Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 191 (2008) 112–126
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Characterization and modelling of guillotine constraints

Said Ben Messaoud [a], Chengbin Chu [a,*], Marie-Laure Espinouse [b]

[a] *Institut Charles Delaunay (ICD), FRE CNRS 2848, Université de technologie de Troyes, Laboratoire d'optimisation des systèmes industriels (LOSI), 12 rue Marie Curie, BP 2060, 10010 Troyes Cedex, France*
[b] *Laboratoire d'Automatique de Grenoble, ENSIEG, Domaine Universitaire BP 46, 38402 Saint Martin d'Hères Cedex, France*

Received 11 August 2004; accepted 14 August 2007
Available online 6 September 2007

Abstract

This paper focuses on guillotine cuts which often arise in real-life cutting stock problems. In order to construct a solution verifying guillotine constraints, the first step is to know how to determine whether a given cutting pattern is a guillotine pattern. For this purpose, we first characterize guillotine patterns by proving a necessary and sufficient condition. Then, we propose a polynomial algorithm to check this condition. Based on this mathematical characterization of guillotine patterns, we then show that guillotine constraints can be formulated into linear inequalities. The performance of the algorithm to check guillotine cutting patterns is evaluated by means of computational results.

Keywords: Cutting stock; Guillotine pattern; Mixed integer programming

1. Introduction

Cutting stock problems often arise in various industrial sectors. They occur, for instance, in steel, paper, adhesive band, glass, textile production systems. In all these cases, the objective usually is to minimize the waste, which implies the minimization of loss of material; hence improves the productivity of the companies and also the environmental impact. Two-dimensional cutting stock or bin packing problems can be formulated as follows. Given a set of "large" rectangles of standard sizes (stock sheets) and a set of "small" rectangles (items), the aim is to pack all the items on (or cut all items from) the sheets while minimizing the total waste (unused part of the stock sheets). Gilmore and Gomory [9,10] gave first models of the cutting and packing problems. The first classification of cutting and packing problems was proposed by Dyckhoff and Finke [6]. Then Lodi et al. [15] gave a more detailed classification. The reader is referred to Dowsland and Dowsland [5] and Dyckhoff et al. [7] for general surveys, and to Lodi et al. [14] for a recent survey.

* Corresponding author. Tel.: +33 325 715630; fax: +33 325 715649.
E-mail address: chengbin.chu@utt.fr (C. Chu).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2007.08.029

In many industrial applications, additional constraints may have to be taken into account, such as

- Fixed orientation of items: In this case, there is no possibility of rotating items. Such constraints are relevant in wallpaper cutting, or packing articles on a newspaper, for instance.
- Guillotine constraint, also called "edge-to-edge" cut. A guillotine cut applied to a given rectangle begins with a point on an edge of this rectangle and finishes at a point of the opposed edge and is parallel to the remaining edges. We will come back later to this constraint in more detail.

In this paper, we are particularly interested in guillotine constraints. We first give a necessary and sufficient condition for a cutting pattern to be a guillotine pattern. Then, we propose a polynomial algorithm to check this condition for any given pattern. Despite the fact that guillotine constraints were introduced at the very beginning of the cutting stock research, no mathematical definition has been given until now. Our research aims at filling up this gap. The result of this paper can be applied in situations where the raw material to be cut is expensive. In this case, it would be reasonable to check whether a given cutting pattern is feasible with respect to the guillotine constraints before implementation. To the best of our knowledge, existing research only considers patterns that we are sure to be guillotine patterns such as those obtained with shelf algorithms [15]. This restriction, however, may lead to suboptimal solutions since only a part of guillotine patterns are examined. The necessary and sufficient condition also makes it possible to formulate guillotine constraints in a mathematical way. Due to this condition, as we will show in this paper, for the first time in the cutting and packing research, guillotine constraints can be formulated into linear inequalities.

The remainder of this paper is organized as follows. Section 2 introduces definitions and notations that will be used in this paper. Section 3 gives a characterization of a guillotine pattern by proving a necessary and sufficient condition. In Section 4, we report an algorithm capable of determining whether a given cutting pattern is a guillotine pattern. Then we present in Section 5 a new model for guillotine strip packing problem, which can be easily extended to general two-dimensional guillotine cutting stock problems. Finally, we present numerical results in Section 6.

2. Definitions and notations

In this section, we introduce some definitions and notations that will be used in the remainder of the paper. Mathematical manipulation of intervals, specularly merging of intervals, plays an important role in this paper. Given (a,b) and (c,d) two segments or *open* intervals of the same nature (both segments are vertical or horizontal) such that $a \leqslant c$, the merging of these two segments is the union of intervals (a,b) and (c,d) denoted $(a,b) \cup (c,d)$:

$$(a,b) \cup (c,d) = \begin{cases} (a, \max[b,d]) & \text{if } b > c \\ (a,b) \text{ and } (c,d) & \text{otherwise.} \end{cases}$$

If $b > c$, $(a,b) \cup (c,d)$ is made up of a single segment $(a, \max[b,d])$. Otherwise, it is made up of two segments (a,b) and (c,d).

Definition 1. A *cutting pattern* is a layout of items on a "large" rectangle (stock sheet).

Let $R = (O,x,y)$ be a two-dimensional orthonormal space such that O coincides with the bottom left corner of the rectangular stock sheet, and the x-axis and the y-axis coincide with the bottom edge and the left edge of the stock sheet, respectively (Fig. 1). In what follows, n represents the number of items packed in the pattern and w_k and h_k, respectively, denote the width and the height of item k with $w_k > 0$ and $h_k > 0$ $(k = 1,2,\ldots,n)$. A cutting pattern is uniquely defined by a couple (α,β), where $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_n)$, $\beta = (\beta_1,\beta_2,\ldots,\beta_n)$, with α_k and β_k being the coordinates of the bottom left corner of item k, in x-axis and y-axis, respectively, and we say that item k is packed at (α_k,β_k).

The *open* interval (or the horizontal segment) $(\alpha_k, \alpha_k + w_k)$ represents the orthogonal projection of item k on the x-axis, and the *open* interval (or the vertical segment) $(\beta_k, \beta_k + h_k)$ is the orthogonal projection of item k on the y-axis.

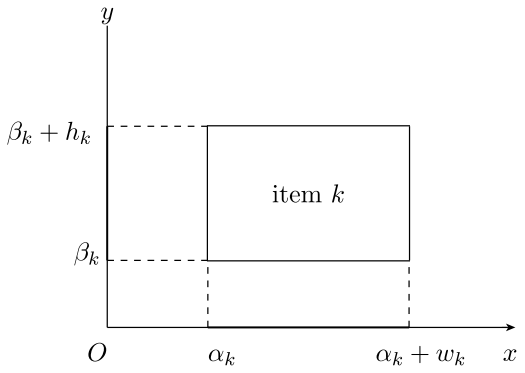


Fig. 1. Location of an item in a two-dimensional orthonormal space.

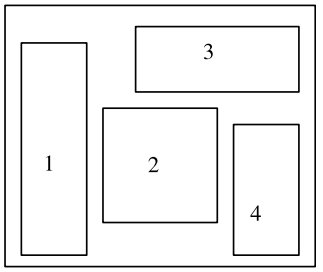


Fig. 2. A guillotine cutting pattern.

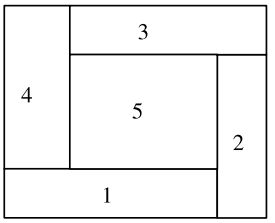


Fig. 3. A non-guillotine cutting pattern.

Every cut carried out on a pattern is orthogonal; i.e., the cuts is either vertical or horizontal and parallel to the edges of the stock sheet. A *guillotine* cut (also called edge-to-edge cut) applied to a given pattern starts on a point of an edge, and ends on a point of the opposed edge. The trajectory of the cut is parallel to the remaining edges. Taking into account this cutting technique, one can distinguish two types of patterns: guillotine patterns (see Fig. 2) and non-guillotine patterns (see Fig. 3). A cutting pattern is a *guillotine pattern* if all the items can be extracted through a sequence of guillotine cuts. In a guillotine cutting pattern containing at least two items, at least one guillotine cut can be performed, dividing the pattern into two subpatterns. It is obvious that such guillotine cuts must not cross any item in the pattern. The same process is repeated on each of the obtained subpatterns containing at least two items until each leftover contains only one item. In what follows, we are interested in a mathematical characterization of guillotine cutting patterns.

3. Characterization of guillotine cutting patterns

Let us consider a cutting pattern composed of n items, such that item k ($k = 1, 2, \ldots, n$) is packed at (α_k, β_k). Without loss if generality, we assume throughout this section that the items are renumbered so that $\alpha_1 \leqslant \alpha_2 \leqslant \cdots \leqslant \alpha_n$. Define a permutation $\sigma : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, n\}$ such that $\beta_{\sigma(1)} \leqslant \beta_{\sigma(2)} \leqslant \cdots \leqslant \beta_{\sigma(n)}$. For any integers i_1, i_2, j_1 and j_2 such that $1 \leqslant i_1 < i_2 \leqslant n$, $1 \leqslant j_1 < j_2 \leqslant n$, we define the following set of items:

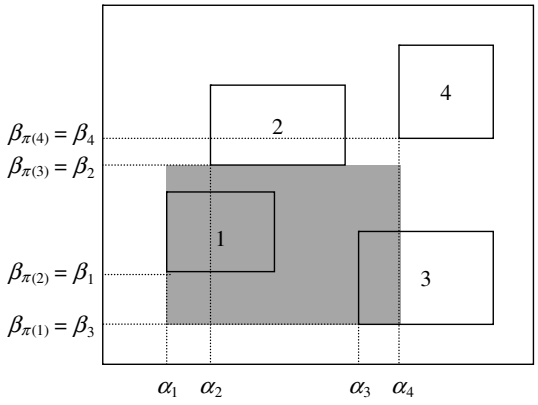


Fig. 4. Definition of $E(1,4,1,3)$.

$E(i_1,i_2,j_1,j_2) = \{i_1,\ldots,i_2\} \cap \{\sigma(j_1),\ldots,\sigma(j_2)\}$, which is the set of items whose bottom-left corners fall into the area delimited by the two vertical axes passing by α_{i_1} and α_{i_2} and the two horizontal axes passing by $\beta_{\sigma(j_1)}$ and $\beta_{\sigma(j_2)}$.

To clarify this notation, we consider the example in Fig. 4. This is a 4-item pattern. The items are renumbered so that $\alpha_1 \leqslant \alpha_2 \leqslant \alpha_3 \leqslant \alpha_4$. But β_1, β_2, β_3 and β_4 do not appear in nondecreasing order. We define a permutation $\sigma : \{1,2,3,4\} \mapsto \{1,2,3,4\}$ such that $\beta_{\sigma(1)} \leqslant \beta_{\sigma(2)} \leqslant \beta_{\sigma(3)} \leqslant \beta_{\sigma(4)}$. Since $\beta_3 \leqslant \beta_1 \leqslant \beta_2 \leqslant \beta_4$, we have $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$ and $\sigma(4) = 4$. Therefore, $E(1,4,1,3) = \{1,2,3,4\} \cap \{\sigma(1),\sigma(2),\sigma(3)\} = \{1,2,3,4\} \cap \{3,1,2\} = \{1,2,3\}$. Indeed, all items 1, 2 and 3 have their bottom-left corner in the gray area delimited by the two vertical axes passing by α_1 and α_4 and the two horizontal axes passing by $\beta_{\sigma(1)} = \beta_3$ and $\beta_{\sigma(3)} = \beta_2$.

By the definition, for any $k \in E(i_1,i_2,j_1,j_2)$, we have $\alpha_{i_1} \leqslant \alpha_k \leqslant \alpha_{i_2}$ and $\beta_{\sigma(j_1)} \leqslant \beta_k \leqslant \beta_{\sigma(j_2)}$.

Theorem 1. *A given cutting pattern with n items is a guillotine pattern if and only if, for any integers i_1, i_2, j_1 and j_2 such that $1 \leqslant i_1 < i_2 \leqslant n$, $1 \leqslant j_1 < j_2 \leqslant n$, at least one of the following conditions is fulfilled*:

1. *$E(i_1,i_2,j_1,j_2)$ contains at most one element (item);*
2. *$\bigcup_{k\in E(i_1,i_2,j_1,j_2)}(\alpha_k,\alpha_k+w_k)$ is made up of at least two disjoint intervals; i.e.*

$$\bigcup_{k\in E(i_1,i_2,j_1,j_2)}(\alpha_k,\alpha_k+w_k) \neq \left(\min_{k\in E(i_1,i_2,j_1,j_2)}\alpha_k, \max_{k\in E(i_1,i_2,j_1,j_2)}(\alpha_k+w_k)\right);$$

3. *$\bigcup_{k\in E(i_1,i_2,j_1,j_2)}(\beta_k,\beta_k+h_k)$ is made up of at least two disjoint intervals; i.e.*

$$\bigcup_{k\in E(i_1,i_2,j_1,j_2)}(\beta_k,\beta_k+h_k) \neq \left(\min_{k\in E(i_1,i_2,j_1,j_2)}\beta_k, \max_{k\in E(i_1,i_2,j_1,j_2)}(\beta_k+h_k)\right).$$

Proof. It is obvious that this condition is necessary, since if neither of these conditions holds for some $E(i_1,i_2,j_1,j_2)$, the items of $E(i_1,i_2,j_1,j_2)$ will not be able to be obtained through guillotine cuts. In fact, in this case, for any y such that $\min_{k\in E(i_1,i_2,j_1,j_2)}\beta_k < y < \max_{k\in E(i_1,i_2,j_1,j_2)}(\beta_k+h_k)$, there is an item $l \in E(i_1,i_2,j_1,j_2)$ such that

$$\min_{k\in E(i_1,i_2,j_1,j_2)}\beta_k \leqslant \beta_l < y < \beta_l + h_l \leqslant \max_{k\in E(i_1,i_2,j_1,j_2)}(\beta_k+h_k).$$

A horizontal cut at y will destroy item l. Similarly, no vertical cut is possible, either.

We now show that this condition is a sufficient condition. We will show that if, for any integers i_1, i_2, j_1 and j_2 such that $1 \leqslant i_1 < i_2 \leqslant n$, $1 \leqslant j_1 < j_2 \leqslant n$, at least one of the conditions 1, 2 and 3 holds, then the cutting pattern is a guillotine pattern. We can prove it by induction.

The condition is obviously sufficient if $n = 1$.

Assume now that there is an integer K such that $K \geqslant 1$ and the condition is sufficient for any n such that $1 \leqslant n \leqslant K$. We show that the condition is also sufficient for $n = K + 1$. We can reorder the items so that

$\alpha_1 \leqslant \alpha_2 \leqslant \cdots \leqslant \alpha_{K+1}$ and we define a permutation $\sigma:\{1,2,\ldots,K+1\} \mapsto \{1,2,\ldots,K+1\}$ such that $\beta_{\sigma(1)} \leqslant \beta_{\sigma(2)} \leqslant \cdots \leqslant \beta_{\sigma(K+1)}$. According to the assumptions, for any integers i_1, i_2, j_1 and j_2 such that $1 \leqslant i_1 < i_2 \leqslant K+1$, $1 \leqslant j_1 < j_2 \leqslant K+1$, at least one of the conditions 1, 2 and 3 holds. In particular, at least one of these conditions holds when $i_1 = 1$, $i_2 = K+1$, $j_1 = 1$ and $j_2 = K+1$. By the definition, we have $E(1,K+1,1,K+1) = \{1,2,3,\ldots,K+1\}$. Since $K \geqslant 1$, $E(1,K+1,1,K+1)$ contains more than one item. Therefore, either condition 2 or condition 3 must hold. By symmetry, it is sufficient to consider the case where condition 2 holds. Due to the fact that $\cup_{k\in E(1,K+1,1,K+1)}(\alpha_k, \alpha_k + w_k)$ is made up of at least two disjoint intervals, any vertical cut between two adjacent disjoint intervals is a guillotine cut. Such a cut separates the pattern into two subpatterns containing at most $K \geqslant 1$ items each. According to the assumption, at least one of the three conditions holds for each of these subpatterns. According to the induction assumption with at most K items, both of these subpatterns are guillotine patterns. As a consequence, the initial pattern with $K+1$ items is also a guillotine pattern. □

The following corollary is a consequence of this theorem. It shows that as soon as a given cutting pattern is known to be a guillotine pattern, we are sure that there is no overlapping between items.

Corollary 1. *In a guillotine pattern, there is no overlapping between any pair of items. In other words, in a guillotine pattern, for any couple of items k_1 and k_2 such that $1 \leqslant k_1 < n$, $k_1 < k_2 \leqslant n$, at least one of the following three conditions holds*:

1. $\alpha_{k_1} + w_{k_1} \leqslant \alpha_{k_2}$;
2. $\beta_{k_2} \geqslant \beta_{k_1} + h_{k_1}$;
3. $\beta_{k_1} \geqslant \beta_{k_2} + h_{k_2}$.

Proof. We show that if there are some items k_1 and k_2 such that $1 \leqslant k_1 < k_2 \leqslant n$, $\alpha_{k_1} + w_{k_1} > \alpha_{k_2}$, $\beta_{k_2} < \beta_{k_1} + h_{k_1}$ and $\beta_{k_1} < \beta_{k_2} + h_{k_2}$; i.e., there is overlapping in the given cutting pattern, then this cutting pattern is not a guillotine pattern. Let $l_1 = \min[\sigma^{-1}(k_1), \sigma^{-1}(k_2)]$ and $l_2 = \max[\sigma^{-1}(k_1), \sigma^{-1}(k_2)]$. It is obvious that $1 \leqslant l_1 < l_2 \leqslant n$. Consider the set of items $E(k_1,k_2,l_1,l_2)$. It is obvious that $\{k_1,k_2\} \subseteq E(k_1,k_2,l_1,l_2)$ which implies that $E(k_1,k_2,l_1,l_2)$ contains at least two elements; i.e., condition 1 of Theorem 1 does not hold. The fact that $\alpha_{k_1} + w_{k_1} > \alpha_{k_2}$ implies that $\alpha_{k_1} + w_{k_1} > \alpha_k$ for any $k \in E(k_1,k_2,l_1,l_2)$. As a consequence, $\bigcup_{k\in E(k_1,k_2,l_1,l_2)}(\alpha_k, \alpha_k + w_k)$ is made up of a single interval; i.e., condition 2 of Theorem 1 does not hold. Similarly, the fact that $\beta_{k_2} < \beta_{k_1} + h_{k_1}$ and $\beta_{k_1} < \beta_{k_2} + h_{k_2}$implies that $\bigcup_{k\in E(k_1,k_2,l_1,l_2)}(\beta_k, \beta_k + h_k)$ is made up of a single interval; i.e., condition 3 of Theorem 1 does not hold. Since neither of the conditions of Theorem 1 holds, the given cutting pattern is not a guillotine pattern. □

4. BMCE: An algorithm for guillotine pattern checking

In this section, we present an algorithm called BMCE based on the characterization of guillotine patterns proved in the previous section to determine whether a given pattern is a guillotine pattern. The originality of the algorithm is in the use of intervals and its polynomial complexity.

At each iteration, the algorithm divides a pattern (the initial one or a subpattern obtained from previous iterations) that contains at least two items into disjoint subpatterns, by performing guillotine cuts, if any. This process is repeated until either all obtained subpatterns contain one item or it is no longer possible to perform guillotine cuts on at least one pattern containing at least two items. In the first case, we can affirm that the initial pattern is a guillotine pattern while it is not a guillotine pattern in the second case.

The division of a given pattern containing at least two items into two subpatterns is composed of two steps: Construction of horizontal and vertical intervals (by projection on x-axis and y-axis, respectively), and merging of intervals of the same nature (all horizontal intervals or all vertical intervals). To be more specific, for a given (sub)pattern P containing at least two items:

1. We start by determining all horizontal segments of the items in the pattern. This can be done by going through the list of items and inserting all the horizontal segments in a list *HorSeg*(P) (for horizontal

segments) in nondecreasing order of their lower bounds. This requires $O(n\log n)$ operations. In the same manner, we determine the list of vertical segments $VerSeg(P)$.
2. Then, merge the sorted list of $HorSeg(P)$ (respectively, $VerSeg(P)$) and put the result into a list called $HorSegF(P)$ (horizontal segments after merging), and respectively, $VerSegF(P)$. It requires $O(n)$ time to merge a list of segments already sorted in nondecreasing order of the lower bounds.

The knowledge of $HorSegF(P)$ and $VerSegF(P)$ after merging enables us to enumerate all possible (horizontal or vertical) guillotine cuts for P. Indeed, if $HorSegF(P)$ is reduced to a single segment, then no vertical cut can be carried out. Otherwise, it is possible to perform vertical guillotine cuts between adjacent segments to divide the pattern into disjoint subpatterns. The same reasoning is valid for $VerSegF(P)$.

To be more precise, the algorithm can be described with the following pseudo-code, where C is the initial pattern, Φ and Ψ, respectively, are the set of (sub)patterns containing at least two items and the set of (sub)patterns containing exactly one item. As a result, Ψ can also be seen as the set of items already extracted.

1. Construct $HorSeg(C)$, $VerSeg(C)$ where the segments are ordered in nondecreasing order of the lower bounds.
2. $\Phi := \{C\}$, $\Psi := \emptyset$.
3. While $\Phi \neq \emptyset$ do
   (a) Select an arbitrary pattern $P \in \Phi$ and remove it from Φ: $\Phi := \Phi - \{P\}$.
   (b) Construct $HorSegF(P)$ and $VerSegF(P)$.
   (c) If $HorSegF(P)$ is made of at least two disjoint segments then divide P into as many subpatterns as segments in $HorSegF(P)$. The subpatterns with at least two items are included in Φ and the other subpatterns are included in Ψ. Construct $HorSeg(Q)$ and $VerSeg(Q)$ for each obtained subpattern Q with at least two items.
   Otherwise,
   if $VerSegF(P)$ is made of at least two disjoint segments then divide P into as many subpatterns as segments in $VerSegF(P)$. The subpatterns with at least two items are included in Φ and the other subpatterns are included in Ψ. Construct $HorSeg(Q)$ and $VerSeg(Q)$ for each obtained subpattern Q with at least two items.
   Otherwise, the pattern is not a guillotine pattern and Stop.
4. Endwhile

The following theorem establishes the computational complexity of algorithm BMCE.

Theorem 2. *The complexity of algorithm BMCE is* $O(n^2)$.

Proof. It is obvious that Step 1 requires $O(n\log n)$ time.

We now show that Step 3 requires at most $2(n+2)(n-1)$ elementary operations. For this purpose, we define the following function:

$$S(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2(n+2)(n-1) & \text{otherwise.} \end{cases}$$

We show that $S(n)$ is an upper bound of the number of elementary operations required in Step 3 for a pattern with n items.

This can be shown by induction. It is obvious that $S(n)$ is an upper bound when $n \in \{0,1\}$. Assume now that there is a K such that $K \geqslant 1$ and the number of operations required for a K-item pattern is at most $S(K)$. Consider now a pattern C with $K+1$ items.

For this pattern, it requires at most $2(K+1)$ operations to obtain $HorSegF(C)$ and $VerSegF(C)$.

If $HorSegF(C)$ and $VerSegF(C)$ are both composed of a single segment, we can conclude that pattern C is not a guillotine pattern and the algorithm stops. The overall number of operations in Step 3 will be $2(K+1) \leqslant 2(K+3)K = S(K+1)$.

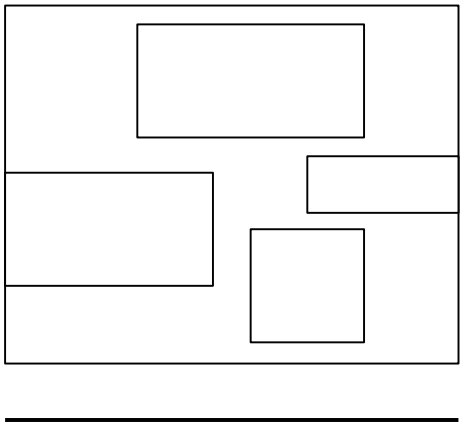


Fig. 5. No vertical guillotine cut possible.

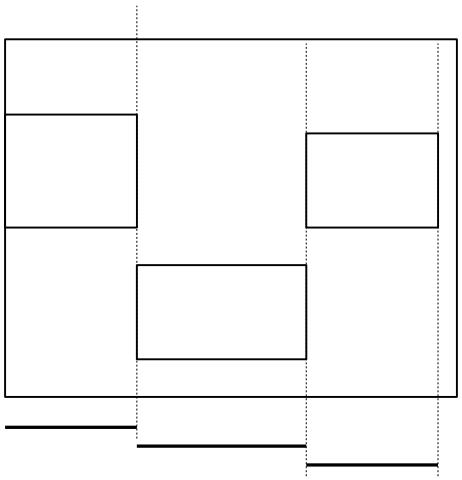


Fig. 6. Three possible vertical cuts.

If either of $HorSegF(C)$ and $VerSegF(C)$ is composed of more than one segment, pattern C is divided into at least two actual subpatterns. Without loss of generality, we can always assume that the number of subpatterns is $K+1$ and denoted as $C_1, C_2, \ldots, C_{K+1}$, where some subpatterns may contain no item. Let n_k be the number of items in subpattern C_k ($k = 1, \ldots, K+1$). We must have $\sum_{k=1}^{K+1} n_k = K+1$. We can assume that without loss of generality that $n_1 \geqslant n_2 \geqslant \cdots \geqslant n_{K+1} \geqslant 0$. Since there are at least two actual patterns, we have $1 \leqslant n_1 \leqslant K$. By using an appropriate data structure (e.g., chain structure) for $HorSeg(P)$ and $VerSeg(P)$ for any (sub)pattern P (note that the segments are ordered in nondecreasing order of lower bounds), the computation of $HorSeg(C_k)$ and $VerSeg(C_k)$ requires $2n_k$ operations. In other words, the computation of all $HorSeg(C_k)$'s and $VerSeg(C_k)$'s requires $2(K+1)$ operations. Subpattern C_k still requires at most $S(n_k)$ operations in subsequent iterations. Therefore, the total number of operations required is at most $4(K+1) + \sum_{k=1}^{K+1} S(n_k)$. Considering the fact that $1 \leqslant n_1 \leqslant K$, $4(K+1) + \sum_{k=1}^{K+1} S(n_k)$ gets its maximum when $n_1 = K,\ n_2 = 1,\ n_3 = \cdots = n_{K+1} = 0$. Therefore, the total number of operations is at most $4(K+1) + S(K) \leqslant 2(K+3)K = S(K+1)$.

As a consequence, in either cases, $S(K+1)$ is an upper bound of the number of elementary operations required in Step 3 for a pattern with $K+1$ items. This ends up the proof that $S(n)$ is an upper bound of the number of elementary operations required in Step 3 for a pattern with n patterns, for any nonnegative n.

Therefore, the complexity of the algorithm is $O(n\log n + 2(n+2)(n-1)) = O(n^2)$. □

Example 1. We will see through an example when a vertical guillotine cut can be performed. For each of the two patterns shown in Figs. 5 and 6, respectively, we will determine the segments resulting from the merging of all the edges of the items.

In the first case (Fig. 5), we obtain one single horizontal segment after merging, therefore we cannot carry out any vertical guillotine cut. In the second case (Fig. 6), three horizontal segments result from merging, which gives us the possibility to cut at the extremities of these segments.

Note that if we apply this algorithm to the pattern given in Fig. 3, we obtain one single horizontal segment, and one single vertical segment. Hence the given pattern is not a guillotine pattern.

We can also modify this algorithm so that it can detect and locate all the non-guillotine zones in the pattern. It can be done in the following way. Each time it is impossible to carry out any (vertical or horizontal) guillotine cut on a rectangle, we insert the rectangle into a list called NGRec (Non-Guillotine Rectangle).

5. Mixed integer programming formulation of guillotine constraints

In previous sections, we proposed a mathematical characterization of the guillotine constraint. In this section, we show for the first time in the cutting stock research that guillotine constraints can be formulated into linear inequalities. Despite the fact that guillotine constraints were introduced at the very beginning of cutting stock studies, they have never been expressed in mathematical way and *a fortiori* with linear inequalities. The necessary and sufficient condition characterizing guillotine patterns (see Theorem 1) makes it possible. For the sake of simplification, we only consider a strip packing problem where the raw material is available in the form of rolls or strip. The result can be easily extended to other guillotine cutting stock problems. This is the problem of cutting a set of rectangular items from a single large rectangle (strip) having a fixed width W and infinite height. Two additional constraints are to be considered: Rotation of items is not allowed and only guillotine patterns are considered. The objective is to cut off a set of rectangular items while minimizing the total height.

We first give a brief survey on existing models and then present our new model taking into account guillotine constraints.

5.1. Existing models

Gilmore and Gomory [9,10] gave the first mathematical model of the one-dimensional cutting and packing problems. In Gilmore and Gomory [11,12] they studied more closely the knapsack problem when it is applied to one and two-dimensional stock cutting.

Biro and Boros [4] characterized the non-guillotine pattern using graph theory. Beasley [1] studied the two-dimensional stock cutting problem that was not restricted to guillotine cuts and the objective was to maximize the value of the items. He gave a linear program to determine discrete coordinates at which items may be located. A similar model was introduced by Hadjiconstantinou and Christofides [13]. Scheithauer and Terno [17] proposed models for packing convex and nonconvex polygons.

Fekete and Schepers [8] used the graph theory to determine a feasible packing without overloapping between items. Note that they did not consider guillotine constraints. Lodi et al. [14,16] presented a model that dealt with a pattern formed by shelves. This latter model explicitly considered guillotine constraints but only a part of guillotine patterns are considered. Recently Beasley [2] presented a new formulation of the non-guillotine cutting. A good review of existing models is available in Lodi et al. [14].

5.2. A new model

We present in this section a new mixed integer programming model for the guillotine strip packing problem. This new model is based on Theorem 1 characterizing guillotine patterns and uses coordinates at which items may be located.

Assume that n items have to be packed on a strip having width W and infinite height. For a pattern where the bottom-left corner of each item k ($k = 1,2,\ldots,n$) is located at (α_k, β_k), we can always obtain two sorted series $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ by sorting in nondecreasing order the series $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and

$(\beta_1, \beta_2, \ldots, \beta_n)$, respectively. As a consequence, we have $0 \leqslant x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n \leqslant W$ and $0 \leqslant y_1 \leqslant y_2 \leqslant \cdots \leqslant y_n$. We can find two permutations $\phi : \{1,2,\cdots,n\} \mapsto \{1,2,\ldots,n\}$ and $\psi : \{1,2,\ldots,n\} \mapsto \{1,2,\ldots,n\}$ such that $\alpha_k = x_{\phi(k)}$ and $\beta_k = y_{\psi(k)}$ for any k such that $1 \leqslant k \leqslant n$.

We say that item k is packed at $(x_{\phi(k)},y_{\psi(k)})$; or simply at $(\phi(k),\psi(k))$. We say that item k is packed between (i_1,j_1) and (i_2,j_2) with $1 \leqslant i_1 \leqslant i_2 \leqslant n$ and $1 \leqslant j_1 \leqslant j_2 \leqslant n$, when there are an i and a j such that $i_1 \leqslant i \leqslant i_2$, $j_1 \leqslant j \leqslant j_2$ and item k is packed at (i,j).

Reversely, if we know two sorted series $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ such that $0 \leqslant x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n \leqslant W$ and $0 \leqslant y_1 \leqslant y_2 \leqslant \cdots \leqslant y_n$, and two permutations $\phi:\{1,2,\ldots,n\} \mapsto \{1,2,\ldots,n\}$ and $\psi:\{1,2,\ldots,n\} \mapsto \{1,2,\ldots,n\}$, we can find the corresponding pattern. As a consequence, a pattern can be defined by $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ and permutations ϕ and ψ. We might use these variables as decision variables. But in this way, the coordinates of item k in a pattern will be $(x_{\phi(k)}, y_{\psi(k)})$ with decision variables as subscripts and the obtained model will not be linear.

To obtain a linear model, we use the following binary variables to characterize the location of the items on the strip:

$$z_{i,j,k} = \begin{cases} 1 & \text{if item } k \text{ is packed at } (i,j) \\ 0 & \text{otherwise.} \end{cases}$$

The following intermediate decision variables are also necessary:

$$u_{i,j,i'} = \begin{cases} 1 & \text{if the item packed at } (i,j), \text{if any, does not exceed(horizontally) } x_{i'} \text{ with } i' > i \\ 0 & \text{otherwise} \end{cases}$$

$$v_{i,j,j'} = \begin{cases} 1 & \text{if the item packed at } (i,j), \text{if any, does not exceed(vertically)} y_{j'} \text{ with } j' > j \\ 0 & \text{otherwise.} \end{cases}$$

The next three sets of binary variables are needed to ensure guillotine constraints:

$$p_{i_1,i',j_1,j_2} = \begin{cases} 1 & \text{if no item packed between } (i_1,j_1) \text{ and } (i'-1,j_2) \text{ exceeds } x_{i'} \text{ (consequently, a} \\ & \text{vertical cut at } x_{i'} \text{ does not cross any item packed between } (i_1,j_1) \text{ and} \\ & (i'-1,j_2)), i' > i_1 \\ 0 & \text{otherwise} \end{cases}$$

$$q_{i_1,i_2,j_1,j'} = \begin{cases} 1 & \text{if no item packed between } (i_1,j_1) \text{ and } (i_2,j'-1) \text{ exceeds } y_{j'} \text{ (consequently, a} \\ & \text{horizontal cut at } y_{j'} \text{ does not cross any item packed between } (i_1,j_1) \text{ and} \\ & (i_2,j'-1)), j' > j_1 \\ 0 & \text{otherwise} \end{cases}$$

$$d_{i_1,i_2,j_1,j_2} = \begin{cases} 1 & \text{if there is at most one item is packed between } (i_1,j_1) \text{ and } (i_2,j_2) \\ 0 & \text{otherwise.} \end{cases}$$

Our complete formulation of the guillotine strip packing is as follows:

$$\text{Minimize} \quad H, \tag{1}$$

$$\text{subject to} \quad 0 \leqslant x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n, \tag{2}$$

$$0 \leqslant y_1 \leqslant y_2 \leqslant \cdots \leqslant y_n, \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} z_{i,j,k} = 1 \quad \forall k, \tag{4}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} z_{i,j,k} = 1 \quad \forall j, \tag{5}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} z_{i,j,k} = 1 \quad \forall i, \tag{6}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \geqslant (u_{i,j,i'} - 1)W \quad \forall i, \forall j, \forall i' > i, \tag{7}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \leqslant u_{i,j,i'} W \quad \forall i, \forall j, \forall i' > i, \tag{8}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \geqslant (v_{i,j,j'} - 1)\overline{H} \quad \forall i, \forall j, \forall j' > j, \tag{9}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \leqslant v_{i,j,j'} \overline{H} \quad \forall i, \forall j, \forall j' > j, \tag{10}$$

$$(1 - d_{i_1,i_2,j_1,j_2})n \geqslant \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i,j,k} - 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \ \forall j_2 > j_1, \tag{11}$$

$$p_{i_1,i',j_1,j_2} \leqslant \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i',j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \ \forall i' > i_1, \tag{12}$$

$$p_{i_1,i',j_1,j_2} \leqslant \sum_{i=i_1}^{i'-1} \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \ \forall i' > i_1, \tag{13}$$

$$(i' - i_1)(j_2 - j_1 + 1)p_{i_1,i',j_1,j_2} \leqslant \sum_{i=i_1}^{i'-1} \sum_{j=j_1}^{j_2} u_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \ \forall i' > i_1, \tag{14}$$

$$q_{i_1,i_2,j_1,j'} \leqslant \sum_{i=i_1}^{i_2} \sum_{k=1}^{n} z_{i,j',k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \ \forall j' > j_1, \tag{15}$$

$$q_{i_1,i_2,j_1,j'} \leqslant \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \ \forall j' > j_1, \tag{16}$$

$$(j' - j_1)(i_2 - i_1 + 1)q_{i_1,i_2,j_1,j'} \leqslant \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} v_{i,j,j'} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \ \forall j' > j_1, \tag{17}$$

$$d_{i_1,i_2,j_1,j_2} + \sum_{i'=i_1+1}^{i_2} p_{i_1,i',j_1,j_2} + \sum_{j'=j_1+1}^{j_2} q_{i_1,i_2,j_1,j'} \geqslant 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \ \forall j_2 > j_1, \tag{18}$$

$$W \geqslant x_i + \sum_{j=1}^{n} \sum_{k=1}^{n} w_k z_{i,j,k} \quad \forall i, \tag{19}$$

$$H \geqslant y_j + \sum_{i=1}^{n} \sum_{k=1}^{n} h_k z_{i,j,k} \quad \forall j, \tag{20}$$

$$z_{i,j,k} \in \{0,1\} \quad \forall i, \forall j, \forall k, \tag{21}$$

$$u_{i,j,i'} \in \{0,1\} \quad \forall i, \forall j, \forall i' > i, \tag{22}$$

$$v_{i,j,j'} \in \{0,1\} \quad \forall i, \forall j, \forall j' > j, \tag{23}$$

$$d_{i_1,i_2,j_1,j_2}, p_{i_1,i_2,j_1,j_2}, q_{i_1,i_2,j_1,j_2} \in \{0,1\} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{24}$$

where

$$\overline{H} = \sum_{k=1}^{n} h_k,$$

which is an upper bound of the objective function value. For the model to be more effective, one can also take as upper bound the strip height in a solution obtained with a heuristic such as the one we proposed (see [3]).

In the formulation above, constraints (4)–(6) ensure that each horizontal or vertical position is occupied by exactly one item and each item is packed exactly once.

Constraints (7) and (8) ensure the coherence of the definition of $u_{i,j,i'}$. Indeed, if $x_{i'} > x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (i.e., if the item packed at position (i,j), if any, does not exceed $x_{i'}$), we must have $u_{i,j,i'} = 1$. Constraint (8) precisely

requires that $u_{i,j,i'} = 1$ whereas constraint (7) is redundant (already satisfied). On the other hand, if $x_{i'} < x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (i.e., if an item is packed at position (i,j) and furthermore this item exceeds $x_{i'}$), then constraint (7) requires $u_{i,j,i'} = 0$ whereas constraint (8) is redundant. Note that $u_{i,j,i'}$ may indifferently take value 0 or value 1 if $x_{i'} - x_i = \sum_{k=1}^{n} w_k z_{i,j,k}$. In such a case, constraints (14) and (18) suggest $u_{i,j,i'}$ to be equal to 1, which is consistent with the definition.

By symmetry, constraints (9) and (10) ensure the coherence of the definition of $v_{i,j,j'}$.

Constraint (11) ensures the coherence of the definition of d_{i_1,i_2,j_1,j_2}. Indeed, if $\sum_{i=i_1}^{i_2}\sum_{j=j_1}^{j_2}\sum_{i=1}^{n} z_{i,j,k} \geqslant 2$ (at least two items are packed between (i_1,j_1) and (i_2,j_2)), constraint (11) requires that $d_{i_1,i_2,j_1,j_2} = 0$. Otherwise, d_{i_1,i_2,j_1,j_2} may be indifferently equal to 0 or 1, but constraint (18) suggests d_{i_1,i_2,j_1,j_2} to be equal to 1, which is consistent with the definition.

Constraints (12)–(18) are guillotine constraints for each rectangular area. If guillotine constraints are satisfied for each rectangular area, then it is also satisfied for the whole cutting pattern.

Indeed, constraints (12)–(14) ensure the coherence of the definition of p_{i_1,i',j_1,j_2}. In particular, if $\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i',j,k} = 0$; i.e., no item is packed between (i',j_1) and (i',j_2), constraint (12) requires $p_{i_1,i',j_1,j_2} = 0$. Similarly, if $\sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{k=1}^{n} z_{i,j,k} = 0$; i.e., no item is packed between (i_1,j_1) and $(i'-1,j_2)$, constraint (12) requires $p_{i_1,i',j_1,j_2} = 0$. In either case, $x_{i'}$ is not the lower bound of an interval disjoint with the items packed between (i_1,j_1) and $(i'-1,j_2)$. In addition, if $p_{i_1,i',j_1,j_2} = 1$ in constraint (14) then $u_{i,j,i'} = 1$ for any i such that $i_1 \leqslant i < i'$ and for any j such that $j_1 \leqslant j \leqslant j_2$. That means that no item packed between (i_1,j_1) and $(i'-1,j_2)$ exceeds $x_{i'}$. The merging of intervals corresponding to the projection of these items on the x-axis forms at least two disjoint intervals. In some cases, p_{i_1,i',j_1,j_2} may be indifferently equal to 0 or 1. In such cases, constraint (18) suggests p_{i_1,i',j_1,j_2} to be equal to 1, which is consistent with the definition.

By symmetry, constraints (15)–(17) ensure the coherence of the definition of $q_{i_1,i_2,j_1,j'}$.

Constraint (18) requires that at least one of the following conditions is fulfilled:

1. $d_{i_1,i_2,j_1,j_2} = 1$ which means that $E(i_1,i_2,j_1,j_2)$ contains at most one item.
2. $p_{i_1,i',j_1,j_2} = 1$ for some i' such that $i_1 < i' \leqslant i_2$, what is equivalent to condition 2 of Theorem 1.
3. $q_{i_1,i_2,j_1,j'} = 1$ for some j' such that $j_1 < j' \leqslant j_2$ what is equivalent to condition 3 of Theorem 1.

And finally constraints (19) and (20) ensure that no item horizontally exceeds the width W and that no item vertically exceeds the height H, whereas the objective function is to minimize the total height used (see Eq. (1)).

Note that there is no contraint requiring there be no overlapping between items. This is due to Corollary 1.

In this model, there are too many binary variables and constraints (about $3n^4/4$ binary variables and about $2n^4$ constraints). Furthermore, many groups of constraints are such that only one constraint of the group is active and the others are redundant (for instance, there is always one redundant constraint among (7) and (8)). For this reason, the LP-relaxation of the model yields a very loose lower bound. All these factors make the model inexploitable in practice for the current performance of computers and softwares for mixed integer programming. This explains why we only tested this model on very small-sized instances.

5.3. Extension to identical stock sheets

In this subsection we deal with a generalization of the two-dimensional cutting stock, in which, we have an infinite number of identical rectangular sheets of raw material having width W and height H. The objective is to cut off a set of rectangular items while minimizing the number of used sheets. Two additional constraints are to be considered: Rotation of items is not allowed and only guillotine patterns are considered.

Our formulation of the previous section can be easily extended to solve this problem. With this intention, we add a subscript corresponding to a sheet, and modify the objective function in such a manner that it will express the minimization of the number of the used sheets. For instance, variable $z_{i,j,k,l}$ is equal to 1, if item k is packed at position (i,j) of sheet l and equal to 0 otherwise. Let Z_l indicate whether sheet l is used. The objective function becomes minimizing $\sum_{l=1}^{n} Z_l$. The following constraints must also be added:

$$Z_l \geqslant z_{i,j,k,l} \quad \forall i, \forall j, \forall k, \forall l.$$

6. Computational results on guillotine pattern checking

6.1. Generation of test problems

To test the effectiveness of our algorithm presented in Section 4 to check whether a given pattern is a guillotine pattern, we generated three types of patterns: guillotine patterns (all the items can be extracted through a sequence of guillotine cuts), entirely non-guillotine patterns (no guillotine cut can be carried out neither horizontally nor vertically), and mixed patterns which are formed by guillotine zones and entirely non-guillotine zones. In the last case, only items belonging to the guillotine zones can be extracted.

Guillotine patterns are generated in a random way. To this end, it is enough to fix dimensions of the stock sheet, which must contain all the items, and by a sequence of random vertical and horizontal cuts, one will obtain the dimensions of the items. We considered three classes of test instances, which are characterized by parameters n, W, and H where n is the number of items in the pattern, W and H are, respectively, the width and the height of the pattern. For every class we considered 50 instances.

In order to generate non-guillotine patterns, we used the following idea: At each step, a rectangle with a certain size is chosen, which is then divided, in a random way, into five smaller rectangles forming a non-guillotine subpattern (as the example shown in Fig. 3). The user input provides an estimate of the problem size. Note that, at each iteration, a "large" rectangle is replaced by five "small" rectangles. Therefore, the number of rectangles increases by four. The total number of rectangles is then re-calculated and may be equal or larger than the user input. The patterns generated by this method are special cases of non-guillotine patterns; i.e., no cut can be carried out on the initial pattern. When our algorithm is applied, it immediately finds that the initial pattern is not a guillotine pattern.

To generate mixed patterns, the previous ideas were combined together. At every step, a rectangle with given dimensions is chosen, then one decides randomly if the chosen rectangle will be divided into two rectangles (forming a guillotine subpattern) or will be divided into five rectangles (forming non-guillotine subpattern). The patterns generated by this way contain guillotine zones and non-guillotine zones, see Fig. 7.

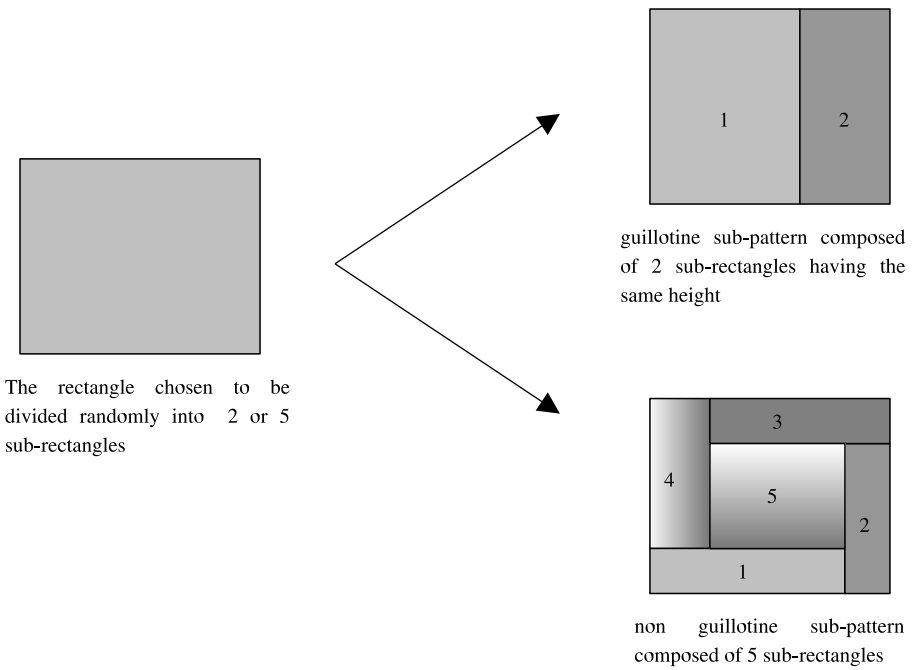


Fig. 7. Mixed pattern generating process.

Table 1
Computation times over guillotine patterns classes

Classes	CPU in seconds
$n = 1000$, $W = 15000$ and $H = 15000$	0.29
$n = 350$, $W = 5000$ and $H = 5000$	0.08
$n = 125$, $W = 750$ and $H = 750$	0.02

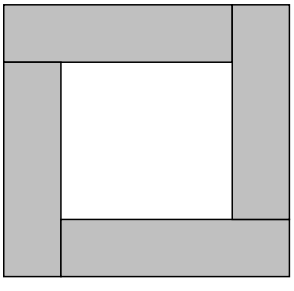
6.2. Computational results

All the algorithms were implemented in C++, and the tests were done under Windows 98, on a Pentium III 700 Mhz, 256 Mo RAM personal computer. The computation time reported hereafter does not include the time to read the generated instance from the input file and to print the final result to the output file. Table 1 gives average computation time over the three classes of guillotine patterns, each class containing 50 instances.

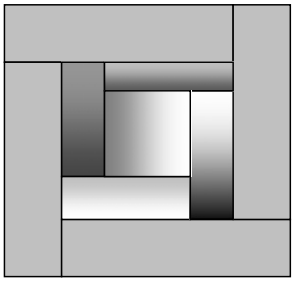
The average computation time over 50 instances of each class of mixed patterns shown in Table 2 is less than guillotine patterns, since in mixed patterns the cutting process stops earlier; i.e., there is a subset of items which cannot be extracted through guillotine cuts, thus the algorithm does not have to cut all the n items. Therefore, we obtain shorter computation time. To prove it empirically, we considered guillotine patterns, and introduced a new parameter noted NNGZ indicating the Number of Non-Guillotine Zones generated in the instance. As we saw before, in our generation method, a non-guillotine zone is composed of five items. Consequently, we obtained exactly 5NNGZ items which cannot be extracted through a sequence of guillotine cuts. However, in some cases, we need less than 5NNGZ items that cannot be extracted, since, by construction, it is possible that a non-guillotine zone is embraced in another one. Indeed, the chosen rectangle to be divided into five sub-rectangles may already belong to a non-guillotine zone. Fig. 8 gives an example of

Table 2
Computation times over mixed guillotine pattern class where $n = 1000$, $W = 15000$ and $H = 15000$

NNGZ	CPU in seconds
20	0.28
40	0.27
60	0.24
80	0.23
100	0.22
120	0.19
140	0.17
160	0.16
180	0.14



The first non-guillotine zone and the chosen rectangle in white color



Division of the chosen rectangle into five sub-rectangles forming the second non-guillotine zone

Fig. 8. An example of embraced non-guillotine zones.

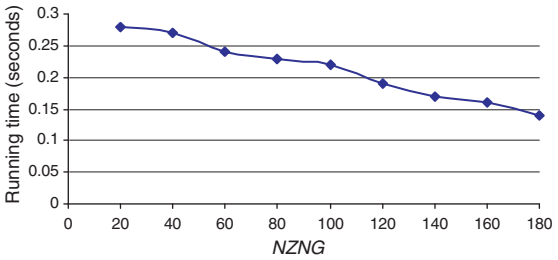


Fig. 9. Simulation results.

Table 3
Comparison with Lodi et al.'s model

A, B	$[\alpha, \beta]$							
	[0.1, 0.5]				[0.3, 0.7]			
	T_LMV	H_LMV	T_BCE	H_BCE	T_LMV	H_LMV	T_BCE	H_BCE
1/2,1/4	2	468.7	5174	426.3	0	866.8	6218	841.9
1/4,1/2	1	426.1	5511	385.4	0	602.4	3299	541.4
1/4,1/4	2	403.2	688	368.1	0	479.7	5993	447.2
1/3,1/3	0	399.8	80	371.3	0	633.4	5169	493.4

generating a particular instance where a non-guillotine zone is all included into another one. In this case, although we have NNGZ = 2, only 9 items cannot be extracted by the algorithm.

The curve obtained in Table 2 is shown in Fig. 9. We note that as the number of non-guillotine zones increases, the computation time decreases.

7. Computational results on the MIP Model

In order to evaluate the MIP model, we compare it with the model of Lodi et al. [16]. This latter model yields a shelf pattern, a special class of guillotine patterns. As indicated above, the MIP model developed in this paper contains a large number of variables and constraints. Therefore, the test instances are of very small size ($n = 5$). They were randomly generated in $U[\alpha \times W, \beta \times W]$, where $U[a, b]$ represents a uniform distribution from a to b. Control parameters α and β allow to control the width of generated items. Since the form of the items is likely to have a great impact on the performance of the model, we considered three classes of items: high items, wide items and almost square items. The high items were generated with $h \in [1.2w, 5w]$ with probability A, almost square items with $h \in [0.8w, 1.2w]$ with probability B and the remaining items are wide ones with $h \in U[0.1w, 0.8w]$. The instances are run with a Cplex Optimizer 9.0.0 on a Pentium 4 Proessor of 2.7GHz with 3.7GB RAM. The computational results are summarized in Table 3 where columns T_LMV and H_LMV, respectively, are the computation time (in CPU milliseconds) and the height of the strip used in the model of Lodi et al. [16] whereas T_BCE and H_BCE, respectively, are the computation time (in CPU seconds) and the height of the strip used with our model.

From the table, we can see that our model is, as expected, much more time consuming than the model of Lodi et al. Our model, however, yields much better solutions, especially when items are relatively small and when there are few wide items. For one series, the improvement is as high as 20%.

8. Conclusion

We have concentrated so far in this study on the guillotine property required by some cutting machines. We proved a theorem which characterizes a guillotine pattern. Then we used the principle of this theorem to give an algorithm which can determine whether a given cutting pattern is a guillotine pattern. This algorithm runs

in at most $O(n^2)$ time and consequently can reasonably be applied in practice. We can also modify this same algorithm so that it will give, in addition, a possible cutting plan to extract all the items from a guillotine pattern and will locate non-guillotine zone(s), in the opposite case.

Based upon the characterization of guillotine patterns, we also presented a new mixed integer programming formulation that explicitly considers guillotine constraints. This is the first mathematical formulation, and furthermore by means of linear inequalities, of guillotine constraints despite the fact that they were introduced at the very beginning of the cutting stock research. To this respect, this formulation presents a very important theoretical interest. However, the practical interest of this formulation is still limited, due to the large number of binary variables (about $3n^4/4$) and constraints (about $2n^4$). Furthermore, the bounds of some constraints are very large in absolute value. All these factors lead to very loose linear relaxation based lower bounds. An interesting prospect is to simplify this model with less variables and/or constraints.

Our work also continues toward further improvement of the algorithm to determine whether a given pattern is a guillotine pattern. One of the directions is to reduce the computational complexity with a more efficient implementation. In fact, this algorithm may be used as a subroutine in implicit enumerative methods to check whether a partial pattern is a guillotine pattern. Reducing the complexity of the algorithm will significantly contribute to the efficiency of the implicit enumerative methods. Another direction is to adapt our algorithm to correct an initially non-guillotine pattern, into a guillotine one, since it is capable of identifying all of the non-guillotine zones in a given pattern. It is also interesting to incorporate the guillotine pattern checking algorithm within a heuristic generating non-guillotine patterns to obtain a heuristic yielding guillotine patterns.

Acknowledgements

The authors would like to thank Mr. Abdelghani Bekrar for his help in the computational experimentation on mixed integer programming models. The authors are also very grateful to anonymous referees for their constructive remarks leading to an improvement of the paper.

References

[1] J.E. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research 33 (1985) 49–64.
[2] J.E. Beasley, A population heuristic for constrained two-dimensional non-guillotine cutting, European Journal of Operational Research 156 (2004) 601–627.
[3] S. Ben Messaoud, C. Chu, M.-L. Espinouse, New concept of the classic shelf algorithm, in: Proceedings of the International Conference on Industrial Engineering and Production Management, Porto, Portugal, 2003.
[4] M. Biro, E. Boros, Network flows and non-guillotine cutting patterns, European Journal of Operational Research 16 (1984) 215–221.
[5] K.A. Dowsland, W.B. Dowsland, Packing problems, European Journal of Operational Research 56 (1992) 2–14.
[6] H. Dyckhoff, U. Finke, Cutting and Packing in Production and Distribution, Physica-Verlag, Heidelberg, 1992.
[7] H. Dyckhoff, G. Scheithauer, J. Terno, Cutting and packing (C&P), in: M. Dell'Amico, F. Maffoli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 393–413.
[8] S.P. Fekete, J. Schepers, A new exact algorithm for general orthogonal d-dimensional knapsack problems, Algorithms – ESA'97, Springer Lecture Notes in Computer Science 1284 (1997) 144–156.
[9] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem, Operations Research 9 (1961) 849–859.
[10] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem – part II, Operations Research 11 (1963) 863–888.
[11] P.C. Gilmore, R.E. Gomory, Multi-stage cutting stock problems of two and more dimensions, Operations Research 13 (1965) 94–120.
[12] P.C. Gilmore, R.E. Gomory, The theory and computation of knapsack functions, Operations Research 14 (1966) 1045–1074.
[13] E. Hadjiconstantinou, N. Christofides, An exact algorithm for general, orthogonal, two-dimensional knapsack problems, European Journal of Operational Research 83 (1) (1995) 39–56.
[14] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: A survey, European Journal of Operational Research 141 (2) (2002) 241–252.
[15] A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal on Computing 11 (1999) 345–357.
[16] A. Lodi, S. Martello, D. Vigo, Models and bounds for the two-dimensional level packing problems, Journal of Combinatorial Optimization 8 (2004) 363–379.
[17] G. Scheithauer, J. Terno, Modeling of packing problems, Optimization 28 (1993) 63–84.