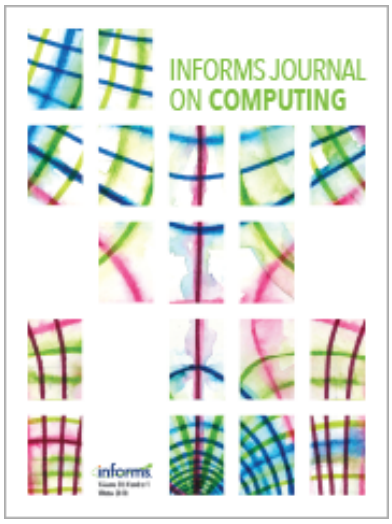

INFORMS Journal on Computing

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

Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming

Fabio Furini, Enrico Malaguti, Dimitri Thomopulos

To cite this article:
Fabio Furini, Enrico Malaguti, Dimitri Thomopulos (2016) Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. INFORMS Journal on Computing 28(4):736-751. https://doi.org/10.1287/ijoc.2016.0710

INFORMS Journal on Computing
Vol. 28, No. 4, Fall 2016, pp. 736–751
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2016.0710

Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming

Fabio Furini
LAMSADE, Université Paris-Dauphine, 75775 Paris, France, fabio.furini@dauphine.fr

Enrico Malaguti, Dimitri Thomopulos
DEI, Università di Bologna, 40136 Bologna, Italy {enrico.malaguti@unibo.it, dimitri.thomopulos@unibo.it}

We propose a framework to model general guillotine restrictions in two-dimensional cutting problems formulated as mixed-integer linear programs (MIPs). The modeling framework requires a pseudopolynomial number of variables and constraints, which can be effectively enumerated for medium-size instances. Our modeling of general guillotine cuts is the first one that, once it is implemented within a state-of-the-art MIP solver, can tackle instances of challenging size. We mainly concentrate our analysis on the guillotine two-dimensional knapsack problem (G2KP), for which a model, and an exact procedure able to significantly improve the computational performance, are given. We also show how the modeling of general guillotine cuts can be extended to other relevant problems such as the guillotine two-dimensional cutting stock problem and the guillotine strip packing problem (GSPP). Finally, we conclude the paper discussing an extensive set of computational experiments on G2KP and GSPP benchmark instances from the literature.

Keywords: integer programming; guillotine cuts; mathematical model
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received January 2015; revised September 2015, February 2016; accepted March 2016. Published online October 5, 2016.

1. Introduction

Two-dimensional cutting problems are about obtaining a set of small (rectangular) *items* from one or more (rectangular) larger *panels*. Cutting problems are of great relevance in metal, wood, paper, and glass industries, but also in loading, transportation, telecommunications, and resource allocation in general (see, e.g., Vanderbeck 2001, Burke et al. 2006, Iori et al. 2007, Lodi et al. 2011, Bennell et al. 2013, Malaguti et al. 2014). Depending on the industry, special features for the cuts may be required; a very common one that applies to glass and wood cutting is to have *guillotine* cuts.

(i) In guillotine cutting, items are obtained from panels through cuts that are parallel to the sides of the panel and cross the panel from one side to the other;

(ii) Cuts are performed in *stages*, where each stage consists of a set of parallel guillotine cuts on the shapes obtained in the previous stages;

(iii) Each cut removes a so-called *strip* from a panel. If during the cut sequence, the width of each cut strip equals the width of the widest item obtained from the strip, then the cut is denoted as *restricted*.

Our objective is to propose a way of modeling general guillotine cuts via mixed-integer linear programs (MIPs); i.e., *we do not limit the number of stages* (restriction (ii)) *or impose that the cuts be restricted* (restriction (iii)). We only ask that the cuts be guillotine (restriction (i)). In the following, we call these kind of cuts *general guillotine cuts* or simply *guillotine cuts*.

In Figure 1 we report, on the left, a pattern (cutting scheme) that cannot be obtained through guillotine cuts; in the center, a pattern that can be obtained through three-stage restricted guillotine cuts; and on the right, a pattern that needs unrestricted guillotine cuts to be obtained, which are the ones we are interested in.

Families of cutting problems. In the following, we will mainly concentrate our analysis on the two-dimensional knapsack problem and briefly discuss extensions of the modeling ideas to the two-dimensional cutting stock problem and the strip packing problem. For convenience, we review the definitions of these problems.

- *Two-Dimensional Knapsack Problem* (2KP): we are given one rectangular panel of length L and width W, and a list of n rectangular items; each item i ($i = 1, \dots, n$) is characterized by a length l_i, a width w_i, and a profit p_i, and is available in u_i copies. The 2KP requires cutting the subset of items of largest profit that can fit in the rectangular panel (without overlapping).
- *Two-Dimensional Cutting Stock Problem* (2CSP): we are given infinitely many identical rectangular panels,

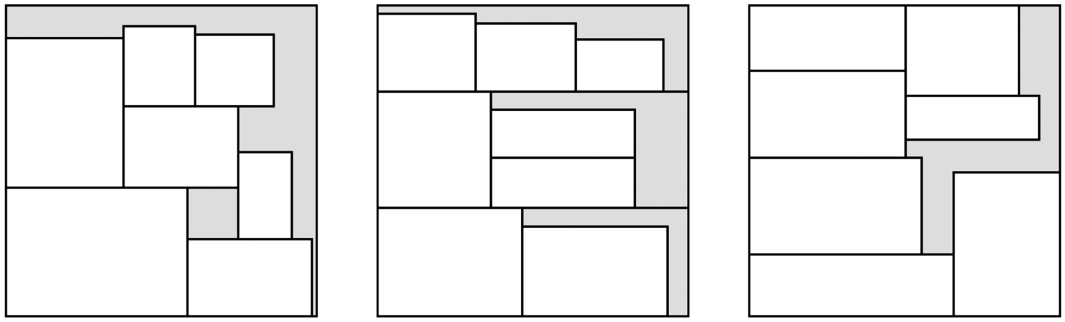

Figure 1 **Examples of Patterns That Can Be Obtained Through Non-Guillotine Cuts (Left), Three-Stage Restricted Guillotine Cuts (Center), and Guillotine Cuts (Right)**

each one having length L and width W and a list of n rectangular items; each item i ($i = 1, \ldots, n$) is characterized by a length l_i and a width w_i, and must be cut in d_i copies. The 2CSP requires cutting all of the items by minimizing the number of used panels. The special case where the demand of each items is equal to 1 is denoted as the two-dimensional bin packing problem (2BPP).

- *Strip Packing Problem* (SPP): we are given a strip having length L and infinite width, and a list of n rectangular items; each item i ($i = 1, \ldots, n$) is characterized by a length l_i and a width w_i, and must be cut in d_i copies. The SPP requires cutting all of the items from the strip by minimizing the used strip width.

In this paper, we consider the *guillotine* versions of these problems (restriction (i))—i.e., the *guillotine* 2KP (G2KP), *guillotine* 2CSP (G2CSP), and *guillotine* SPP (GSPP). All of these problems are NP-hard.

Structure of general guillotine cuts. Let us consider an example that shows the differences among the optimal solutions of the G2KP obtained by imposing decreasing restrictions to the cuts performed. We compare the structure of the optimal solutions for an unweighted instance of two-dimensional knapsack of seven items, where item profits equal their areas. In the left of Figure 2, we report the optimal solution of the guillotine *two-stage* 2KP, where the rectangular panel is first divided into horizontal strips, and then items are obtained from the strips by vertical cuts. Further horizontal cuts (trimming) may be necessary to obtain the final items. In the center of the figure, we represent the optimal solution of the G2KP when we consider guillotine cuts with an *unlimited number of stages*, but cuts are *restricted* (i.e., they define strips whose width (resp., length) equals the width (resp., length) of some item that is obtained from the strip). The profit of this solution is 9.97% larger than the profit of the two-stage solution. Finally, in the right of the figure we report the optimal solution of the G2KP studied in this paper: the only restriction imposed on the cuts is that they be guillotine; they are not restricted or limited in the number of stages. The profit of this solution, where all of the seven items are obtained, is 17.04% larger than the profit of the two-stage solution. The example shows a case where the tree problems have different optimal solutions of strictly increasing profit.

Literature review. Cutting problems were introduced by Gilmore and Gomory (1965), who considered the G2CSP and proposed the *k-stage* version of the problem. The authors introduced the well known exponential-size model that is usually solved via column generation, where the pricing problem is a one-dimensional knapsack problem. Since the seminal work of Gilmore and Gomory (1965), a relevant body of literature on two-dimensional cutting has been developed; thus, we mainly concentrate this review on 2KPs, and on guillotine cutting. For a more comprehensive survey on two-dimensional cutting and packing, the reader is referred to Lodi et al. (2002) and Wäscher et al. (2007).

Concerning the 2*KP* (with no specific restrictions on the cut features), Boschetti et al. (2002) proposed a branch-and-bound algorithm based on a MIP

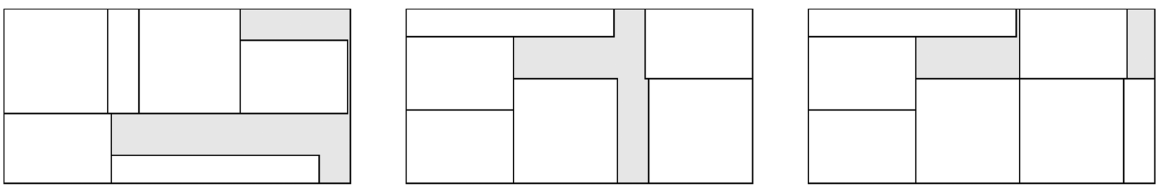

Figure 2 **Optimal Solutions for an Unweighted Seven Items Instance: Two-Stage 2KP (Left), Restricted Guillotine 2KP (Center), Guillotine 2KP (Right)**

formulation. Caprara and Monaci (2004) and Fekete et al. (2007) proposed exact algorithms. The first one is based on a relaxation given by the KP instance with item weights coincident with the rectangle areas; for this relaxation a worst-case performance ratio of three is proved. The latter is based on bounding procedures exploiting dual feasible functions.

Restricting the attention to *guillotine cutting*, the majority of contributions in the recent literature considered the case where the *number of stages is limited to two or three*. Unless explicitly stated, three-stage approaches are for the restricted case. Pisinger and Sigurd (2007) consider the G2CSP and solve the pricing problem as a constraint satisfaction problem, by considering among others the case of guillotine cutting (with limited and unlimited number of stages). Puchinger and Raidl (2007) propose compact models and a branch-and-price algorithm for the three-stage G2BPP. They consider the unrestricted case as well. A more application-oriented study is presented in Vanderbeck (2001), where a real-world G2CSP with multiple panel size and additional features is solved via column generation in an approximate fashion. A similar real-world problem, with the additional feature that identical cutting patterns can be processed in parallel, was recently considered by Malaguti et al. (2014).

In terms of *optimization models* not based on the Gilmore and Gomory (exponential size) formulation, Lodi and Monaci (2003) presented a compact model for the guillotine two-stage 2KP. Macedo et al. (2010) solved the guillotine two-stage 2CSP by extending a MIP formulation proposed by Valério de Carvalho (2002) for the one-dimensional CSP. The extension of the model to two dimensions asks us to define a set of flow problems to determine a set of horizontal strips and a flow problem to determine how the strips fit into the rectangular panel. Silva et al. (2010) presented a pseudopolynomial size model for the guillotine two- and three-stage 2CSP based on the concepts of item to be cut and residual plates, obtained after the cut. Recently, Furini and Malaguti (2016) extended this idea to model the guillotine two-stage 2KP. Finally, a computational comparison of compact, pseudopolynomial, and exponential size (based on the Gilmore and Gomory formulation) models for the guillotine two-stage 2CSP with multiple panel size is presented in Furini and Malaguti (2013).

Few contributions are available in the literature for *guillotine cutting problems* with an *unlimited number of stages*. This is probably due to the intrinsic difficulty of modeling guillotine restrictions. In addition to the mentioned paper by Pisinger and Sigurd (2007), where guillotine restrictions are tackled through constraint satisfaction techniques, exact approaches for G2KP have been proposed by Christofides and Whitlock (1977), Christofides and Hadjiconstantinou (1995), and Cung et al. (2000). Cintra and Wakabayashi (2004) proposed a recursive exact algorithm for the unconstrained case of G2KP—i.e., the case where no upper bound on the number of items of each type exists. A dynamic programming algorithm able to solve large-size instances of the latter problem was recently proposed by Russo et al. (2014). The most recent exact approach to G2KP is due to Dolatabadi et al. (2012), where a recursive procedure is presented that, given a set of items and a rectangular panel, constructs the set of associated guillotine packings. This procedure is then embedded into two exact algorithms and computationally tested on a set of instances from the literature. In addition to the reported exact methods, Hifi (1997) proposed two upper bounding procedures; concerning heuristic algorithms, we mention the hybrid algorithm by Hifi (2004) and the recursive algorithm by Chen (2008). The related problem of determining if a given set of items can be obtained from a given large rectangle by means of guillotine cuts was modeled through oriented graphs by Clautiaux et al. (2013). Recently, an exact algorithm for the GSPP was proposed by Fleszar (2016). This algorithm constructs guillotine cutting patterns starting from single items and repeatedly combines items into partial patterns, and partial patterns (and items) into larger ones, as proposed by Wang (1983). Key features of the algorithm are the generation of patterns in the order of nonincreasing waste and the discarding of dominated patterns. In this way, the number of generated partial patterns can be considerably reduced.

Despite the relevance of general guillotine cutting problems, the only MIP model in the literature we are aware of was proposed by Ben Messaoud et al. (2008) and solves the guillotine GSPP. The model is polynomial in the input size, but in practice it has a very large number of variables and constraints, and as observed in Ben Messaoud et al. (2008), its linear programming relaxation produces a "very loose lower bound." For these reasons the authors report computational experiments where instances with five items are solved in nonnegligible computing time.

Paper Contribution. The main contribution of this paper is to propose a way of modeling general guillotine cuts via MIPs. The modeling framework requires a pseudopolynomial number of variables and constraints, which can all be explicitly enumerated for medium-size instances. To the best of our knowledge, this is the first attempt to model guillotine restrictions via MIPs that works in practice—i.e., once it is implemented within a state-of-the-art solver, it can tackle instances of challenging size. In this paper, we mainly concentrate on the G2KP. We model the problem as a MIP and propose an effective exact method for selecting a subset of the variables containing an optimal solution. Then, the resulting model can be solved by

a general purpose MIP solver by only considering the subset of selected variables. This exact procedure is able to significantly increase the number of instances solved to proven optimality. In addition, we propose a number of procedures to further reduce the number of variables and constraints, and discuss conditions under which these reductions preserve the optimality of the solutions. We show how the modeling of guillotine cuts can be extended to other relevant problems such as the G2CSP and the GSPP. Finally, we conclude the paper by an extensive set of computational experiments on benchmark G2KP and GSPP instances from the literature.

2. MIP Modeling of Guillotine Cuts

Dyckhoff (1981) proposed a pseudopolynomial size model for the (one-dimensional) CSP, based on the concepts of *cut* and *residual element*. The idea is the following: each time a stock of length L is cut to produce an item i of length l_i, a residual element of size $L-l_i$ is obtained, which can be further used to produce additional items. The model associates a decision variable to each item and each (stock or residual) element, and feasible solutions are obtained by imposing balance constraints on the number of residual elements, while the cost of a solution is given by the number of used stock elements.

We extend the approach of Dyckhoff (1981) to two dimensions by using the concepts of *cut* and *plate*, where a plate can be either the original rectangular panel or a smaller rectangular residual plate obtained from the panel as a result of a sequence of guillotine cuts. We concentrate on the G2KP. The main idea of the model we propose is the following: starting from the initial rectangular panel, we obtain two smaller plates through a horizontal or vertical guillotine cut; for each obtained plate, we need to decide where to perform further cuts or eventually to keep the plate as it is when its dimensions equal the dimensions of one of the items to obtain. The process is iterated until the plates are large enough to fit some item.

In the model we propose, each cut decision is represented by a triple (q, j, o), where *position* q denotes the distance from the bottom-left corner of a *plate* j, where a cut with *orientation* o is performed. In the left panel of Figure 3 we depict a vertical cut performed at position q on a generic plate j, producing two smaller plates j_1 and j_2. We depict a horizontal cut in the right panel of the figure.

Without loss of generality we can assume that all problem data are positive integers. We denote by J the set of plates where the rectangular panel, indexed by $j=0$, has dimensions L, W and each plate j has

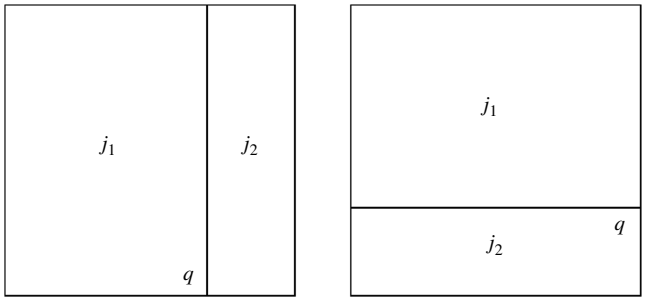

Figure 3 **Vertical (Left) and Horizontal (Right) Cut at Position q Producing Two Plates j_1 and j_2**

dimensions (l_j, w_j), with $1 \le w_j \le W$ and $1 \le l_j \le L$. The actual values of plate dimensions are discussed in the next section. We denote by $O=\{h, v\}$ the set of possible orientations for a cut (horizontal and vertical, respectively) and by $o \in O$ the generic orientation. We denote by $\bar{J} \subset J$ the subset of plates having dimensions equal to one of the items; thus, with a slight abuse of notation, $\bar{J}$ also denotes the set of items. Without loss of generality, we assume $0 \in \bar{J}$ (in case the rectangular panel does not correspond to an item to obtain, we set $u_0 = 0$). For a plate j, we define by $Q(j, o)$ the set of positions where we can cut j with orientation $o \in O$. We have $Q(j, h) \subseteq \{1, \dots, w_j - 1\}$ and $Q(j, v) \subseteq \{1, \dots, l_j - 1\}$.

The model has integer variables x^o_{qj} denoting the number of times a plate of type j is cut at position q through a guillotine cut with orientation o. Let a^o_{qkj} be a coefficient taking value 1 when a plate of type k is obtained by cutting at position q a plate of type j by a cut with orientation o, and 0 otherwise. In addition, we use the integer variables y_j, $j \in \bar{J}$, denoting the number of plates of type j that are kept as final items or, equivalently, the number of items of type j that are obtained.

The G2KP can be modeled as follows:

$$\text{(PP-G2KP):} \quad \max \sum_{j \in \bar{J}} p_j y_j \tag{1}$$

$$\sum_{k \in J} \sum_{o \in O} \sum_{q \in Q(k,\, o)} a^o_{qkj} x^o_{qk} - \sum_{o \in O} \sum_{q \in Q(j,\, o)} x^o_{qj} - y_j \ge 0 \qquad j \in \bar{J},\ j \ne 0 \tag{2}$$

$$\sum_{k \in J} \sum_{o \in O} \sum_{q \in Q(k,\, o)} a^o_{qkj} x^o_{qk} - \sum_{o \in O} \sum_{q \in Q(j,\, o)} x^o_{qj} \ge 0 \quad j \in J \setminus \bar{J} \tag{3}$$

$$\sum_{o \in O} \sum_{q \in Q(0,\, o)} x^o_{q0} + y_0 \le 1 \tag{4}$$

$$y_j \le u_j \quad j \in \bar{J} \tag{5}$$

$$x^o_{qj} \ge 0 \text{ integer} \quad j \in J, o \in O, q \in Q(j, o) \tag{6}$$

$$y_j \ge 0 \text{ integer} \quad j \in \bar{J}, \tag{7}$$

where the objective function (1) maximizes the profit of cut items; constraints (2) impose that the number

of plates j that are cut or kept as items does not exceed the number of plates j obtained through the cut of some other plates; constraints (3) are equivalent to the previous constraints for plates $j \notin \bar{J}$ (hence, the corresponding y_j variables are not defined); constraint (4) imposes that the original rectangular panel is not used more than once; constraints (5) impose not to exceed the maximum number of items which can be obtained; and finally, (6) and (7) force the variables to be nonnegative integers.

The model has a pseudopolynomial size; indeed, in the worst case the number of plates is WL, and each plate can be horizontally cut in $O(W)$ positions and vertically cut in $O(L)$ positions. The overall number of x variables is thus $O(WL(W+L))$, in addition to the y variables of which there are n. In the following, we denote this pseudopolynomial size model as *PP-G2KP Model*. Indeed, not all of the plates (accordingly, the variables and the constraints of the *PP-G2KP Model*) are necessary to preserve the optimality of the solutions. In the following, we discuss different ways of safely reducing the number of variables and constraints for the *PP-G2KP Model*.

The plates and, accordingly, the model variables can be enumerated by processing the item set $\bar{J}$ and the rectangular panel (plate 0) as described in Procedure 1. Starting from plate 0, new plates are obtained through vertical and horizontal cuts (line 7) and stored in set J when their size is such that they can fit some item (lines 9–12); otherwise, the new plate is discarded. The definition of the set of positions $Q(j,o)$ where plate j is cut with orientation o (line 5) is discussed in Section 2.1.

Procedure 1 (`Plate-and-variable enumeration`)
Input: plate 0, items set $\bar{J}$
Output: plates set J, variables x
1: initialize $J=\{0\}$, mark 0 as nonprocessed;
2: **while** J contains non-processed plates **do**
3: select a nonprocessed $j \in J$;
4: **for all** $o \in \{h, v\}$ **do**
5: compute the set of cut positions $Q(j,o)$;
6: **for all** positions $q \in Q(j,o)$ **do**
7: cut j at q with orientation o, generate plates j_1, j_2;
8: **if** $j_1 \notin$ J and j_1 can fit some item **then**
9: set $J = J \cup \{j_1\}$;
10: **end if**
11: **if** $j_2 \notin$ J and j_2 can fit some item **then**
12: set $J = J \cup \{j_2\}$;
13: **end if**
14: create x^o_{qj};
15: **end for**
16: **end for**
17: mark j as processed;
18: **end while**
19: **return** J, x.

A related extension of the model in Dyckhoff (1981) was proposed by Silva et al. (2010) to model two- and three-stage restricted G2CSPs. In Silva et al. (2010), a decision variable defines the cut of an *item* from a *plate* through two orthogonal guillotine cuts, which in addition to the item produce (up to) two residual plates. This idea cannot be extended to the unrestricted case, where the position of a cut may not correspond to the size of an item. In Section 2.1, we provide a lower bound on the largest profit loss that is incurred by considering the restricted case instead of the unrestricted case. An extension of the model of Silva et al. (2010) to two-stage guillotine knapsack problems is discussed in Furini and Malaguti (2016).

Finally, we mention Arbib et al. (2002), where a further extension of the model in Dyckhoff (1981) is presented. The authors consider a one-dimensional CSP where the residual elements can be reused and, in the specific application case, combined, so as to obtain the requested items.

2.1. Definition of the Cut Position Set

Model (1)–(7) can have very large size, depending on the cardinality of sets J and $Q(j,o)$, $j \in J$, $o \in O$. The number of plates that we consider and the number of cuts performed on each plate (eventually producing new plates) determine, in practice, the size of the model. Thus, a crucial question to be answered is the following:

Given a plate j of length l_j and width w_j, how should $Q(j,o)$, $o \in O$ be defined to minimize the number of variables and plates of the model, while preserving the optimality of the solution?

Let I_j be the set of items that can fit into plate j; i.e., $I_j = \{i \in \bar{J}\colon l_i \le l_j, w_i \le w_j\}$. The *complete position set* (Q) where a cut can be performed includes the dimensions of items $i \in I_j$, and all combinations of the items $i \in I_j$ dimensions, and is defined as follows:

$$Q(j,h) = \Big\{q\colon 0 < q < w_j;\ \forall i \in I_j,\ \exists n_i \in \mathbb{N},\ n_i \le u_i,\ q = \textstyle\sum_{i\in I_j} n_i w_i\Big\} \tag{8}$$

and

$$Q(j,v) = \Big\{q\colon 0 < q < l_j;\ \forall i \in I_j,\ \exists n_i \in \mathbb{N},\ n_i \le u_i,\ q = \textstyle\sum_{i\in I_j} n_i l_i\Big\}. \tag{9}$$

These positions are known in the literature as *discretization points*, and a pattern where cuts are performed at discretization points is known as a *normal* or *canonical* pattern (see Herz 1972, Christofides and Whitlock 1977). All of the combinations of items defining the complete position set can be effectively obtained by a dynamic programming (DP) algorithm. The DP algorithm we used is an extension to the case of items available in several copies of the one described in Trick (2003) (which only considers single items).

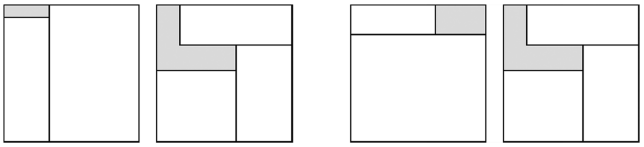

Figure 4 **Possible Configurations After Separating Five Items with a Vertical Guillotine Cut**

Let us also define the *restricted position set* (Q_R), including only the dimensions of items $i \in I_j$, as

$$Q_R(j,h) = \{q\colon \exists\, i \in I_j,\ q = w_i\},$$
$$Q_R(j,v) = \{q\colon \exists\, i \in I_j,\ q = l_i\}. \tag{10}$$

Note that one can remove symmetric cut positions for a plate j from set $Q(j,o)$, $o \in O$ (resp. $Q_R(j,o)$); i.e.,

$$w_j - q \notin Q(j,h), \quad \forall\, q \in Q(j,h),\ q < w_j/2$$
$$l_j - q \notin Q(j,v), \quad \forall\, q \in Q(j,v),\ q < l_j/2.$$

These positions are automatically discarded by the DP algorithm.

Considering the *PP-G2KP Model* with the restricted position set only does not guarantee in general the optimality of the corresponding solution. The following theorem states a condition under which the optimality is preserved.

Theorem 1. *If a plate j can fit at most five items by guillotine cuts, then an optimal solution to the PP-G2KP Model exists by considering the positions q for plate j restricted to $Q_R(j,h)$ and $Q_R(j,v)$.*

Proof of Theorem 1. We need to show that, when cutting five items from a plate, any packing can be obtained by considering restricted cut positions only. Without loss of generality, consider the first cut to be vertical. When performing the first guillotine cut, either we obtain two new plates that contain two and three items, respectively, or we obtain two new plates that contain one and four items, respectively. In the latter case, the first cut can be performed at a position corresponding to the length of the item that is alone in the new plate. Assume instead the first case holds, and consider the new plate containing two items. If they are placed one on the side of the other (as in the left panel of Figure 4), it was possible to separate one of them with the first cut. When the two items are placed one on top of the other (as in the right panel of Figure 4), the first cut could be performed at a position equal to the length of the longest of the two. Similar considerations apply when cutting four items from a plate. □

Consider the following example.

Example 1. Consider an instance of the G2KP of six items with dimensions $l = [34, 30, 30, 8, 31, 60]$ and $w = [47, 40, 40, 40, 11, 4]$, and a rectangular panel of dimensions $L = 102$, $W = 51$ (see Figure 5). All items can be obtained from the rectangular panel through guillotine cuts (in Figure 5, the first cut is vertical and separates the panel in two new plates containing three items each), but no feasible solution to the *PP-G2KP Model* with q restricted to $Q_R(j,o)$, $o \in O$ allows us to obtain the six items.

From Example 1 it follows that:

Remark 1. The value of five items in Theorem 1 is tight.

Given the result of Theorem 1, a reduction in the number of variables of the *PP-G2KP Model* can be obtained by considering positions in set Q_R for plates with the property that they can fit at most five items. Since exactly checking this condition can be computationally expensive, we considered the following relaxation. A sufficient condition for this property to hold is that the cumulative area of the six smallest items exceeds the plate area. In the following, we denote the reduction obtained by checking the previous sufficient condition as `Cut-Position` reduction.

Restricting the positions to $Q_R(j,h)$ and $Q_R(j,v)$ for *all* plates has a large impact on the number of variables and plates of the *PP-G2KP Model* (see Section 4.1) but potentially leads to suboptimal solutions. Thus, it is natural to wonder what the loss of profit is in the worst case. In the following, we denote the *PP-G2KP Model* with variables restricted to the position sets Q_R as *Restricted PP-G2KP Model*. Let z_R be the optimal solution of the *Restricted PP-G2KP Model* and z_U the optimal solution of the *PP-G2KP Model* (with complete position and Q). The following proposition provides an upper bound on the profit in the worst case.

Remark 2. In the worst case, $z_R/z_U \leq 5/6$.

Proof of Remark 2. The result follows from Example 1 when the profit is the same for all of the six items. □

Notice that the *Restricted PP-G2KP Model* allows us to perform a cut on a plate without obtaining a final

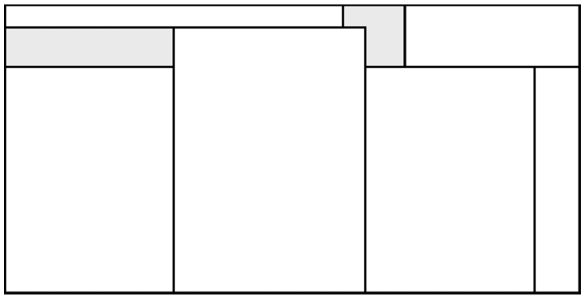

Figure 5 **A Six-Items Packing That Cannot Be Obtained by Only Considering Restricted Positions Q_R**

item: this may happen each time the dimension of an item is the combination of the dimensions of two or more smaller items. As an example, if there are three items with widths $2, 3, 5$, the *Restricted PP-G2KP Model* would allow us to cut at position $q = 5$ and then perform a further cut at position $q = 2$ on the obtained plate. Hence, the width of the strip obtained by cutting at position 5 would not correspond to the width of one of the obtained items. For this reason, the *Restricted PP-G2KP Model* can produce solutions that do not satisfy the definition of restricted guillotine cuts given in Section 1.

To solve the *restricted* G2KP, one possibility is to extend the modeling ideas presented by Silva et al. (2010) for the Guillotine Two- and Three-Stage 2CSP and adapted by Furini and Malaguti (2016) to the 2KP. By removing the limitation on the number of stages, the model in Furini and Malaguti (2016), which cuts *items* from *plates*, can solve the restricted G2KP. Another reduction of the model size can be obtained by removing redundant cuts (denoted as `Redundant-Cut` reduction in the following). Note that, while the `Cut-Position` reduction can reduce the number of plates in the model, the `Redundant-Cut` reduction does not affect the number of plates but only the number of cut positions (and thus the variables of the model).

We say that q is a *trim cut* on plate j when cutting plate j at position q produces a single useful plate j_1 (the second produced plate j_2 is waste).

Remark 3. Given a plate j, one can remove a trim cut at position q with orientation o from $Q(j, o)$, while preserving the optimality of the solution of the *PP-G2KP Model*, in the following cases:

1. Plate j can *only* be obtained through a sequence of two orthogonal trim cuts on a larger plate.
2. Plate j can be obtained from one or more larger plates, but *always* through trim cuts, and at least one of these trim cuts has orientation o.

Proof of Remark 3. In both cases, plate j is obtained anyway through an alternative cut sequence, and thus the corresponding variable can be removed from the model preserving optimality. □

Conditions 1 and 2 of Remark 3 are checked during the enumeration of plates and variables through Procedure 1. To check these conditions, we associate flags to each plate, denoting that a trim cut with orientation h or v was used during the generation of the plate being processed, or during the generation all plates form which it is obtained. Algorithmic details are given in the online supplement (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2016.0710) to this paper.

The computational effect on the number of variables and plates of the reductions discussed in this section are highlighted in Section 4.

2.2. Model Extensions: Cutting Stock Problem and Strip Packing Problem

The modeling ideas of Section 2 can be extended to model other two-dimensional guillotine cutting problems. We present in this section two MIP models for the G2CSP and the GSPP.

By using the same variables defined for the *PP-G2KP Model*, a MIP formulation for the G2CSP reads as follows:

$$\text{(PP-G2CSP):}\quad \min \sum_{o\in O}\sum_{q\in Q(0,\,o)} x^o_{q0} + y_0 \tag{11}$$

$$(2),\ (3)$$

$$y_j \geq d_j \quad j \in \bar{J}, \tag{12}$$

$$x^o_{qj} \geq 0 \text{ integer} \quad j \in J,\ o \in O,\ q \in Q(j, o) \tag{13}$$

$$y_j \geq 0 \text{ integer} \quad j \in \bar{J}, \tag{14}$$

where the objective function (11) minimizes the number of rectangular panels that are used and constraints (12) impose to satisfy the demand associated with the items. The remaining constraints have the same meaning as in the *PP-G2KP Model*.

The PP-G2CSP model can be extended to the case of panels available in p different sizes, by defining p initial panels 0^t and a coefficient c_t, $t = 1, \ldots, p$, specifying the corresponding cost. The objective function is promptly modified to

$$\min \sum_{t=1,\ldots,p} c_t \left(\sum_{o\in O}\sum_{q\in Q(0^t,\,o)} x^o_{q0^t} + y_{0^t} \right). \tag{15}$$

To model the GSPP, we first need an upper bound W on the optimal solution value. We consider the first cut performed on the strip ($j = 0$) to be horizontal (h), with $Q(0, h) = \{1, \ldots, W\}$ and do not define vertical cuts for the strip (i.e., $Q(0, v) = \{\varnothing\}$); in addition, out of the two parts obtained from the first cut, only the bottom one is a finite rectangle that can be used, while the top part is the residual of the infinite strip. The width of the obtained initial rectangle is in $Q(0, h)$ and equals the solution value. We use a variable z denoting the solution value, in addition to the variables defined for the *PP-G2KP Model*. A MIP formulation for the GSPP is then

$$\text{(PP-GSPP):}\quad \min z \tag{16}$$

$$z \geq q x^h_{q0} \quad q \in Q(0, h) \tag{17}$$

$$\sum_{q\in Q(0,\,h)} x^h_{q0} = 1 \tag{18}$$

$$(2),\ (3)$$

$$y_j \geq d_j \quad j \in \bar{J} \tag{19}$$

$$x^o_{qj} \geq 0 \text{ integer} \quad j \in J,\ o \in O,\ q \in Q(j, o) \tag{20}$$

$$y_j \geq 0 \text{ integer} \quad j \in \bar{J}, \tag{21}$$

where the objective function (16) and constraints (17) minimize the (vertical) distance of the first cut from the bottom of the strip and constraint (18) imposes having one horizontal first cut (where q is the width of the cut, with $q \in Q(0, h)$). The remaining constraints have the same meaning as in the previous models.

3. An Effective Solution Procedure for the *PP-G2KP Model*

Tackling directly the *PP-G2KP Model* through a general-purpose MIP solver can be out of reach for medium-size instances due to the large number of variables and constraints; thus, in this section we describe an effective exact solution procedure based on *variable pricing*, aimed at reducing the number of variables and quickening the computational convergence.

The procedure starts by enumerating all of the *PP-G2KP Model* variables by means of Algorithm 1, considering the complete position set Q (see Section 2). Symmetric cut positions are not generated (Section 2.1). Variables are then stored in a *variable pool*. We denote the *PP-G2KP Model* with all of these variables as the *Complete PP-G2KP Model*. The variable pool of the *Complete PP-G2KP Model* can be preprocessed by means of the `Cut-Position` and `Redundant-Cut` reductions, so as to reduce its size.

We perform two subsequent variable pricing procedures executed in cascade. The first one concerns the solution of the linear relaxation of the *PP-G2KP Model*, where variables having positive reduced profit are iteratively selected from the variable pool. The value of the linear programming relaxation of the *PP-G2KP Model*, denoted as *LP* in the following, gives an upper bound on the optimal integer solution value. By exploiting the dual information from the linear programming relaxation, and by computing a feasible solution of value *LB*, a second pricing of the variables can be performed. This second variable pricing allows us to select from the variable pool all of the variables that, by entering in the optimal base with an integer value (e.g., after a branching decision), could potentially improve on the incumbent solution of value *LB*. We denote the *PP-G2KP Model* after the second variables pricing as the *Priced PP-G2KP Model*.

The details on the two variable pricing procedures are given in Section 3.1.

3.1. Variable Pricing Procedures

The linear programming relaxation of the *PP-G2KP Model* can be solved by variable pricing, where we iteratively solve the model with a subset of variables and exploit dual information to add variables with positive reduced profit. Initially, we relax the integrality requirements for the variables (constraints (6) and (7)) to

$$x^o_{qj} \geq 0 \quad j \in J,\ o \in O,\ q \in Q(j, o), \tag{22}$$

$$y_j \geq 0 \quad j \in \bar{J}, \tag{23}$$

and we initialize the resulting linear program (1)–(5) and (22), (23) with all of the y_j, $j \in \bar{J}$ variables and the x^o_{qj}, $j \in J$, $o \in O$, $q \in Q_R(j, o)$ variables; i.e., variables corresponding to cuts in the restricted positions set Q_R (see Section 2). The solution of the resulting linear programming relaxation provides optimal dual variables π_j associated with constraints (2) and (3).

The reduced profit of a variable x^o_{qj}, associated with a cut of plate j producing (up to) two new plates j_1 and j_2, is readily computed as

$$\tilde{p}(x^o_{qj}) = \pi_{j_1} + \pi_{j_2} - \pi_j \tag{24}$$

and can be evaluated for all of the variables of the *Complete PP-G2KP Model*, stored in the pool, in linear time in the size of the pool. The reduced profit $\tilde{p}(x^o_{qj})$ represents the change in the objective function for a unitary increase in the value of the corresponding variable x^o_{qj}.

We optimally solve the linear relaxation of the *PP-G2KP Model* by iteratively adding variables with positive reduced profit—i.e., a subset of

$$\{x^o_{qj},\ j \in J,\ o \in O,\ q \in Q(j, o)\colon \tilde{p}(x^o_{qj}) > 0\},$$

and then reoptimizing the corresponding linear program. When all variables in the variable pool have a nonpositive reduced profit, then the linear programming relaxation of the *Complete PP-G2KP Model* is optimally solved, providing us an upper bound of value *LP*.

Given a feasible solution of value *LB*, the optimal value *LP* of the linear programming relaxation, and the optimal value of the dual variables π^*_j, $j \in J$, we perform a last round of pricing. The *Priced PP-G2KP Model* is defined by including the y variables and the subset of the x variables

$$\{x^o_{qj},\ j \in J,\ o \in O,\ q \in Q(j, o)\colon \lfloor \tilde{p}(x^o_{qj}) + LP \rfloor > LB\}.$$

This includes all of the variables in base plus the variables that, by entering in the current basic solution with value 1 (i.e., at the minimal nonzero integer value), would produce a solution of value $z > LB$.

The effectiveness of the last round of variable pricing in reducing the number of variables of the *Priced PP-G2KP Model* largely depends on the gap between the upper bound value *LP* and the value of a feasible solution *LB*. Heuristic feasible solutions of excellent quality can be computed by solving the *Restricted PP-G2KP Model*, defined by the variables from the restricted position set Q_r (see Section 4.1).

The exact solution procedure is summarized in Algorithm 2.

Procedure 2 (`Solution procedure for the` *PP-G2KP Model*)

1: Set *LB* to the value of a feasible solution to the *PP-G2KP Model*;
2: Generate the variable pool through Algorithm 1;
3: Apply the `Cut-Position` and `Redundant-Cut` reductions to the pool variables;
4: Initialize the model with the variables of the restricted position set Q_R;
5: **repeat**
6: Solve the linear programming relaxation, compute reduced profits, add the variables with positive reduced profit from the pool,
7: **until** variables with positive reduced profit exist;
8: Let *LP* be the optimal solution value of the linear relaxation of the *PP-G2KP Model*;
9: *Final Pricing*: define the *Priced PP-G2KP Model* by including the x variables with reduced profit $\tilde{p}(x)$ such that $\lfloor \tilde{p}(x) + LP \rfloor > LB$, and all of the y variables;
10: Solve the *Priced PP-G2KP Model* with a MIP solver.

4. Computational Experiments

To the best of our knowledge, the modeling framework of guillotine restrictions that we propose in this paper is the first approach based on mixed-integer linear programming that is able to solve benchmark instances to optimality. Thus, the scope of the reported computational experiments is broader than simply comparing the obtained results against previous approaches. Namely, with these experiments we wish to evaluate

- the size and practical solvability of the *Complete PP-G2KP Model* for a set of G2KP benchmark instances by means of a general-purpose MIP solver;
- the effectiveness of the proposed `Cut-Position` and `Redundant-Cut` reductions in removing variables and constraints from the *Complete PP-G2KP Model*;
- the capability to solve the *PP-G2KP Model* by means of the pricing procedure described in Section 3 (*Priced PP-G2KP Model*);
- the quality of the solutions that are obtained by optimally solving the *PP-G2KP Model* by considering the restricted position set Q_R only;
- finally, we wish to discuss the computational performance of our framework with respect to a state-of-the-art combinatorial algorithm for the G2KP (Dolatabadi et al. 2012), and with the only alternative MIP formulation of guillotine restrictions we are aware of, which is described for the GSPP (Ben Messaoud et al. 2008).

We performed all of the computational experiments on one core of a Core2 Quad Q9300 2.50 GHz computer with 8 GB RAM, under the Linux operating system. As linear programming and MIP solver, we used IBM ILOG CPLEX 12.5.

In the computational experiments, we considered two sets of classical two-dimensional instances, listed in Table 1. The first set of 21 instances, for which Dolatabadi et al. (2012) reports computational results as well, is from the OR-Library (Beasley 2016); the second set of 38 instances is from Hifi and Roucairol (2001). Both sets include weighted and unweighted instances, where the profit of each item is given or equals the item area, respectively.

To classify the instances according to the size, we generated the *Complete PP-G2KP Model*, and we grouped the instances into two sets according to the corresponding number of variables. Table 2 reports the features of 33 *large*-size instances, having more than 200,000 variables. Features of the remaining 26 smaller instances, which are easier, are given in the online supplement.

For each instance, the table reports the name (*name*), the optimal solution value (*opt*), the number of different items n, the total number of items $\bar{n}$, and the largest ratio between the length or width of the rectangular panel and an item length or width (ρ). Then, the table reports the number of variables of the *Complete PP-G2KP Model* (*vars*) and the corresponding number of plates (*plates*). We solved the model with the CPLEX MIP solver by allowing one hour of computing time. The table reports the effective computing time (t) and the corresponding percentage optimality gap (*gap*), computed as $100((UB_{MIP} - LB_{MIP})/UB_{MIP})$ (where UB_{MIP} and LB_{MIP} are the lower and upper bound achieved by the MIP solver at the end of the computation).

The size of the *Complete PP-G2KP Model* can be very large; in particular, for the *gcut* instances, the largest

Table 1 List of the Considered G2KP Instances

	Unweighted	Weighted
Beasley (2016)	*gcut1–gcut12*, *wang20*,	*cgcut1–cgcut3*, *okp1–okp5*,
Hifi and Roucairol (2001)	2s, 3s, A1s, A2s, STS2s, STS4s, OF1,OF2,W,CHL1s,CHL2s, A3, A4, A5, CHL5, CHL6, CHL7, CU1, CU2, Hchl3s, Hchl4s, Hchl6s, Hchl7s, Hchl8s.	HH, 2, 3, A1, A2, STS2, STS4, CHL1, CHL2, CW1, CW2, CW3, Hchl2, Hchl9.

Table 2 Large-Size Instances

Instance features					Complete PP-G2KP model			
Name	*opt*	*n*	$\bar{n}$	ρ	*vars*	*plates*	*t*	*gap*
gcut5	195,582	10	10	3.8	250,327	25,336	*t/*	1.4
okp1	27,589	15	50	100.0	255,497	7,947	490.5	0.0
okp3	24,019	30	30	33.3	261,074	8,356	*t/*	8.8
okp4	32,893	33	61	100.0	287,773	9,049	684.4	0.0
okp2	22,503	30	30	100.0	289,825	9,506	*t/*	30.0
CU1	12,330	25	82	5.0	335,415	7,700	1,460.8	0.0
STS4	9,700	20	50	7.1	352,590	7,224	2,476.6	0.0
STS4s	9,770	20	50	7.1	352,590	7,224	2,452.4	0.0
okp5	27,923	29	97	100.0	368,529	9,506	2,118.6	0.0
CW1	6,402	25	67	5.0	410,004	8,407	*t/*	70.7
gcut9	919,476	10	10	3.4	474,360	38,936	2,847.9	0.0
A5	12,985	20	45	10.2	494,455	10,402	*t/*	100.0
Hchl4s	12,006	10	32	8.5	525,902	9,670	*t/*	100.0
Hchl3s	12,215	10	51	8.5	526,403	9,670	*t/*	100.0
CHL1	8,699	30	63	10.2	549,019	10,402	*t/*	100.0
CHL1s	13,099	30	63	10.2	549,019	10,402	*t/*	100.0
CHL6	16,869	30	65	10.8	817,604	13,170	*t/*	100.0
Hchl2	9,954	35	75	7.2	828,561	12,594	*t/*	100.0
CHL7	16,881	35	75	7.2	831,884	12,594	*t/*	100.0
CW2	5,354	35	63	4.9	884,289	14,664	*t/*	100.0
CU2	26,100	35	90	5.0	951,889	16,068	*t/*	100.0
gcut2	59,307	20	20	3.8	1,105,140	28,793	*t/*	100.0
gcut6	236,305	20	20	3.8	1,717,110	74,797	*t/*	100.0
gcut3	60,241	30	30	4.0	1,969,390	32,681	*t/*	100.0
gcut10	903,435	20	20	3.8	2,625,443	138,062	*t/*	100.0
CW3	5,689	40	96	4.6	2,837,862	33,224	*t/*	100.0
gcut4	60,942	50	50	4.0	2,963,221	35,176	*t/*	100.0
Hchl6s	61,040	22	60	7.2	4,730,229	44,593	*t/*	100.0
gcut7	238,974	30	30	3.0	4,908,322	100,800	*t/*	100.0
Hchl7s	63,112	40	90	8.0	5,722,617	46,491	*t/*	100.0
gcut8	245,758	50	50	3.9	16,329,925	136,524	*t/*	100.0
gcut11	955,389	30	30	3.8	32,997,962	400,070	*t/*	100.0
gcut12	970,744	50	50	3.9	66,415,467	489,428	*t/*	100.0

model (*gcut*12) has more than 66 million variables and almost 0.5 million constraints. In addition to a very large size, since no reductions are applied to the *Complete PP-G2KP Model*, it may contain equivalent solutions. The CPLEX MIP solver can solve to optimality 24 of 26 small-size instances and 7 of the 33 large-size instances. For instance *A*5 and all of the larger ones, gaps at time limit are 100%, meaning that no feasible solution is found by the solver.

4.1. Lower Bound (Feasible Solution) Computation

Computing a feasible solution is the first step of the solution procedure for the *PP-G2KP Model*, summarized in Algorithm 2. A possible fast way of computing a feasible solution is by the iterated greedy algorithm proposed by Dolatabadi et al. (2012): given a random order of the items, the algorithm selects the first k items in the ordering whose cumulative profit would improve on the incumbent solution. The algorithm then tries to pack the selected items into the rectangular panel, according to a First Fit Decreasing strategy (see Coffman et al. 1980); in case of success, the incumbent solution is updated (the attempt is not performed if the sum of the areas of the selected items is larger than the area of the panel). In our implementation, we allow one million iterations after the last update of the incumbent solution.

Improved feasible solutions can be obtained by considering the optimal solution of the *PP-G2KP Model* with cut positions in the restricted position set Q_R; i.e., the *Restricted PP-G2KP Model*. Example 1 tells us that the *Restricted PP-G2KP Model* might not contain the optimal solution. However, this occurrence is rare in practice, and in any case, the optimal solution value of the *Restricted PP-G2KP Model* is a valid lower bound *LB* on the optimal solution value.

To show the relative size of the *Restricted PP-G2KP Model*, in Figure 6 we use performance profiles to depict the percentage of variables and plates of the *Restricted PP-G2KP Model* with respect to variables and plates of the *Complete PP-G2KP Model*. In the horizontal axis the figure reports the percentage of variables (resp., plates), and in the vertical axis the

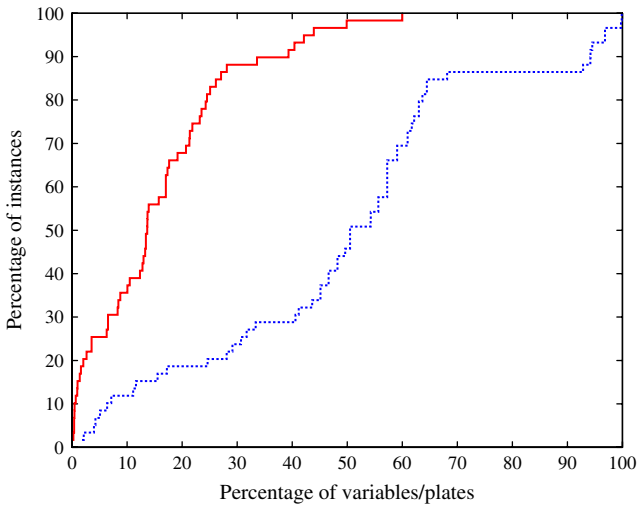

Figure 6 **(Color online) Variables and Plates of the Restricted PP-G2KP Model with Respect to the Complete PP-G2KP Model**

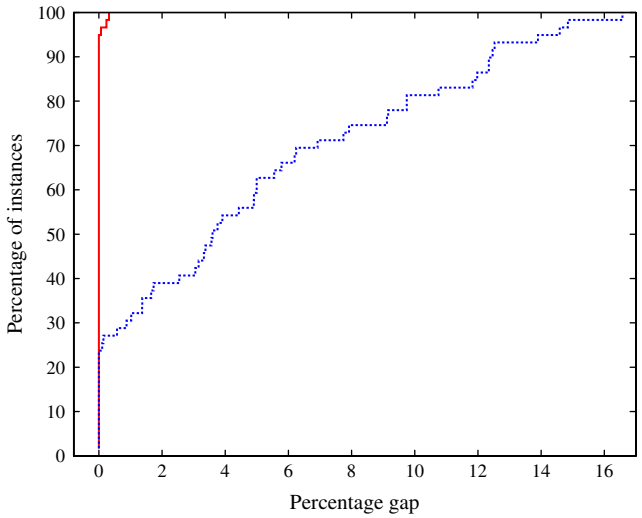

Figure 7 **(Color online) Value of the Greedy Heuristic Solution (Dashed Line) and Value of the Restricted PP-G2KP Model Optimal Solution (Continuous Line), Percentage Gap with Respect to the Best Computed Solution**

percentage of instances for which the *Restricted PP-G2KP Model* has no more than the corresponding percentage of variables (resp., plates). The continuous line is for the variables and the dashed line for the plates. We see that for 80% of the instances, the *Restricted PP-G2KP Model* has at most 25% of the variables and 65% of the plates of the *Complete PP-G2KP Model*.

Despite the large reduction in the number of variables and plates, solving the *Restricted PP-G2KP Model* by using a MIP solver can be very time consuming; hence, we adapted the pricing procedure of Algorithm 2 to this case. We compute an initial feasible solution by means of the iterated greedy algorithm, we solve the linear programming relaxation of the *Restricted PP-G2KP Model*, and we price the model variables, thus defining a *Restricted Priced PP-G2KP Model* containing only the variables that, by entering in base with value 1 or larger, could improve on the incumbent solution. Then, we solve the resulting *Restricted Priced PP-G2KP Model* by means of the CPLEX MIP solver.

In Figure 7 we use performance profiles to represent the gaps between the values of the greedy heuristic solution (LB_g) and the *Restricted PP-G2KP Model* (LB_R) optimal solution, and the value of the best solution we can compute. In the horizontal axis of the figure we report the percentage gap, computed as $100(best - LB)/best$, and in the vertical axis the percentage of instances for which the corresponding or a smaller gap is obtained. The dashed line denotes the greedy heuristic solution and the continuous line the *Restricted PP-G2KP Model*, solved through the pricing procedure.

The quality of the solutions obtained by solving the *Restricted PP-G2KP Model* is very good: for all but three cases it coincides with the best solution we could compute. This suggests that, even though in principle the *Complete PP-G2KP Model* solution can have a profit that is at least 20% larger than the profit of the *Restricted PP-G2KP Model* (see Remark 2), in many cases one could consider solving (exactly or heuristically) the latter, still obtaining very good solutions. Concerning the greedy heuristic, the largest gap with respect to the best computed solution does not exceed 17%.

Concerning the computational effort, the greedy heuristic takes a few seconds, wheras solving the *Restricted PP-G2KP Model* can be time consuming, even if the pricing procedure is used. More details on the computing times are reported next in Table 3.

4.2. Iterative Variable Pricing

Solving the linear programming relaxation of the *PP-G2KP Model* through the variable pricing procedure asks to iteratively add variables with positive reduced profit and then reoptimize the linear program, as explained in Algorithm 2. The actual way variables are added has practical relevance, because it heavily impacts on the computing time and number of iterations of the procedure. On the one hand, one could add at each iteration all of the variables having positive reduced profit, solving the linear programming relaxation in fewer iterations (but eventually solving large LPs). On the other hand, one could add a single variable to the linear program at each time, eventually performing more iterations.

We designed an optimized procedure for iteratively adding variables with positive reduced profit to the linear programming relaxation of the *PP-G2KP*

Table 3 Solution of the Large-Size Instances

				Linear relaxation		Priced PP-G2KP Model	
Name	t_{tot}	$t_{gen(\%)}$	$t_{LB(\%)}$	$t_{LP(\%)}$	gap_{LP}	$t_{(\%)}$	gap
gcut5	558.3	0.1	0.5	1.4	11.7	98.0	0.0
okp1	319.4	0.5	64.2	31.8	11.2	3.5	0.0
okp3	7,429.2	0.0	48.4*	3.1	11.4	48.4*	8.4
okp4	685.5	0.4	32.5	66.6	5.6	0.4	0.0
okp2	7,667.2	0.0	46.9*	6.1	15.9	46.9*	15.9
CU1	7.4	3.7	11.7	83.9	0.1	0.8	0.0
STS4	205.5	0.4	66.8	31.3	5.4	1.6	0.0
STS4s	197.2	0.4	69.1	30.5	5.9	0.1	0.0
okp5	1,276.2	0.3	29.9	69.8	12.6	0.0	0.0
CW1	318.9	0.2	25.2	6.2	17.2	68.4	0.0
gcut9	20.5	4.2	2.3	76.1	1.9	17.4	0.0
A5	700.2	0.2	37.1	17.2	4.1	45.5	0.0
Hchl4s	7,602.5	0.0	47.3*	5.3	9.0	47.3*	9.0
Hchl3s	2,518.0	0.0	15.7	12.7	4.4	71.6	0.0
CHL1	4,577.6	0.1	33.3	7.5	11.9	59.2	0.0
CHL1s	1,520.0	0.2	59.9	22.8	4.1	17.1	0.0
CHL6	2,576.8	0.2	67.5	32.2	3.9	0.1	0.0
Hchl2	4,346.3	0.1	38.6	34.2	10.0	27.0	0.0
CHL7	3,801.3	0.1	51.5	43.3	5.0	5.1	0.0
CW2	3,932.4	0.0	7.6	0.9	11.9	91.5*	5.5
CU2	51.8	2.5	3.8	93.4	1.7	0.3	0.0
gcut2	17.6	4.7	2.6	78.9	1.5	13.9	0.0
gcut6	46.5	8.9	0.9	87.2	0.6	3.1	0.0
gcut3	34.9	4.0	2.0	89.9	1.2	4.1	0.0
gcut10	4.9	2.4	46.4	9.9	6.1	41.3	0.0
CW3	1,264.8	0.3	70.7	22.3	15.4	6.6	0.0
gcut4	181.7	2.1	5.9	71.8	1.1	20.2	0.0
Hchl6s	7,234.1	0.3	49.8*	50.0*	—	—	—
gcut7	117.1	7.3	0.5	72.5	1.3	19.7	0.0
Hchl7s	7,270.4	0.6	49.8*	49.5*	—	—	—
gcut8	666.4	3.6	0.1	95.1	0.2	1.2	0.0
gcut11	4,522.0	2.8	0.2	17.2	2.1	79.7*	2.1
gcut12	2,454.6	8.0	0.1	89.8	0.8	2.1	0.0

Model. The procedure uses two parameters; namely, the maximum number $n_{\max}$ of variables added at each iteration and a threshold $\bar{p}$ for the reduced profit of the variables to be added. At each iteration, the first $n_{\max}$ variables in the pool, having reduced profit larger than $\bar{p}$, are added to the linear programming relaxation of the *PP-G2KP Model*. If no variable with reduced profit larger than $\bar{p}$ exists, the first $n_{\max}$ variables with positive reduced profit are added. Variable number $n_{\max}$ is defined as the number of variables that have positive reduced profit at the first iteration, times a parameter α. Threshold $\bar{p}$ is defined as the sum of the profits of all of the available items, times a parameter β. The values of parameters α and β were experimentally tuned, so as to minimize the cumulative computing time of the variable pricing procedure for the whole instance set. As result of these experiments, we choose the following values of the parameters: $\alpha = 0.20$ and $\beta = 0.25$.

4.3. Model Size and Reductions

In this section, we discuss the size of the various models we work with; namely, the *Complete PP-G2KP Model*, the *Complete PP-G2KP Model* after applying the `Cut-Position` and `Redundant-Cut` reductions, and the *Priced PP-G2KP Model*.

We use performance profiles to represent the percentage of variables of each considered model with respect to the number of variables of the *Complete PP-G2KP Model*. Figure 8 reports on the horizontal axis the percentage of residual variables and on the vertical axis the percentage of instances having no more than the specified percentage of variables. From right to left, the lines in the figure correspond to the application of the `Redundant-Cut` reduction, the `Cut-Position` reduction, and the two reductions together. The reduction strategies appear to be very effective for the *gcut* instances, where in several cases the number of variables is halved, while for the rest of the instance set the percentage of residual variables ranges between 72% and 96%. The leftmost line of Figure 8 depicts the percentage of residual variables of the *Priced PP-G2KP Model* (i.e., after the variable pricing procedures are applied). For approximately 80% of the instances, the percentage of residual variables in the *Priced PP-G2KP Model* is smaller than 40%.

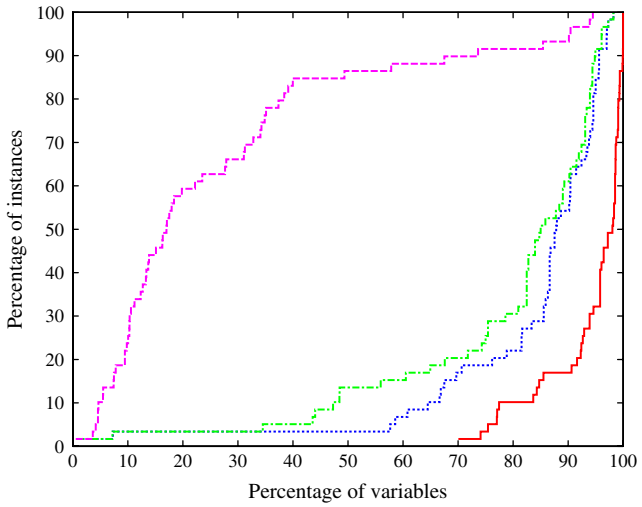

Figure 8 (Color online) Percentage of Residual Variables of the PP-G2KP Model After the Reductions

Notes. From right to left: variables after the `Redundant-Cut` reduction, the `Cut-Position` reduction, the two reductions together. Leftmost line: percentage of variables of the *Priced PP-G2KP Model*.

The benefit of such a reduction in the model size is discussed in the next section.

Concerning the number of plates, as anticipated only the `Cut-Position` reductions can affect it. In practice this happens for a small fraction of the instance set, where there is a large reduction: for 3% of the instances, the percentage of residual plates is no larger than 10%; for 90% of the instances, the number of plates is unchanged.

4.4. Overall Solution Procedure

Finally, Table 3 reports the results obtained by applying the proposed solution procedure to the large-size instances. Detailed results for small-size instances are reported in the online supplement. The table follows the steps of the solution procedure:

- After the instance name, the table reports the overall computing time in seconds spent for the corresponding instance (t_{tot}). The procedure has three time limits of 1 hour, for solving the *Restricted PP-G2KP Model*, for the variable pricing, and for solving the *Priced PP-G2KP Model*, respectively. If a time limit is incurred, the corresponding value is marked with a "$*$" in the table.
- The first step of the solution procedure is to enumerate the problem variables (and plates) to be stored in the variable pool, and to apply the reductions (`Cut-Position` and `Redundant-Cut`). The percentage of computing time (with respect to the total t_{tot}) is reported in column three ($\%t_{\text{gen}}$).
- The solution procedure first asks for a feasible solution, which we compute by solving the *Restricted PP-G2KP Model*. As anticipated, the *Restricted PP-G2KP Model* is tackled by solving the corresponding MIP through CPLEX, after a pricing of the variables has been performed. The pricing asks for a feasible solution, which is obtained through the greedy heuristic. The percentage computing time $\%t_{LB}$ in column four accounts for all of these computations. Note that the aim of this step is producing a good-quality feasible solution; thus, even if the step reaches the time limit, the correctness of the procedure is maintained.
- The next step of the procedure is to solve the linear programming relaxation of the *PP-G2KP Model* through iterative pricing of the pool variables. The table reports, in column five, the percentage of the overall computing time devoted to this task. No useful upper bound is available if this step reaches the time limit. Column six (gap_{LP}) reports the percentage optimality gap at this step of the procedure, computed as $100((UB_{LP}-LB)/UB_{LP})$, where LB is the value of the feasible solution computed in the previous step and UB_{LP} is the upper bound obtained by rounding down the value of the linear programming relaxation.
- After applying a final round of pricing, the procedure defines the *Priced PP-G2KP Model*. The *Priced PP-G2KP Model* is then tackled by the CPLEX MIP solver with a time limit of 1 hour. We report the percentage computing time in column seven and the final percentage optimality gap in column eight—i.e., $100((UB_{best}-LB_{best})/UB_{best})$, where LB_{best} and UB_{best} are the values of the best feasible solution and upper bound computed during the overall procedure, respectively.

We compare the results obtained by applying the proposed solution procedure with the performance of the CPLEX MIP solver in solving the *Complete PP-G2KP Model*. Concerning small-size instances, two instances remain unsolved, but the final optimality gaps are reduced. The computing times are reduced for almost all other instances, in several cases of one order of magnitude (see the online supplement for detailed results). Concerning the 33 large-size instances, by applying the solution procedure based on pricing, we could solve to optimality 26 of them, and for five out of seven unsolved instances, the final optimality gap is largely reduced. For only two instances—namely, *Hchl6s* and *Hchl7s*—the time limit is reached during the solution of the linear programming relaxation, and a valid upper bound is not provided.

To give a graphical representation of the performance of the two methods—namely, solving the *Complete PP-G2KP Model* directly by the CPLEX MIP solver and applying the proposed solution procedure—we report a performance profile in Figure 9. For each instance, we compute a normalized time τ as the ratio of the computing time of the considered solution method over the minimum computing time for solving the instance to optimality. For

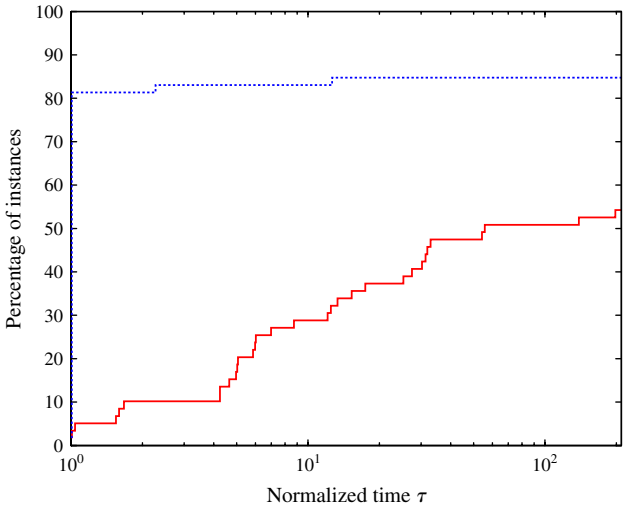

Figure 9 (Color online) Percentage of Solved Instances with Respect to the Normalized Computing Time

Note. Complete PP-G2KP Model (continuous line) and Priced PP-G2KP Model (dashed line).

each value of τ in the horizontal axis, the vertical axis reports the percentage of the instances for which the corresponding method spent at most τ times the computing time of the fastest method. The proposed solution procedure based on pricing (dashed line) has a much better performance than solving the *Complete PP-G2KP Model* directly with the CPLEX MIP solver (continuous line). The performance profile of both methods reaches a horizontal line denoting the percentage of instances that could be solved within the time limit.

We conclude this section by commenting on the quality of the solutions obtained by solving the *Restricted PP-G2KP Model* to optimality: quite often this is the optimal solution; among 50 instances solved to optimality, the solution of the *Restricted PP-G2KP Model* is optimal in 47 cases. For three instances only, the incumbent is improved—namely, A5, CHL1, and CHL7. For the remaining instances, the value of the lower bound *LB* is not improved when solving the *Priced PP-G2KP Model*. In other words, solving the *Priced PP-G2KP Model* is quite often only needed to certify the optimality of the incumbent solution.

4.5. Comparison with State-of-the-Art Approaches

The most recent and effective combinatorial algorithm for the 2GKP is the A1 algorithm by Dolatabadi et al. (2012), which embeds a recursive procedure to enumerate all possible packings of the items within a given rectangular panel. Algorithm A1 needs as input an upper bound *UB* on the maximum profit that can be obtained from the original rectangular panel, which is set to $UB = \min\{U_{kp}, U_{unc}\}$, where U_{kp} is the optimal solution of the associated non-guillotine problem and U_{unc} is the optimal solution value of the associated unconstrained two-dimensional problem (i.e., the problem in which infinite copies of each item are available). Algorithm A1 also needs a lower bound that is computed by running the greedy heuristic described in Section 4.1. We ran an implementation of the algorithm received from the authors of Dolatabadi et al. (2012), which computes internally the *UB*, while the lower bound is obtained by running for 60 seconds (as in Dolatabadi et al. 2012) our implementation of the greedy heuristic.

In our experiments, A1 could solve all 59 instances we considered with an average computing time of 62.6 seconds, with a minimum of 60 and a maximum of 99.21 seconds. It has a strictly better performance (in terms of computing time or optimality of the solution) in 31 cases and a worse performance (in terms of computing time) in 28 cases. Most of the computing time (60 seconds) is spent by the initial greedy heuristic; however, the quality of this information is crucial for the performance of the algorithm. If the initial greedy heuristic is removed and a trivial lower bound of value 0 is used, the algorithm runs into the time limit for 14 of 59 considered instances.

In addition, as reported in Dolatabadi et al. (2012), A1 can also solve many of the APT problems from Alvarez-Valdes et al. (2002). Because of their structure, where very small items need to be packed into large rectangular panels, these instances are intractable for the *PP-G2KP Model*, whose number of variables would be too large in that case.

Concerning the modeling of guillotine restrictions through MIPs, the only alternative framework we are aware of was proposed by Ben Messaoud et al. (2008) and applied to the GSPP. The model was tested on instances having five items, to be packed into strips of length $L = 300$. Items lengths l were randomly generated with uniform distribution in $[\alpha L, \beta L]$, with $(\alpha, \beta) \in \{(0.1, 0.5), (0.3, 0.7)\}$. Three shape classes were considered for the items: wide items, long items, and almost-square items. Wide items, with $w \in [1.2l, 5l]$, were generated with probability A; almost-square items, with $w \in [0.8l, 1.2l]$, where generated with probability B; and long items, with $w \in [0.1l, 0.8l]$, were generated with residual probability. Ben Messaoud et al. (2008) considered the following set of values for A and B: $(A, B) \in \{(1/2, 1/4), (1/4, 1/2), (1/4, 1/4), (1/3, 1/3)\}$.

In Table 4, we report on some experiments we performed by generating instances with these features. We implemented the *PPS-GSPP Model* (16)–(21) and computed the upper bound W on the optimal solution by using a greedy first-fit algorithm for the two-stage version of the problem. In the table, for each combination of the A, B, α, and β parameters, we report the average computing time for solving a set of ten homogeneous instances: t_{FMT} is the time need

Table 4 **Average Computing Times for Solving Homogeneous Classes of Five-Items Strip Packing Instances**

	$[\alpha, \beta]$			
	[0.1, 0.5]		[0.3, 0.7]	
A, B	t_{FMT}	t_{BCE}	t_{FMT}	t_{BCE}
1/2, 1/4	37.0	5,174	13.9	6,218
1/4, 1/2	74.0	5,511	10.4	3,299
1/4, 1/4	50.1	688	22.5	5,993
1/3, 1/3	58.8	80	18.6	5,169

by our approach for solving the instances we generated, while t_{BCE} is the time reported in Ben Messaoud et al. (2008) for solving instances generated with the same features. Our approach is two orders of magnitude faster in solving six of eight problem classes, and still faster on the remaining two classes. This testifies to the better performance of our approach, although Ben Messaoud et al. (2008) used an older version of the CPLEX MIP solver and a slightly slower computer. We are confident that our results could still be improved by refining the value of the initial upper bound W.

4.6. Relevance of Guillotine Cuts

Guillotine cuts potentially allow a better use of the panels and can reduce the waste of raw material when compared with more constrained cut paradigms. General guillotine cuts, being less constrained than restricted guillotine cuts, potentially are the most efficient in this respect.

In our computational experiments, we considered 59 *general* (unrestricted) G2KP from the literature. We tackled the same instances with the model described in Furini and Malaguti (2016), after removing all limitations on the number of stages—i.e., we solved the *restricted* G2KP. For four of 56 instances, we could solve to optimality for the latter problem; the optimal solution of the *general* G2KP has larger profit than the solution of the *restricted* G2KP. The profit increase is between 0.07% and 0.45%. Considering the overall set of 59 instances, instead, there is a profit increase with respect to the corresponding two-stage optimal solution in 48 cases, with an overall increase of 2.05%, and a profit increase with respect to the corresponding three-stage optimal solution in 46 cases, with an overall increase of 1.17%.

In Figure 5 of Section 2.1, we depicted an instance for which the profit of the optimal solution of the *general* G2KP is 20% larger than the profit of the optimal solution obtained by using restricted guillotine cuts, three-stage cuts, or two-stage cuts.

These examples shows that, even though in general the profit increase obtained by allowing general instead of restricted guillotine cuts is limited (at least for the considered set of instances), there exist cases in which it can be quite large. On the other hand, the profit increase obtained by allowing general guillotine cuts instead of two- or three-stage cuts is often substantial for the considered set of instances.

As a final consideration, the practical suitability of general (instead of restricted) guillotine cuts depends on the cutting technology. When in a general pattern the height (or width) of a strip is determined by more than just one element, there might be precision problems with the size of the elements. Problems occur when it is possible to cut off from the raw material a waste part in a precise way but the width of the cut and therefore the size of the remaining part are not known with full certainty in advance. In this case, it is preferable to avoid cuts from which a part is obtained that needs to be further cut into multiple elements without any remaining waste.

5. Conclusions

In this paper, we proposed a way of modeling (general) guillotine cuts in mixed-integer linear programs, without limiting the number of stages or imposing that the cuts be restricted. We concentrated on the guillotine two-dimensional knapsack problem (G2KP), and we discussed extensions of the approach to guillotine two-dimensional cutting stock and guillotine strip packing problems. Because our framework, based on the concepts of *cuts* and residual *plates*, can lead to a very large (pseudopolynomial) number of variables, we proposed effective procedures for generating, managing, and solving the obtained models.

Specifically, we devised a solution procedure based on the computation of a feasible solution of very good quality, which is obtained by restricting the modeling to consider only cuts that coincide with the size of some item. By exploiting dual information, we then perform a pricing of the variables that allows us to define smaller-size models while preserving the optimality of the solutions.

We reported extensive computational experiments, where the approach we proposed solved to optimality several benchmark instances from the literature. Compared with the state-of-the-art combinatorial approach for the G2KP, the proposed approach had a satisfactory performance, and it outperformed the only alternative framework based on mixed-integer programming of which we are aware.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2016.0710.

Acknowledgments

The authors thank José Manuel Valério de Carvalho for stimulating discussions on cutting problems; Mohammad

Dolatabadi, Andrea Lodi, and Michele Monaci for sharing the code of algorithm A1; and Glauber Cintra for sharing the code of the unconstrained version of the G2KP. Thanks are also due to two anonymous referees for their careful reading and useful comments.

References

Alvarez-Valdes R, Parajon A, Tamarit JM (2002) A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Comput. Oper. Res.* 29(7):925–947.

Arbib C, Marinelli F, Rossi F, di Iorio F (2002) Cutting and reuse: An application from automobile component manufacturing. *Oper. Res.* 50(6):923–934.

Beasley JE (2016) OR-Library. Accessed February 1, 2016, http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

Ben Messaoud S, Chu C, Espinouse M (2008) Characterization and modelling of guillotine constraints. *Eur. J. Oper. Res.* 191(1): 112–126.

Bennell JA, Oliveira JF, Wäscher G (2013) Cutting and packing. *Internat. J. Production Econom.* 145(2):449–450.

Boschetti M, Hadjiconstantinou E, Mingozzi A (2002) New upper bounds for the two-dimensional orthogonal non guillotine cutting stock problem. *IMA J. Management Math.* 13(2):95–119.

Burke E, Hellier R, Kendall G, Whitwell G (2006) A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Oper. Res.* 54(3):587–601.

Caprara A, Monaci M (2004) On the two-dimensional knapsack problem. *Oper. Res. Lett.* 32(1):5–14.

Chen Y (2008) A recursive algorithm for constrained two-dimensional cutting problems. *Comput. Optim. Appl.* 41(3): 337–347.

Christofides N, Hadjiconstantinou E (1995) An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts. *Eur. J. Oper. Res.* 83(1):21–38.

Christofides N, Whitlock C (1977) An algorithm for two-dimensional cutting problems. *Oper. Res.* 25(1):30–44.

Cintra G, Wakabayashi Y (2004) Dynamic programming and column generation based approaches for two-dimensional guillotine cutting problems. *Experimental and Efficient Algorithms*, Lecture Notes in Computer Science, Vol. 3059 (Springer, Berlin), 175–190.

Clautiaux F, Jouglet A, Moukrim A (2013) A new graph-theoretical model for the guillotine-cutting problem. *INFORMS J. Comput.* 25(1):72–86.

Coffman EG, Garey MR, Johnson DS, Tarjan RE (1980) Performance bounds for level- oriented two-dimensional packing algorithms. *SIAM J. Comput.* 9(4):808–826.

Cung V-D, Hifi M, Le Cun B (2000) Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *Internat. Trans. Oper. Res.* 7(3):185–210.

Dolatabadi M, Lodi A, Monaci M (2012) Exact algorithms for the two-dimensional guillotine knapsack. *Comput. Oper. Res.* 39(1): 48–53.

Dyckhoff H (1981) A new linear programming approach to the cutting stock problem. *Oper. Res.* 29(6):1092–1104.

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

Fleszar K (2016) An exact algorithm for the two-dimensional stage-unrestricted guillotine cutting/packing decision problem. *INFORMS J. Comput.* 28(4):703–720.

Furini F, Malaguti E (2013) Models for the two-dimensional two-stage cutting stock problem with multiple stock size. *Comput. Oper. Res.* 40(8):1953–1962.

Furini F, Malaguti E (2016) A pseudo-polynomial size formulation for 2-stage 2-dimensional knapsack problems. *Proc. 45th Internat. Conf. Comput. Indust. Engrg.* (Curran Associates, Red Hook, NY), 833–840.

Gilmore PC, Gomory RE (1965) Multistage cutting stock problems of two and more dimensions. *Oper. Res.* 13(1):94–120.

Herz JC (1972) Recursive computational procedure for two-dimensional stock-cutting. *IBM J. Res. Development* 16(5):462–469.

Hifi M (1997) An improvement of Viswanathan and Bagchi's exact algorithm for constrained two-dimensional cutting stock. *Comput. Oper. Res.* 24(8):727–736.

Hifi M (2004) Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems. *J. Combin. Optim.* 8(1):65–84.

Hifi M, Roucairol C (2001) Approximate and exact algorithm for constrained (un)weighted two-dimensional two-staged cutting stock problems. *J. Combin. Optim.* 5(4):465–494.

Iori M, Salazar-González J-J, Vigo D (2007) An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Sci.* 41(2):253–264.

Lodi A, Monaci M (2003) Integer linear programming models for 2-staged two-dimensional knapsack problems. *Math. Program.* 94(2):257–278.

Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: A survey. *Eur. J. Oper. Res.* 141(2):241–252.

Lodi A, Martello S, Monaci M, Cicconetti C, Lenzini L, Mingozzi E, Eklund C, Moilanen J (2011) Efficient two-dimensional packing algorithms for mobile WiMAX. *Management Sci.* 57(12): 2130–2144.

Macedo R, Alves C, Valério de Carvalho JM (2010) Arc-flow model for the two-dimensional guillotine cutting stock problem. *Comput. Oper. Res.* 37(6):991–1001.

Malaguti E, Medina Durán R, Toth P (2014) Approaches to real world two-dimensional cutting problems. *Omega* 47:99–115.

Pisinger D, Sigurd M (2007) Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. Comput.* 19(1):36–51.

Puchinger J, Raidl GR (2007) Models and algorithms for three-stage two-dimensional bin packing. *Eur. J. Oper. Res.* 183(3): 1304–1327.

Russo M, Sforza A, Sterle C (2014) An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems. *Comput. Oper. Res.* 50:97–114.

Silva E, Alvelos F, Valério de Carvalho JM (2010) An integer programming model for two- and three-stage two-dimensional cutting stock problems. *Eur. J. Oper. Res.* 205(3):699–708.

Trick M (2003) A dynamic programming approach for consistency and propagation for knapsack constraints. *Ann. Oper. Res.* 118(1):73–84.

Valério de Carvalho JM (2002) LP models for bin packing and cutting stock problems. *Eur. J. Oper. Res.* 141(2):253–273.

Vanderbeck F (2001) A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. *Management Sci.* 47(6):864–879.

Wang PY (1983) Two algorithms for constrained two-dimensional cutting stock problems. *Oper. Res.* 31(3):573–586.

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