Constructive procedures to solve 2-Dimensional Bin Packing Problems with Irregular Pieces and Guillotine Cuts

Antonio Martinez[a], **Ramon Alvarez-Valdes**[b], **Julia Bennell**[a], **Jose Manuel Tamarit**[b]
a University of Southampton
b University of Valencia

Abstract

This paper presents an approach for solving a new real problem in Cutting and Packing. At its core is an innovative mixed integer programme model that places irregular pieces and defines guillotine cuts. The two-dimensional irregular shape bin packing problem with guillotine constraints arises in the glass cutting industry, for example, the cutting of glass for conservatories. Almost all cutting and packing problems that include guillotine cuts deal with rectangles only, where all cuts are orthogonal to the edges of the stock sheet and a maximum of two angles of rotation are permitted. The literature tackling packing problems with irregular shapes largely focused on strip packing i.e. minimizing the length of a single fixed width stock sheet, and does not consider guillotine cuts. Hence, this problem combines the challenges of tackling the complexity of packing irregular pieces with free rotation, guaranteeing guillotine cuts that are not always orthogonal to the edges of the stock sheet, and allocating pieces to bins. To our knowledge only one other recent paper tackles this problem. We present a hybrid algorithm that is a constructive heuristic that determines the relative position of pieces in the bin and guillotine constraints via a mixed integer programme model. We investigate two approaches for allocating guillotine cuts at the same time as determining the placement of the piece, and a two phase approach that delays the allocation of cuts to provide flexibility in space usage. Finally we describe an improvement procedure that is applied to each bin before it is closed. This approach improves on the results of the only other publication on this problem, and gives competitive results for the classic rectangle bin packing problem with guillotine constraints.
Keywords: irregular packing, bin packing, guillotine cuts, mixed integer models.

1 Introduction

The paper focuses on the two-dimensional bin packing problem(BBP) with irregular convex pieces and guillotine cuts. Guillotine cuts arise due to the cutting process of certain materials, where cuts are restricted to extend from one edge of the stock-sheet to another. This is commonly found in the glass cutting industry and research has almost exclusively focused on the cutting of rectangles where the cuts are orthogonal to the edges of the stock sheet.The problem tackled here is a generalized version of the rectangle single size stock sheet bin packing problem with guillotine constraints, where the pieces to be cut are irregular convex polygons and the guillotine cuts are not constrained to be orthogonal to the edges of the bin. We present a number of constructive heuristics that exploit a mixed integer programming(MIP) model for placing the pieces in the bin while satisfying all the geometric constraints and the guillotine cuts and provide comprehensive results for benchmark instances for this problem and the rectangular bin packing problem with guillotine cuts. The results are compared with the best in the literature and new lower bound is presented.

The motivation for tackling this problems arises in the cutting of glass sheets, for example, the glass panes for conservatories. Glass is cut by scoring and snapping the material. Each cut divides the sheet in two and is called a guillotine cut. As a result, only convex pieces can be cut directly. Since the properties of the material are homogeneous, pieces can be cut at any angle of rotation, and in some cases reflected. Typically, cutting plans involve multiple stock sheets. An example solution to this problem is given in Figure 1. Note that the pieces must be removed by the guillotine cuts (in bold) in a specific order so that each cut begins and ends at the edge of the complete or divided stock sheet.

The requirement for guillotine cuts arise in many industries such as glass, wood boards or fibre glass for printed circuit boards. Each industrial setting implies a particular class of cutting and packing problem. For example, Malaguti et al. (2014) consider a cutting stock problem with multiple stock sheets found in the wood industry, while Cui (2014) tackles the cutting of metal of coils where a stock sheet is considered to have unrestricted length.

The literature tackling packing problems with irregular shapes largely focus on strip packing i.e. minimizing the length of a single fixed width stock sheet, see Bennell and Oliveira (2009), and does not consider guillotine cuts. Further, almost all cutting and packing problems that include guillotine cuts deal with rectangles only, where all cuts are orthogonal to the edges of the stock sheet and a maximum of two angles of rotation of the pieces are permitted. Hence, this problem combines the challenges of tackling the complexity of packing irregular pieces with free rotation, guaranteeing guillotine cuts that may or may not be orthogonal to the edges of the stock sheet, and allocating pieces to bins or stock sheets. To our knowledge, Han et al. (2012) is the only other paper that tackles this problem. They propose two different construction heuristics. The first heuristic, they call a one-stage algorithm, constructs multiple parallel clusters of pieces by iteratively joining two blocks, where a block is either a piece or the convex hull of a cluster of pieces. Blocks are combined by matching pairs of edges and selecting the best match according to a dynamic solution evaluation function. Since each accepted match, starting with two individual pieces, is transformed into its convex block by taking the convex hull, the guillotine cut structure is always satisfied. The approach uses threshold parameters to control the number of blocks generated and as a result the computational time. The second constructive heuristic, a two-stage algorithm, combines pairs of pieces into rectangles by using the phi-functions presented by Bennell et al. (2011), and then uses the guillotine bin packing algorithm developed by Charalambous and Fleszar (2011) to pack the rectangles. The combination of the pieces into rectangles using phi-functions requires significant computational effort, up to eighteen hours in the largest instance (149 pieces), although the packing algorithm by Charalambous and Fleszar (2011) takes less than one second. Most recently, Fleszar (2013) developed an heuristic algorithm based on the insertions of the pieces one at a time obtaining slightly worse results but the computational time is much shorter.

A key theme in the cutting and packing literature for irregular shape packing, is concerned with efficiently identifying overlap between pieces. Bennell and Oliveira (2008) provide a useful review of this aspect of the problem. In recent years the most common approach is to use the Nofit Polygon (NFP). The NFP is a polygon derived from two component polygons in such

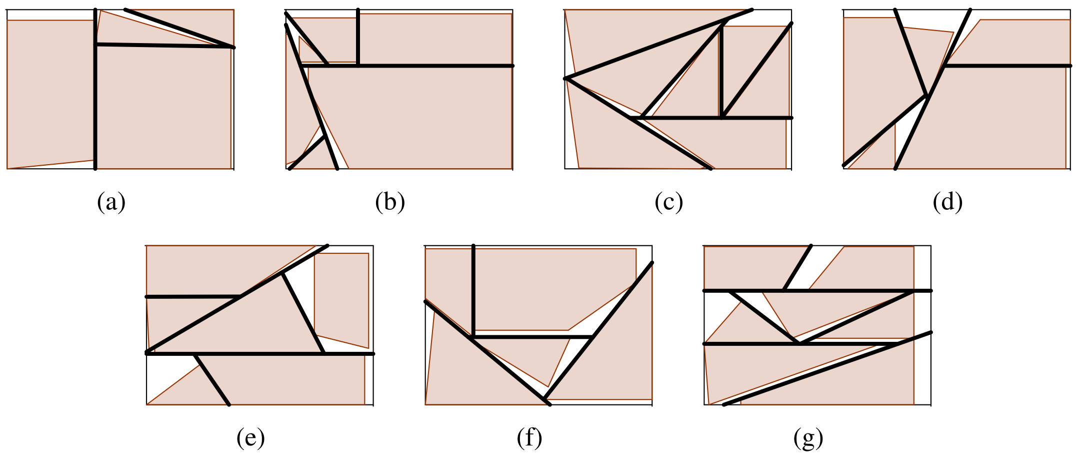


Figure 1: *Real example (J40)*

a way that its interior represents all overlapping position between the component polygons, and its boundary represents all touching positions. By determining the location of the co-ordinate point arising from the difference between the co-ordinate position of the reference point of the two component polygons and comparing with the correctly located NFP, the overlap test reduces to a simple point inclusion test while the NFP has almost exclusively been used for placing polygons, Burke et al. (2010) show that it can be extended to include arced edges.

In this paper we present a constructive heuristic that builds a solution by sequentially adding pieces to the current bin, where a MIP model determines the position of pieces in the bin and the corresponding guillotine constraints. Once a piece is placed in the bin, the relative position of the piece with respect to the other pieces is fixed, but the absolute position in the bin may change each time a new pieces is added. We investigate two approaches for allocating guillotine cuts, and a two phase approach that delays the allocation of cuts to provide flexibility in space usage. Finally we describe an improvement procedure that is applied to each bin before it is closed. The contributions of the paper are many. It is the first paper to model and solve guillotine cuts for irregular shapes as a MIP. It is only the second paper to tackle this practical problem in the literature and improves on the results of the only other publication on this problem. It provides a new lower bound for the problem. Finally it benchmarks the proposed algorithm against well known rectangle guillotine packing instances and gives competitive results.

In the next section we give a detailed description of the problem and some notation. Section 3 presents the *horizontal slices* MIP model for the insertion of each piece into a bin. The two different guillotine cuts structures are presented in Section 4. In Section 5 we describe our approach for reducing the rotation angles to a finite set. In Section 6 we introduce the constructive algorithm scheme, which is extended in Section 6.1 with an alternative approach for creating the guillotine cut structure and in Section 6.2 by embeding an improvement procedure into the constructive process. We derive a lower bound for the problem in section 7. Section 8 contains the computational study. Finally, in Section 9 we draw some conclusions.

2 Problem description

The problem objective is to minimize the number of stock sheets required to cut all the demand pieces, hence it is an input minimization problem. There are sufficient standard size rectangular stock-sheets available to meet demand, where the stock sheet has length L and width W. Let P be the demand set of pieces, and each piece is considered to be unique. According to the typology proposed by Wäscher et al. (2007), which improves the typology presented in Dyckhoff (1990, 1985), this is a single bin size bin packing problem (SBSBPP). The further refinements are that all pieces are convex and usually irregular, pieces can be rotated continuously and reflected, and only guillotine cuts are allowed. In this case, the cutting line is not constrained to be parallel to an edge of the stock-sheet and there are no limits on the number of cuts.

Let B denote the set of bins, where all bins are identical in size with width W and length L. The bottom left corner of the bin is the origin of the cartesian co-ordinate system used to locate each piece. Each bin, $b_i(P_i, X_i, Y_i, R_i, M_i, G_i) \in B$, has an associated set of pieces, $P_i \subseteq P$ in such a way each piece belong exactly to one of these subsets, $\bigcup_{i=1}^{|B|} P_i = P$ and $P_i \cap P_j = \emptyset$. Each piece $p \in P_i$ is given by an ordered list of vertices, $p = (v_1, \ldots, v_{n_p})$, and its edges can be expressed by $e_j = (v_j, v_{j+1})$, where $j = 1, \ldots, n_p - 1$ and the n_pth edge is $e_n = (v_{n_p}, v_1)$. In addition, each piece has a reference point (x_p, y_p), which corresponds to the bottom-left corner of the enclosing rectangle of the piece. The position of each piece is the coordinate position of the reference point of the piece. Then, the position of all the pieces in the bin b_i is given by the two vectors $X_i \in \mathbb{R}^{|P_i|}$ and $Y_i \in \mathbb{R}^{|P_i|}$, which denotes the $x-$coordinates and the $y-$ coordinates of all the pieces in the bin respectively. The rotation angle and the reflection (mirror transformation) of the pieces are given by vector sets $R_i \in \mathbb{R}^{|P_i|}$ and $M_i \in \mathbb{B}^{|P_i|}$, where 1 indicates that the original piece is reflected. Finally, $G_i = (g_i^1 \ldots g_i^{|P_i|-1})$ is an ordered set of guillotine cuts in such a way that the first guillotine cut, $g_i^1 \in G_i$ divides b_i into two parts. The second guillotine cut, $g_i^2 \in G_i$, divides one of the parts, and so on. Note that apart from the first cut, either endpoint of a cut can lie on one of the previous cuts instead of an edge of the bin.

Each guillotine cut, $g_i^k(p, e_l) \in G_i$, where $k \in \{1, \ldots, |P_i| - 1\}$ is the order of the cut, is associated to piece $p \in P_i$ and runs concurrent with edge e_l. This means that we have to know the position of piece p in order to know where $g_i^k(p, e_l)$ is placed in the bin. Note that $g_i^k(p, e_l)$ has *order k* if it is the k^{th} guillotine cut. The order of the guillotine cuts is important because the endpoints of the guillotine cuts are given by the intersection with either the closest guillotine cut with a lower order or the edges of the bin.

While the primary objective is to minimize the total number of bins (stock sheets), in practice an offcut of a partially packed bin can be reused in subsequent cutting patterns. An offcut is the remaining material after applying a guillotine cut to separate the packed and unpacked area. The cut may be horizontal or vertical, and selected to give the largest reusable rectangle area. We denote by R^* to the packed area after applying the offcut. So the objective is to minimize the fractional number of bins (F). If N is the total number of bins used in the solution, then:

$$F = N - 1 + \frac{R^*}{LW} \tag{1}$$

Alternatively, the objective could be to maximize the stock sheet utilization (U), defined by:

$$U = \frac{\sum_{i=1}^{|P|} Area(p_i)}{((N-1)LW) + R^*} \tag{2}$$

Either of these measures (U) and (F) is helpful for differentiating the quality of competing methods when they produce solutions with the same number of bins. There is a close relation between (F) and (U). If we consider two different solutions s_1 and s_2 such that $U(s_1) > U(s_2)$, indicating that the usage obtained by solution s_1 is better, then s_1 has a smaller fractional number of bins $F(s_1) < F(s_2)$.

3 Mixed integer formulation for the insertion of one piece

In this section we describe how an individual piece is located in a bin. Initially, all the pieces are sorted by a certain criterion in order to be inserted into the bins. If the next piece in the sorted list does not fit into the current bin, we try to insert the remaining pieces in the list before opening a new bin.

In order to determine whether a piece can be inserted feasibly into a bin, and if so where, a Mixed Integer Problem MIP is solved to optimality. Note, the MIP model permits moving the position of the pieces already in the bin, and must respect the existing guillotine cut structure defined in previous steps. A feasible solution for the model is the co-ordinate positions of all the pieces already in the bin and the newly inserted piece that results in: no overlap between pieces, all pieces are contained in the bin, and no piece crosses any of the existing guillotine cuts. The formulation assumes all pieces have fixed rotation and a fixed reflection in order to calculate the NFPs between the new piece and the pieces already placed. Hence, the model is solved several times for different rotation angles for the original and the reflection of the next piece. Clearly, the MIPs become harder to solve as the number of pieces in the bin increases.

In Figure 2 we show a simple example of a NFP given by a square(p_i) and a diamond (p_j). If we fix the position of piece p_i then the NFP is defined as the set of points such that the placement of the reference point of p_j (shown in the figure) would produce overlap between p_i and p_j. When calculating the edges of the NFP we assume that the reference point of p_i is placed at the origin.

The NFP is a core concept of the model and therefore worth further explanation. Given two convex polygons p_i and p_j, and a reference point on p_j, the NFP is formed by tracing the path of the reference point on p_j as p_j slides around the boundary of p_i in such a way that the polygons always touch and never overlap, see 2. Intuitively it is clear that if the reference point moves inside the NFP then p_i and p_j overlap, hence the interior of the NFP represents all overlapping positions. Equally, when the reference point is on the boundary of the NFP the polygons touch and when it is outside the NFP the polygons are separated. This result holds when the origin of p_i and the origin of the NFP coincide. If p_i is moved to position (x_i, y_i) then the position of the reference point of p_j must be transposed by $(-x_i, -y_i)$ before testing p_j's relative position with the NFP. Hence overlap between p_i and p_j can be identified by testing whether point $(x_j - x_i, y_j - y_i)$ is inside the NFP.

Let $p \in P \setminus \{\bigcup_{j=1}^{i} P_j\}$ be the next piece to be inserted into bin $b_i \in B$ with a fixed orientation and reflection (the piece is reduced into a polygon). We denote by $INT(p)$ the interior of polygon p. The following model seeks to feasibly insert p into the bin with position that minimizes the weighted combination of the dimensions of the bounding box of the packed area, defined by L_c and W_c. Further explanation of the model follows the formulation given by the following equations:

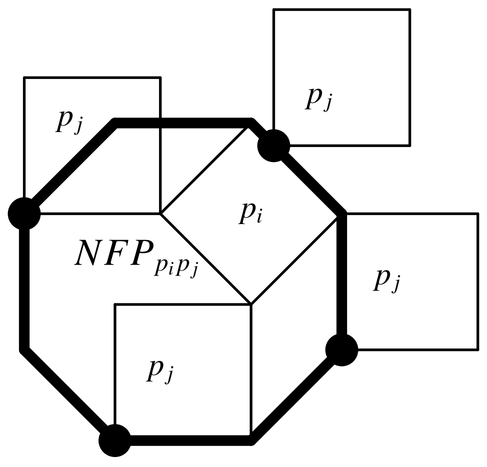


Figure 2: *Example of the NFP of a square and a diamond.*

$$
\begin{array}{llll}
Min & \omega L_c + (1-\omega)W_c & & (3) \\
s.t. & L_c \le L & & (4) \\
 & W_c \le W & & (5) \\
 & x_j \le L_c - l_j & p_j \in P_i \cup \{p\} & (6) \\
 & y_j \le W_c - w_j & p_j \in P_i \cup \{p\} & (7) \\
 & \alpha^r_{kt}(x_k - x_p) + \beta^r_{kt}(y_k - y_p) \le \sum_{h=1}^{\mu_k} \delta^{rh}_{kt} v_{kt} & 1 \le k \le |P_i|, 1 \le t \le \mu_k, r = 1,\dots,4 & (8) \\
 & \sum_{t=1}^{\mu_k} v_{kt} = 1 & 1 \le k \le |P_i| & (9) \\
 & \alpha_{kj}(x_k - x_j) + \beta_{kj}(y_k - y_j) \le \gamma_{kj} & 1 \le j < k \le |P_i| & (10) \\
 & INT(p) \cap g^k_i = \emptyset & k = 1,\dots,|P_i| - 1 & (11) \\
 & v_{kt} \in \{0,1\} & 1 \le k \le |P_i|, 1 \le t \le \mu_k & (12) \\
 & x_j,\ y_j \ge 0 & p_j \in P_i \cup \{p\} & (13)
\end{array}
$$

- *Objective function*

  The objective function (3) is a weighted combination of the length and width of the minimum size bounding box that contains the placed pieces, represented by L_c and W_c, respectively. Using this objective function formulation, the intention is to pack the pieces as tightly as possible. We consider three different alternatives for weight coefficient ω:

  - *FO0*: $\omega = \frac{1}{(W/L)+1}$
    In this case, the rectangle (L_c, W_c) grows in proportions to the bin dimensions.
  - *FO1*: $\omega = 0.01$
    The objective is to minimize the width, and the length is used as a tie-breaker.

  - *FO2*: $\omega = 0.99$
    The objective is to minimize the length, and the width is used as a tie-breaker.

- *Containment constraints*

  Constraints (4) and (5) ensure that pieces are contained entirely within the bin dimensions. Constraint sets (6) and (7) define L_c and W_c as follows. For each piece p_j, the bottom left corner of its enclosing rectangle is placed at co-ordinate position (x_j, y_j) in the bin. Let the length and width of the enclosing rectangle be l_j nd w_j, then these constraints set the upper-right corner to be less than or equal (L_c, W_c), for all the pieces.

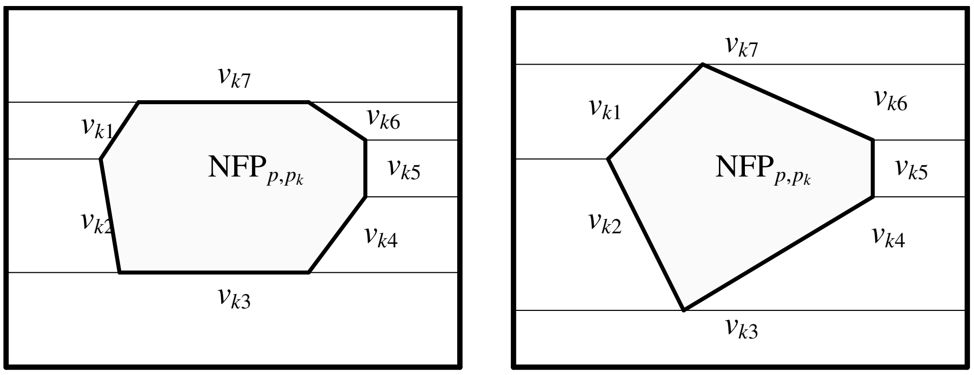
- *Non-overlapping constraints*

  Constraint sets (8) and (9) are defined to ensure that the new piece p is not placed in an overlapping position with any other piece already in the bin. These constraints were proposed by Alvarez-Valdes et al. (2013) in their *horizontal slices formulation* and use the edges of the NFP to form the no-overlap constraints. Prior to solving the MIP, construct the NFP of p and each piece p_k, $k \in \{1, \dots, |P_i|\}$ (NFP$_{p,p_k}$). Since all the pieces are convex polygons, the NFP construction is simply a case of sorting the edges of both polygons by slope order into one edge list, where p has counterclockwise orientation (fixed polygon) and p_k has clockwise orientation (orbiting polygon). The NFP is constructed from this edge list. Care must be taken to correctly locate the origin of the NFP. When the origin of the NFP coincides with the origin of the co-ordinate system, overlap is detected when the difference of the reference points of pieces p and p_k is inside the NFP. Note, that the NFP of two convex polygons is also convex. The exterior of NFP$_{p,p_k}$ indicates a non-overlapping placement between p and p_k, which is a non-convex region. The horizontal slices formulation address this problem by partitioning the exterior into convex polygons, called slices, using horizontal lines coinciding with the vertices of the polygon (see Figure 3). These slices are bounded by the dimensions of the bin (L and W). The number of slices needed to cover all the possible relative positions between p and p_k is denoted by μ_k.

  We associate one binary variable v_{kt} to each slice. $v_{kt} = 1$ if the reference point of $p - p_k$ lies on slice t and 0 otherwise. If $v_{kt} = 1$, the reference point of $p - p_k$ must satisfy the constraints defining the slice. For instance, in Figure 3(b) if $v_{k1} = 1$ the four constraints defined by the horizontal lines, the limit of the bin and the edge of NFP$_{p,p_k}$ must be satisfied. Constraint set (9) guarantees that one and only one variable v_{kt} takes the value 1 for each pair (p, p_k). When $v_{kt} = 0$, the associated constraints in (8) must be relaxed, this is commonly done using the well known Big-M method. Here we use the same approach as Alvarez-Valdes et al. (2013) who follow the proposal by Fischetti and Luzzi (2008). In order to write one no-overlapping constraint, we need four inequalities indexed by $r \in \{1, \dots, 4\}$, to describe slice $t \in \{1, \dots, \mu_k\}$. Note that the number of inequalities needed to describe one slice is always 4 because the slices are defined as quadrilaterals. The no-overlapping inequality can be written with big M constant as:

  $$\alpha^r_{kt}(x_k - x_p) + \beta^r_{kt}(y_k - y_p) \leq \delta^r_{kt} + (1 - v_{kt})M$$

  where the coefficients α^r_{kt} and β^r_{kt} are such that $\frac{\alpha^r_{kt}}{\beta^r_{kt}}$ matches with the slope of edge r in slice t, and in case that the edge r is vertical (horizontal) then $\alpha^r_{kt} = 0$ and $\beta^r_{kt} = 1$ ($\alpha^r_{kt} = 1$



(a) (b)

Figure 3: *Two examples of the horizontal slices partition.*

and $\beta^r_{kt} = 0$). Coefficient δ^r_{kt} refers to the y-intercept of edge r in slice t. Fischetti and Luzzi (2008) describe how to eliminate the M constant by including on the right hand side of the inequality all the binary variables, defined by the current NFP, multiplied by a given constant. These constants are obtained by solving the following problem:

$$\delta^{rh}_{kt} := \max_{p_k \in S^h} \alpha^r_{kt}(x_k - x_p) + \beta^r_{kt}(y_k - y_p),$$

which corresponds to the maximum value of the left hand side when piece k lies on slice S^h, $h \in \{1, \dots, \mu_k\}$. Note that $\delta^{rt}_{kt} = \delta^r_{kt}$ in the previous inequality when $h = t$, i.e, the slice is active.

Finally, including $\sum_{h=1}^{\mu_k} v_{kh} = 1$ as constraint (9) assures the validity of the following constraints:

$$\alpha^r_{kt}(x_k - x_p) + \beta^r_{kt}(y_k - y_p) \leq \sum_{h=1}^{\mu_k} \delta^{rh}_{kt} v_{kt}$$

Note that many of these constraints are the same as redundant because the slices have parallel or concurrent edges. Hence, we only need to consider two inequalities in order to write all the horizontal inequalities of the NFP.

These constraints only apply between the next piece and the pieces in the bin. Overlap between pieces already placed is dealt with by the guillotine constraints, described next.

- *Guillotine cut constraints*

  The next set of constraints (10) is the guillotine cut constraints that ensures that all pieces can be removed from the stock sheet through a series of guillotine cuts. These constraints arise from the previous insertions of pieces and apply to all the pieces already placed. After inserting the first piece into the bin, all subsequent insertions are followed by identifying the guillotine cut that separates the inserted piece from the other pieces. For each insertion, the MIP model will insert the piece in a segment of the stock sheet, partitioned by guillotine cuts, that includes one other piece. Outside of the MIP, we define a guillotine cut that divides this segment separating the piece initially occupying the segment from the newly inserted piece. Hence every guillotine cut is separating two pieces. Further, once a guillotine cut is defined, pieces must remain entirely on that side of the guillotine cut for future insertions. As a result, these constraints ensure the pieces already placed do not overlap. Therefore, the coefficients α_{kj}, β_{kj} and γ_{kj} corresponds to the coefficients of one of the inequalities defined in (8) in the MIP model used for the insertion of the previous piece, which are given by one of the edges of the NFP of the corresponding pair of pieces.

In addition, the next insertion, piece p, must respect all the existing cuts. This is more difficult to formulate since we do not know which side of the guillotine cut the piece will be placed. Constraint set (11) is written to indicate the inclusion of the constraints to prevent piece p being placed across an existing guillotine cut. In the next section we will provide a detailed discussion and formulation of these constraints.

- *Lifting the bound constraints*

  Constraints (6) and (7) apply to the new piece p. These inequalities can be improved using the interaction of piece p with the pieces already placed $p_k \in P_i$, expressed by the variables v_{kt} which have been defined for the non-overlapping constraints. For piece p, inequalities (6) and (7) are transformed into:

$$\textstyle\sum_t \overline{\alpha}_{kt} v_{kt} \le x_p \le L_c - l_p - \sum_t \underline{\alpha}_{kt} v_{kt} \quad 1 \le k \le |P_i| \qquad (14)$$

$$\textstyle\sum_t \overline{\beta}_{kt} v_{kt} \le y_p \le W_c - w_p - \sum_t \underline{\beta}_{kt} v_{kt} \quad 1 \le k \le |P_i| \qquad (15)$$

  where $\overline{\alpha}_{kt}$ and $\underline{\alpha}_{kt}$ represent the minimal horizontal displacement, to the right or to the left, of the reference point of piece p if $v_{kt} = 1$, and $\overline{\beta}_{kt}$ and $\underline{\beta}_{kt}$ represent the minimal vertical displacement, up or down, of the reference point of piece p if $v_{kt} = 1$.

4 Guillotine cuts structure

The previous section describes two sets of constraints in the MIP formulation related to guillotine cuts. Constraint set (10) are defined outside of the MIP, where a new cut constraint is added after the next piece is inserted and the relative position of all pieces is known. Hence it is straight forward to derive the relationship between the cuts and the pieces to form constraints in the model. Constraint set (11) are to prevent the insertion of the next piece, p, in a position that crosses the guillotine cuts. In this section we describe how the guillotine cuts are derived outside of the model and how to formulate constraint set (11).

Since each cut separates two pieces, it is convenient to assign a guillotine cut to a piece. One way to achieve this is to associate the guillotine cut to the piece that contains the concurrent edge, called *associated guillotine cuts* (AGC). In this case some pieces may have no associated cuts and others may have one or more. The other approach is to associate the new guillotine cut with the last piece inserted, called *iterated guillotine cuts* (IGC), where every piece apart from the first will have an associated guillotine cut. We describe each strategy below.

4.1 AGC

The following explanation uses the example presented in Figure 4. For AGC and IGC, the first piece (p_1) is placed at the bottom left corner of the bin, testing each rotation angle (see section 5) and selecting the one that minimizes the objective function. Ties are broken by selecting the orientation that maximizes the length of the edges that are concurrent with the edges of the stock sheet. There is no need to define a guillotine cut as it does not need to be separated from another piece. So the model, defined in Section 3, to insert the second piece (p_2) only has the containment constraints (4), (5), (6) and (7) for both pieces, and the non-overlapping constraints, (8) and (9), between the placed piece, p_1, and the new piece, p_2. The solution of this model provides the coordinates of both pieces in the bin and we have to identify a guillotine cut.

Since pieces are convex polygons, there is at least one edge of one piece which can be used as a valid guillotine cut. The first valid guillotine cut found is used. For AGC, the guillotine cut is associated with the piece that contains the edge that defines the guillotine cut. In the example in Figure 4, the guillotine cut is concurrent with an edge of p_2, so it is associated with p_2. Therefore, the relative position between p_2 and this guillotine cut is fixed throughout the construction process. In other words, wherever piece p_2 is moved after the solution of successive MIPs, the guillotine cut will be moved with it.

The first guillotine cut in this example is $g^1(p_2, e_l^2)$, where p_2 is the associated piece and e_l^2 is the edge of p_2 that defines the guillotine cut. The endpoints of this guillotine cut are determined by the intersection between the cut and the edges of the bin. The inequality defined by $g^1(p_2, e_l^2)$ which separates p_1 and p_2 (inequality (10)) has the following structure, where $\alpha_{p_1,p_2,1}$, $\beta_{p_1,p_2,1}$ and $\gamma_{p_1,p_2,1}$ are the coefficients needed to define the inequality and the third subscript indicates that this is the first constraint associated with p_2:

$$\alpha_{p_1,p_2,1}(x_{p_1} - x_{p_2}) + \beta_{p_1,p_2,1}(y_{p_1} - y_{p_2}) \leq \gamma_{p_1,p_2,1} \tag{16}$$

The next model will insert p_3 and contain all constraint sets. Constraint set (10) includes the only guillotine cut $g^1(p_2, e_l^2)$ defined by (16). Constraint set (11) must prevent piece p_3 crossing $g^1(p_2, e_l^2)$. Since we do not know which side of g^1 piece p_3 will lie, we add the following three inequalities that ensure it will lie entirely on one side or the other:

$$\alpha_{p_2,p_3,1}(x_{p_3} - x_{p_2}) + \beta_{p_2,p_3,1}(y_{p_3} - y_{p_2}) \leq \gamma^R_{p_2,p_3,1} + (1 - \chi^R_{p_2,p_3,1})M \tag{17}$$

$$\alpha_{p_2,p_3,1}(x_{p_3} - x_{p_2}) + \beta_{p_3,p_2,1}(y_{p_3} - y_{p_2}) \leq \gamma^L_{p_2,p_3,1} + (1 - \chi^L_{p_2,p_3,1})M \tag{18}$$

$$\chi^R_{p_2,p_3,1} + \chi^L_{p_2,p_3,1} = 1 \tag{19}$$

Note that the lines that define the guillotine cuts have the same orientation as the edge of the piece. All pieces are assumed to have counterclockwise orientation. Hence, in Figure 4 the guillotine cut runs from the top left of the stock sheet with decreasing gradient towards the bottom right, so p_2 is to the left of the cut and p_1 is to the right. Constraint (17) forces p_3 to be placed to the right of $g^1(p_2, e_l^2)$ when the corresponding binary variable $\chi^R_{p_2,p_3,1} = 1$, while constraint (18) forces p_3 to be placed to the left of $g^1(p_2, e_l^2)$ when the corresponding binary variable $\chi^L_{p_2,p_3,1} = 1$. When binary variable $\chi^R_{p_2,p_3,1}$ or $\chi^L_{p_2,p_3,1}$ takes the value 0, then the big-M constant relaxes the constraint. Constraint (19) allows one of constraints (17) and (18) to hold true.

The model is free to move the position of p_1 and p_2 while respecting that they must remain on opposite sides of $g^1(p_2, e_l^2)$, and $g^1(p_2, e_l^2)$ remains concurrent with edge e_l^2 of p_2. Recall that each piece has an origin at the bottom left corner of their bounding box and all calculations are made relative to that origin. As a result, coefficients $\alpha_{p_2,p_3,1}$ and $\beta_{p_2,p_3,1}$ are the same in both inequalities (17) and (18) because the lines are parallel, while coefficients $\gamma^R_{p_2,p_3,1}$ and $\gamma^L_{p_2,p_3,1}$ are different because the vertex which touches the guillotine cut and maintains all the vertices on one side, is different depending on the side the piece lies.

Figure 4(c) shows that the model places p_3 on the same side of $g^1(p_2, e_l^2)$ as p_1. Note that the position of pieces p_1 and p_2 and guillotine cut $g^1(p_2, e_l^2)$ have changed when piece p_3 is inserted. The position of p_3 arose from setting $\chi^R_{p_2,p_3,1} = 1$. For the next model, $\chi^R_{p_2,p_3,1} = 1$ is fixed, constraint (17) becomes part of constraint set (10) and constraints (18) and (19) are discarded. Next, we identify guillotine cut, g^2, which separates p_3 from p_1. This time an edge of p_3 defines the next guillotine cut, $g^2(p_3, e_l^3)$, which is associated with p_3. Here, the top and the bottom limits of $g^2(p_3, e_l^3)$ are given by the intersections with the edges of the bin.

The model to insert p_4 includes three constraints in set (10): two arising from cut g^1: (16) between p_1 and p_2 and (17) between p_2 and p_3, and one arising from g^2 that separate p_1 and p_3. In order to guarantee that p_4 is going to respect $g^1(p_2, e_l^2)$ and $g^2(p_3, e_l^3)$, constraint set (11) is as follows:

$$\alpha_{p_2,p_4,1}(x_{p_4} - x_{p_2}) + \beta_{p_2,p_4,1}(y_{p_4} - y_{p_2}) \leq \gamma^R_{p_2,p_4,1} + (1 - \chi^R_{p_2,p_4,1})M \tag{20}$$

$$\alpha_{p_2,p_4,1}(x_{p_4} - x_{p_2}) + \beta_{p_2,p_4,1}(y_{p_4} - y_{p_2}) \leq \gamma^L_{p_2,p_4,1} + (1 - \chi^L_{p_2,p_4,1})M \tag{21}$$

$$\chi^R_{p_2,p_4,1} + \chi^L_{p_2,p_4,1} = 1 \tag{22}$$

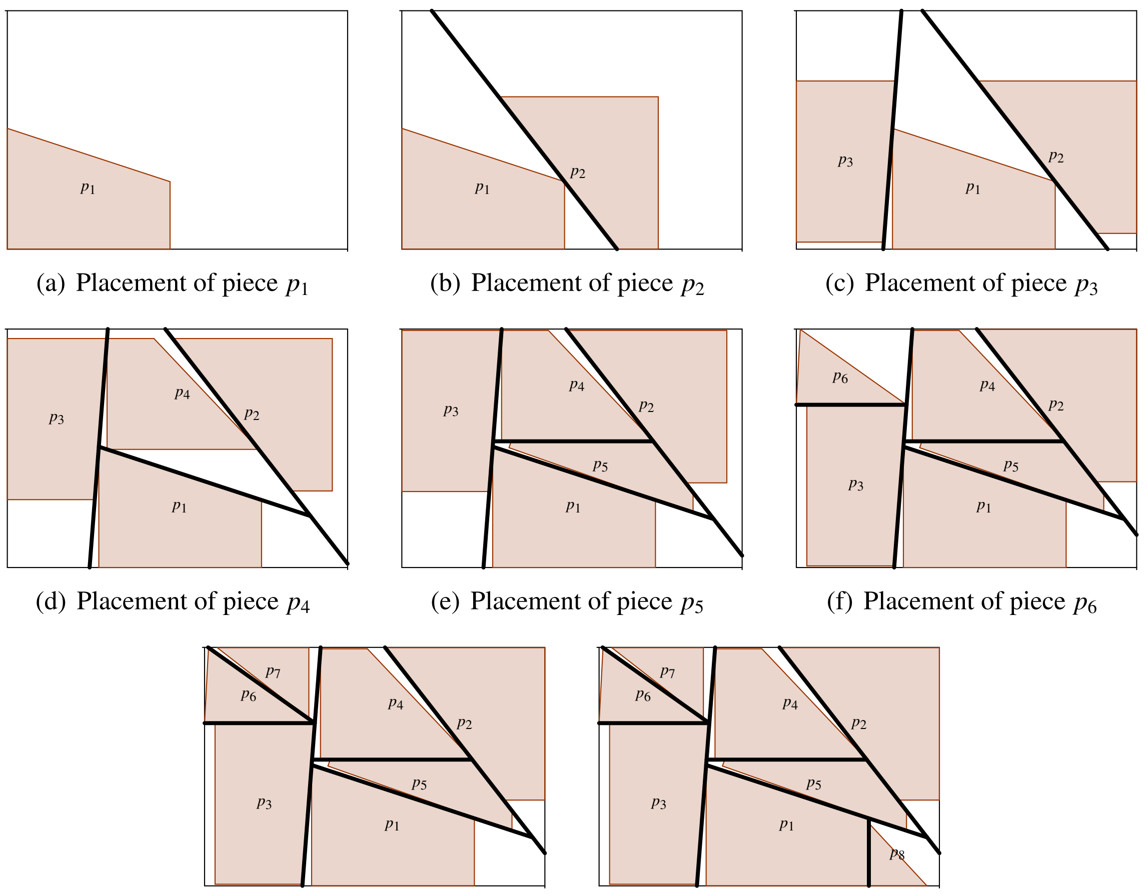
$$\alpha_{p_3,p_4,1}(x_{p_4} - x_{p_3}) + \beta_{p_3,p_4,1}(y_{p_4} - y_{p_3}) \leq \gamma^R_{p_3,p_4,1} + (1 - \chi^R_{p_3,p_4,1})M \tag{23}$$

$$\alpha_{p_3,p_4,1}(x_{p_4} - x_{p_3}) + \beta_{p_3,p_4,1}(y_{p_4} - y_{p_3}) \leq \gamma^L_{p_3,p_4,1} + (1 - \chi^L_{p_3,p_4,1})M \tag{24}$$

$$\chi^R_{p_2,p_4,1} = \chi^R_{p_3,p_4,1} + \chi^L_{p_3,p_4,1} \tag{25}$$

As before, the model is seeking to place p_4 entirely on one side of each of the guillotine cuts. If it is placed to the left of $g^1(p_2, e_l^2)$ (with p_2), then $\chi^L_{p_2,p_4,1} = 1$, $g^1(p_2, e_l^2)$ holds by constraint (21) and $g^2(p_3, e_l^3)$ is redundant by constraints (22) and (25). If it is placed to the right of $g^1(p_2, e_l^2)$ (not with p_2), then $\chi^R_{p_2,p_4,1} = 1$, $g^1(p_2, e_l^2)$ holds by constraint (20) and $g^2(p_3, e_l^3)$ holds where either $\chi^R_{p_3,p_4,1} = 1$ or $\chi^L_{p_3,p_4,1} = 1$, but not both, permitted by constraint (25). In this case p_4 may be placed to the left of $g^2(p_3, e_l)$ (with p_3), then $\chi^L_{p_3,p_4,1} = 1$ and constraint (24) holds, or to the right of $g^2(p_3, e_l^3)$ (with p_1), then $\chi^R_{p_3,p_4,1} = 1$ and constraint (23) holds.

Figure 4(d) shows the insertion of p_4. We can see that the new guillotine cut that separates p_1 and p_4 ends at intersections with previous guillotine cuts instead of with the edges of the bin. This new guillotine cut is associated to p_1, so will be denoted as $g^3(p_1, e_l^1)$. Since the guillotine cuts are performed in order, $g^1(p_2, e_l)$ and $g^2(p_3, e_l)$ are made before performing this cut, the cut is valid. Also observe that while maintaining their relative positions, p_2 and p_3 and their associated guillotine cuts have moved upwards making room for p_4. The position of p_4 arose from setting $\chi^R_{p_2,p_4,1} = 1$, $\chi^R_{p_3,p_4,1} = 1$. For the next model relative position between p_4 and the three guillotine cuts are fixed, so constraints (20), (23) and the one given by the new guillotine cut $g^3(p_1, e_l^1)$ needed to separate p_1 and p_4, which match with one inequality defined in (8), become part of constraint set (10) and the other constraints involving p_4 with any other piece



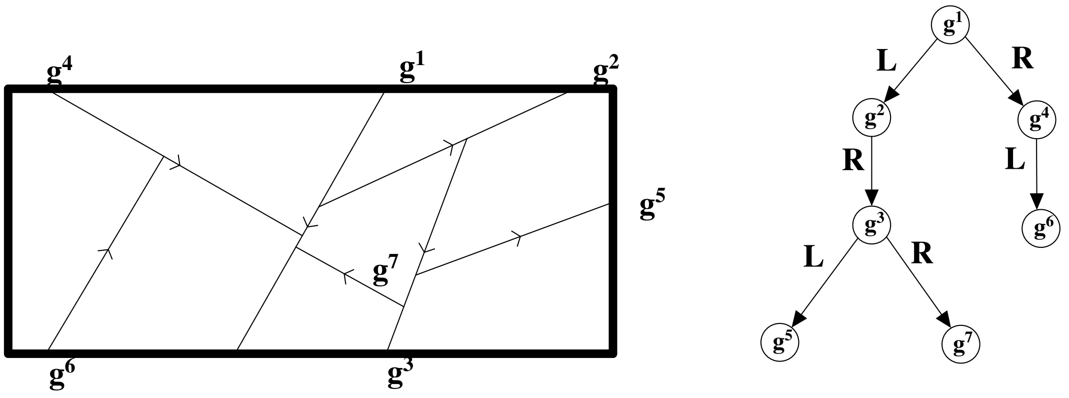
(a) Placement of piece p_1 (b) Placement of piece p_2 (c) Placement of piece p_3

(d) Placement of piece p_4 (e) Placement of piece p_5 (f) Placement of piece p_6

(g) Placement of piece p_7 (h) Placement of piece p_8

Figure 4: *Example of the packing of a bin with the AGC structure.*

are discarded. Therefore, if piece p_h has been inserted, the model we build for the insertion of p_{h+1} will have $h-1$ more inequalities in (10) than in the previous iteration.



(a) (b)

Figure 5: *Hierarchical structure of the guillotine cuts*

We can add the rest of the pieces iteratively, as shown in the other drawings in Figure 4. In order to provide a general form for defining the guillotine cuts, it is worth highlighting that the relationship between cuts is hierarchical, and when inserting a piece on a given side of a guillotine cut, the predecessors of the cut are included in the set of active guillotine constraints. For example see Figure 5. The guillotine cuts are illustrated in 5a without showing the pieces, the corresponding hierarchy is illustrated in 5b. If we were placing a piece on the right side of g^3, where a feasible placement would result in cut g^7, the active guillotine cuts are predecessors g^2 and g^1. The other cuts do not impose any restriction on this placement. Also note that g^1 is always active.

In general, let p be the next piece to be inserted, where there are $l-1$ pieces in the bin and separated by $l-2$ guillotine cuts. For each guillotine cut g^t, $t = 2, \ldots, l-2$, we know which of the guillotine cuts is the immediate predecessor of t denoted by $t^* \in \{1, \ldots, t-1\}$.

We denote by p_t the piece associated to the guillotine cut $g^t, t = 1, \ldots, l-2$. Since it might be possible that two or more guillotine cuts are associated with one piece then we use a three index notation in order to determine which guillotine cut of those associated to piece p_t is being used. Therefore, $\alpha_{p_t,p,s}, \beta_{p_t,p,s}$ and $\gamma^R_{p_t,p,s}$ are the coefficients needed to write the constraint to ensure that piece p is going to be placed on the right hand side of g^t, which corresponds to the s-th guillotine cut associated to piece p_t. Then, in a general form, inequalities (11) can be written as follows:

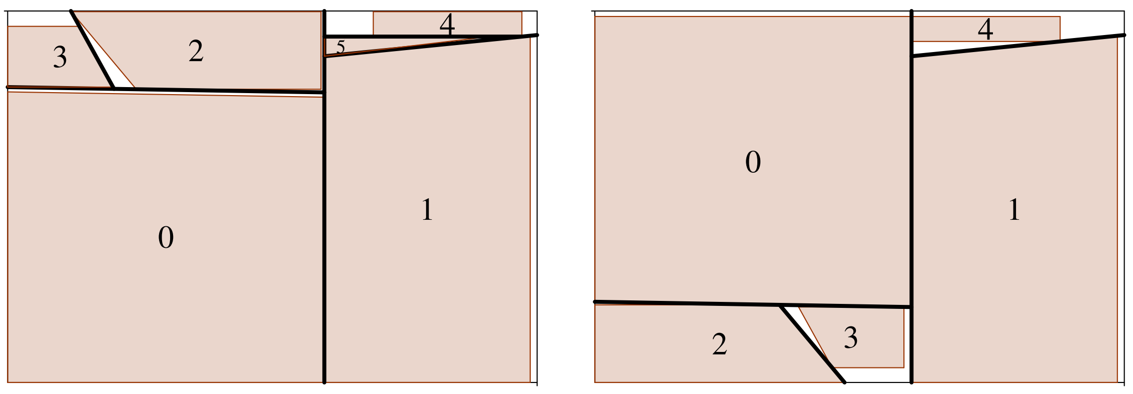
$$\alpha_{p_t,p,s_t}(x_p - x_{p_t}) + \beta_{p_t,p,s_t}(y_p - y_{p_t}) \leq \gamma^R_{p_t,p,s_t} + (1 - \chi^R_{p_t,p,s_t})M \qquad t = 1, \ldots, l-2 \tag{26}$$

$$\alpha_{p_t,p,s_t}(x_p - x_{p_t}) + \beta_{p_t,p,s_t}(y_p - y_{p_t}) \leq \gamma^L_{p_t,p,s_t} + (1 - \chi^L_{p_t,p,s_t})M \qquad t = 1, \ldots, l-2 \tag{27}$$

$$\chi^R_{p_1,p,1} + \chi^L_{p_1,p,1} = 1 \tag{28}$$

$$\chi^\sigma_{p_{t^*},p,s_t} = \chi^R_{p_t,p} + \chi^L_{p_t,p} \qquad t = 2, \ldots, l-2 \tag{29}$$

Inequalities (26) and (27) define the guillotine cuts, which are activated by binary variables $\chi^R_{p_k,p_t,s}$ and $\chi^L_{p_k,p_t,s}$, respectively. Equality (28) ensures that g^1 is always satisfied by the new piece (p). The remaining guillotine cuts are imposed on p using equalities (29) in the same way as



(a) AGC (b) IGC

Figure 6: *Difference between AGC and IGC (real example on instance H120).*

described in the example above, where σ will take the term R or L depending on whether the immediate predecessor cut is on the right (R) or left (L). For example in Figure 5, when placing piece 8 in such a way that it would result in defining cut g^7, $\sigma = R$.

4.2 IGC

The *Iterated Guillotine Cut* structure associates each guillotine cut with the last piece inserted into the bin. While the cuts are still defined by an edge of one of the pieces it separates, the edge may be from a previously placed piece but associated to the newly placed piece. The advantage is that this association is easier to implement since each piece has only one cut association and the cuts are in the same order as the piece allocation. The MIP formulation remains the same, i.e. inequalities (10) and (11) are the same in both structures.

Figure 6(a) shows two different arrangement of pieces generated by each type of association. With the AGC structure the constructive algorithm places five pieces instead of the four placed when using the IGC structure. The algorithm works similarly in both cases until piece four is placed. The first two pieces are separated in both cases by a vertical line which is associated with piece one. The second guillotine cut separates pieces zero and two and is associated with piece two in both cases. The third guillotine cut is associated with piece three in both cases. When piece four is placed, with the AGC structure the new guillotine cut is associated with piece one with which it has a concurrent edge, while with the IGC structure it is associated with piece four, the last piece placed. Then, piece five (a tiny triangle) fits into the packing on the left-hand side of Figure 6(a) because piece four can be moved to the top of the bin while the cut stays with piece one, making room for piece five. In the packing on the right-hand side of Figure 6(b), if piece four is moved to the top of the bin, the guillotine cut has also to be moved upwards and there is no feasible placement for piece five.

5 Rotations and reflections

The above discussion assumes a fixed rotation and reflection of each piece. However, in reality a piece may be freely rotated and, in some cases, may be reflected (mirror image). In this section we describe how we identify a finite set of orientations, reflected and not reflected, for each piece. Since we are selecting a small number of orientations from an infinite set, it is important to identify orientations that are likely to fit well with the rectangular bin and the pieces already in the bin. With this aim in mind, algorithm *Get Rotations* (GR), identifies rotation angles by matching the edges of the next piece with the edges of the bin and the pieces in the bin.

Let n_r be the maximum number of different orientations we wish to test, p be the next piece and n_p the number of edges of p. Generate all the rotation angles for p so that each of its edges coincide with each edge of the bin ($4 * n_p$ matches) and with each edge of each piece already placed into the bin ($n_p(n_1 + n_2+, \dots)$). Note that some rotation angles will be generated several times, only unique angles are stored along with the number of times they are generated. If the total number of angles is less than n_r, then algorithm GR returns all these angles. If it is greater than n_r, select n_r rotation angles according to the following sorting criteria:

a) Non-increasing number of times the rotation angle is generated.

b) In the case of a tie in *(a)*, the largest total length of the edges that match at that rotation angle.

We test a number of different strategies for setting n_r and for testing the value of using the mirror image of the piece, as follows:

- *3R&3R*: $n_r = 6$, three orientations for the original and three for the reflected piece.
- *6R*: $n_r = 6$, six orientation for the original and no reflection.
- *3R+1&3R+1*: same as *3R&3R* for the first piece inserted, then increase by one for each reflection every time a new piece is inserted.
- *3R+3&3R+3*: same as *3R&3R* for the first piece inserted, then increase by three for each reflection every time a new piece is inserted.
- *5R+5&5R+5*: same as *3R+3&3R+3* but starting with five orientations per reflections and increasing by five for each reflection every time a new piece is inserted.
- *E30*: Every 30^o for the original and the reflected piece. The first rotation is given by matching the longest edge of the piece with the bottom edge of the bin.
- *E10*: Similar to *E30*, but considering the rotations every 10^o.

6 Constructive algorithm

The constructive algorithm uses the set of angles of rotation and reflection for each piece to sequentially place the pieces in the bin using the MIP, described above. Algorithm 1 details the basic procedure for the constructive heuristic.

Given an initial permutation of pieces and an open bin, for the next piece generate the rotation angles using algorithm *GR*. For each rotation solve the MIP. If the piece can be feasibly placed in the bin, select the placement arising from the rotation angle that gives the best objective function value, otherwise leave the piece unpacked and move on to the next. Once no other piece can feasibly fit in the bin, close the bin and open a new bin. Start packing from the first unpacked piece in the list in the new bin. Once all pieces are packed, rebuild the bin with the lowest utilization, once using objective functions $FO1$ and once using objective function $FO2$. Apply a horizontal or vertical cut respectively to find the fractional number of bins.

In order to solve a problem instance, we call the MIP several times for each piece. The computational effort of the constructive algorithm depends on the number of pieces in a problem instance and on the number of rotations permitted for each piece. Setting good upper bounds improves the efficiency of the MIP. The first attempt to insert a piece at a given angle of rotation has an upper bound equal to the value of the objective function when $L_c = L$ and $W_c = W$. If the MIP provides a feasible solution, then the objective function value can be used as an upper bound for the following insertion of the same piece in the remaining angles of rotation. Otherwise, the bounds remain the same as for the first insertion.

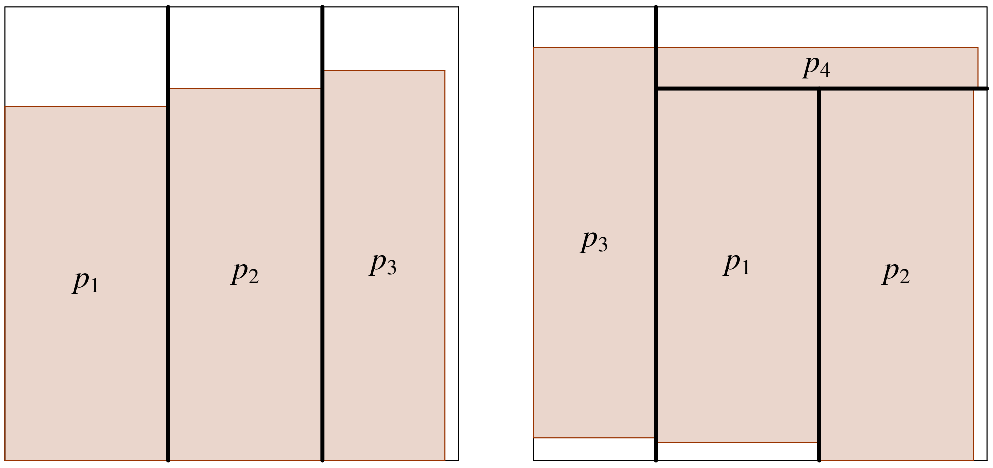
6.1 Two phase constructive algorithm

The constructive algorithm described above identifies the guillotine cuts after each piece insertion, these are then set as several constraints in the MIP model, described in Section 4, which force the next piece to respect all the guillotine cuts previously defined. Setting the guillotine cuts in this way can over constrain the solutions space. For example see figure 7(a), where the first set of guillotine cuts prevents p_4 from being placed, while the second layout allows the placement with a feasible set of guillotine cuts. Since the guillotine cut inequalities reduce the feasible region for placing the new piece, we propose a modification to the constructive heuristic that inserts the next piece ignoring the guillotine cut constraints (inequalities 11 in Section 3). After the best insertion is found, we then perform a procedure to identify a new guillotine cut structure, if one exists. This is called the two phase constructive algorithm.

The modification to the original MIP formulation (defined in Section 3) is straight forward. The new MIP is simply a relaxation by removing constraint set (11). We refer to it as MIP$'$. Since MIP$'$ is a relaxation of the original MIP, if MIP$'$ is infeasible then so is the original MIP.

Given a feasible solution to MIP$'$, we search for a guillotine cut structure using a simple algorithm, called *Finding a Guillotine Cut Structure* (FGCS), that works as follows. Test each edge of every piece, in the order the pieces are inserted, as a feasible guillotine cut i.e. a cut concurrent with the edge that does not divide any piece and pieces lie on both sides of the cut. On finding a feasible cut, divide the bin into two parts using the cut. For each part, repeat the same procedure until all the pieces are separated, or stop if no cut can be found.

In general the two-phase procedure for inserting a piece p works as follows: For each rotation, insert p using MIP$'$. Given a feasible solution, call FGCS. If MIP$'$ fails to find a feasible solution for all rotations, move on to the next piece. If FGCS fails to find a feasible guillotine cut structure for all feasible solutions to MIP$'$, call the original MIP and proceed in the same way as the basic construction heuristic for this piece.



(a) *Constructive Algorithm* (b) *Constructive Algorithm with two phases*

Figure 7: *Solutions obtained by a constructive algorithm and a constructive algorithm with two phases.*

6.2 Embedding an improvement procedure into the constructive algorithm

Local search has been successfully applied to a wide range of packing problems and instinctively would seem like an interesting avenue to pursue here. However, all our experimentation over a wide range of neighborhood structures resulted in little benefit and high computational times. Instead, we developed an improvement procedure embedded in the construction heuristic to improve the bin utilization before it is closed and a new bin opened. In order to avoid wasting computational time, the procedure is only applied to bins with utilization below a given threshold, κ. In our experiments we set κ to 0.95.

The guillotine cuts effectively define containment polygons around each piece. Comparing the area of the piece to the area of the containing polygon provides a measure of space utilization of the specific piece. Figure 8 shows the containment polygon of piece p_6, which has the lowest space utilization. In this example, the utilization in the containment polygon is lower than 0.5. The improvement procedure removes the piece with the worst utilization and rebuilds the bin, without this piece, using the same angles of rotation and reflection for the retained pieces. Then we try to insert the remaining pieces on the unpacked list. Finally we try to insert the removed piece last considering ten alternative rotations for each reflection. As before, algorithm GR produces the angles of rotation by matching the edges of the piece with the edges of the other pieces, but in this case, ignoring the edges of the bin. If the utilization of the repacked bin is higher, then accept the solution and identify the next piece with the lowest utilization of its containing polygon. The procedure continues until no improvement can be found.

7 Lower bounds for the total number of bins

In order to assess the quality of the solutions obtained, we have computed a simple lower bound for N by solving a 1-dimensional bin packing model. This model uses an upper bound of N_{ub}, given by the constructive algorithm. In order to indicate that bin i is open we use a binary variable y_i, which takes the value 1 if the bin is used in the solution. We consider binary

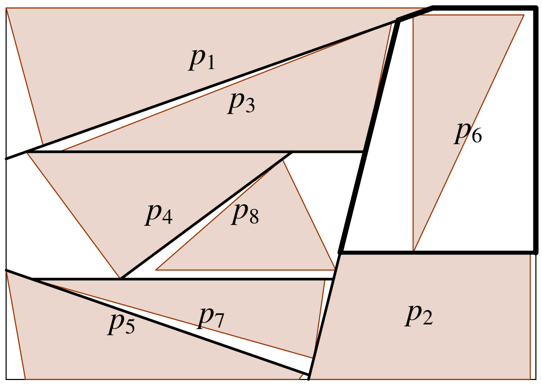


Figure 8: *Containment polygon of piece p_6*

variables x_{ij}, which take the value 1 if piece j is placed on bin i, and 0 otherwise. The model can be written as follows:

$$
\begin{array}{llll}
 & Min \ \sum_{i=1}^{N_{ub}} y_i & & (30) \\
s.t. & \sum_{j=1}^{n} a_j x_{ij} \leq a_b y_i & i = 1, ..., N_{ub} & (31) \\
 & \sum_{i=1}^{N_{ub}} x_{ij} = 1 & j = 1, ..., n & (32) \\
 & x_{ij} \in \{0,1\}, y_i \in \{0,1\}, & 1 \leq j \leq n, 1 \leq i \leq N_{ub} & (33)
\end{array}
$$

where n denotes the total number of pieces, N_{ub} is an upper bound for the total number of bins, a_j is the area of piece $j \in \{1, \ldots, n\}$ and a_b the area of one bin. In the objective function we try to minimize the total number of bins used. Inequalities (31) ensure that the total area of pieces placed in bin $i \in \{1, \ldots, N_{ub}\}$ must be less than or equal to the area of the bin. Finally, equalities (32) force each piece to be placed exactly in one open bin.

8 Computational experiments

We divide the computational experiments into four parts. In subsection 8.1 we compare the different versions of the constructive algorithm. Once the best version is identified, in subsection 8.2 we show the improvement achieved by the constructive algorithm with two phases and the improvement procedure. Finally, in subsections 8.3 and 8.4 we compare with the state of the art algorithms on irregular packing and rectangular packing respectively. The computational experiments were coded in Microsoft Visual Studio C++ 2008 and run on a Intel core i5 with 2.4 GHz PC with 4 GB of RAM. The MIP models were solved using CPLEX solver, version 12.5.0.0.

We test the performance of the alternative variants using the test data presented by Han et al. (2012). They consider eight instances, four provided by a company in glass cutting for conservatories and another four generated using properties of the industrial data. The number of pieces ranges between 40 and 149. The instance name is coded by a letter and a number: the letter can be J or H depending on whether the instance is provided by a company (J) or is generated (H); the number represents the total number of pieces to be packed into the bins. For each data set the pieces have between 3.5 and 4 edges on average. The data defining the pieces is available on the ESICUP website (http://paginas.fe.up.pt/ esicup/tiki-index.php), and the length and the width of the bin are $L = 225$ and $W = 321$ respectively.

8.1 Finding the best constructive algorithm

In this subsection we study the basic constructive algorithm and test alternative design decisions. All the different versions follow the structure described in Algorithm 1 while testing alternative sorting criteria of pieces, orientation and reflections (see Section 5), objective function (see Section 3), and association of guillotine cuts (AGC and IGC).

We investigate three alternative sorting criteria for the pieces at the beginning of the process, these are: (i) Randomly. (ii) Non-increasing area. (ii) By shape: pieces are classified by those similar to a rectangle and those that are not (usually triangular). Given the enclosing rectangle of a piece, if the area of the piece occupies more that 70% of the enclosing rectangle then it is classified rectangular. Within each class, pieces are sorted by non-increasing area. Rectangular pieces are packed first.

The default settings for the constructive algorithm (CA1), sorts the pieces by non-increasing area as the initial permutation, uses the best 3 angles of rotations for both the original and reflected piece (*3Rx3R*), objective function $FO0$ and the AGC guillotine cut structure. The other variants are the same as CA1 with the following modifications:

CA2: initially pieces are sorted randomly.

CA3: initially pieces are sorted by shape.

CA4: objective function $FO1$.

CA5: objective function $FO2$ (see Section 3).

CA6: piece rotation strategy is *6R*.

CA7: piece rotation strategy is *3R+1&3R+1*.

CA8: piece rotation strategy is *3R+3&3R+3*.

CA9: piece rotation strategy is *5R+5&5R+5*.

CA10: piece rotation strategy is *E30*.

CA11: piece rotation strategy is *E10*.

CA12: guillotine cuts association is IGC.

Table 1 shows the total number of bins used to pack all the pieces using these versions of the constructive algorithm and the lower bound (LB). Since $CA2$ is random, the result is the average of 10 random permutations. Table 2 shows the fractional number of bins used. Finally, Table 3 shows the computational time in seconds.

Table 1: *Number of bins used*

Instances	CA1	CA2	CA3	CA4	CA5	CA6	CA7	CA8	CA9	CA10	CA11	CA12	LB
J40	8	9	8	8	8	8	8	8	8	8	8	8	7
J50	10	11	10	10	10	10	10	10	10	10	10	10	8
J60	11	12	11	11	11	11	11	11	11	11	11	11	9
J70	12	13.5	12	12	12	13	12	12	12	12	12	12	10
H80	10	11	10	10	10	10	10	10	10	10	10	10	9
H100	16	17.7	17	16	16	17	16	16	16	16	16	16	14
H120	16	17.6	17	16	17	17	16	16	16	17	16	17	14
H149	22	24.5	23	23	23	23	22	22	22	22	22	22	20
Total	105	116.3	108	106	107	109	105	105	105	106	105	106	91

Table 2: *Fractional number of bins used*

Instances	CA1	CA2	CA3	CA4	CA5	CA6	CA7	CA8	CA9	CA10	CA11	CA12
J40	7.40	8.45	7.38	7.45	7.43	7.70	7.25	7.21	7.21	7.33	7.25	7.52
J50	9.27	10.50	9.37	9.24	9.31	9.53	9.31	9.16	9.16	9.36	9.25	9.38
J60	10.35	11.54	10.54	10.52	10.40	10.68	10.35	10.21	10.22	10.36	10.28	10.41
J70	11.63	13.03	11.85	11.80	11.78	12.18	11.57	11.66	11.62	11.79	11.74	11.79
H80	9.46	10.44	9.47	9.48	9.46	9.34	9.40	9.30	9.35	9.43	9.33	9.55
H100	15.56	17.27	16.12	15.87	15.82	16.16	15.62	15.43	15.47	15.94	15.54	15.75
H120	15.75	17.19	16.17	15.65	16.24	16.39	15.69	15.65	15.74	16.14	15.95	16.21
H149	21.83	24.03	22.39	22.09	22.24	22.09	21.88	21.72	21.77	21.87	21.75	21.83
Total	101.27	112.45	103.30	102.10	102.67	104.06	101.07	100.33	100.54	102.21	101.08	102.43

The comparison between $CA1$, $CA2$ and $CA3$ in Table 1 shows that the best sorting criterion is non-increasing area ($CA1$). We can see that $CA2$ is always inferior to $CA1$, and $CA3$ is inferior to $CA1$ in the last three instances. Furthermore, Table 2 shows that $CA1$ obtains better results than $CA3$ in all but instance $J40$ where it is slightly inferior.

In order to decide which objective function produces better results, we compare $CA1$, $CA4$ and $CA5$. According to Table 1 there is little to chose between the objective functions. Table 2 shows that $CA1$ is better than $CA4$ and $CA5$ apart from instances $J50$ and $H120$, therefore the weighted objective function $FO0$ works better than both $FO1$ and $FO2$.

The advantages of using reflection can be seen by comparing $CA1$ and $CA6$. Each insertion tries 6 different orientations of one piece, $CA1$ considers the best three rotations of both original and reflected polygons and $CA6$ does not take into account the reflection. We can see from Table 2 that the results are clearly better if we consider the reflected polygons.

Algorithm $CA1$ considers 6 different orientations of the piece which is going to be inserted. This means that 6 MIP models are solved to optimality in order to decide the relative position between the new inserted piece and the pieces already placed. Algorithms $CA7$, $CA8$, $CA9$, $CA10$ and $CA11$ consider more rotations for both polygons, original and reflected, of a given piece. In Table 3 we can see that the computational time increases, $CA11$ being the slowest algorithm (note that at each insertion $CA11$ 72 MIPs are solved to optimality). Table 1 shows that all these algorithms produce results with the same number of bins with the exception of $CA10$ which obtains a worse result on instance $H120$. It is interesting to note that matching edges ($CA7$, $CA8$, $CA9$) works better than a fixed increment in the angle of rotation ($CA10$, $CA11$). The best results are given by $CA8$, which produces the best results for 6 of 8 instances. However, the computational time of $CA8$ increases considerably in comparison with $CA1$.

Finally, as suspected AGC structure produces better solutions than the IGC, shown by comparing $CA1$ using AGC with $CA12$ using IGC.

Table 3: *Time in seconds*

Instances	CA1	CA2	CA3	CA4	CA5	CA6	CA7	CA8	CA9	CA10	CA11	CA12
J40	14	28	15	14	16	41	31	83	105	58	177	15
J50	18	45	22	20	19	24	52	89	108	95	237	22
J60	32	67	45	73	36	65	96	204	180	162	440	39
J70	62	100	121	93	129	44	142	334	305	281	609	55
H80	155	227	131	112	158	117	354	610	545	449	1257	99
H100	124	215	97	223	177	92	294	582	587	720	1286	108
H120	326	456	149	288	186	158	771	1198	1424	782	2769	194
H149	624	627	235	225	208	193	1042	1529	1525	1307	3369	250
Total	1355	1765	813	1048	928	734	2782	4627	4779	3855	10145	783

8.2 Comparison of the constructive algorithms with two phases and the improvement procedure

Table 4 shows the comparison between the original $CA1$, the $CA1$ configuration using the two-phase constructive algorithm ($CA1 - 2Ph$) and $CA1 - 2Ph$ with the improvement procedure ($CA1 - 2Ph - imp$) described in Section 6.2. Comparing $CA1$ and $CA1 - 2Ph$, the computational time remains similar and the quality of the solutions is slightly improved. Adding the improvement phase, shows a more significant improvement in solution quality. The number of bins in instances $J40$, $J50$ and $J60$ are reduced and in instances $H100$ and $H120$ the fractional number of bins is reduced. That is, on 5 instances this improvement procedure obtains a better solution. However, there is at a significant increase in computational times. Note that solution using $CA2 - 2Ph - imp$ find an optimal solution in instance J40 (respect the total number of bins) and the three algorithms find solutions at most two bins above the lower bound.

Table 4: *Comparing the initial and the two-phase constructive algorithms*

	CA1				CA1-2Ph				CA1-2Ph-imp				LB
	N	U	F	T	N	U	F	T	N	U	F	T	N
J40	8	0.82	7.40	14	8	0.83	7.31	21	7	0.88	6.92	168	7
J50	10	0.82	9.27	18	10	0.83	9.23	26	9	0.85	8.97	344	8
J60	11	0.84	10.35	32	11	0.83	10.49	68	10	0.87	9.99	445	9
J70	12	0.85	11.63	62	12	0.88	11.46	99	12	0.86	11.54	703	10
H80	10	0.86	9.46	155	10	0.89	9.20	121	10	0.89	9.21	1275	9
H100	16	0.86	15.56	124	16	0.87	15.33	165	16	0.88	15.27	1412	14
H120	16	0.87	15.75	326	16	0.87	15.74	488	16	0.89	15.37	2406	14
H149	22	0.88	21.83	624	22	0.89	21.57	524	22	0.89	21.59	3314	20
Total	**105**		**101.27**	**1355**	**105**		**100.13**	**1513**	**102**		**98.87**	**10068**	

8.3 Comparison with the best known algorithms

Han et al. (2012) propose several versions for the one step algorithm depending on two parameters: θ is the threshold for accepting matches of blocks and K controls the linearity of the dynamic weighting scheme. The best two algorithms using the one-step approach are given by the following combinations: *1S-0.94-5*, with $\theta = 0.94$ and $K = 5$ and *1S-0.97-3*, with $\theta = 0.97$ and $K = 3$.

The two-step algorithm (*2S*) also proposed by Han et al. (2012) is in general inferior to the one-step algorithms, with the exception of one instance (see *J70* in Table 5) where the two-step algorithm found a better solution with fewer bins than all the one-step algorithms.

Table 5 shows the computational results obtained by the two-step algorithm *2S* and the one-step algorithms *1S-0.94-5* and *1S-0.97-3*. The two last columns correspond to $CA1$ and $CA1 - 2Ph - imp$. The total number of bins is denoted by N, F is the fractional number of bins and T is the computational time measured in seconds. Algorithm $CA1-2Ph-imp$ produces the best known results for five of the eight instances (the best known solution of instance $J70$, $H80$ and $h149$ is given by $CA1 - 2Ph$ in Table 4). The behavior of algorithm $CA1$ is also interesting because on average it works better than the algorithms proposed by Han et al. (2012), and it is faster than CA1-2Ph-imp requiring a similar computational time than *1S-0.97-3* on average. In fact, we can see that $CA1$ reduces the number of bins used in *1S-0.94-5* and *1S-0.97-3* in 4 instances.

8.4 Comparison with the state of the art algorithms in rectangular bin packing problems

The constructive algorithms proposed in this paper deal with irregular pieces and use a mathematical model which is hard to solve to optimality in each step. Nevertheless, the same approach can be directly applied to the standard bin packing problem with rectangular pieces. While there are few competing algorithms for irregular pieces to use as a benchmark, it is interesting to compare with the state of the art in rectangle bin packing. However, we emphasize that our algorithm was designed for irregular pieces.

For the bin packing problem with rectangular pieces, there is a standard benchmark set composed of 500 instances divided into 10 classes. The first 6 classes were proposed by Berkey and Wang (1987) and the last 4 classes by Lodi et al. (1999). We consider two rotations for the insertion of each piece (0^o and 90^o) and therefore we are solving the $2DBP|R|G$ problem in the typology presented in Dyckhoff (1990).

Table 6 compares the total number of bins used by the constructive algorithm $CA1$ with fast heuristic algorithms: the Knapsack-Problem-based heuristics of Lodi et al. (1999) (KP), the Guillotine Bottom-Left heuristic of Polyakovsky and MHallah (2009) (GBL) and the Constructive Heuristic of Charalambous and Fleszar (2011) (CH).

We can observe that the constructive algorithm $CA1$ is competitive, working better than GBL, slightly worse than KP and clearly worse than CH. Algorithm $CA1 - 2Ph$ produces better results than any other constructive algorithm. However, the state of the art procedure on rectangular bin packing problems with guillotine cuts is the CHBP algorithm proposed by Charalambous and Fleszar (2011), in which their constructive algorithm (CH) is followed by a post-optimization phase. CHBP requires only 7064 bins. The heuristic algorithm CFIH+J4 proposed by Fleszar (2013) obtains similar solutions than the CHBP but in a small fraction of time (the average time of the CFIH+J4 is 0.000314 seconds while CHBP requires an average of 0.66278 seconds). Besides, CA1-2Ph requires a high computational effort, being the average

Table 5: *Comparison with the algorithms proposed by Han et al. (2012)*

		2S	1S-0.94-5	1S-0.97-3	CA1	CA1-2Ph-imp
	N	8	8	8	8	7
J40	F	7.72	7.39	7.32	7.40	6.92
	T	$> 1h$	110	47	14	168
	N	10	11	10	10	9
J50	F	9.66	10.43	9.8	9.27	8.97
	T	$> 1h$	130	74	18	344
	N	11	11	11	11	10
J60	F	10.90	10.23	10.51	10.35	9.99
	T	$> 1h$	170	60	32	204
	N	12	13	13	12	12
J70	F	11.95	12.75	12.65	11.63	11.54
	T	$> 1h$	200	98	62	703
	N	10	10	10	10	10
H80	F	9.63	9.25	9.45	9.46	9.21
	T	$> 1h$	405	187	155	1275
	N	17	17	17	16	16
H100	F	16.40	16.31	16.35	15.56	15.27
	T	$> 1h$	700	201	124	1412
	N	17	17	17	16	16
H120	F	16.14	16.25	16.58	15.75	15.37
	T	$> 1h$	714	247	326	2406
	N	23	23	23	22	22
H149	F	22.29	22.33	22.41	21.83	21.59
	T	$> 1h$	947	389	624	3314
TOTAL	N	108	108	108	105	102
	F	104.69	103.06	103.72	101.27	98.87
	T	$> 8h$	3006	1145	1261	10067

time 151.68 seconds.

Table 7 shows the total number of bins used by algorithms CH, $CA1$, $CA1-2Ph$ and CHBP for each class of instances. The main differences between algorithms CH and $CA1$ appear in classes 7 and 8, the performance in the rest of the classes being similar. The differences disappears if we consider algorithm $CA1 - 2Ph$, which seems especially well suited for these types of instances. Note that $CA1$ and the other constructive algorithms presented in this paper are carefully designed to decide the position of the pieces in a given bin and do not focus on the assignment of pieces to bins. Nevertheless, they work well on rectangular bin packing problems.

9 Conclusions

In this paper we tackle a new problem in the cutting and packing literature (only one other published article) that has a real application. Our contribution includes a new approach to modelling and solving a guillotine cut structure using a mathematical programming model. To our knowledge, this is the first paper in the cutting and packing literature presenting a mathematical

Table 6: *Total number of bins in the* 10 *classes.*

	Total number of bins	CPU TIME (Milliseconds)	Processor
KP	7297	< 500	SG 195 MHz
GBL	7367	< 500	SG 195 MHz
CH	7191	5.37	Core i3 2.13 GHz
CA1	7303	104170.42	Core i5 2.4 GHz
CA1-2Ph	7146	151678.63	Core i5 2.4 GHz
CHBP	7064	66.28	Core i3 2.13 GHz
CFIH+J4	7080	0.314	Core2 2.33 GHz

Table 7: *Total number of bins in each class.*

Class	1	2	3	4	5	6	7	8	9	10
CH	997	127	705	126	894	115	792	792	2131	512
CA1	994	130	718	127	896	116	844	843	2124	511
CA1(2 phases)	984	128	695	126	878	116	792	795	2124	508
CHBP	975	124	687	125	872	113	770	776	2119	503

model which considers guillotine cuts. In addition we present a lower bound for this problem.

In modelling the problem we propose an approach for determining a finite set of rotations in algorithm (GR) and comprehensively test a range of rotation setting and the value of reflecting polygons. We also test different objective functions, initial permutations and approaches for associating a guillotine cut to a piece. We enhance the construction heuristic with a two-phase approach and an improvement heuristic. Both improve the solution quality, although the latter at a notable computational cost.

The constructive algorithm proposed obtains high quality results on the bin packing problem with guillotine cuts and irregular convex pieces, improving the best known solutions in all of the eight benchmark instances and it produces competitive results for the rectangular bin packing problem with guillotine cuts.

Acknowledgements: This study has been partially supported by the Spanish Ministry of Economy and Competitiveness, DPI2011-24977, cofinanced by FEDER funds.

Algorithm 1 Constructive algorithm structure

```
Require: $P$, $L$, $W$;
  Set $P'$ (initial permutation of pieces);
  Set $n_r$ (number of rotations);
  Set $OF$ (objective function);
  Set guillotine cut structure;
  $B = \emptyset$, $cont = 0$;
  while $P' \neq \emptyset$ do
    Create a new bin $b_{cont}$.
    for $i = 0, \ldots, |P'| - 1$ do
      Set $bestOFvalue = \omega L + (1 - \omega)W$ ($\omega$ is given by $OF$);
      $IN = false$;
      $P^*$ is the set of all polygons obtained by the different rotations of $p'_i = P'[i]$;
      if $p'_i$ has no symmetries and reflection is allowed then
        $P^*_m$ is the set of all polygons obtained by the different rotations of $m(p'_i)$ (reflected
        polygon);
      end if
      for each polygon $p \in P^* \cup P^*_m$ do
        Add $p$ to the MIP model;
        Solve the MIP model using as upper bound $bestOFvalue$;
        if model is feasible then
          $IN = true$;
          Update best rotation (and reflection) of $p'_i$.
          $bestOFvalue$ =current objective function value;
        end if
        Remove $p$ from the model.
      end for
      if $IN = true$ then
        Add $p'_i$ to $b_{cont}$
        Insert the piece into the MIP model with the best rotation (and reflection).
        Identify the new guillotine cut and update the guillotine cut constraints of the model.
        $P' = P' \setminus \{p'_i\}$
      end if
    end for
    $B = B \cup \{b_{cont}\}$;
    $cont = cont + 1$;
  end while
  Sort bins $B$ by non-decreasing waste;
  Rebuild last bin of $B$ with objectives functions $FO1$ and $FO2$ and choose the best configur-
  ation.
  return $B$;
```

References

Alvarez-Valdes, R., A. Martinez, J.M. Tamarit. 2013. A branch & bound algorithm for cutting and packing irregularly-shaped pieces. *International Journal of Production Economics.* **145** 463–477.

Bennell, J.A., J.F. Oliveira. 2008. A tutorial in irregular shaped packing problems. *European Journal of Operational Research* **184** 397–415.

Bennell, J.A., J.F. Oliveira. 2009. The geometry of nesting problems: a tutorial. *Journal of the Operational Research Society* **60** S93–S105.

Bennell, J.A., Scheithauer, G., Romanova, T., Stoyan, Y., Pankratov, A. 2011. Optimal clustering of a pair of irregular objects. *Journal of Global Optimization* `doi: http://dx.doi.org/10.1007/s10898-014-0192-0.`.

Berkey, J.O., P.Y. Wang. 1987. Two-dimensional finite bin-packing algorithms. *Journal of Operational Research Society* **38** 423–429.

Burke, E.K., R.S.R. Hellier, G. Kendall, G. Whitwell. 2010. Irregular packing using the line and arc no-fir polygon. *Operations Research* **58** 948–970.

Charalambous, C., K. Fleszar. 2011. A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. *Computers and Operational Research* **38** 1443–1451.

Cui, Y. 2014. Heuristic for the cutting and purchasing decisions of multiple metal coils. *Omega* **46** 117–125.

Dyckhoff, H. 1990. A typology of cutting and packing problems. *European Jouranl of Operational Research* **44** 144–159.

Dyckhoff, H., Kruse, H-J., Abel, D., Gal, T. 1990. Trim loss and related problems. *Omega* **13** 59–72.

Fischetti, M., I. Luzzi. 2008. Mixed-integer programming models for nesting problems. *J Heuristics* **15** 201–226.

K. Fleszar. 2013. Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. *Computers & Operations Research* **40** 463–474.

Han, W., J.A. Bennell, X. Zhao, X. Song. 2012. Construction heuristics for two dimensional irregular shape bin packing with guillotine constraints. *European Journal of Operational Research* dx.doi.org/10.1016/j.ejor.2013.04.04.

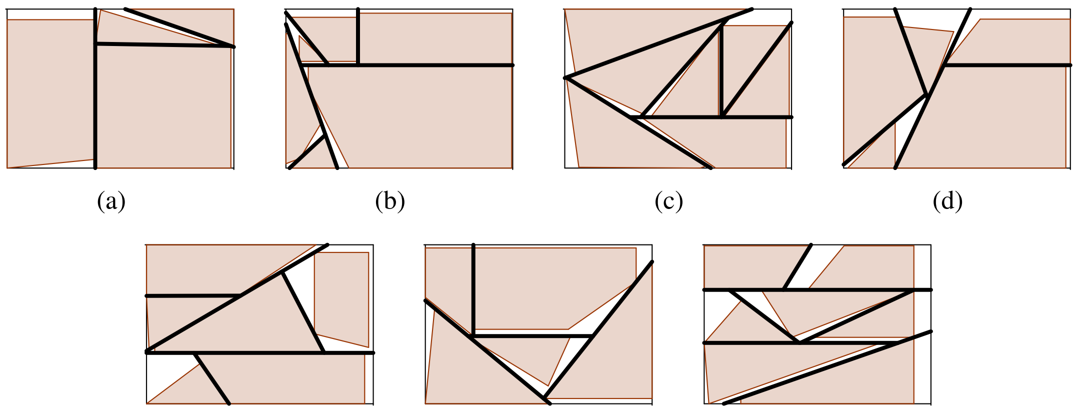
Lodi, A., S. Martello, D.Vigo. 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *Informs Journal on Computing* **11** 345–357.

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

Polyakovsky, S., R. MHallah. 2009. An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research* **192** 767–781.

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

Appendix: Solutions representation.


(a) (b) (c) (d)

(e) (f) (g)

Figure 9: *J40*

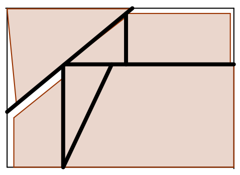
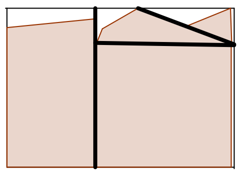
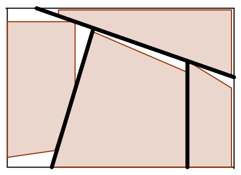
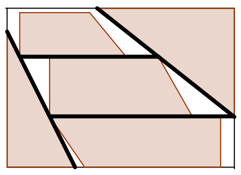
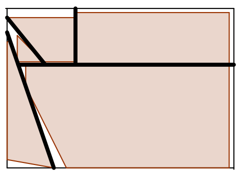
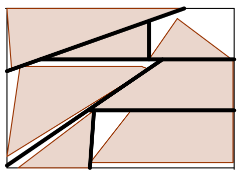
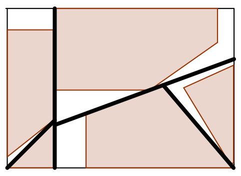
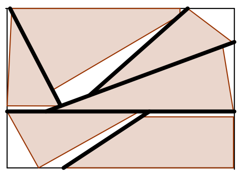
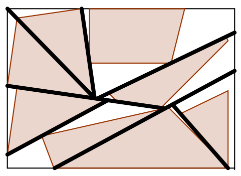

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

Figure 10: *J50*

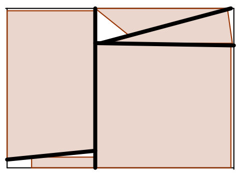
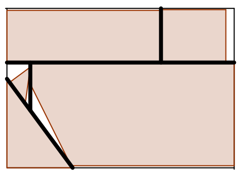
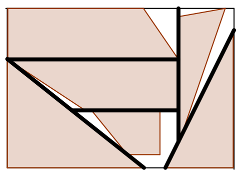
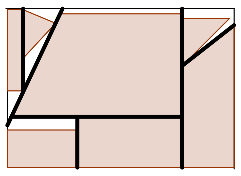
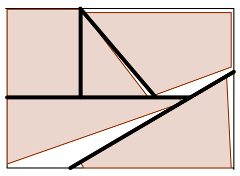
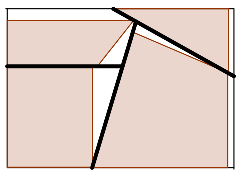
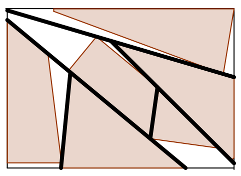
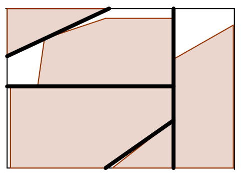
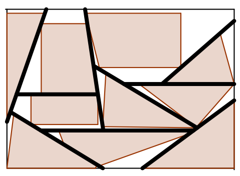
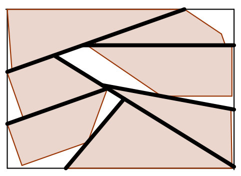

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

Figure 11: *J60*

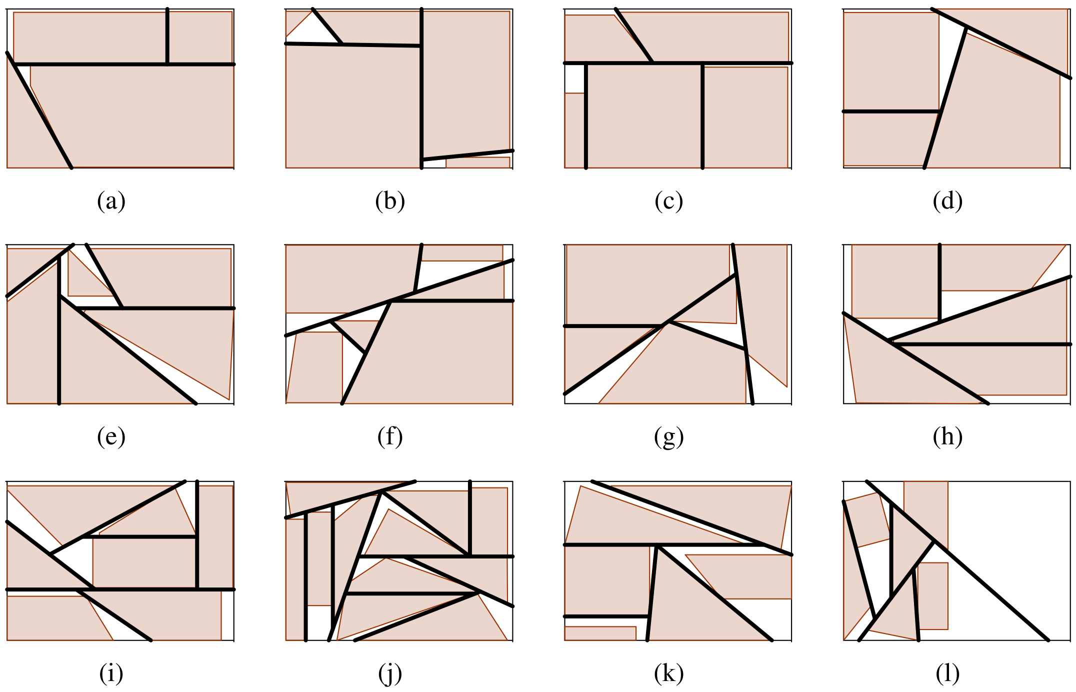


Figure 12: *J70*

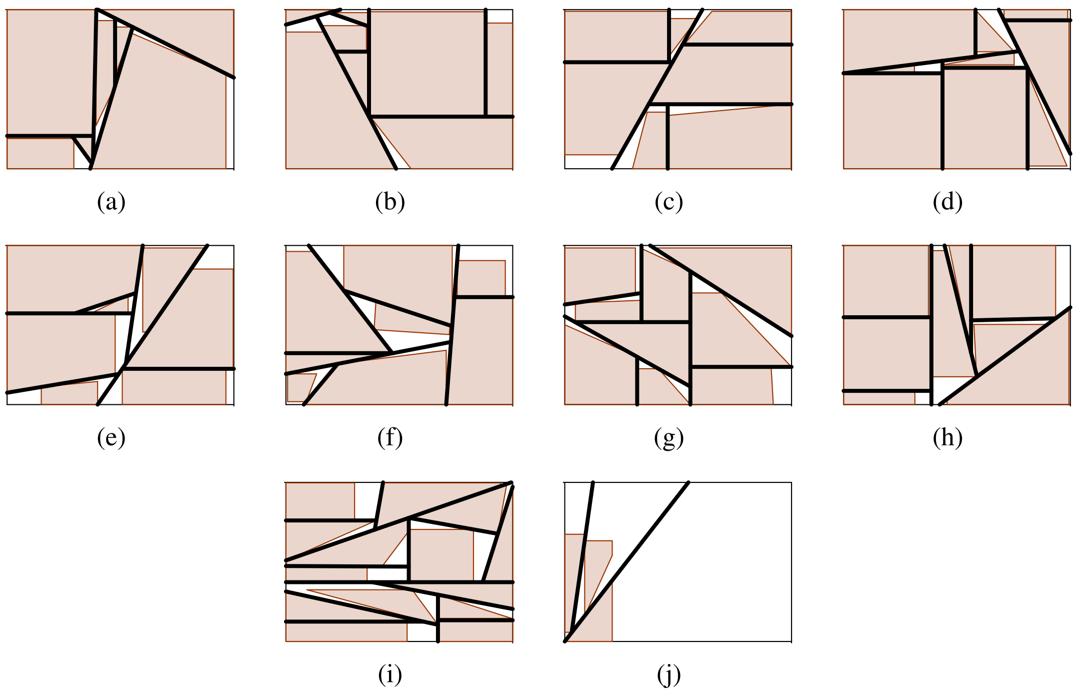


Figure 13: *H80*

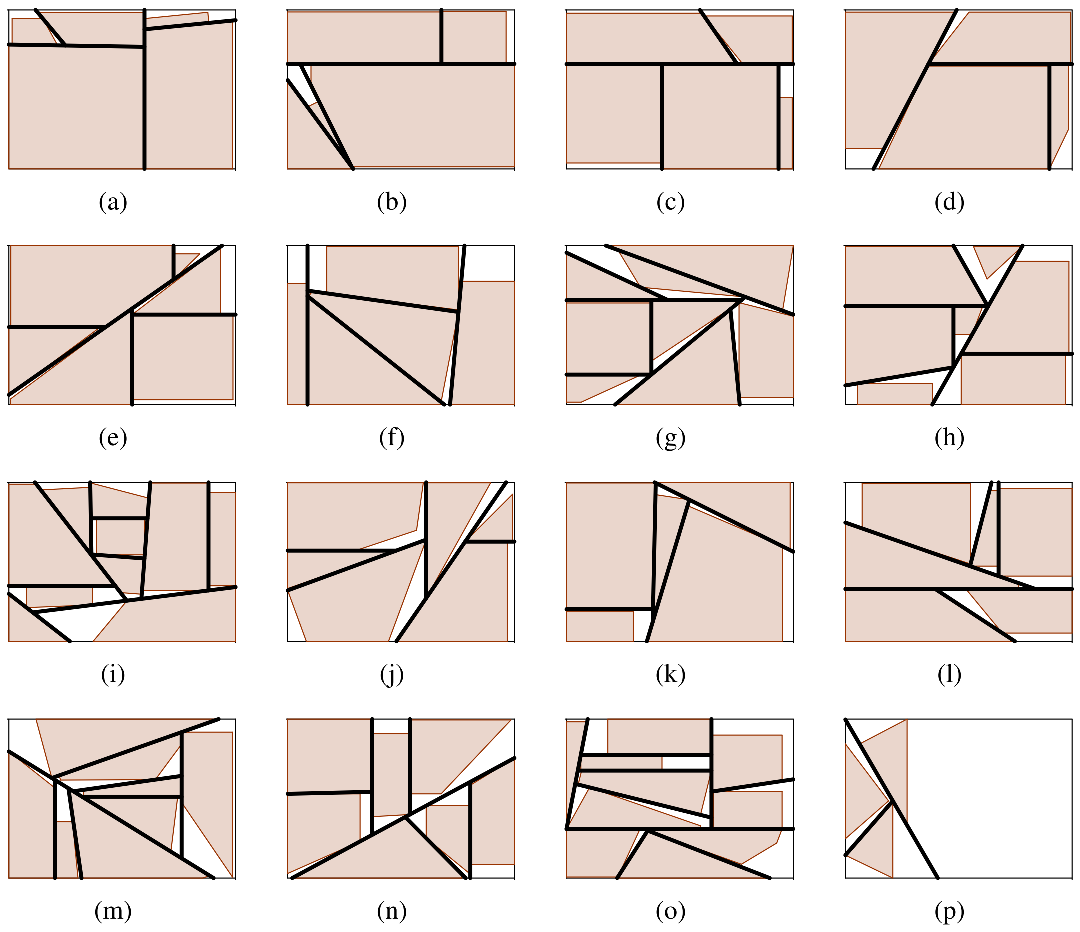


Figure 14: *H100*

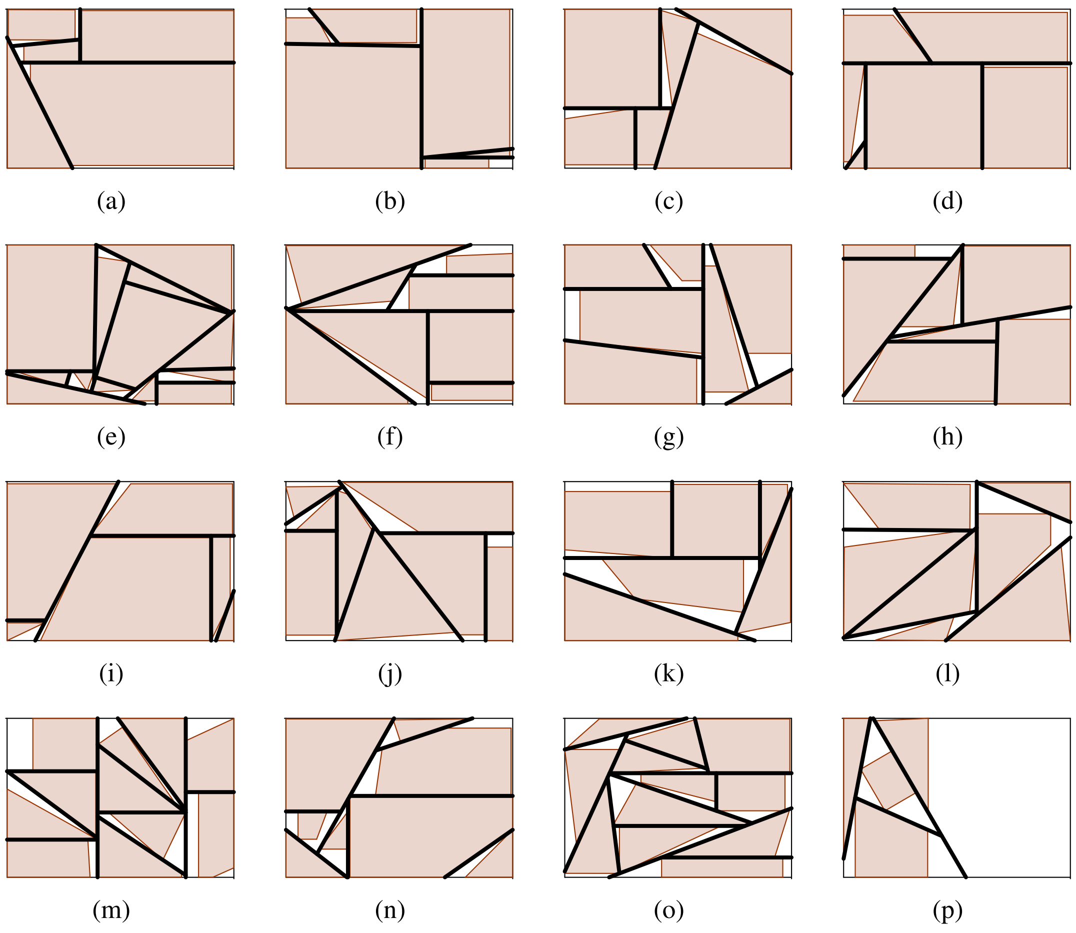


Figure 15: *H120*

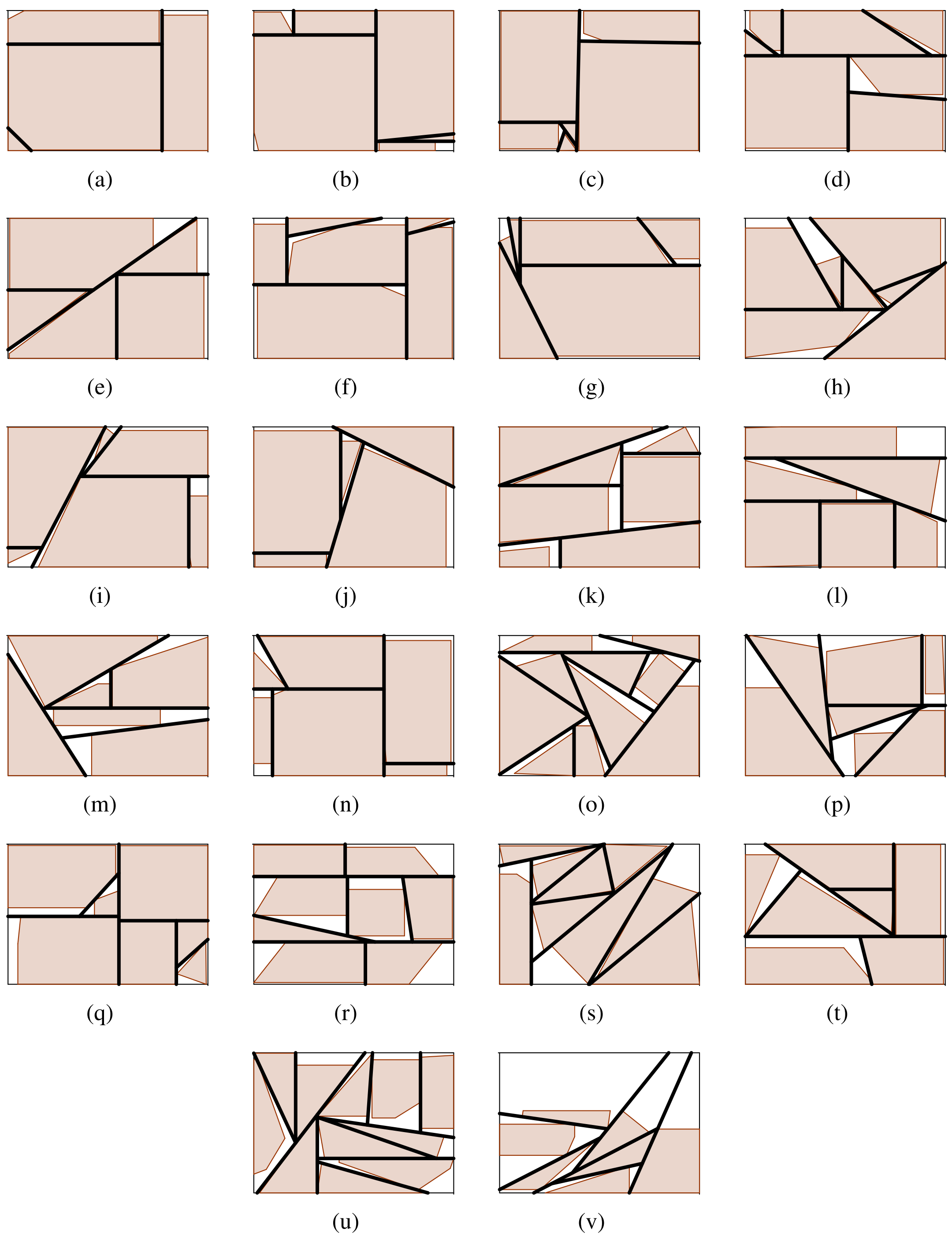


Figure 16: *H149*