A new graph-theoretical model for k-dimensional guillotine-cutting problems

François Clautiaux[1]*, Antoine Jouglet[2], Aziz Moukrim[2]

1: LIFL, UMR CNRS 8022, Université des Sciences et Technologies de Lille

2: HeuDiaSyC, UMR CNRS 6599, Université de Technologie de Compiègne

January 15, 2008

Abstract

We consider the problem of determining if a given set of rectangular items can be cut in a large rectangle, using guillotine cuts only. We introduce a new class of arc-colored and oriented graphs, named *guillotine graphs*, which model guillotine patterns. Then we show that an uncolored and non-oriented multigraph is sufficient to obtain any guillotine pattern. We propose linear algorithms for recognizing these graphs, and computing the corresponding patterns.

1 Introduction

The two-dimensional orthogonal guillotine-cutting problem consists in determining if a given set of small rectangles (items) can be cut from a larger rectangle (bin), using *guillotine cuts* only. A guillotine cut is straight and has to be performed from one edge to the opposite edge of a current available rectangle. As it is a decision problem, it can be seen either as an *open-dimension packing* problem (such as the *strip-packing* problem), or as a *knapsack* problem [19].

It occurs in industry if pieces of steel, wood, or paper have to be cut from larger pieces. It is sometimes cited as the unconstrained two-dimensional guillotine-cutting problem (UTDGC). This problem is *NP-complete* as it generalizes the classical bin-packing problem $1BP$ (see [13]).

The two-dimensional guillotine-cutting problem has been widely studied in the OR literature. A first way of tackling this problem efficiently is to use restrictions on the cutting patterns, such as *staged cutting patterns* (see [1,5,14], or *two-section cutting patterns* (see [10]).

For the non-guillotine case, the best exact results are obtained using constraint-programming techniques [2–4,9]. For the guillotine case, other models have been used.

*Corresponding author: François Clautiaux, INRIA, Parc de la Haute Borne, 59650, Villeneuve d'Ascq, France, Email: `francois.clautiaux@univ-lille1.fr`

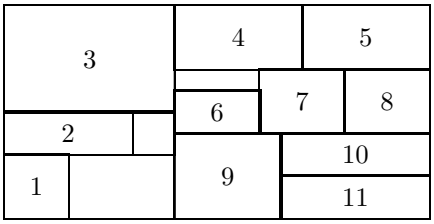


Figure 1: A guillotine pattern

Two ways are used in the literature for seeking an exact solution for the guillotine case (see [16]). The first approach [7] consists in iteratively cutting the bin into two rectangles, following an horizontal or a vertical cut, until all rectangles are obtained. The second approach [17] recursively fusions items into larger rectangles, using so-called horizontal or vertical *builds* [18].

In this paper we propose a new graph-theoretical model for the two-dimensional guillotine-cutting problem. This approach has been used successfully by Fekete *et al.* [11, 12] for the non-guillotine case. This model can be adapted for the guillotine case. Since the problem we consider has a simpler recursive combinatorial structure, we think that it is interesting to deal with a model that is hand-tailored for the guillotine case.

First we introduce the notion of *guillotine-cutting class*, which is similar to the concept of *packing class* [11, 12], but leads to less redundancies for the guillotine-case. Then we describe a model that is based on an unique oriented arc-colored graph, where circuits are related to horizontal or vertical builds. Finally we show that an unique undirected uncolored multigraph is sufficient to represent exactly two guillotine-cutting classes.

We describe several algorithms of linear complexity for recognizing these graphs and for constructing the guillotine pattern related to a given graph.

In Section 2, we describe our notation and some previous results. Section 3 is devoted to our new model, based on colored and oriented graphs. In Section 4, we deal with the undirected uncolored multigraph model. In Section 5 we propose an algorithm to recover a guillotine pattern from such a graph.

2 Notation and graph-theoretical concepts

2.1 Problem formulation and notation

A guillotine-cutting instance D is a pair (I, B). I is the set of n items i to cut. An item i has a width w_i and an height h_i ($w_i, h_i \in \mathbb{N}$). The bin B is of width W and height H. All items have to be cut, items cannot be rotated, all sizes are discrete, and only guillotine cuts are allowed (see Figure 1). A *cutting pattern* is a set of coordinates for the items to cut. A pattern is *guillotine* if it can be obtained using guillotine cuts only. It can be checked in $O(n^2)$ time [6].

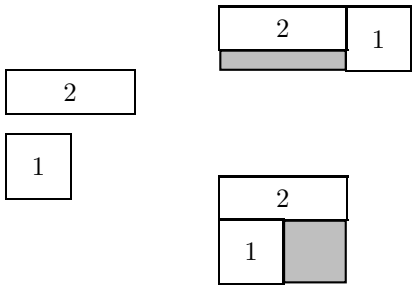


Figure 2: Vertical and horizontal builds

A build [18] consists in creating a new item by combining two items (see Figure 2). The result of an horizontal build of two items i and j, denoted $build(i, j, horizontal)$, is an item of label $min\{i, j\}$, of width $w_i + w_j$, and of height $\max\{h_i, h_j\}$. A vertical build can be defined similarly. A valid sequence of builds is an ordered list $b_1, b_2, \ldots, b_{n-1}$, such that for $k = 1 \ldots, n-1$, $b_k = build(i_k, j_k, o_k)$, i_k and j_k two valid labels for step k, and o_k an orientation (horizontal or vertical). A valid sequence of builds is said to be *normal* if for any b_k, $i_k < j_k$. Any guillotine pattern can be obtained by a normal sequence of builds.

2.2 Graph-theoretical concepts

An *undirected graph* G is a pair (V, E), and a *directed graph* G is a pair (X, A). The notation $[u, v]$ is used for an edge in an undirected graph, and (u, v) for an arc in a directed graph. For a given vertex v, its *neighborhood* $N(v)$ is the set of vertices u such that $[v, u] \in E$. When an edge may appear more than once, we have a multigraph. An *Hamiltonian chain* (resp. cycle) μ is a chain (resp. cycle) that visits each vertex exactly once. A *stable set* V_1 is a set of vertices such that there are no edges between any pair of vertices of V_1.

Interval graphs are used in the model of Fekete and Schepers [11]. Roughly speaking, a graph $G = (V, E)$ is an *interval graph* if it captures the intersection relation for some set of intervals on the real line.

Definition 2.1. *G is an interval graph if one can assign to each vertex v of V an interval I_v such that $I_u \cap I_v$ is nonempty if and only if $[u, v] \in E$.*

Interval graphs are triangulated (there is no chordless cycle of size at least four). See the book by Golumbic [15] for more details.

We give the definition of the classical concept of *edge-contraction* that plays an important role in this paper (see [20] for a recent and extensive study on edge contraction). Contracting an edge $e = (u, v)$ in the graph $G = (V, E)$ consists in deleting u, v and all edges incident to u or incident to v and introducing a new vertex v_e and new edges, such that v_e is incident to all vertices that were incident to u or incident to v.

2.3 The model of Fekete and Schepers

Fekete and Schepers [11] show that a pair of interval graphs can be associated with any *packing class* (*i.e.* a set of patterns sharing a certain combinatorial structure). Their model reduces the number of symmetries by a wide range compared to classical methods, since only one packing by class is enumerated. Two interval graphs $G^W = (V, E^W)$ and $G^H = (V, E^H)$ are respectively associated with the width and with the height. A vertex v_i of V is associated with each item i. An edge is added in the graph G^W (respectively G^H) between two vertices v_i and v_j if the projections of items i and j on the horizontal (respectively vertical) axis overlap. The authors prove that a pair of interval graphs G^W and G^H is a packing class if and only if

1. each stable set S of G^W is such that $\sum_{v_i \in S} w_i \leq W$
2. each stable set S of G^H is such that $\sum_{v_i \in S} h_i \leq H$
3. $E^H \cap E^W = \emptyset$

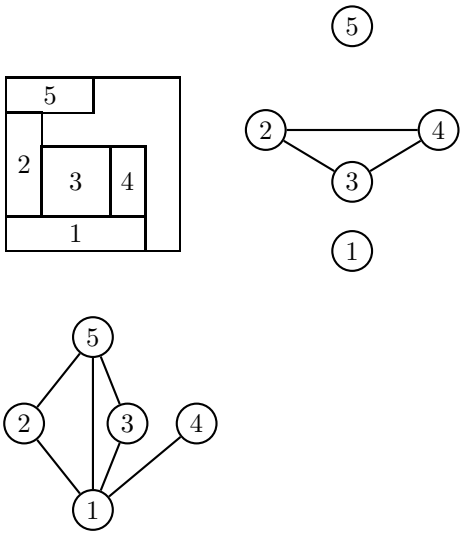


Figure 3: The graph-theoretical model of Fekete and Schepers [11]

Fekete *et al.* propose a method [12] to seek a pair of interval graphs with the properties described above. In comparison with the classical algorithms it avoids a large number of redundancies, and is still competitive compared to more recent methods [8, 9].

This model can be adapted to the guillotine case, by using the algorithm of [6] for example.

When guillotine cuts are applied, many edges do not have any interest. For example, if two items are separated by a vertical cut, knowing whether they are on the same y-coordinate or not does not change the configuration, it only generates equivalent configurations. If this situation rarely happens in the non-guillotine case, the interest of dealing with it is large when guillotine cuts are applied. So for the model of Fekete and Schepers [11] to be efficient for solving this problem, it should use an effective handling method for these edges. In the sequel, we chose another path. We proposed a brand new graph-theoretical model, which takes into account the simpler combinatorial structure of the guillotine-cutting problem.

3 A new graph-theoretical model for the guillotine-cutting problem

In this section, we propose a new graph-theoretical model for the guillotine-cutting problem. For this purpose, we introduce the concept of *guillotine-cutting classes*, which are similar to the *packing classes* of [11]. We study a new class of oriented and arc-colored graphs, and show that such graphs can be associated with a guillotine-cutting class. We name these graphs *guillotine graphs*.

3.1 Guillotine-cutting classes

In order to avoid equivalent patterns in the non-guillotine orthogonal-packing problem ($2OPP$), Fekete and Schepers [11] proposed the concept of *packing class*. Packing classes are general and model any pattern, guillotine or not. When only guillotine patterns are sought, packing classes may not be suited to the problem, as two different packing classes may lead to patterns with the same combinatorial structure. In order to gather such guillotine patterns, we introduce the concept of *guillotine-cutting class*. It takes into account the fact that exchanging the positions of two rectangular blocks of items does not change the combinatorial structure of the solution. The definition uses the notion of builds reminded in Section 2. Note that from a given normal sequence of builds, one may obtain several patterns: for each vertical build, one may choose to cut the first item above the second, or the opposite (the same applies to horizontal cuts).

Definition 3.1. *Two solutions pertain to the same* guillotine-cutting *class if they can be obtained from the same normal sequence of horizontal and vertical builds.*

Clearly if a member of a guillotine-cutting class is feasible, also are the other members. In this case, we say that the guillotine-cutting class is feasible. This concept takes into account the specificity of the guillotine cuts and reduces dramatically the number of equivalent patterns compared to a direct application of the model of [11]. This is not surprising, since the concept of packing classes is

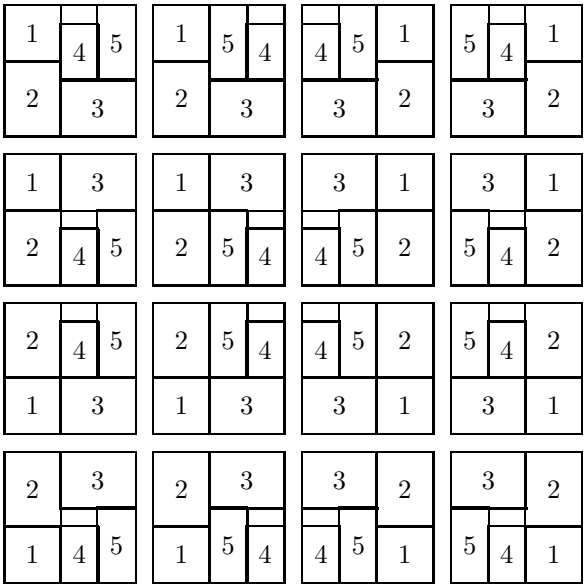


Figure 4: A guillotine-cutting class

not designed for this specific problem. On the contrary, the model of guillotine-cutting class cannot model any non-guillotine orthogonal-packing patterns.

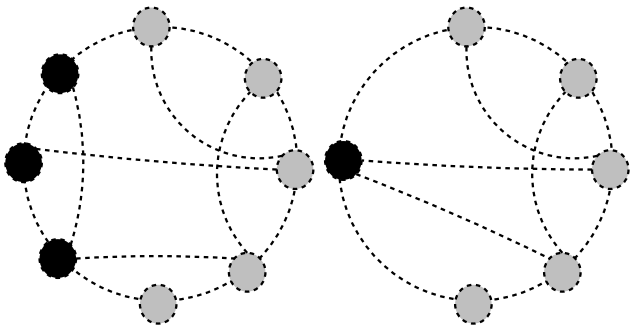
Note that there may still be redundancies. Two different sequences of builds may lead to solutions with the same combinatorial structure, for example when there is a partial pattern that can be obtained either with an horizontal cut first, or with a vertical cut first. This means that a given pattern may pertain to several guillotine-cutting classes.

3.2 A new graph-theoretical model

In the sequel we propose a new class of graphs that represent guillotine-cutting classes. In order to give a definition of this new class, we introduce the concept of *circuit contraction*, similarly to the classical concept of arc contraction used in graph theory (see Section 2).

Definition 3.2. *Let $G = (V, E)$ be a graph, and $\mu = [v_{i_1}, v_{i_2}, \ldots, v_{i_k}, v_{i_1}]$ a circuit of G. Contracting μ is equivalent to iteratively contracting each arc of μ.*

The same concept can be applied to directed graphs, in which case we will use the term *circuit-contraction*. In Figure 5, contracting the black cycle of the left-hand graph leads to the right-hand graph. Note that the order in which the arcs/edges are contracted does not change the resulting graph. The index of



(a) initial graph (b) after cycle-contraction

Figure 5: Cycle-contraction

the vertex obtained by contracting a circuit μ is the smallest index of an item in μ.

In our new model, a vertex x_i is associated with each item i, and a circuit is associated with a list of horizontal or vertical builds. Let $G = (X, A)$ be a directed graph. In order to make a difference between horizontal and vertical builds, we associate the color red to horizontal builds and the color green to vertical builds. Let $\xi : A \to \{red, green\}$ be a coloration of the arcs of G. We say that a circuit is monochromatic if all arcs of the circuit have the same color. In the graph, circuit-contracting a red (resp. green) circuit corresponds with a list of horizontal (resp. vertical) builds. When a circuit μ is contracted, the size associated with the residual vertex is the size of the item built, and its label is the smallest label of a vertex of μ. We now give a definition of *guillotine graphs*, which model guillotine patterns.

Definition 3.3. *Let G be an arc-colored oriented graph. G is a* guillotine graph *if the two following conditions hold:*

1. *G can be reduced to a single vertex x by iterative contractions of monochromatic circuits*
2. *no steps are encountered where a vertex pertains to two different monochromatic circuits*

In Figure 6, we depict the dominant graph that models the configuration of Figure 1. Many equivalent graphs can be associated with a given guillotine-cutting class, depending on the order of the vertices in the circuits. To avoid equivalent graphs, we only consider *dominant guillotine graphs*.

Definition 3.4. *Let G be a valid guillotine graph. G is a* dominant guillotine graph *if in all graphs obtained by applying circuit-contractions to G, vertices in a monochromatic circuit are ordered by increasing index and when a circuit is contracted, only two vertices are of degree greater than two.*

Algorithm 1 checks that the input graph verifies the properties of Definition 3.3. If at the end of the algorithm, the unique vertex x_j is such that $w_j \leq W$ and $h_j \leq H$, the guillotine-cutting class associated with G is feasible. This algorithm is presented to help the reader understanding the process, and to simplify the proofs.

Algorithm 1: A naïve algorithm for testing if a graph is a guillotine graph

```
Data: I = 1, 2, ..., n: a list of items;
G: a directed and colored-arc graph;
while the current graph has more than one item do
    if a vertex x_i is in two elementary monochromatic circuits then
        return false;
    Let μ be a monochromatic circuit of G and j the smallest index of an
    item in μ;
    if there is no such circuit in G then return false;
    contract μ into a single vertex x_j;
    if the circuit is red then
        w_j ← Σ_{i∈μ} w_i;
        h_j ← max_{i∈μ} h_i;
    else
        h_j ← Σ_{i∈μ} h_i;
        w_j ← max_{i∈μ} w_i;
return true;
```

In order to show that a dominant guillotine graph leads to an unique guillotine-cutting class and vice-versa, we introduce Algorithm 2, which constructs a dominant guillotine graph from a normal sequence of builds. Algorithm 2 is initialized with an empty graph, and iteratively adds arcs related to the input sequence of builds. At each step of the algorithm, two current connected components are connected to each other: the resulting component is labeled by the smallest label of a vertex pertaining to it. This algorithm uses two tables $final$ and $color$, which are associated to each connected component. They are related to the last circuit $\mu_i = [x_i, \ldots, x_f, x_i]$ obtained when all possible recursive circuit-contractions are performed on the corresponding component. The value $final[i]$ is the last vertex x_f of the circuit, and $color[i]$ is the color of circuit μ_i (by construction, it is monochromatic).

Lemma 3.1. *If the input sequence ψ of builds is a normal sequence, Algorithm 2 constructs a dominant guillotine graph.*

Proof. We show that at each step of the algorithm, all subgraphs have the dominance property. Initially, there are only circuits of size zero, so the property is true. Now suppose the property is true at step k. It means that each pair of connected components containing respectively i and j are dominant cycle-contractable graphs. When the build $b_{k+1} = (i, j, c)$ is considered, three cases can occur.

Algorithm 2: Creating a dominant guillotine graph from a normal sequence of builds

Data: ψ: a normal sequence of builds;
for $i \in I$ **do**
  $final[i] = x_i$;
  $color[i] = 0$;
$X = \{x_1, \ldots, x_n\}$;
$A \leftarrow \emptyset$;
for $k = 1$ *to* $n - 1$ **do**
  Let b_k be a triplet (i, j, c), $i < j$;
  $A \leftarrow A \cup \{(final[i], x_j, c), (final[j], x_i, c)\}$;
  if $c = color[i]$ **then** $A \leftarrow A \setminus \{(final[i], x_i, color[i])\}$;
  if $c = color[j]$ **then** $A \leftarrow A \setminus \{(final[j], x_j, color[j])\}$;
  $final[i] \leftarrow final[j]$;
  $color[i] \leftarrow c$;

The first case occurs when the color of the two circuits related to i and j is different from the color c. By construction, these colors are stored in variables $color[i]$ and $color[j]$. In this case a new graph is obtained, which can be cycle-contracted to a monochromatic circuit of size two. So it is cycle-contractable, and dominant.

The second case occurs when exactly one of the two circuits related to i and j is of color c. Without loss of generality, we consider that it is the circuit $\mu_i = [x_i, \ldots, final[i], x_i]$. In this case, the arc $(final[i], x_i)$ is removed, and two new arcs $(final[j], x_i)$ and $(final[i], x_j)$ are added. Consider the new connected subgraph created. By assumption, the previous subgraph related to j is a dominant cycle-contractable graph. When this subgraph is cycle-contracted, a single vertex x_j is obtained. When the subgraph previously associated with i is recursively cycle-contracted, a new monochromatic circuit $[x_i, \ldots, final[i], x_j, x_i]$ is obtained. By assumption, the vertices from x_i to $final[i]$ are ordered by increasing label. As ψ is a normal sequence (see Section 2), $final[i] < j$. Consequently, the property remains valid.

The third case occurs when the two circuits related to i and j are of color c. In this case, the arcs $(final[i], x_i)$ and $(final[j], x_j)$ are removed, and two new arcs $(final[j], x_i)$ and $(final[i], x_j)$ are added. Now consider the graph obtained by recursively cycle-contracting the new connected subgraph. It is composed of an unique monochromatic circuit $[x_i, \ldots, final[i], x_j, \ldots, final[j], x_i]$ By assumption, $i, \ldots, final[i]$ are sorted by increasing order of label, so are $j, \ldots, final[j]$. As ψ is a normal sequence, $final[i] < j$, so the property remains valid.

Note that if ψ is a valid sequence of builds, the graph is connected at the end of the algorithm. Consequently, at the end of the algorithm, only one graph remains, which is a dominant cycle-contractable graph.

□

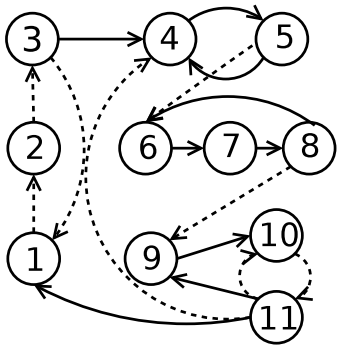


Figure 6: Modeling the pattern of Figure 1 with a guillotine graph

Theorem 3.1. *If G is a dominant guillotine graph, G can be associated with an unique guillotine-cutting class. Moreover for each normal sequence of builds, there is exactly one dominant guillotine graph.*

Proof. A guillotine pattern is obtained if horizontal and vertical builds are performed (see [17]). In the method of Definition 3.3, circuits are contracted by necessary condition. When a build is performed, two positions are possible for the items. It corresponds with the same horizontal or vertical build, so it does not modify the resulting guillotine-cutting class. Consequently a guillotine graph leads to an unique guillotine-cutting class.

From any such sequence, Algorithm 2 constructs a dominant guillotine graph (see Lemma 3.1). Algorithm 2 always adds arcs by necessary condition, so a normal sequence of builds always leads to an unique dominant guillotine graph □

If several normal sequences of builds lead to the same guillotine pattern there will be several graphs associated with the same pattern. This may occur when items of the same size are cut such that the order of a vertical and an horizontal cut can be exchanged. Handling these symmetries is an issue that we can only handle using algorithmic methods.

4 Cycle-contractable graphs

Now we show that colors and orientations are not mandatory in our model. For this purpose, we introduce a new class of undirected multi-graphs, which are named *cycle-contractable graphs* and we show that these graphs are undirected and uncolored guillotine graphs.

A cycle-contractable graph is associated with two guillotine patterns, depending on the chosen coloring. These graphs have many interesting properties,

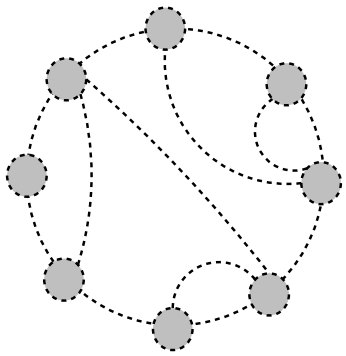


Figure 7: A cycle-contractable graph

which are taken into account to design algorithms of linear complexity for recognizing them and computing the corresponding guillotine patterns.

4.1 Cycle-contractable graphs and guillotine graphs

Let $G = (V, E)$ be an undirected multigraph. If there is an Hamiltonian cycle $\mu = [v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_1}]$, a corresponding ordering σ can be associated with the vertices of V: $\sigma(i_1) = 1$, and $\sigma(i_{j+1}) = \sigma(i_j) + 1$ for $j = 1, \ldots, n-1$. In the sequel, when a graph G has an Hamiltonian cycle, any edge that is not included in the cycle is named a *backward edge*.

Definition 4.1. *Let $G = (V, E)$ be an undirected multigraph. G is a* cycle-contractable graph *if G contains an Hamiltonian cycle μ with a corresponding ordering σ such that*

1. *G does not include two backward edges $[v_i, v_j]$ and $[v_i, v_j]$ connecting the same pair of vertices*
2. *G does not include two backward edges $[v_i, v_j]$ and $[v_k, v_l]$ such that $\sigma(i) < \sigma(k) \leq \sigma(j) < \sigma(l)$*

The graph of Figure 7 is a cycle-contractable graph. It can be pictured as a circle of vertices and non-crossing chords.

We now show that dominant guillotine graphs have the structure of cycle-contractable graphs. For this purpose we consider the graph obtained by removing the color and the orientation of the considered guillotine graph.

Lemma 4.1. *If G is a guillotine graph, it contains an Hamiltonian circuit.*

Proof. G is a guillotine graph if it can be reduced to one vertex by iteratively contracting r circuits. We show the result by recurrence on r.

At step r of the algorithm, the r circuit have been contracted, thus the current graph G_r only contains one vertex. Consequently the property is true.

Now assume that the property is true at a given step k ($1 \leq k \leq r$). It means that there is an Hamiltonian circuit $\mu = [x_1, x_2, \ldots, x_p, \ldots, x_{n^r}, x_1]$ in the current graph G_k (where n^r is the number of vertices in G_k). At step $k-1$, the graph G_{k-1} can be obtained from G_k by replacing a vertex x_p by a circuit $\mu^1 = [x_{p_1}, x_{p_2}, \ldots, x_{p_q}, x_{p_1}]$. A Hamiltonian circuit in G_{k-1} is obtained by replacing in μ the vertex x_p by the corresponding path $[x_{p_1}, x_{p_2}, \ldots, x_{p_q}]$. So by recurrence, we obtain that the initial graph G_0 contains an Hamiltonian circuit. □

Theorem 4.1. *An uncolored non-oriented dominant guillotine graph is a cycle-contractable multigraph.*

Proof. From Lemma 4.1, we know that if G is a guillotine graph, there is an Hamiltonian circuit $\mu = [x_{i_1}, x_{i_2}, \ldots, x_{i_n}, x_{i_1}]$ in G. In the corresponding undirected multigraph, the circuit becomes a cycle.

Now we have to show that there cannot be crossing arcs in a dominant guillotine graph. Suppose there exist two arcs $[x_{i_j}, x_{i_k}]$ and $[x_{i_p}, x_{i_q}]$ with $j < p \leq k < q$. It means that there exist two circuits μ_1 and μ_2 in the guillotine graph. $\mu_1 = (x_{i_j}, \ldots, x_{i_p} \ldots, x_{i_k}, x_{i_j})$ and $\mu_2 = (x_{i_p}, \ldots, x_{i_k} \ldots, x_{i_q}, x_{i_p})$.

Consider Algorithm 1 applied on G. Without loss of generality, we consider that circuit μ_1 is contracted before μ_2. When this contraction is performed, x_{i_p}, x_{i_j} and x_{i_k} are of degree greater than two. It contradicts the fact that G is a dominant guillotine graph. Consequently the graph obtained from a dominant guillotine graph G by removing the color and the orientation of the arcs of G is a cycle-contractable graph. □

The next proposition states that the number of edges in a cycle-contractable graph is in $O(n)$. This result allows us to propose $O(n)$ algorithms in the sequel.

Proposition 4.1. *In a guillotine graph G with at least two vertices, the number m of arcs in G is in $[n, 2n-2]$, and the bounds are tight.*

Proof. When a circuit μ is contracted, the number of deleted arcs in the graph is n_1 where n_1 is the number of vertices included in this circuit. When Algorithm 1 is applied, it removes $(n_1-1)+(n_2-1)+\ldots+(n_r-1)$ vertices, and $n_1+n_2+\ldots+n_r$ arcs corresponding with circuits $\mu_1, \mu_2, \ldots, \mu_r$.

After r iterations, $n-1$ vertices are removed, so $n-1+r$ arcs are also removed. Since each arc is removed once, the initial number of arcs is equal to $n-1+r$. The minimum value of r is 1 (when all items are cut side by side) and the maximum value of r is $n-1$ (when only one vertex is removed at each step). Thus we obtain the following relation: $n \leq m \leq 2n-2$. If all items are packed side by side, the number of arcs is equal to n. If no circuit of size at least 3 is encountered during Algorithm 1 the number of arcs is equal to $2n-2$. □

In the remainder, we will consider that any graph considered is connected and has less than $n-1$ edges.

Algorithm 3: Finding the Hamiltonian cycle in a cycle-contractable graph

Data: $G = (V, E)$: multigraph;
$\mu \leftarrow \emptyset$;
$L \leftarrow \emptyset$;
forall *edge that appears twice in* E **do** delete one of the two edges $[v_i, v_j]$;
forall i *such that* $|N(v_i)| = 2$ **do** $L \leftarrow L \cup \{v_i\}$;
repeat
  Let v_i be a vertex in L and let v_j and v_k be its two neighbors;
  $L \leftarrow L \setminus \{v_i\}$;
  if $[v_i, v_j]$ *is not backward* **then** $\mu \leftarrow \mu \cup \{[v_i, v_j]\}$;
  if $[v_i, v_k]$ *is not backward* **then** $\mu \leftarrow \mu \cup \{[v_i, v_k]\}$;
  $G \leftarrow G \setminus \{v_i\}$;
  if $[v_j, v_k] \notin G$ **then** $G \leftarrow G \cup \{[v_j, v_k]\}$;
  mark $[v_j, v_k]$ as *backward*;
  if $|N(v_j)| = 2$ **then** $L \leftarrow L \cup \{v_j\}$;
  if $|N(v_k)| = 2$ **then** $L \leftarrow L \cup \{v_k\}$;
until $n = 3$ *or* L *is empty*;
if $n > 3$ **then** exit with the **FAIL** status;
add each remaining edge in μ if it is not backward;
return μ;

4.2 Finding the Hamiltonian cycle in a cycle-contractable graph

Cycle-contractable graphs can be recognized in linear time. When such a graph is considered, the first step for computing the corresponding guillotine graphs is to determine which edges pertain to the Hamiltonian cycle μ, and which edges are backward. Algorithm 3 finds the cycle μ in linear time, by using the fact that: if there are two edges $[v_i, v_j]$ then one of them is in μ; if a vertex v_i has two neighbors, the two edges incident to v_i pertain to μ. We also use the fact that removing these edges leads to a graph that is still cycle contractable.

In order to show the validity of Algorithm 3, we first prove the following lemmas.

Lemma 4.2. *The graph G' obtained from the cycle-contractable graph G by performing one of the two following modifications:*

1. *removing an edge $[v_j, v_k]$ that appears twice in G;*
2. *deleting a vertex v_i of degree two and its two incident edges $[v_i, v_j]$ and $[v_i, v_k]$, and adding an edge $[v_j, v_k]$ if it is not already in the graph.*

is a cycle-contractable graph. Moreover, any edge pertaining to the Hamiltonian cycle of G' and to G also pertains to the Hamiltonian cycle of G.

Proof. When the first modification is performed, if there is a Hamiltonian cycle μ in G, it crosses one of the two considered edges, thus removing one of them

conserves the Hamiltonian cycle. Since no edges are added, no *crossing edges* are added. So the new graph is still cycle contractable.

For the second modification, since we have $\mu = [v_1, \ldots, v_j, v_i, v_k, \ldots, v_t, v_1]$, we know that $\mu' = [v_1, \ldots, v_j, v_k, \ldots, v_t, v_1]$ is an Hamiltonian cycle for $G' = G \setminus \{v_i\}$. If the edge $[v_j, v_k]$ is added, it pertains to μ, thus cannot be backward.

Now consider μ' again. If the first transformation was performed, one can build an Hamiltonian cycle for G by replacing the $[v_j, v_k]$ by its double (which practically leads to the same cycle). If the second transformation is performed, one can recover the initial Hamiltonian cycle by replacing the new edge $[v_j, v_k]$ by $[v_j, v_i, v_k]$. In both cases, any edge that pertains to both μ' and the initial E also pertains to μ.

□

Lemma 4.3. *Let G be a cycle-contractable graph. If G has at least three vertices, and no cycle of size two, it has at least one vertex of degree two.*

Proof. If there are no backward edges in G, then the lemma is directly true. If there are backward edges, there are several cycles in the graph. Let $\mu = [v_1, v_2, \ldots, v_k, v_1]$ be the smallest cycle which does not contain any smaller cycles (*i.e.* for any $1 < i < j < k$ there is no cycle $[v_i, \ldots, v_j, v_i]$). By initial assumption, it is at least of size three. This means that $v_2, \ldots, v_{k-1}$ each have two neighbors.

□

Proposition 4.2. *If G is cycle-contractable, Algorithm 3 finds the Hamiltonian cycle of G.*

Proof. Clearly if there is a cycle of size two, one of the two edges pertains to the Hamiltonian chain. When the other edge is removed, a cycle-contractable graph G' remains. All edges pertaining to the Hamiltonian cycle of G' pertain to the Hamiltonian cycle of G. The remaining edges of the cycle of size two is marked backward, so it will not be added twice.

If a vertex v_i has only two neighbors, then the two edges incident to v_i belong to the Hamiltonian cycle. Thus it is valid to add these edges to the current cycle. Lemma 4.3 ensures that there is always such a vertex in the initial graph. The obtained graph is still cycle-contractable (Lemma 4.2), and no cycle of size two was added so it still contains a vertex of degree two. Consequently the method can be repeated until the obtained graph is of size three.

□

The following lemma is necessary for the proof of Proposition 4.3.

Lemma 4.4. *Let $G = (V, E)$ be a cycle-contractable graph with more than three vertices, and v a vertex with two neighbors w and t. If $[t, w] \in E$, either $|N(w)| = 3$ or $|N(t)| = 3$.*

Proof. First note that since v has only two neighbors, $[w, v, t]$ is a part of the Hamiltonian cycle of G. Thus $[t, w]$ is a backward edge. This means that w and t have at least three neighbors.

Note that if the graph is oriented, w can only be the final extremity of backward arcs, and that t can only be the initial extremity of backward arcs. Now we show that there cannot be a backward arc pointing to t and a backward arc from w in the corresponding oriented graph.

If w is the final extremity of backward arcs, it means that there exists at least one arc (u, w), where $\sigma(u) > \sigma(t)$. If t is the initial extremity of a backward arc, it means that there exists (t, s), where $\sigma(s) < \sigma(w)$ The coexistence of these two arcs would contradict the fact that G is guillotine.

□

Proposition 4.3. *The complexity of Algorithm 3 is $O(n)$.*

Proof. In the first phase, the set of edges is studied once to find the cycles of size two, which leads to a global complexity of $O(m)$. Recording vertices of degree two takes $O(n)$ time.

Now let us consider the third phase (recursively removing the vertices of degree two). Removing v_i and adding the two edges takes $O(n)$ time. Removing v_i consists in deleting the two edges $[v_i, v_j]$ and $[v_i, v_k]$ in the list of successors of v_i, v_j, and v_k. This can be done in linear time if successors are stored in double-chained lists and if the same edge in two lists are linked. Lemma 4.4 ensures that either one of the two neighbors has less than three neighbors, or the edge $[v_j, v_k]$ is not in G. Thus, verifying that this edge exists is in $O(1)$. Determining the number of neighbors of v_j and v_k takes $0(1)$. Consequently $O(m)$ operations are needed for the third phase.

Thus the complexity of Algorithm 3 in $O(m)$. Using Proposition 4.1, we deduce that its complexity is $O(n)$. □

5 From cycle-contractable graphs to guillotine patterns

We have shown that a given dominant guillotine graph leads to an unique cycle-contractable graph. In this section we show that a given cycle-contractable graph leads to at most two guillotine-cutting classes. The first step consists in deducing the unique suitable orientation of the edges. Then a choice remains for the coloring of the arcs. The two possible colorings lead to two possible guillotine graphs.

5.1 Finding suitable orientation and coloring for the edges

Not all cycle-contractable graphs lead to dominant guillotine graphs, depending on the possible ordering of the vertices in the cycles. In order to avoid non-dominant solutions, we introduce the *dominant cycle-contractable graphs*, which lead to dominant guillotine graphs.

Definition 5.1. *Let G be a cycle-contractable graph. If for one of the two possible orientations, for all obtained arcs (x_i, x_j) of the main circuit ($j \neq 1$), either $i < j$, or there is a backward arc (x_l, x_i) such that $l < j$, G is a dominant cycle-contractable graph.*

Proposition 5.1. *A cycle-contractable graph leads to a dominant guillotine-graph if and only if it is a dominant cycle-contractable graph.*

Proof. It is immediate to see that the condition above is necessary. We have to show that it is sufficient to ensure that the graph is dominant. We show this result by recurrence on the number of circuits in which vertices are involved.

If we consider a circuit with no backward arc, the result is immediate. Now consider a circuit $\mu = (x_1, x_2, \ldots, x_p, x_1)$ with only one backward arc (x_p, x_1). If the condition above is true, it means that all arcs (x_i, x_j) pertaining to the Hamiltonian circuit are such that $i < j$ (as there are no backward arcs). So the vertices of the circuit are ordered by increasing value of label.

Now we assume that the proposition is true for all circuits including vertices involved in k circuits or less. We show that the property holds for any circuit including vertices involved in at most $k + 1$ circuits.

Let $\mu = (x_1, \ldots, x_p, x_1)$ be a circuit including vertices involved in at most $k+1$ circuits. By assumption, if the condition stated above is true then all subgraphs induced by circuits involving a vertex of μ different from μ are dominant. Consider μ' the circuit obtained by circuit-contracting all circuits different from μ involving vertices of μ. Only the remaining arcs have to be considered. For each arc (x_i, x_j) of μ', we have either $i < j$, and it is directly dominant, or $i > j$. In this case, it means that there exists a backward arc (x_l, x_i) such that $l < j$. This backward arc pertains to a contracted circuit, so the label of the corresponding circuit is l. This concludes the proof.

□

Algorithm 4 returns true if and only if the input cycle-contractable graph is dominant. This property is checked using the result of Proposition 5.1. In this case, the algorithm finds a suitable orientation and coloring for determining the corresponding guillotine graph. The algorithm visits the vertices v of the obtained directed graph following the Hamiltonian circuit. Each time a backward arc has v as final extremity, it means that a new circuit is included in the current circuit, so the current color is changed. This color is changed as many times as there are such backward edges. Similarly, when backward edges have v as initial extremity, each backward edge is considered by decreasing value of index and is colored with the current color, and then the color is changed.

Proposition 5.2. *Algorithm 4 colors the arcs of a dominant cycle-contractable graph in such a way that this graph is a dominant guillotine graph.*

Proof. We show this result by recurrence on the size of the graph. If the graph is of size less than or equal to 2, the result is immediate.

Algorithm 4: Orienting and coloring a cycle-contractable graph

```
Data: G = (V, E): a cycle-contractable graph;
Use Algorithm 3 to determine the backward edges;
Choose an orientation for the edges;
forall arc (v_i, v_j) of the Hamiltonian circuit do
    if j < i and ∄k < j s.t. (v_k, v_i) is a backward edge then test ← fail;
if test = fail then
    choose the other orientation for the edges;
    forall arc (v_i, v_j) of the Hamiltonian circuit do
        if j < i and ∄k < j s.t. (v_k, v_i) is a backward edge then return
        false;
compute the corresponding ordering σ;
choose a color;
for i : 1 → n do
    v ← σ(i);
    Let S^+ be the set of backward arcs a such that a = (u, v);
    forall a ∈ S^+ do change the current color;
    Let S^- be the set of backward arcs a such that a = (v, u);
    foreach backward arc a of S^- by decreasing value of label do
        color a with the current color;
        change the current color;
    u = σ(i + 1);
    color the arc (v, u) with the current color;
return true;
```

Assume the proposition is true for a graph of size less than or equal to k. Now consider a graph of size $k+1$. Two cases are possible. If the graph is composed of an unique circuit, the result is trivial. In the other case, the graph can be decomposed in two circuit-contractable graphs connected by two monochromatic chains μ_1 and μ_2. Using the assumption, the two subgraphs are correctly colored. Algorithm 4 changes the current color only when backward arcs are encountered. As each backward arc is encountered twice (once forward, once backward), the current color before and after a circuit-contractable subgraph is the same. Chains μ_1 and μ_2 are colored with the same color. So the graph is correctly colored.

□

Corollary 5.1. *Given the color of one arc, there is only one valid coloring for a cycle-contractable graph.*

The results above can be summarized by Theorem 5.1, which is the main result of this section. As we have shown that each guillotine pattern can be obtained from a cycle-contractable graph, we can state the guillotine-cutting problem in a new way: finding a dominant cycle-contractable graph leading to a guillotine pattern fitting the input bin.

Algorithm 5: Computing the size of the guillotine pattern related to a guillotine graph

Data: G: a valid guillotine graph;
σ: the corresponding ordering on the items ($\sigma(1) = 1$);
Let S be an initially empty stack of builds b_k;
for $i : 1 \to n$ **do**
  $v_j \leftarrow \sigma(i)$;
  Let b_j be a new build of size $w_j \times h_j$ and of label j;
  foreach *backward arc* (v_j, v_k) *of color* c *by decreasing value of* $\sigma^{-1}(v_k)$ **do**
    repeat
      remove from S its top element b_t;
      $b_j \leftarrow build(b_j, b_t, c)$;
    until b_j *has for label* v_k;
  push b_j on the top of S;
return *the unique element of* S;

Theorem 5.1. *Each dominant cycle-contractable graph is related to two guillotine-cutting classes and every dominant sequence of builds is related to one dominant cycle-contractable graph.*

Proof. We showed that a guillotine-cutting class led to an unique dominant dominant guillotine-graph, and a guillotine graph trivially leads to an unique cycle-contractable graph.

When a dominant cycle-contractable graph is considered, there is only one valid orientation for the edges. There are two possible colorings (Corollary 5.1). So there are only two possible guillotine-cutting classes for a given dominant cycle-contractable graph.

□

5.2 Computing the size of the guillotine pattern

Algorithm 5 computes the width and the height of the guillotine pattern associated with the guillotine graph G. First the ordering σ is computed using Algorithm 3. Then the vertices are considered following σ. First a dummy build b is created, with the current item only. When there is a backward arc, the new build associated with the corresponding circuit is computed and stored in b, and then pushed on the top of S. At the end of the algorithm, S only contains one element, which corresponds with the guillotine pattern.

The following theorem summarizes the different complexity results described in this paper.

Theorem 5.2. *Recognizing a cycle-contractable graph, and computing the two guillotine-cutting classes related to this graphe takes $O(n)$ time and space.*

Proof. In Algorithm 3, finding the Hamiltonian cycle takes $O(n)$ time (Proposition 4.3).

For finding the valid orientation (Algorithm 4), each vertex is considered once, following the circuit, and its neighborhood is checked once. Thus this phase is in $O(n+m)$.

For the coloration phase (second phase of Algorithm 4), each vertex and its neighborhood is visited once, leading again to a complexity of $O(n+m)$. Consequently, Algorithm 4 is in $O(n+m)$.

In the main phase of Algorithm 5, each vertex is considered once, and then each arc is considered twice: when the corresponding build is added to the stack, and when it is removed from the stack. Consequently, computing the guillotine pattern from each guillotine graph is in $O(n+m)$.

Proposition 4.1 indicates that $m \in O(n)$. Consequently, the overall complexity is $O(n)$.

□

6 Conclusion

We have proposed a new graph-theoretical model for the two-dimensional guillotine-cutting problem. It uses an arc-colored directed graph, which can be replaced by an uncolored undirected multigraph, called cycle-contractable graph. We show that a cycle-contractable graph is related to two classes of guillotine patterns. We also propose linear algorithms for recognizing cycle-contractable graphs and for computing these guillotine patterns.

Our model can be used to design heuristic and exact methods and can be extended to the $k-$dimensional case. We plan to use it in a constraint-programming framework.

References

[1] J. E. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36:297–306, 1985.

[2] N. Beldiceanu and M. Carlsson. Sweep as a generic pruning technique applied to the non-overlapping rectangles constraints. *Principles and Practice of Constraint Programming (CP'2001), Lecture Notes in Computer Science*, 2239:377–391, 2001.

[3] N. Beldiceanu, M. Carlsson, E. Poder, R. Sadek, and C. Truchet. A generic geometrical constraint kernel in space and time for handling polymorphic k - dimensional objects. In *Principles and Practice of Constraint Programming - CP 2007, LNCS 4741*, pages 180–194, 2007.

[4] N. Beldiceanu, M. Carlsson, and S. Thiel. Sweep synchronization as a global propagation mechanism. *Computers and Operations Research*, 33(10):2835–2851, 2006.

[5] G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research*, 171:85–106, 2006.

[6] S. Ben Messaoud. *Caractérisation, modélisation et algorithmes pour des problèmes de découpe guillotine*. PhD thesis, Université de Technologie de Troyes, 2004.

[7] N. Christofides and E. Hadjiconstantinou. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, 83:21–38, 1995.

[8] F. Clautiaux, J. Carlier, and A. Moukrim. A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, 183(3):1196–1211, 2007.

[9] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, 35(3):944–959, 2008.

[10] Y. Cui, H. Dongli, and X Song. Generating optimal two-section cutting patterns for rectangular blanks. *Computers and Operations Research*, 33:1505,1520, 2006.

[11] S. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29:353–368, 2004.

[12] S. Fekete, J. Schepers, and J. Van Der Ween. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587, 2007.

[13] M. R. Garey and D. S. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. Freeman, New York, 1979.

[14] P. Gilmore and R. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13:94–120, 1965.

[15] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[16] M. Hifi. Exact algorithms for the guillotine strip cutting/packing problem. *Computers and Operations Research*, 25(11):925–940, 1998.

[17] K.V. Viswanathan and A. Bagchi. Best-first search methods for constrained two-dimensional cutting stock problem. *Operations Research*, 41:768–776, 1993.

[18] P. Y. Wang. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, 31:573–586, 1983.

[19] G. Wäscher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130, 2007.

[20] T. Wolle and H. Bodlaender. A note on edge contraction. Technical Report UU-CS-2004-028, Institute of information and computing sciences, Utrecht University, 2004.