grid`, while CPLEX ran out of memory using `FMT`. Note that for the gcut instances, as the value of $m$ increases, the processing times also increase. These instances are characterized by small items that are large with respect to the object, resulting in fewer elements in sets $\bar{X}$ and $\bar{Y}$ and hence fewer variables and constraints in the model. The experiments also highlight the relevance of the EVP. For example, instance gcut2 would have 742,201 variables and 2,770,181 constraints in the absence of

lines 7 to 24 of Algorithm 1. Hence, for this instance, the EVP leads to reductions of 98.70% and 99.06% in the numbers of variables and constraints, respectively.

Regarding the LP-relaxation, `Grid` provided bounds significantly better than those provided by `BMCE`, but inferior or similar to those of `FMT`. On the other hand, `Grid` does not seem to be suitable in scenarios with many small item types, as it presented memory issues in all instances with $m = 50$ (gcut4, gcut8 and gcut12) and for instance gcut11. Particularly, `FMT` outperforms `Grid` in the first six instances in Table 2 (sets cgcut, OF and wang), which are characterized by items that are smaller than those in the gcut instances. Despite the relevance of avoiding symmetrical solutions in ILP formulations, we are not aware of any other type of symmetry breaking procedures for the model of Beasley (1985) and/or `Grid`, apart from the one we adopted in this paper. Therefore, even though we restrict the guillotine cuts up to the half of the plates, a plate (rectangle) with length $i_2 - i_1$ and width $j_2 - j_1$ may be represented several times by variables $p_{i_1 i_2 j_1 j_2}$ according to its position.

Table 2: Results of the three ILP formulations of the 2D_R_CG_SLOPP.

| Instance | BMCE (Ben Messaoud et al, 2008) | | | | | FMT (Furini et al, 2016) | | | | | Grid | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | cons | gap | solr | t[s] | var | cons | gap | solr | t[s] | var | cons | gap | solr | t[s] |
| cgcut1 | 23,010 | 52,820 | 0.0 | 346.0 | tl | 752 | 147 | 0.0 | 244.5 | 0.02 | 1,417 | 3,979 | 0.0 | 244.5 | 0.15 |
| cgcut2 | 83,223 | 198,417 | 100.0 | * | * | 49,886 | 2,027 | 0.0 | 2,897.4 | 10.49 | 24,860 | 68,500 | 9.9 | 2,906.1 | tl |
| cgcut3 | 17,809 | 26,746 | 6.5 | 5,240.0 | tl | 32,511 | 1,879 | 0.0 | 1,919.2 | 8.35 | 5,793 | 15,520 | 0.0 | 1,962.5 | 121.24 |
| OF1 | 13,090 | 26,707 | 0.9 | 4,735.0 | tl | 34,876 | 2,108 | 0.0 | 2,754.6 | 3.73 | 15,051 | 41,229 | 0.9 | 2,789.9 | tl |
| OF2 | 9,395 | 18,082 | 6.2 | 4,747.0 | tl | 31,889 | 2,120 | 0.0 | 2,734.1 | 5.64 | 8,746 | 21,938 | 0.0 | 2,747.2 | 124.61 |
| wang20 | 15,389 | 26,726 | 3.6 | 10,834.0 | tl | 32,511 | 1,879 | 0.0 | 2,721.0 | 5.53 | 5,793 | 15,520 | 0.0 | 2,784.5 | 32.61 |
| gcut1 | 660 | 1,038 | 0.0 | 112,502.0 | 94.05 | 2,809 | 525 | 0.0 | 50,739.5 | 0.28 | 520 | 1,553 | 0.0 | 50,823.5 | 0.09 |
| gcut2 | 2,611 | 4,176 | 0.0 | 138,133.0 | tl | 536,469 | 28,813 | 0.0 | 60,188.5 | 2,429.35 | 9,626 | 26,148 | 0.0 | 61,240.4 | 84.59 |
| gcut3 | 4,736 | 7,244 | 0.6 | 165,699.0 | tl | 947,384 | 32,711 | 0.0 | 60,638.5 | 2,705.92 | 29,591 | 78,053 | 0.0 | 61,731.7 | 1,550.54 |
| gcut4 | 8,604 | 11,766 | 0.0 | 226,788.0 | tl | 1,800,416 | 35,226 | 0.0 | 61,640.7 | tl | 105,456 | 276,254 | 100.0 | * | * |
| gcut5 | 1,227 | 2,204 | 0.0 | 414,013.0 | tl | 169,204 | 24,480 | 0.0 | 218,543.0 | 481.30 | 1,408 | 3,987 | 0.0 | 228,381.9 | 4.42 |
| gcut6 | 1,587 | 2,214 | 0.0 | 509,976.0 | tl | 960,704 | 73,834 | 0.0 | 237,651.5 | tl | 4,103 | 12,087 | 0.0 | 242,292.2 | 7.36 |
| gcut7 | 3,101 | 4,186 | 0.0 | 771,605.0 | tl | 2,162,047 | 100,575 | 100.0 | * | * | 12,094 | 34,089 | 0.0 | 242,976.0 | 67.66 |
| gcut8 | 8,604 | 11,766 | 0.0 | 859,788.0 | tl | 8,008,691 | 136,574 | 100.0 | * | * | 145,214 | 379,646 | 100.0 | * | * |
| gcut9 | 1,227 | 2,204 | 0.0 | 1,478,097.0 | tl | 308,140 | 36,162 | 0.0 | 937,266.0 | 1,243.86 | 1,803 | 4,720 | 0.0 | 947,014.0 | 1.30 |
| gcut10 | 910 | 1,048 | 0.0 | 2,126,738.0 | 1,758.86 | 21,137 | 2,556 | 0.0 | 962,059.7 | 6.11 | 3,180 | 9,600 | 0.0 | 962,059.7 | 10.95 |
| gcut11 | 4,736 | 7,244 | 0.1 | 2,660,197.0 | tl | 14,269,396 | 400,100 | 100.0 | * | * | 42,477 | 118,138 | 100.0 | * | * |
| gcut12 | 4,081 | 4,206 | 0.0 | 2,999,501.0 | tl | 23,073,016 | 489,478 | 100.0 | * | * | 91,051 | 261,285 | 100.0 | * | * |

Table 3: Results of the BBC algorithm.

| Instance | $\lvert\bar{X}\rvert$ | $\lvert\bar{Y}\rvert$ | BBC algorithm | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | var | cons | gap | t[s] | trials | cuts |
| cgcut1 | 12 | 11 | 470 | 117 | 0.0 | 0.02 | 0 | 0 |
| cgcut2 | 22 | 51 | 5,920 | 1,060 | 0.0 | 213.47 | 7 | 4 |
| cgcut3 | 18 | 46 | 3,029 | 784 | 0.0 | 183.46 | 116 | 114 |
| OF1 | 41 | 30 | 3,659 | 1,170 | 0.0 | 254.58 | 109 | 108 |
| OF2 | 30 | 30 | 3,243 | 851 | 0.0 | 14.99 | 1 | 0 |
| wang20 | 46 | 18 | 3,029 | 784 | 0.0 | 29.25 | 168 | 165 |
| gcut1 | 23 | 9 | 361 | 186 | 0.0 | 0.06 | 0 | 0 |
| gcut2 | 35 | 48 | 6,443 | 1,618 | 0.0 | 407.88 | 1,872 | 1,872 |
| gcut3 | 78 | 41 | 19,636 | 3,110 | 0.0 | tl | 3,100 | 3,099 |
| gcut4 | 85 | 81 | 71,393 | 6,770 | 100.0 | * | * | * |
| gcut5 | 16 | 23 | 892 | 340 | 0.0 | 1.91 | 2 | 1 |
| gcut6 | 33 | 38 | 3,053 | 1,204 | 0.0 | 13.43 | 0 | 0 |
| gcut7 | 63 | 33 | 9,244 | 2,014 | 0.0 | 99.66 | 156 | 156 |
| gcut8 | 95 | 133 | 97,795 | 12,458 | 100.0 | * | * | * |
| gcut9 | 26 | 11 | 1,137 | 260 | 0.0 | 7.45 | 881 | 881 |
| gcut10 | 28 | 49 | 2,335 | 1,316 | 0.0 | 357.56 | 5,472 | 5,472 |
| gcut11 | 63 | 104 | 31,800 | 6,416 | 0.0 | tl | 1,414 | 1,413 |
| gcut12 | 148 | 115 | 73,295 | 16,808 | 100.0 | * | * | * |

Table 3 shows the results obtained by the proposed BBC algorithm. For each instance, we also report the size of sets $\bar{X}$ and $\bar{Y}$, the number of trial solutions found by the BMP and verified in the BSP (column trials) and the number of cuts added to the BMP (column cuts). The number of variables and constraints in this table correspond to the initial BMP. In comparison to `Grid`, the BBC algorithm was able to prove optimality faster than this model in 5 instances, namely cgcut1, OF2, wang20, gcut1 ang gcut5. One of these instances is characterized by having its optimal value (OPT) close to the optimal value of the problem with non-guillotine pattern and, thus, it needs few feasibility cuts to converge to an optimal solution (see columns "trials" and "cuts"). The BBC algorithm provided a solution to gcut11, one of the instances in which CPLEX failed due to memory issues when using both `Grid` and `FMT`. The BBC algorithm proved optimality in two of the medium-size instances that `Grid` reached the time limit (cgcut2 and OF1), but the processing times were worse than those of `FMT`. Notice that the BBC algorithm obtained optimal solutions of all instances in which the method did not run out of memory, although it was not able to prove optimality for two of them (gcut3 and gcut11) within the time limit. As expected, the method was not able to provide optimal solutions for all instances with $m = 30$ or $m = 50$ item types in the set gcut. As the number of small item types increases, the number of (non-guillotine and guillotine) symmetrical solutions also increase in the BMP, which results in the generation of significantly more cuts in order to remove the non-guillotine trial solutions. It is worth mentioning that, in an experiment using a time limit of just a few minutes, the method was able to provide good-quality solutions for all those instances, but without the certificate of optimality. However, for the time limit of 1 hour, the number of cuts in the BPM led to the method running out of memory. Additionally, we note the processing time spent in the BSP is negligible, and the number of trials and cuts in Table 3 may differ as a trial solution is not cut if it is of guillotine-type, and the provided initial solution is not checked in the BSP.

### 5.2 Results for 2-staged and 1-group versions of 2D_R_CG_SLOPP

In this section, we analyse the performance of Model (13) for 2-staged patterns (`Grid-2staged`) and Model (16) for 1-group patterns (`Grid-1group`) with respect to the models proposed by Lodi et al (2002) and Yanasse and Morabito (2006), which were specifically designed to these types of guillotine cutting patterns, respectively. In the models for 2-staged patterns, we consider horizontal shelves only.

Tables 4 and 5 show the results of the four models. As they indicate, the models of Lodi et al (2002) and Yanasse and Morabito (2006) outperform `Grid-2staged` and `Grid-1group`, mainly regarding the processing solution time. For example, in Table 5, `Grid-1group` resulted in less processing time only for instance gcut9. Furthermore, the numbers of variables and constraints of Lodi et al (2002)'s model are significantly smaller than those of `Grid-2staged`, while for Yanasse and Morabito (2006)'s model and `Grid-1group` these numbers are closer. This behavior is somehow expected, as the model of Beasley (1985) – a sub-part of models `Grid-2staged` and `Grid-1group` – would already lead by itself to larger times than the benchmark models for some of these instances, which indicates the difficulty that the non-overlapping constraints bring to our models.

On the other hand, `Grid-2staged` and `Grid-1group` provided better LP-relaxation bounds than the other two models, especially with respect to Yanasse and Morabito (2006)'s model, which tends to be useful in the development of solution methods. Therefore, `Grid-2staged` and `Grid-1group` are likely to be suitable in to cases where the discretization of the large object provides an interesting feature. For instance, if the value of a small item type depends on its area and position due to different thicknesses in the large object, as discussed by Gilmore and Gomory (1965), then one can still use models `Grid-2staged` or `Grid-1group` after easily replacing parameter $v_k$ by $v_{kij}$. However, we believe that the other models from the literature cannot be be straightforwardly adapted for this case.

## 6 Conclusions

We addressed the Constrained Two-dimensional Rectangular Guillotine Single Large Placement Problem (2D_R_CG_SLOPP), a variant that has been broadly studied in the field of *Cutting & Packing* problems. Its applicability in different productive environment motivates the proposition of several solution methods since the seminal papers of Christofides and Whitlock (1977) and Wang (1983) for this problem. However,

Table 4: Comparison between Lodi et al (2002)'s model and `Grid-2staged`.

| Instance | Lodi et al (2002) | | | | | `Grid-2staged` | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | var | cons | gap | solr | t[s] | var | cons | gap | solr | t[s] |
| cgcut1 | 136 | 32 | 0.0 | 257.8 | 0.10 | 480 | 136 | 0.0 | 244.5 | 2.45 |
| cgcut2 | 276 | 46 | 0.0 | 2,878.0 | 0.18 | 5,970 | 1,159 | 0.0 | 2,906.1 | 1,158.17 |
| cgcut3 | 1,953 | 124 | 0.0 | 2,005.3 | 0.43 | 3,074 | 873 | 0.0 | 1,995.5 | 105.33 |
| OF1 | 276 | 46 | 0.0 | 2,800.0 | 0.47 | 3,688 | 1,227 | 0.0 | 2,789.9 | 138.83 |
| OF2 | 300 | 48 | 0.0 | 2,800.0 | 1.78 | 3,272 | 908 | 0.0 | 2,747.2 | 41.20 |
| wang20 | 903 | 84 | 0.0 | 2,800.0 | 1.98 | 3,046 | 817 | 0.0 | 2,791.1 | 39.67 |
| gcut1 | 55 | 20 | 0.0 | 62,192.6 | 0.17 | 369 | 201 | 0.0 | 50,960.0 | 1.72 |
| gcut2 | 210 | 40 | 0.0 | 62,397.1 | 0.45 | 6,490 | 1,711 | 0.0 | 61,240.4 | 39.02 |
| gcut3 | 465 | 60 | 0.0 | 62,500.0 | 1.07 | 19,676 | 3,189 | 0.0 | 61,785.3 | 571.51 |
| gcut4 | 1,275 | 100 | 0.0 | 62,500.0 | 1.34 | 71,473 | 6,929 | 100.0 | * | * |
| gcut5 | 55 | 20 | 0.0 | 247,475.5 | 0.60 | 914 | 383 | 0.0 | 228,381.9 | 1.96 |
| gcut6 | 210 | 40 | 0.0 | 249,494.5 | 0.05 | 3,090 | 1,277 | 0.0 | 242,556.1 | 32.44 |
| gcut7 | 465 | 60 | 0.0 | 250,000.0 | 0.06 | 9,276 | 2,077 | 0.0 | 243,671.0 | 96.12 |
| gcut8 | 1,275 | 100 | 0.0 | 250,000.0 | 1.33 | 97,927 | 12,721 | 100.0 | * | * |
| gcut9 | 55 | 20 | 0.0 | 990,591.4 | 0.11 | 1,147 | 279 | 0.0 | 947,014.0 | 1.35 |
| gcut10 | 210 | 40 | 0.0 | 997,715.9 | 1.21 | 2,383 | 1,411 | 0.0 | 967,987.4 | 15.32 |
| gcut11 | 465 | 60 | 0.0 | 999,177.6 | 2.91 | 31,903 | 6,621 | 0.0 | 983,752.5 | 2,244.23 |
| gcut12 | 1,275 | 100 | 0.0 | 1,000,000.0 | 1.21 | 73,409 | 17,035 | 100.0 | * | * |

Table 5: Comparison between Yanasse and Morabito (2006)'s model and `Grid-1group`.

| Instance | Yanasse and Morabito (2006) | | | | | `Grid-1group` | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | var | cons | gap | solr | t[s] | var | cons | gap | solr | t[s] |
| cgcut1 | 170 | 174 | 0.0 | 364.0 | 0.03 | 491 | 157 | 0.0 | 244.5 | 2.15 |
| cgcut2 | 837 | 600 | 0.0 | 4,449.0 | 0.57 | 5,991 | 1,200 | 0.0 | 2,906.1 | tl |
| cgcut3 | 6,956 | 2,624 | 0.0 | 20,631.7 | 3.61 | 3,091 | 906 | 0.0 | 1,995.5 | 491.27 |
| OF1 | 1,096 | 872 | 0.0 | 7,670.0 | 0.20 | 3,728 | 1,306 | 0.0 | 2,789.9 | 304.17 |
| OF2 | 962 | 708 | 0.0 | 8,652.0 | 0.19 | 3,301 | 965 | 0.0 | 2,747.2 | 207.12 |
| wang20 | 7,126 | 3,098 | 0.0 | 23,910.0 | 6.44 | 3,091 | 906 | 0.0 | 2,791.1 | 109.15 |
| gcut1 | 1,185 | 852 | 0.0 | 163,562.0 | 0.44 | 391 | 244 | 0.0 | 50,960.0 | 1.71 |
| gcut2 | 7,779 | 2,986 | 0.0 | 274,563.0 | 5.52 | 6,524 | 1,778 | 0.0 | 61,240.4 | 250.47 |
| gcut3 | 23,945 | 6,766 | 0.0 | 407,651.0 | 144.70 | 19,753 | 3,342 | 0.0 | 61,785.3 | 675.46 |
| gcut4 | 101,794 | 16,508 | 1.2 | 731,408.0 | tl | 71,557 | 7,096 | 100.0 | * | * |
| gcut5 | 1,274 | 832 | 0.0 | 545,300.0 | 2.68 | 929 | 412 | 0.0 | 228,381.9 | 2.78 |
| gcut6 | 8,159 | 3,005 | 0.0 | 1,232,057.0 | 15.42 | 3,122 | 1,340 | 0.0 | 242,556.1 | 59.77 |
| gcut7 | 24,709 | 6,584 | 0.0 | 2,004,791.0 | 103.97 | 9,338 | 2,200 | 0.0 | 243,671.0 | 164.93 |
| gcut8 | 106,416 | 17,256 | 3.2 | 2,805,462.0 | tl | 98,021 | 12,908 | 100.0 | * | * |
| gcut9 | 1,318 | 952 | 0.0 | 2,021,830.0 | 8.45 | 1,172 | 328 | 0.0 | 947,014.0 | 8.35 |
| gcut10 | 9,049 | 3,362 | 0.0 | 5,355,377.0 | 18.16 | 2,410 | 1,464 | 0.0 | 967,987.4 | 92.85 |
| gcut11 | 28,331 | 7,292 | 0.0 | 6,536,520.0 | 136.76 | 31,965 | 6,744 | 100.0 | * | * |
| gcut12 | 121,197 | 19,980 | 0.0 | 12,521,992.0 | 1,327.06 | 73,556 | 17,328 | 100.0 | * | * |

the first mathematical models to completely formulate this problem only appeared in the literature more than 30 years after the mentioned seminal papers (Ben Messaoud et al, 2008; Furini et al, 2016).

We proposed a compact INLP formulation that completely models the 2D_R_CG_SLOPP and an equivalent compact ILP formulation, both based on the model of Beasley (1985). They are extensions of this model to the guillotine case, as the original formulation was designed for non-guillotine patterns. In addition, we proposed a procedure to, without loss of generality, reduce the numbers of variables and constraints in the ILP formulation. By reformulating this ILP model, we developed two new compact formulations for special cases of guillotine cutting patterns (2-staged and 1-group). As a solution method for the 2D_R_CG_SLOPP, we proposed a branch-and-Benders-cut (BBC) algorithm, based on applying Benders decomposition to the proposed ILP formulation. The resulting Benders-Master-Problem corresponds to the model of Beasley (1985).

The results of computational experiments using benchmark instances from the literature pointed out that the proposed ILP formulation is appropriate for scenarios with small items that are large with respect to the object (as in the gcut instances), because they result in fewer allocation decisions (variables). In

this case, our model outperforms the other two formulations from the literature. On the other hand, it is not indicated in scenarios with many item types, as the opposite behavior tends to appear.

We also verified that our BBC algorithm presented a good average performance with respect to our ILP formulation, especially for instances in which the optimal value (OPT) is close to the optimal value of the model of Beasley (1985) (i.e., allowing non-guillotine patterns). However, the performance of the method is adversely affected by scenarios with many small item types, as the number of (guillotine and non-guillotine) symmetrical solutions tends to be large and hence the method needs significantly more combinatorial cuts to converge. With respect to the special cases of guillotine patterns (2-staged and 1-group), the proposed models achieved an inferior performance with respect to other models from the literature.

As future research, the proposed formulations can be enhanced with additional symmetry breaking techniques for the discretized object. Particularly, one could extend our BBC algorithm to the special cases of the 2D_R_CG_SLOPP that strictly generate 2-staged or 1-group patterns. The compact INLP formulation could be more explored in the field of Constraint Programming techniques by developing tailored-approaches, rather than relying on a general purpose solver. Adaptating the formulations in order to consider different applications is also relevant. For instance, suppose the distance of the guillotine cuts is somehow constrained as, e.g., in the glass industry; these models seem to be suitable to address such a case. The developments of other solution methods are also interesting considering different decomposition techniques (e.g., Lagrangean relaxation or Dantzig-Wolfe decomposition) or variable reduction procedures that are likely to enable the solution of larger instances.

**Acknowledgements** The authors would like to thank the Sao Paulo Research Foundation (FAPESP) [grant numbers 16/08039-1, 16/01860-1, 13/07375-0]. This study was financed in part by CNPq and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

## References

Alvarez-Valdés R, Parajón A, Tamarit JM (2002) A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. Computers and Operations Research 29(7):925–947, DOI 10.1016/S0305-0548(00)00095-2

Beasley JE (1985) An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure. Operations Research 33(1):49–64, DOI 10.1287/opre.33.1.49

Belov G, Scheithauer G (2006) A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research 171(1):85 – 106, DOI https://doi.org/10.1016/j.ejor.2004.08.036, URL http://www.sciencedirect.com/science/article/pii/S0377221704006150

Ben Messaoud S, Chu C, Espinouse ML (2008) Characterization and modelling of guillotine constraints. European Journal of Operational Research 191(1):110–124, DOI 10.1016/j.ejor.2007.08.029

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4(1):238–252, DOI 10.1007/BF01386316, URL http://dx.doi.org/10.1007/BF01386316

Christofides N, Hadjiconstantinou E (1995) An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. European Journal of Operational Research 83(1):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(1):30–44, DOI 10.1287/opre.25.1.30

Costa AM (2005) A survey on Benders decomposition applied to fixed-charge network design problems. Computers & Operations Research 32(6):1429–1450

Côté J, Guastaroba G, Speranza M (2017) The value of integrating loading and routing. European Journal of Operational Research 257(1):89 – 105, DOI https://doi.org/10.1016/j.ejor.2016.06.072, URL http://www.sciencedirect.com/science/article/pii/S0377221716305355

de Queiroz TA, Hokama PHDB, Schouery RCS, Miyazawa FK (2017) Two-dimensional disjunctively constrained knapsack problem: Heuristic and exact approaches. Computers & Industrial Engineering 105:313 – 328, DOI http://dx.doi.org/10.1016/j.cie.2017.01.015, URL http://www.sciencedirect.com/science/article/pii/S0360835217300347

Dolatabadi M, Lodi A, Monaci M (2012) Exact algorithms for the two-dimensional guillotine knapsack. Computers and Operations Research 39(1):48–53, DOI 10.1016/j.cor.2010.12.018, URL http://dx.doi.org/10.1016/j.cor.2010.12.018

Dyckhoff H (1981) A new linear programming approach to the cutting stock problem. Operations Research 29(6):1092–1104, URL http://www.jstor.org/stable/170363

Furini F, Malaguti E, Thomopulos D (2016) Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. INFORMS Journal on Computing 28(4):736–751, DOI 10.1287/ijoc.2016.0710, URL http://pubsonline.informs.org/doi/10.1287/ijoc.2016.0710

Gilmore PC, Gomory RE (1965) Multistage Cutting Stock Problems of Two and More Dimensions. Operations Research 13(1):94–120, DOI 10.1287/opre.13.1.94, URL http://pubsonline.informs.org/doi/abs/10.1287/opre.13.1.94

Henn S, Wäscher G (2013) Extensions of cutting problems: setups. Pesquisa Operacional 33:133–162, DOI 10.1590/S0101-74382013000200001, URL http://www.scielo.br/scielo.php?script=sci{_}arttext{&}pid=S0101-74382013000200001{&}nrm=iso

Herz JC (1972) Recursive Computational Procedure for Two-dimensional Stock Cutting. IBM Journal of Research and Development 16(5):462–469, DOI 10.1147/rd.165.0462

Hifi M (2004) Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. Journal of Combinatorial Optimization 8(1):65–84, DOI 10.1023/B:JOCO.0000021938.49750.91

Linhares A, Yanasse HH (2002) Connections between cutting-pattern sequencing, vlsi design, and flexible machines. Computers & Operations Research 29(12):1759 – 1772, DOI https://doi.org/10.1016/S0305-0548(01)00054-5, URL http://www.sciencedirect.com/science/article/pii/S0305054801000545

Lodi A, Monaci M (2003) Integer linear programming models for 2-staged two-dimensional Knapsack problems. Mathematical Programming 94(2-3):257–278, DOI 10.1007/s10107-002-0319-9, URL http://link.springer.com/10.1007/s10107-002-0319-9

Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: A survey. European Journal of Operational Research 141(2):241–252, DOI 10.1016/S0377-2217(02)00123-6

Melega GM, de Araujo SA, Jans R (2018) Classification and literature review of integrated lot-sizing and cutting stock problems. European Journal of Operational Research DOI https://doi.org/10.1016/j.ejor.2018.01.002, URL http://www.sciencedirect.com/science/article/pii/S037722171830002X

Misener R, Floudas CA (2013) Glomiqo: Global mixed-integer quadratic optimizer. Journal of Global Optimization 57(1):3–50, DOI 10.1007/s10898-012-9874-7, URL https://doi.org/10.1007/s10898-012-9874-7

Misener R, Floudas CA (2014) Antigone: Algorithms for continuous / integer global optimization of nonlinear equations. Journal of Global Optimization 59(2):503–526, DOI 10.1007/s10898-014-0166-2, URL https://doi.org/10.1007/s10898-014-0166-2

Morabito R, Arenales M (2000) Optimizing the cutting of stock plates in a furniture company. International Journal of Production Research 38(12):2725–2742, DOI 10.1080/002075400411457, URL https://doi.org/10.1080/002075400411457, https://doi.org/10.1080/002075400411457

Morabito R, Arenales MN (1996) Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. European Journal of Operational Research 94(3):548–560, DOI 10.1016/0377-2217(95)00128-X

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(1):297–315, DOI 10.1007/s10479-008-0457-4

Moreno A, Munari P, Alem D (2018) Branch-and-Benders-cut algorithm for the Crew Scheduling and Routing Problem in Network Repair, Technical Report 001/2018, Operations Research Group, Production Engineering Department, Federal University of Sao Carlos, URL http://www.optimization-online.org/DB_HTML/2018/01/6422.html

Oliveira J, Ferreira J (1990) An improved version of Wang's algorithm for two-dimensional cutting problems. European Journal of Operational Research 44(2):256–266, DOI 10.1016/0377-2217(90)90361-E

Parada V, Gómes A, de Diego J (1995) Exact solutions for constrained two-dimensional cutting problems. European Journal of Operational Research 84(3):633–644, DOI 10.1016/0377-2217(95)00028-O

Parada V, Sepúlveda M, Solar M, Gómes A (1998) Solution for the constrained guillotine cutting problem by simulated annealing. Computers & Operations Research 25(I):37–47, URL http://www.

sciencedirect.com/science/article/pii/S0305054898800063

Puchinger J, Raidl GR (2007) Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research 183(3):1304–1327, DOI 10.1016/j.ejor.2005.11.064

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. European Journal of Operational Research 259(3):801–817, DOI 10.1016/j.ejor.2016.12.005

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

Silva E, Alvelos F, Valério de Carvalho JM (2010) An integer programming model for two- and three-stage two-dimensional cutting stock problems. European Journal of Operational Research 205(3):699–708, DOI 10.1016/j.ejor.2010.01.039, URL http://dx.doi.org/10.1016/j.ejor.2010.01.039

Tawarmalani M, Sahinidis NV (2005) A polyhedral branch-and-cut approach to global optimization. Mathematical Programming 103(2):225–249, DOI 10.1007/s10107-005-0581-8, URL https://doi.org/10.1007/s10107-005-0581-8

Velasco AS, Uchoa E (2018) Improved state space relaxation for constrained two-dimensional guillotine cutting problems. European Journal of Operational Research DOI https://doi.org/10.1016/j.ejor.2018.06.016, URL http://www.sciencedirect.com/science/article/pii/S0377221718305393

Vigerske S, Heinz S, Gleixner A, Berthold T (2012) Analyzing the computational impact of MIQCP solver components. Numerical Algebra, Control and Optimization 2(4):739–748, DOI 10.3934/naco.2012.2.739, URL https://doi.org/10.3934%2Fnaco.2012.2.739

Viswanathan KV, Bagchi A (1993) Best-First Search Methods for Constrained Two-Dimensional Cutting Stock Problems. Operations Research 41(4):768–776, DOI 10.1287/opre.41.4.768, URL http://dx.doi.org/10.1287/opre.41.4.768

Wang PY (1983) Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. Operations Research 31(3):573–586, DOI 10.1287/opre.31.3.573, URL http://dx.doi.org/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(3):1109–1130, DOI 10.1016/j.ejor.2005.12.047

Yanasse HH, Katsurayama DM (2005) Checkerboard pattern: proposals for its generation. International Transactions in Operational Research 12(1):21–45, DOI 10.1111/j.1475-3995.2005.00488.x, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2005.00488.x, https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2005.00488.x

Yanasse HH, Katsurayama DM (2008) An enumeration scheme to generate constrained exact checkerboard patterns. Computers & Operations Research 35(6):2114 – 2128, DOI https://doi.org/10.1016/j.cor.2006.10.018, URL http://www.sciencedirect.com/science/article/pii/S0305054806002759, part Special Issue: OR Applications in the Military and in Counter-Terrorism

Yanasse HH, Morabito R (2006) Linear models for 1-group two-dimensional guillotine cutting problems. International Journal of Production Research 44(17):3471–3491, DOI 10.1080/00207540500478603, URL http://www.tandfonline.com/doi/abs/10.1080/00207540500478603

Yanasse HH, Morabito R (2008) A note on linear models for two-group and three-group two-dimensional guillotine cutting problems. International Journal of Production Research 46(21):6189–6206, DOI 10.1080/00207540601011543, URL http://www.tandfonline.com/doi/abs/10.1080/00207540601011543