A branch & bound algorithm for the nesting problem

R. Alvarez-Valdes†, † A. Martinez†, J.M. Tamarit†

† University of Valencia, Department of Statistics and Operations Research, Burjassot, Valencia, Spain

Abstract

Cutting and packing problems involving irregular shapes, usually known as Nesting Problems, are common in industries ranging from clothing and footwear to engineering and shipbuilding. Research publications on these problems are relatively scarce compared with other cutting and packing problems with rectangular shapes, and have been mostly focused on heuristic approaches. In this paper we propose a new mixed integer formulation for the problem and use it to develop an exact Branch & Bound Algorithm. A computational experiment using known and new test instances shows the performance of the algorithm.

Keywords: Cutting & packing; Nesting; Integer formulation; Branch & Bound

1 Introduction

Nesting problems are two-dimensional cutting and packing problems involving irregular shapes. These problems arise in a wide variety of industries like garment manufacturing, sheet metal cutting, furniture making and shoe manufacturing. The wide range of applications implies many different variants of the problem. The placement areas into which the pieces have to be packed or cut can be rectangular, as in the case of materials provided in rolls, or can have irregular shapes, as in the case of leather hides for making shoes. This placement area can have a uniform quality or different qualities depending on the region, including sometimes defective parts that cannot be used for pieces. The pieces to be cut can be described by polygons, convex or not, or can include curved edges. Depending on the real application, the pieces can be rotated freely, only at specific angles (90°, 180°, ...) or not rotated at all. There may also be different objectives, usually involving the minimization of the area required for cutting the pieces or the maximization of the value of the pieces cut. Using the typology proposed by Waescher et al. (2007), nesting problems, in general, are open-dimension problems with irregular pieces.

In this paper we consider a nesting problem in which we have to arrange a set of two-dimensional irregular shapes without overlapping in a rectangular stock sheet of a fixed width, where the objective is to minimize the required length. We will consider the pieces to be described by polygons, not necessarily convex, which cannot be rotated.

The main difficulty of nesting problems is to ensure that the pieces have a non-overlapping configuration. This question has been studied extensively in recent years and there are several approaches which determine when two polygons overlap. Bennell and Oliveira (2008) give a tutorial of the different approaches which study the geometry of nesting problems. The problem is NP-complete and as a result solution methodologies predominantly utilize heuristics.

Pixel/Raster methods transform the continuous stock sheet into a discrete grid represented by a matrix, and the position of each piece adds a given coded value to each matrix element. Identifying possible overlapping comes down to checking the matrix values. There are three known codification schemes: the first one proposed by Segenreich and Braga (1986), the second one by Oliveira and Ferreira (1993), and the last one by Babu and Babu (2001).
When pieces are represented (or approximated) by polygons, there are tests for edge intersection and for point inclusion that can be used for identifying overlapping (Konopasek 1985, Alves et al. 1998). The most widely used tool for checking whether two polygons overlap is the Non Fit Polygon (NFP). It can be used, along with the vector difference of the position of the two polygons, to determine whether these polygons overlap, touch, or are separated, by conducting a simple test to identify whether the resulting vector is inside the NFP. There are three approaches to generating the NFP: the orbiting algorithm by Mahadevan (1984), improved by Burke et al. (2007); the Minkowski sums used by Milenkovic et al. (1991) and Bennell and Dowsland (2001); and the decomposition into star-shaped polygons (Daniels et al., 1994) or convex polygons (Watson and Tobias 1999, Agarwal et al. 2002).

A more general tool that generalizes the NFP is the Phi-function. Its purpose is to represent all the mutual positions of two polygons. The Phi-functions for cutting and packing were conceived and applied by Stoyan et al. (2001, 2004, 2005). Bennell et al. (2010) provide a good explanation of how the Phi-functions can be built, as well as some applications for them.

Many different heuristic and metaheuristic approaches have been proposed for solving nesting problems. Simple heuristic rules are used for building a solution step by step, placing one piece at a time, using different placement procedures and different piece sequences. Metaheuristic procedures are used for working with complete solutions and iteratively modifying them in order to find improvements.

Very few integer linear programming formulations have been proposed to date. One of them appears in the Simulated Annealing Algorithm proposed by Gomes and Oliveira (2006). In their algorithm, they use a compaction phase in which they solve a linear program which is a relaxation of a mixed-integer formulation of the problem. A different approach, based on Daniels et al. (1994) and Li (1994), is proposed by Fischetti and Luzzi (2008), introducing the concept of slices which, for each pair of pieces, partition the region outside the corresponding NFP, that is the region in which the second piece can be placed without overlapping the first piece.

The remainder of this paper is organized into five sections. In Section 2 we describe in detail the elements of the problem, including the NFP, as it will be the basic tool for developing our formulation. Section 3 is devoted to the mixed integer linear formulation. We review the Gomes and Oliveira (2006) approach and introduce two new formulations, using the main ideas of Fischetti and Luzzi (2008) model. In Section 4 we present our own Branch and Bound algorithm, describing several branching strategies, lower bounds and variable-reduction procedures. Section 5 contains an extensive computational experiment, using known and new test instances, which allows us to compare the proposed alternatives and the overall efficiency of the algorithm. Finally, in Section 6, we draw some conclusions and outline future work.

2 The problem

Let $P = \{p_1, \ldots, p_N\}$ be the set of pieces to arrange in the strip. We consider the reference point of each piece to be the bottom-left corner of the enclosing rectangle. We denote by $(x_i, y_i)$ the coordinates of the reference point of piece $p_i$, by $l_i$ its length and by $w_i$ its width (see Figure 1). The dimensions of the strip are its width $W$ (fixed) and its length $L$ (to be determined). We consider that the bottom-left corner of the strip is placed at the origin.

![Figure 1: Reference point of piece $p_i$](image-url)
Piece \( j \) moves around \( i \) 

\[ \text{NFP}_{ij} \]

Figure 2: Building the NFP of pieces \( i \) and \( j \)

Figure 3: Special cases of NFP when non-convex pieces are involved

For each pair of pieces, \( p_i \) and \( p_j \), the No-Fit Polygon, \( \text{NFP}_{ij} \), is the region in which the reference point of piece \( p_j \) cannot be placed because it would overlap piece \( p_i \). The feasible zone to place \( p_j \) with respect to \( p_i \), outside \( \text{NFP}_{ij} \), is a non-convex polygon or it could be unconnected. Figure 2 shows a simple case in which piece \( i \) is a square and piece \( j \) a triangle. To build \( \text{NFP}_{ij} \), the reference point of piece \( i \) is placed at the origin and the reference point of piece \( j \) slides around piece \( i \) in such a way that a point of piece \( j \) is always touching the border of piece \( i \). The left-hand side of Figure 2 shows several positions of piece \( j \) moving around piece \( i \). The right-hand side of the figure shows \( \text{NFP}_{ij} \), the forbidden region for the reference point of piece \( j \), relative to piece \( i \), if no overlapping is allowed.

When one or both polygons are non-convex, building the NFP is more complex. Figure 3, taken from Bennell and Oliveira (2008), shows more complicated cases. In Figure 3(a), piece \( B \) has some feasible positions within the concavity of \( A \) and therefore \( \text{NFP}_{AB} \) includes a small triangle of feasible placements for the reference point of \( B \). In Figure 3(b), the width of \( B \) fits exactly into the concavity of \( A \) and its feasible positions in the concavity produce a segment of feasible positions for the reference point of \( B \) in \( \text{NFP}_{AB} \). In Figure 3(c), there is exactly one position in which \( B \) fits into the concavity of \( A \) and then \( \text{NFP}_{AB} \) includes a single feasible point for the reference point of \( B \). The methods mentioned in the previous section are progressively more complex for building NFP even in the most difficult cases. In this paper we assume that for each pair of pieces \( i, j \), \( \text{NFP}_{ij} \) is given by a polygon plus, if necessary, some points (as in Figure 3(c)), some segments (as in Figure 3(b)) or some enclosed polygons (as in Figure 3(a)).

### 3 Mixed integer formulations

In this section we will first describe the formulation used by Gomes and Oliveira (2006) and then our two proposals, based on the ideas of Fischetti and Luzzi (2008). In all cases, the objective function will be the minimization of \( L \), the strip length required to accommodate all the pieces without overlapping. Also, all formulations contain two types of constraints: those preventing the pieces from exceeding
the dimensions of the strip and those forbidding the pieces from overlapping. The differences between formulations lie in the way these constraints are defined.

### 3.1 Formulation GO (Gomes and Oliveira)

Let us consider the simple example in Figure 4. Pieces $p_i$ and $p_j$ are rectangles, and then $NFP_{ij}$ is a rectangle. Associated with each edge of $NFP_{ij}$, a binary variable $b_{ijk}$ is defined. Variable $b_{ijk}$ takes value 1 if piece $j$ is separated from piece $i$ by the line defined by the $k$th edge of $NFP_{ij}$, otherwise it takes value 0. In Figure 4, if $b_{ij1} = 1$, $p_j$ is placed above $p_i$. Similarly, variables $b_{ij2} = 1$, $b_{ij3} = 1$, $b_{ij4} = 1$ place $p_j$ to the left, below, or to the right of $p_i$, respectively. As at least one of these variables must take value 1 to prevent overlapping, $\sum_{k=1}^{4} b_{ijk} \geq 1$.

![Figure 4: Definition of variables $b_{ijk}$](image)

If $b_{ij1} = 1$, one way of expressing that $p_j$ has to be placed above $p_i$ would be:

$$y_j - y_i \geq w_i b_{ij1}$$

where $w_i$ is the width of $p_i$. However, this inequality is not valid if $b_{ij1} = 0$. In order to transform (1) into a valid inequality, Gomes and Oliveira (2006) include a big-M term (where $M$ is large positive number). The valid constraint would be:

$$y_j - y_i \geq w_i - (1 - b_{ij1})M$$

The general form of the constraints preventing overlapping is:

$$\alpha_{ijk}(x_j - x_i) + \beta_{ijk}(y_j - y_i) \leq q_{ijk} + M(1 - b_{ijk})$$

where $\alpha_{ijk}(x_j - x_i) + \beta_{ijk}(y_j - y_i) = q_{ijk}$ is the equation of the line including the $k$th of the $m_{ij}$ edges of $NFP_{ij}$. The complete GO formulation is:

\[
\begin{align*}
    & \text{Min } L \\
    \text{s.t. } & x_i \leq L - l_i & i = 1, \ldots, N \tag{5} \\
    & y_i \leq W - w_i & i = 1, \ldots, N \tag{6} \\
    & \alpha_{ijk}(x_j - x_i) + \beta_{ijk}(y_j - y_i) \leq q_{ijk} + M(1 - b_{ijk}) & 1 \leq i < j \leq N, k = 1, \ldots, m_{ij} \tag{7} \\
    & \sum_{k=1}^{m_{ij}} b_{ijk} \geq 1 & 1 \leq i < j \leq N \tag{8} \\
    & b_{ijk} \in \{0, 1\} & 1 \leq i < j \leq N \tag{9} \\
    & x_i, y_i \geq 0 & 1 \leq i \leq N \tag{10}
\end{align*}
\]
The objective function (4) minimizes the required length of the strip. Constraints (5), (6) and (10) are bound constraints, keeping the pieces inside the strip. Constraints (7) prevent overlapping. Variables $b_{ijk}$ are integer (constraints 9) and at least one of them must take value 1 for each $NFP$ (constraints 8).

One potential disadvantage of this formulation is that the above definition of non-overlapping constraints does not limit the relative position of the pieces very strictly. Let us consider the more complex situation in Figure 5. As $NFP_{12}$ has 8 edges, 8 binary variables are defined. If $p_2$ is placed into the shared region of the upper left-hand side, two variables $b_{121}$ and $b_{122}$ can take value 1. In fact, if $p_2$ is placed well to the left of $p_1$ in that zone, even variable $b_{123}$ can take value 1. If we use this formulation in a branch and bound procedure, many different branches can contain the same solution and that can slow down the search.

![Figure 5: Variables of GO formulation](image)

### 3.2 Formulation HS1, using the Fischetti and Luzzi slices

The formulation proposed by Fischetti and Luzzi (2008) differs from the Gomes and Oliveira model in their use of $NFP$ for the definition of variables. They consider the region outside each $NFP_{ij}$ partitioned into $k$ convex disjoint regions called slices, $S_{ij}^k$, and define a variable $b_{ijk}$ so that $b_{ijk} = 1$ if the reference point of $p_j$ is placed into $S_{ij}^k$, and 0 otherwise. Some examples of partitions appear in Figure 6.

Each slice $S_{ij}^k$ is a 2-dimensional polyhedron, defined by $m_{ij}^k$ linear inequalities. Initially, one way of writing valid inequalities would be using $big-M$ constants, as in the GO formulation:

$$
\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq q_{ij}^{kf} + M(1 - b_{ijk}), \forall f = 1 \ldots m_{ij}^k
$$

The coefficients of $b_{ijk}$ can be lifted (and the $big-M$ avoided) in the following way. First, we substitute $M$ by a set of coefficients:

$$
\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq q_{ij}^{kf} + \sum_{h=1}^{m_{ij}} \delta_{ijfh}^k b_{ijh}
$$

Then, taking into account that $\sum_{h=1}^{m_{ij}} b_{ijh} = 1$, and substituting $q_{ij}^{kf}$ by $q_{ij}^{kf} \sum_{h=1}^{m_{ij}} b_{ijh} = 1$, we get:

$$
\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \delta_{ijfh}^k b_{ijh}
$$
Finally, in order to have a valid inequality, coefficients \( \delta_{ij}^{kfh} \) are obtained by computing the maximal value of the left-hand side where each variable \( b_{ijh} \) takes value 1:

\[
\delta_{ij}^{kfh} := \max_{(v_j - v_i) \in S_{ijh} \cap C} \alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i)
\]

where \( C \) is a box which is large enough to include all the possible placements of \( p_i \) and \( p_j \), of width \( 2W \) and length \( 2L \), and where \( L \) is an upper bound on \( L \). This maximization problem can be solved by evaluating the function on the vertices of the closed region \( S_{ijh} \cap C \).

Fischetti and Luzzi (2008) do not specify the way in which the slices are defined. We use their slices and define the variables and the non-overlapping constraints in the following way:

1. **Step 1: Consider the special cases of \( NFP_{ij} \)**

   As was shown in Figure 3, when building the \( NFP_{ij} \) some special elements can appear: points, segments and inner polygons. We define a variable \( b_{ijk} \) for each point, for each segment and for each convex inner polygon, so that the variable takes value 1 if the reference point of \( p_j \) is placed at this point or segment, or inside the polygon. If an inner polygon is non-convex, it is previously decomposed into convex polygons and a variable is associated with each convex polygon.

   Once these elements have been considered, \( NFP_{ij} \) is just a polygon, convex or not.

2. **Step 2: Closing the concavities of \( NFP_{ij} \)**

   If \( NFP_{ij} \) is non-convex, we close its concavities by adding convex polygons in a recursive way until the resulting polygon is convex. We go through the ordered set of vertices counterclockwise, identifying concavities. A concavity starts at vertex \( v_n \) when not all the vertices are to the left-hand side of the line defined by \( v_n \) and \( v_{n+1} \). The concavity ends when the next vertex \( v_{n+r} \) is not to the right-hand side of the line defined by \( v_{n+r-2} \) and \( v_{n+r-1} \). In Figure 7 there is just one concavity, \( C = \{v_1, v_5, v_6, v_7\} \), but in the general case many concavities can appear.

   In the general case, from the list of concavities, we choose that with the largest number of vertices and close it by adding a segment from its first to its last vertex. Other concavities can also be closed if they are disjoint with those already closed. The list of \( NFP_{ij} \) vertices is updated, eliminating the intermediate vertices, and a variable \( b_{ijk} \) is associated with each closed concavity. The process is repeated until the resulting polygon \( NFP_{ij}^c \) is convex. Figure 8 illustrates the process. In a first iteration several concavities are found: \( C_1 = \{v_2, v_3, v_4\} \), \( C_2 = \{v_5, v_6, v_7\} \), \( C_3 = \{v_8, v_9, v_{10}\} \), \( C_4 = \{v_{10}, v_{11}, v_{12}, v_{13}\} \), \( C_5 = \{v_{13}, v_{14}, v_{15}\} \), \( C_6 = \{v_{16}, v_{17}, v_{18}\} \) and \( C_7 = \{v_{19}, v_{20}, v_1\} \).
$C_1 = \{v_4, v_5, v_6, v_7\}$

Figure 7: Closing the concavities of $NFP$

Figure 8: Transforming $NFP$ into a convex set

$C_4$ with 4 vertices is chosen to be closed, as are $C_1$, $C_2$, $C_6$ and $C_7$, which are disjoint with $C_4$ and with each other. Binary variables $b_{ij1}$ to $b_{ij5}$ are associated to these closed regions. The updated $NFP$ appears in Figure 8(c). As this polygon is still non-convex, the process is repeated in Figure 8(d), until a convex polygon is obtained.

3. Step 3: Defining horizontal slices for the edges of $NFP_{ij}^c$

For each edge of every $NFP_{ij}^c$, we define a horizontal slice by drawing two horizontal lines, one from each vertex, in the opposite direction to $NFP_{ij}^c$. If at the bottom (top) there is a horizontal edge, the slice goes from the line containing that edge to the bottom (top) edge of the strip. If at the bottom (top) there is no horizontal edge, an additional slice is built by drawing a horizontal line at the bottommost (topmost) vertex, stretching the whole length of the strip. An example can be seen at Figure 9.

The variables are defined in that way for two main reasons. First, using variables associated with slices overcomes the disadvantages of the Gomes and Oliveira definition (2006). Each feasible position of piece $j$ with respect to piece $i$ corresponds to a unique variable (except for the unavoidable common border between slices). Second, defining the slices in a horizontal way helps us to control the relative vertical position of the pieces. We will take advantage of that when developing the branch and bound algorithm.

The complete HS1 formulation is:
Figure 9: Horizontal slices

\[\begin{align*}
Ml \in L \\
\text{s.t.} & \quad x_i \leq L - l_i \quad i = 1, \ldots, N \\
& \quad y_i \leq W - w_i \quad i = 1, \ldots, N \\
& \quad \alpha_{ij}^k (x_j - x_i) + \beta_{ij}^k (y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \delta_{ij}^{kh} b_{ijk} \\
& \quad 1 \leq i < j \leq N, \quad k = 1, \ldots, m_{ij}, \quad f = 1, \ldots, t_{ij} \\
& \quad \sum_{k=1}^{m_{ij}} b_{ijk} = 1 \\
b_{ijk} \in \{0, 1\} & \quad 1 \leq i < j \leq N \\
x_i, y_i \geq 0 & \quad 1 \leq i \leq N
\end{align*}\]

3.3 Formulation HS2, lifting the bound constraints

The bound constraints (12), (13), (17) of the HS1 formulation are the same as those of the GO formulation (5), (6), (10). In this section we lift constraints (12), (13), (17) using the interaction between pairs of pieces.

3.3.1 Relative position of pieces

Let \(NFP_{ij}\) be the No-Fit Polygon associated with pieces \(p_i\) and \(p_j\). We say we are working in the \(NFP_{ij}\)-coordinate system when we fix the reference point of \(p_i\) at the origin and let \(p_j\) move around in all the possible positions in the strip. We denote by \(Y_{ij}^\alpha \) (\(Y_{ij}^\beta\)) the maximum (minimum) value of the \(y\)-coordinate of \(NFP_{ij}\). In a similar way, \(X_{ij}^\alpha \) (\(X_{ij}^\beta\)) is the maximum (minimum) value of the \(X\)-coordinate of \(NFP_{ij}\) (see Figure 10).

Let us consider the slice \(k \in \{1, \ldots, m_{ij}\}\). We denote by \(x_{ijk} \) (\(x_{ijk}\)) the maximum (minimum) value \(x_j\) can take in the \(NFP_{ij}\)-coordinate system when \(b_{ijk} = 1\). Analogously, \(y_{ijk} \) (\(y_{ijk}\)) is the maximum (minimum) value \(y_j\) can take in the \(NFP_{ij}\)-coordinate system when \(b_{ijk} = 1\). In the example in Figure 10, these values are represented for slice \(k = 2\), considering \(W = 10\) and an upper bound on \(L, T = 11\). In this case, if the reference point of \(p_i\) is placed at the origin, and as \(l_i = 4\) and \(T = 11\), there is a space of 4 units to the left of \(NFP_{ij}\) and 3 units to its right into which the reference point of piece \(p_j\) can be placed. Similarly, as \(w_i = 4\) and \(W = 10\), the space above \(NFP_{ij}\) for the reference point of \(p_i\) has 3 units and below it also has 3 units. Looking at \(NFP_{ij}\), \(X_{ij}^\alpha = 4\), \(X_{ij}^\beta = -3\), \(X_{ij}^\alpha = 4\) and \(X_{ij}^\beta = -3\). Looking at slice \(k = 2\), \(x_{i2} = -1\), \(x_{i2} = -7\), \(y_{i2} = 4\) and \(y_{i2} = 2\).
Figure 10: Definitions in the $NFP_{ij}$-coordinate system
The subset of variables associated with $NFP_{ij}$ which force the reference point of $p_j$ to be above (below) the reference point of $p_i$ is denoted as $U_{ij}$ ($D_{ij}$). Analogously, the subset of variables which force the reference point of $p_j$ to be to the right (left) of the reference point of $p_i$ is denoted as $R_{ij}$ ($L_{ij}$). If $VNFP_{ij}$ is the set of all the variables associated with $NFP_{ij}$, these sets can be described as follows:

- $U_{ij} := \{ b_{ijk} \in VNFP_{ij} \mid y_{ijk} \geq 0 \}.$
- $D_{ij} := \{ b_{ijk} \in VNFP_{ij} \mid y_{ijk} \leq 0 \}.$
- $R_{ij} := \{ b_{ijk} \in VNFP_{ij} \mid x_{ijk} \geq 0 \}.$
- $L_{ij} := \{ b_{ijk} \in VNFP_{ij} \mid x_{ijk} \leq 0 \}.$

In the example in Figure 10, we have: $U_{ij} := \{ b_{ij1}, b_{ij2}, b_{ij8} \}$, $D_{ij} := \{ b_{ij4}, b_{ij5}, b_{ij6} \}$, $R_{ij} := \{ b_{ij6}, b_{ij7}, b_{ij8} \}$, $L_{ij} := \{ b_{ij2}, b_{ij3}, b_{ij4} \}$.

### 3.3.2 Lifted bound constraints

For each piece $p_j$, the formulations GO and HS1 include four bound constraints, ensuring that the piece does not exceed the left, right, top and bottom limits of the strip. We can lift each of these constraints by considering the variables in sets $U_{ij}$, $D_{ij}$, $R_{ij}$, $L_{ij}$, for each piece $p_i$, $i \neq j$.

- **Left-hand side bound**
  
  In the constraint we include the variables forcing $p_j$ to be to the right of $p_i$, that is, the variables in $R_{ij}$. The coefficient of each variable, that is, the forced displacement of $p_j$ to the right of $p_i$ is given by $x_{ijk}$. The lifted constraint will be:

  $$
x_j \geq \sum_{k \in R_{ij}} x_{ijk} b_{ijk}
  $$

  In Figure 10, the constraint will be: $x_j \geq 2b_{ij6} + 4b_{ij7} + 2b_{ij8}$.

- **Right-hand side bound**

  In this case, the variables to be included are those forcing $p_j$ to be to the left of $p_i$, those in $L_{ij}$. The corresponding coefficient, the minimum distance from the reference point of $p_j$ to $L$ forced by the variable, is $\lambda_{ijk} = l_i - (x_{ijk} - X^{\uparrow})$. This value can be seen as the length of piece $p_i$, $l_i$, which would be the extra separation if $p_j$ were completely to the right of $p_i$, minus the maximum amount of parallelism between both pieces which is allowed in slice $k$, given by $x_{ijk} - X^{\uparrow}$. Then, the lifted right-hand side bound constraint is:

  $$
x_j \leq L - l_j - \sum_{k \in L_{S_{ij}}} \lambda_{ijk} b_{ijk}
  $$

  where $L_{S_{ij}} := \{ k \mid \lambda_{ijk} > 0 \}$. In the example in Figure 10: $x_j \leq L - 3 - 2b_{ij2} - 4b_{ij3} - 2b_{ij4}$. In slice 3, $p_j$ is completely to the left of $p_i$ and the coefficient of $b_{ij3}$ is 4, corresponding to $l_i$. In slice 2, 2 units of this initial separation can be eliminated if piece $p_j$ is placed at the rightmost point inside the slice. Then, the coefficient of $b_{ij2}$ is 2.

- **Bottom-side bound**

  The variables forcing $p_j$ to be above $p_i$, are those in $U_{ij}$. The coefficient of each variable, that is, the forced displacement of $p_j$ on top of $p_i$, is given by $y_{ijk}$. The lifted constraint will be:
\[ y_j \geq \sum_{k \in U_{ij}} y_{ijk} b_{ijk} \]  

(20)

In the example in Figure 10 we have: \( y_j \geq 4b_{ij1} + 2b_{ij2} + 2b_{ij8} \).

- Upper-side bound

The variables to be included are those forcing \( p_j \) to be at the bottom of \( p_i \), those of \( D_{ij} \). The corresponding coefficient, the minimum distance from the reference point of \( p_j \) to \( W \) forced by the variable, is \( \mu_{ijk} = w_i - (y_{ijk} - \Sigma^{ij}) \). The constraint will be:

\[ y_j \leq W - w_j - \sum_{k \in DS_{ij}} \mu_{ijk} b_{ijk} \]  

(21)

where \( DS_{ij} := \{k \mid \mu_{ijk} > 0\} \). In the example in Figure 10: \( y_j \leq W - w_j - 2b_{ij4} - 4b_{ij5} - 2b_{ij6} \).

### 3.3.3 Formulation HS2

The complete HS2 formulation is:

\[ \text{Min } L \]  

s.t. \[ \sum_{k \in R_{ij}} x_{ijk} b_{ijk} \leq x_i \leq L - l_i - \sum_{k \in LS_{ij}} \Delta x_{ijk} b_{ijk} \]  

(23)

\[ \sum_{k \in U_{ij}} y_{ijk} b_{ijk} \leq y_i \leq W - w_i - \sum_{k \in DS_{ij}} \mu_{ijk} b_{ijk} \]  

(24)

\[ \alpha_{ij}^k (x_j - x_i) + \beta_{ij}^k (y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \delta_{ij}^k h b_{ijh} \]  

\( 1 \leq i < j \leq N, \ k = 1, \ldots, m_{ij}, \ f = 1, \ldots, t_{ij} \)  

(25)

\[ \sum_{k=1}^{m_{ij}} b_{ijk} = 1 \]  

(27)

\[ b_{ijk} \in \{0, 1\} \]  

(28)

\[ x_i, y_i \geq 0 \]  

(29)

where constraints (23) and (24) are the lifted bound constraints which substitute the initial bound constraints (12) and (13) of the HS1 formulation.

### 4 A branch and bound algorithm

The formulations in the previous section can be used in a branch and bound algorithm in which at each node of the search tree the linear relaxation provides a lower bound and, if the node is not fathomed, branching will consist in building two nodes, one with a variable \( b_{ijk} = 1 \) and the other with \( b_{ijk} = 0 \). In this section we consider several aspects of the procedure that can influence its performance significantly. First, we study several branching strategies, that is, several alternative ways of selecting the branching variable at each level. Second, we develop some procedures to update the upper and lower bounds of the pieces. Fixing the variables to 1 imposes some specific relative position for the pair of pieces involved and that can restrict the position of the pieces in the strip. Updating the bounds can have a positive influence on the branching strategy and on the enhancement of the formulation at each node. One of these enhancements which will be studied is the case in which the variables already fixed to 1 imply that some other variables must be fixed to 0.
4.1 Branching strategies

One obvious strategy is leaving the integer linear code, in our case CPLEX, to decide the branching variable, using its internal strategy in which some priorities are assigned to the non-integer variables based on the information provided by the linear solution of each node. However, this strategy does not take into account any problem-specific information which could be useful in guiding the search process. Therefore, we study three specific strategies: the first one is based on the branching procedure proposed by Fischetti and Luzzi (2008). Then we consider a dynamic strategy and finally an alternative branching on constraints procedure.

4.1.1 The Fischetti and Luzzi strategy

The strategy followed by Fischetti and Luzzi (2008) is first to determine the relative positions of 2 pieces (say A and B), then those of 3 pieces (A, B and, say, C), of 4 pieces (A, B, C and, say, D), and so on. To do that, a simple procedure assigns priorities to the variables, in decreasing order, starting from variables separating A and B, then variables separating A and C and B and C, then variables separating A and D, B and D, C and D, and so on. Doing that, they try to avoid the visit of subtrees that are unfeasible because of inconsistent variable fixings that could have been detected at higher levels in the tree.

Fischetti and Luzzi do not specify any particular order for the pieces. Nevertheless, as their procedure builds an increasingly large clique of non-overlapping pieces (allowing the other pieces to overlap with them and among themselves), it could be interesting to separate large pieces first. If the growing clique is made of large pieces, the lower bound could increase faster than when small pieces are involved. Defining a large piece is not obvious in the case of irregular pieces. Two approximations could be the length and the area of each piece. Therefore, when studying the behavior of the Fischetti and Luzzi branching strategy computationally, we will consider three alternatives:

- **FL**: Fischetti and Luzzi’s priorities without any initial ordering of the pieces
- **FL,L**: Fischetti and Luzzi’s priorities ordering pieces by non-increasing length
- **FL,A**: Fischetti and Luzzi’s priorities ordering pieces by non-increasing area

Even when we specify the order in which the pieces are considered for assigning priorities, there is still a degree of freedom concerning the order in which the variables of the corresponding NFP are considered. One possibility for ordering the variables of an NFP is the area of the corresponding slice, giving more priority to variables associated to larger slices. Then, a fourth strategy to be studied is **FL,A,SA**, in which we first order the pieces by area and then the variables by the area of the slice. A last strategy in this group could be **SA**, ordering all the variables according to the area of their slices, but not using Fischetti and Luzzi’s priorities.

4.1.2 Dynamic branching

Let us consider the 3-pieces example in Figure 11. The pieces are already ordered by some priority criterion and let us suppose that in the first branching level $b_{128} = 1$. That means that $p_1$ and $p_2$ are separated and the relative position of $p_2$ with respect to $p_1$ is restricted to the corresponding slice. In a static branching strategy the variable used at the second branching level would be given by some predefined criterion. But we can use a dynamic strategy, taking advantage of the information in the solution of the node. In particular, we can take into account the relative position of the pieces and choose a branching variable such that when it is fixed to one, more than two pieces are separated and feasible solutions are obtained faster. In the example, we see that if we branch on variables $b_{132}$, $b_{133}$, $b_{134}$, $b_{135}$ or $b_{136}$ from NFP$_{13}$, or variables $b_{231}$ or $b_{236}$ from NFP$_{23}$, the three pieces would be separated and fixing just two variables to value 1 will produce a feasible solution.
We can generalize this idea. Let us consider the pieces ordered by non-increasing value of a given priority criterion. At each node of the search tree, we read the solution and go through the ordered list of pieces until we find a piece \( p_j \) overlapping with some of the previous pieces. Let \( S = \{ p_1, \ldots, p_k \} \) be the set of these pieces, \( 1 \leq i_1 < \ldots < i_k \). For each piece \( i \in S \) we compute \( u_i, \text{down}_i, \text{left}_i, \text{right}_i \), the number of pieces separated from \( i \) from above, from below, from the left and from the right, respectively. We consider that a piece \( p_k \) is separated above to piece \( p_i \) if there is a variable \( b_{ijkl} = 1 \) for some \( l \), such that \( y_{ijkl} > 0 \). Similarly, it is separated from below if \( y_{ijkl} < 0 \), to the right if \( x_{ijkl} > 0 \) and to the left if \( x_{ijkl} < 0 \).

When choosing a variable to separate piece \( p_j \) from \( S \), the lower the values of \( u_i, \text{down}_i, \text{left}_i, \text{right}_i \) for some \( i \in S \), the more adequate this position is for piece \( p_j \) and hence the branching variable should separate \( p_j \) from \( p_i \) in this direction. For instance, if for some \( i \in S \), \( u_i = 0 \), none of the other pieces in \( S \) is above \( p_i \) and then that would be a good position for \( p_j \). Separating \( p_j \) from \( p_i \) in this direction could possibly separate \( p_j \) from some other pieces in \( S \).

### 4.1.3 Branching on constraints

An alternative branching strategy is to branch on constraints. In order to do that, we slightly modify the formulation \( HS2 \) so that all the slices are on one side of the \( y \)-axis, which means that some of the slices defined in Section 3.2 are divided into two, as can be seen in Figure 12. Associated with the new slices, new variables are defined.

At each node of the search tree we look for a pair of pieces \( p_i, p_j \) for which

\[
0 < \sum_{k \mid \xi_{ijk} \geq 0} b_{ijk} < 1
\]  

(30)

Then, in one of the branches we set \( \sum_{k \mid \xi_{ijk} \geq 0} b_{ijk} = 0 \) and in the other \( \sum_{k \mid \xi_{ijk} \geq 0} b_{ijk} = 1 \). In the example, one branch would have \( b_{ij3} + b_{ij4} + b_{ij5} + b_{ij6} + b_{ij7} + b_{ij8} = 0 \) and the other \( b_{ij3} + b_{ij4} + b_{ij5} + b_{ij6} + b_{ij7} + b_{ij8} = 1 \). If no pair of pieces satisfy (30), we branch on variables using strategy FLA.
One advantage of this branching strategy is that the formulation can be locally enhanced. In the example, if we are in the node in which we have set $b_{ij3} + b_{ij4} + b_{ij5} + b_{ij6} + b_{ij7} + b_{ij8} = 1$, piece $p_i$ must be to the left of piece $p_j$. Then, constraint $x_i \leq x_j$ is satisfied in all the successor nodes. This constraint can be lifted, considering variables $b_{ijk}$ for which $x_{ijk} \geq 0$. The lifted constraint would be:

$$x_i + \sum_{k : x_{ijk} \geq 0} x_{ijk} b_{ijk} \leq x_j$$

(31)

In the example, $x_i + 6b_{ij5} + 12b_{ij6} + 6b_{ij7} \leq x_j$.

### 4.2 Lower bounds

There are two obvious lower bounds for the nesting problem. The first one, the length of the longest piece, is already included in the formulation. Another lower bound can be obtained by calculating the area of the pieces, adding them up and dividing this value by $W$, the strip width. This bound can be easily obtained, but it is usually very loose, except for the artificial “broken glass” instances.

A third alternative we have considered is solving a special case of a 1-Contiguous Bin Packing Problem (1-CBPP), which was shown to be very effective for the Strip Packing Problem with rectangular pieces (Martello et al., 2003; Alvarez-Valdes et al., 2009). Each piece is divided into a set of vertical slices of width $w$. From each slice the maximum embedded rectangle is obtained (see Figure 13). The problem is then to pack all these rectangles into the minimum number of levels $N$, putting the rectangles corresponding to one piece into contiguous levels. The value $w/N$ is a lower bound for the original problem. An integer formulation for the 1-CBPP appears in Alvarez-Valdes et al. (2009). We solve this formulation using CPLEX, with limited time, and obtain the corresponding lower bound.
4.3 Updating the bounds on the pieces

When variables are fixed to 1, the relative position of the pieces is constrained and the lower and upper bounds of the pieces can be updated. We have developed two methods:

1. Method 1: Using the non-overlapping constraints

   Let us suppose that \( b_{ijk} = 1 \). The corresponding slice has \( f^k_{ij} \) constraints in the formulation:

   \[
   \alpha_{ij}^k (x_j - x_i) + \beta_{ij}^k (y_j - y_i) \leq \delta_{ij}^k b_{ijk} + \sum_{h=1, h \neq k}^{m_{ij}} \delta_{ij}^{kh} b_{ijh}
   \]  

   (32)

   where \( \sum_{h=1, h \neq k}^{m_{ij}} \delta_{ij}^{kh} b_{ijh} = 0 \) since \( b_{ijk} = 1 \).

   Let us consider the case in which \( \alpha_{ij}^k > 0 \) and \( \beta_{ij}^k > 0 \) (The other cases are studied using similar arguments). The constraint can be written as:

   \[
   \alpha_{ij}^k x_j \leq \beta_{ij}^k y_i - \beta_{ij}^k y_j + \alpha_{ij}^k x_i + \delta_{ij}^k.
   \]  

   (33)

   The maximum value the right hand side can attain is \( \beta_{ij}^k U_i - \beta_{ij}^k L_j + \alpha_{ij}^k U_i + \delta_{ij}^k \) (where \( U \) stands for upper bound and \( L \) for lower bound). Then

   \[
   x_j \leq \frac{\beta_{ij}^k U_i - \beta_{ij}^k L_j + \alpha_{ij}^k U_i + \delta_{ij}^k}{\alpha_{ij}^k}
   \]

   and

   \[
   UX_j = \min\{UX_j, \frac{\beta_{ij}^k U_i - \beta_{ij}^k L_j + \alpha_{ij}^k U_i + \delta_{ij}^k}{\alpha_{ij}^k}\}
   \]

   2. Method 2: Extended slices

   Let us consider a pair of pieces \( p_i \) and \( p_j \) and one slice \( S_{ijk} \) associated with a binary variable \( b_{ijk} \). We define the extended slice \( S^*_{ijk} \) as the minimum rectangle enclosing \( S_{ijk} \). Let us denote the minimum and maximum coordinates of \( S^*_{ijk} \) by \( x_{min}, x_{max}, y_{min}, y_{max} \). Then, if \( b_{ijk} = 1 \), the bounds for piece \( p_j \) can be updated as follows:

   • \( LX_j = \max\{LX_j, LX_i + x_{min}\} \)
   • \( UX_j = \min\{UX_j, UX_i + x_{max}\} \)
   • \( LY_j = \max\{LY_j, LY_i + y_{min}\} \)
   • \( UY_j = \min\{UY_j, UY_i + y_{max}\} \)

   Analogously, the bounds for piece \( p_i \) are:

   • \( LX_i = \max\{LX_i, LX_j - x_{max}\} \)
   • \( UX_i = \min\{UX_i, UX_j - x_{min}\} \)
   • \( LY_i = \max\{LY_i, LY_j - y_{min}\} \)
   • \( UY_i = \min\{UY_i, UY_j - y_{min}\} \)

   Both methods are used in an iterative way, going through the list of pieces until no bounds are updated. The second method is less accurate, because the extended slice may allow overlapping and the calculated bounds are looser, but it is very fast because the values \( x_{min}, x_{max}, y_{min}, y_{max} \) are calculated just once, when \( NFP_{ij} \) is built.
4.4 Fixing variables to 0

At each node of the tree we study whether some of the variables not yet fixed cannot take value 1, because that would produce an unfeasible solution. In this case, the variable can be fixed to 0, reducing the size of the problems to be solved in successor nodes and focusing the search on variables which really can take value 1.

One simple way of fixing variables is using the second method of updating bounds from the previous subsection. For each pair of pieces \( p_i \) and \( p_j \) and each variable \( b_{ijk} \), we suppose \( b_{ijk} = 1 \) and update the bounds. If an upper bound is lower than the corresponding lower bound, \( b_{ijk} \) can be fixed to 0.

A more complex method is based on transitivity. Let us consider three pieces, \( p_i \), \( p_j \) and \( p_k \) and let \( b_{ij1} \), \( b_{ik1} \), \( b_{jk1} \) be one variable of each non-fit polygon. A sufficient condition for these three variables to be incompatible is that one of these cases is satisfied:

1. \( x_{ik1} < x_{ij1} + x_{jk1} \)
2. \( x_{ik1} > x_{ij1} + x_{jk1} \)
3. \( y_{ik1} < y_{ij1} + y_{jk1} \)
4. \( y_{ik1} > y_{ij1} + y_{jk1} \)

We consider the extended slices associated with the three variables. On the one hand, if \( b_{ij1} = 1 \), piece \( p_j \) must have its reference point inside \( S^*_{ij1} = \{(x_{ij1, \text{min}}, y_{ij1, \text{min}}), (x_{ij1, \text{max}}, y_{ij1, \text{max}})\} \). If also \( b_{jk1} = 1 \), the possible positions for the reference point of \( p_k \) are obtained by considering, for each point in \( S^*_{ij1} \) the points in \( S^*_{jk1} \), that is, all the points in the rectangle \( S^*_{ij1, jk1} = \{(x_{ij1, \text{min}} + x_{jk1, \text{min}}, y_{ij1, \text{min}} + y_{jk1, \text{min}}), (x_{ij1, \text{max}} + x_{jk1, \text{max}}, y_{ij1, \text{max}} + y_{jk1, \text{max}})\} \). On the other hand, if \( b_{ik1} = 1 \), the reference point of \( p_k \) must be in \( S^*_{ik1} = \{(x_{ik1, \text{min}}, y_{ik1, \text{min}}), (x_{ik1, \text{max}}, y_{ik1, \text{max}})\} \). Therefore, for the 3 variables taking value 1 simultaneously, \( S^*_{ij1, jk1} \) and \( S^*_{ik1} \) must intersect and none of the above conditions may hold.

4.5 Avoiding duplicate solutions

It is quite common in nesting instances that several copies of the same piece type have to be packed or cut. In this case, if these copies are considered as different pieces, the algorithm will have to study partial and complete solutions which are really identical to other solutions already studied elsewhere in the search tree, just changing one piece for another belonging to the same type. In order to partially avoid this unnecessary effort, for each piece type \( i \) consisting of \( n \) copies of the same piece, we add a set of inequalities to the formulation:

\[
x_{i1} \leq x_{i2} \leq \ldots \leq x_{in}
\]

imposing a left-to-right order in the position of these pieces.

5 Experimental results

In this section we will present the results of the computational experiments. All the procedures described in previous sections were coded in C++ and run on a computer with an Intel Core i7 processor, at 3.4 GHz with 16 Gb of RAM. To solve the linear and integer models we used CPLEX 12.1 serial model.
5.1 Instances

In the only paper in which an exact algorithm has been implemented and tested, Fischetti and Luzzi (2008) generated three broken glass instances, glass1, glass2, glass3, with 5, 7 and 9 pieces respectively, dividing a square into polygonal pieces. In addition, other authors who have developed and tested heuristic algorithms have used many instances, some of them taken from real problems, and others artificially generated, ranging from 10 to 99 pieces. In some cases, some kind of rotation is allowed. Approximate results on these instances can be found in Gomes and Oliveira (2006), Burke et al. (2006), Bennell and Song (2010) and Imamochi et al. (2009).

In our computational study, we use the three broken glass instances and some of the other instances, those with the fewest number of pieces (dighe2, fu, poly1a). However, in order to assess the behavior and the limits of our algorithm we needed a larger set of instances. Instead of generating new problems, we have taken existing instances and extracted subsets of pieces, solving, so to say, subproblems of known problems. In the tables in this section, these instances are referred to using the name of the original instance and a number indicating the number of pieces taken into account. As an example, instance fu has 12 pieces. It has been included in the study, but also subsets of its first 5, 6, 7, 8, 9, 10 and 11 pieces. From the instances Jakob1 and Jakob2 (Jakobs, 1996), we have generated 30 instances of 10, 12 and 14 pieces. In this case, the selection of pieces has been made at random, generating 5 replications for each problem and size. The generated instances are denoted as Ja-b-c-d, where a indicates the original instance (1 or 2), b the number of pieces, c the strip width and d the replica. Overall, we use 50 instances, some of them very small, to follow the search tree closely, and some others which are larger, to explore the limits of the proposed algorithm.

An initial feasible solution providing a value for $L$ was obtained for each instance using a fast and simple heuristic provided by Gomes and Oliveira (2006).

5.2 Comparing formulations

Table 1 compares the three mixed integer formulations described in Section 3. For each instance, the corresponding formulation has been generated and input to CPLEX 12.1 serial model, without modifying any of its default parameters, and setting the time limit at 3600 seconds. Therefore, when in columns 5, 8 and 11 the reported time is 3600, the execution has stopped before the search tree has been completely explored. In this case, GAP shows the ratio (Best Known Solution - Lower Bound)/Lower Bound. The Best Known Solution is the best feasible solution found so far in the search. When GAP is 0, the optimal solution has been found and proved and its value appears in the LB column. The bottom rows show the average GAP and time and the number of optimal solutions found in each case.

The overall results clearly show the superiority of HS2 formulation, in terms of the number of instances optimally solved, GAP and running time. However, the behavior is not uniform. In some cases the GAP produced by HS1 or GO is lower, usually because HS2 has more difficulties in finding good feasible solutions. Nevertheless, we will use HS2 as the best formulation in the next steps of our study.

5.3 Comparing branching strategies

Table 2 shows the comparison between the branching strategies described in Section 4.1. The performance of each strategy is summarized in three values: the number of optimal solutions, the average GAP and the average CPU time. The strategies compared are:

- CPLEX: the default strategy provided by CPLEX, as shown in the final columns of Table 1.
- FL: the Fischetti and Luzzi strategy, taking the pieces as they appear in the data file
Table 1: Comparing formulations \textit{GO}, \textit{HS1} and \textit{HS2}

<table>
<thead>
<tr>
<th>Instance</th>
<th>Pieces</th>
<th>\textit{GO}</th>
<th></th>
<th>\textit{HS1}</th>
<th></th>
<th>\textit{HS2}</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>three</td>
<td>3</td>
<td>6.00</td>
<td>0.00</td>
<td>0.02</td>
<td>6.00</td>
<td>0.00</td>
<td>0.22</td>
</tr>
<tr>
<td>shapes_4</td>
<td>4</td>
<td>24.00</td>
<td>0.00</td>
<td>0.08</td>
<td>24.00</td>
<td>0.00</td>
<td>0.03</td>
</tr>
<tr>
<td>fu_5</td>
<td>5</td>
<td>17.89</td>
<td>0.00</td>
<td>1.40</td>
<td>17.89</td>
<td>0.00</td>
<td>0.94</td>
</tr>
<tr>
<td>glass1</td>
<td>5</td>
<td>45.00</td>
<td>0.00</td>
<td>0.02</td>
<td>45.00</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>fu_6</td>
<td>6</td>
<td>23.00</td>
<td>0.00</td>
<td>0.11</td>
<td>23.00</td>
<td>0.00</td>
<td>0.12</td>
</tr>
<tr>
<td>threep2</td>
<td>6</td>
<td>9.33</td>
<td>0.00</td>
<td>20</td>
<td>9.33</td>
<td>0.00</td>
<td>3.2</td>
</tr>
<tr>
<td>threep2w9</td>
<td>6</td>
<td>8.00</td>
<td>0.00</td>
<td>96</td>
<td>8.00</td>
<td>0.00</td>
<td>11.4</td>
</tr>
<tr>
<td>fu_7</td>
<td>7</td>
<td>24.00</td>
<td>0.00</td>
<td>0.67</td>
<td>24.00</td>
<td>0.00</td>
<td>1.5</td>
</tr>
<tr>
<td>glass2</td>
<td>7</td>
<td>45.00</td>
<td>0.00</td>
<td>0.72</td>
<td>45.00</td>
<td>0.00</td>
<td>7.0</td>
</tr>
<tr>
<td>fu_8</td>
<td>8</td>
<td>24.00</td>
<td>0.00</td>
<td>0.76</td>
<td>24.00</td>
<td>0.00</td>
<td>2.6</td>
</tr>
<tr>
<td>shapes_8</td>
<td>8</td>
<td>25.00</td>
<td>0.04</td>
<td>3600</td>
<td>26.00</td>
<td>0.00</td>
<td>126</td>
</tr>
<tr>
<td>fu_9</td>
<td>9</td>
<td>25.00</td>
<td>0.00</td>
<td>161</td>
<td>25.00</td>
<td>0.00</td>
<td>257</td>
</tr>
<tr>
<td>threep3</td>
<td>9</td>
<td>9.15</td>
<td>0.32</td>
<td>3600</td>
<td>13.53</td>
<td>0.00</td>
<td>2565</td>
</tr>
<tr>
<td>threep3w9</td>
<td>9</td>
<td>7.00</td>
<td>0.36</td>
<td>3600</td>
<td>8.67</td>
<td>0.22</td>
<td>3600</td>
</tr>
<tr>
<td>glass3</td>
<td>9</td>
<td>100.00</td>
<td>0.00</td>
<td>2909</td>
<td>100.00</td>
<td>0.00</td>
<td>175</td>
</tr>
<tr>
<td>fu_10</td>
<td>10</td>
<td>28.00</td>
<td>0.02</td>
<td>3600</td>
<td>28.00</td>
<td>0.03</td>
<td>3600</td>
</tr>
<tr>
<td>dighe2</td>
<td>10</td>
<td>96.27</td>
<td>0.04</td>
<td>3600</td>
<td>100.00</td>
<td>0.00</td>
<td>359</td>
</tr>
<tr>
<td>J1-10-20-0</td>
<td>10</td>
<td>16.00</td>
<td>0.11</td>
<td>3600</td>
<td>18.00</td>
<td>0.00</td>
<td>82</td>
</tr>
<tr>
<td>J1-10-20-1</td>
<td>10</td>
<td>17.00</td>
<td>0.00</td>
<td>442</td>
<td>17.00</td>
<td>0.00</td>
<td>69</td>
</tr>
<tr>
<td>J1-10-20-2</td>
<td>10</td>
<td>20.00</td>
<td>0.00</td>
<td>428</td>
<td>20.00</td>
<td>0.00</td>
<td>124</td>
</tr>
<tr>
<td>J1-10-20-3</td>
<td>10</td>
<td>18.00</td>
<td>0.14</td>
<td>3600</td>
<td>19.00</td>
<td>0.10</td>
<td>3600</td>
</tr>
<tr>
<td>J1-10-20-4</td>
<td>10</td>
<td>11.00</td>
<td>0.15</td>
<td>3600</td>
<td>12.50</td>
<td>0.00</td>
<td>3413</td>
</tr>
<tr>
<td>J2-10-35-0</td>
<td>10</td>
<td>20.00</td>
<td>0.18</td>
<td>3600</td>
<td>20.25</td>
<td>0.16</td>
<td>3600</td>
</tr>
<tr>
<td>J2-10-35-1</td>
<td>10</td>
<td>20.00</td>
<td>0.12</td>
<td>3600</td>
<td>20.00</td>
<td>0.11</td>
<td>3600</td>
</tr>
<tr>
<td>J2-10-35-2</td>
<td>10</td>
<td>18.00</td>
<td>0.10</td>
<td>3600</td>
<td>18.38</td>
<td>0.13</td>
<td>3600</td>
</tr>
<tr>
<td>J2-10-35-3</td>
<td>10</td>
<td>18.00</td>
<td>0.13</td>
<td>3600</td>
<td>20.00</td>
<td>0.03</td>
<td>3600</td>
</tr>
<tr>
<td>J2-10-35-4</td>
<td>10</td>
<td>18.00</td>
<td>0.08</td>
<td>3600</td>
<td>18.00</td>
<td>0.10</td>
<td>3600</td>
</tr>
<tr>
<td>J1-12-20-0</td>
<td>12</td>
<td>10.00</td>
<td>0.17</td>
<td>3600</td>
<td>11.56</td>
<td>0.04</td>
<td>3600</td>
</tr>
<tr>
<td>J1-12-20-1</td>
<td>12</td>
<td>9.00</td>
<td>0.18</td>
<td>3600</td>
<td>9.00</td>
<td>0.25</td>
<td>3600</td>
</tr>
<tr>
<td>J1-12-20-2</td>
<td>12</td>
<td>10.00</td>
<td>0.23</td>
<td>3600</td>
<td>12.00</td>
<td>0.00</td>
<td>22056</td>
</tr>
<tr>
<td>J1-12-20-3</td>
<td>12</td>
<td>7.00</td>
<td>0.22</td>
<td>3600</td>
<td>7.00</td>
<td>0.22</td>
<td>3600</td>
</tr>
<tr>
<td>J1-12-20-4</td>
<td>12</td>
<td>8.00</td>
<td>0.43</td>
<td>3600</td>
<td>9.01</td>
<td>0.40</td>
<td>3600</td>
</tr>
<tr>
<td>J2-12-35-0</td>
<td>12</td>
<td>20.25</td>
<td>0.26</td>
<td>3600</td>
<td>20.00</td>
<td>0.35</td>
<td>3600</td>
</tr>
<tr>
<td>J2-12-35-1</td>
<td>12</td>
<td>17.00</td>
<td>0.35</td>
<td>3600</td>
<td>16.37</td>
<td>0.49</td>
<td>3600</td>
</tr>
<tr>
<td>J2-12-35-2</td>
<td>12</td>
<td>15.50</td>
<td>0.30</td>
<td>3600</td>
<td>14.00</td>
<td>0.43</td>
<td>3600</td>
</tr>
<tr>
<td>J2-12-35-3</td>
<td>12</td>
<td>16.00</td>
<td>0.29</td>
<td>3600</td>
<td>16.00</td>
<td>0.30</td>
<td>3600</td>
</tr>
<tr>
<td>J2-12-35-4</td>
<td>12</td>
<td>18.00</td>
<td>0.26</td>
<td>3600</td>
<td>15.20</td>
<td>0.42</td>
<td>3600</td>
</tr>
<tr>
<td>fu</td>
<td>12</td>
<td>28.00</td>
<td>0.17</td>
<td>3600</td>
<td>24.00</td>
<td>0.29</td>
<td>3600</td>
</tr>
<tr>
<td>J1-14-20-0</td>
<td>14</td>
<td>7.00</td>
<td>0.46</td>
<td>3600</td>
<td>10.00</td>
<td>0.33</td>
<td>3600</td>
</tr>
<tr>
<td>J1-14-20-1</td>
<td>14</td>
<td>6.08</td>
<td>0.49</td>
<td>3600</td>
<td>8.00</td>
<td>0.43</td>
<td>3600</td>
</tr>
<tr>
<td>J1-14-20-2</td>
<td>14</td>
<td>8.00</td>
<td>0.47</td>
<td>3600</td>
<td>8.30</td>
<td>0.48</td>
<td>3600</td>
</tr>
<tr>
<td>J1-14-20-3</td>
<td>14</td>
<td>6.25</td>
<td>0.43</td>
<td>3600</td>
<td>7.00</td>
<td>0.42</td>
<td>3600</td>
</tr>
<tr>
<td>J1-14-20-4</td>
<td>14</td>
<td>8.00</td>
<td>0.47</td>
<td>3600</td>
<td>10.00</td>
<td>0.33</td>
<td>3600</td>
</tr>
<tr>
<td>J2-14-35-0</td>
<td>14</td>
<td>12.78</td>
<td>0.57</td>
<td>3600</td>
<td>16.67</td>
<td>0.52</td>
<td>3600</td>
</tr>
<tr>
<td>J2-14-35-1</td>
<td>14</td>
<td>12.00</td>
<td>0.59</td>
<td>3600</td>
<td>14.50</td>
<td>0.59</td>
<td>3600</td>
</tr>
<tr>
<td>J2-14-35-2</td>
<td>14</td>
<td>13.00</td>
<td>0.48</td>
<td>3600</td>
<td>14.18</td>
<td>0.53</td>
<td>3600</td>
</tr>
<tr>
<td>J2-14-35-3</td>
<td>14</td>
<td>12.00</td>
<td>0.54</td>
<td>3600</td>
<td>13.00</td>
<td>0.59</td>
<td>3600</td>
</tr>
<tr>
<td>J2-14-35-4</td>
<td>14</td>
<td>12.00</td>
<td>0.54</td>
<td>3600</td>
<td>14.00</td>
<td>0.50</td>
<td>3600</td>
</tr>
<tr>
<td>poly1a</td>
<td>15</td>
<td>13.00</td>
<td>0.21</td>
<td>3600</td>
<td>13.00</td>
<td>0.28</td>
<td>3600</td>
</tr>
<tr>
<td>dighe1ok</td>
<td>16</td>
<td>56.00</td>
<td>0.58</td>
<td>3600</td>
<td>65.62</td>
<td>0.57</td>
<td>3600</td>
</tr>
</tbody>
</table>

Average #opt | 0.21 | 2621 | 0.19 | 2302 | 0.14 | 1970
Table 2: Comparing branching strategies

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Optimal solutions</th>
<th>Average GAP</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPLEX</td>
<td>28</td>
<td>0.15</td>
<td>2003</td>
</tr>
<tr>
<td>FL</td>
<td>15</td>
<td>0.22</td>
<td>2566</td>
</tr>
<tr>
<td>FL_L</td>
<td>31</td>
<td>0.07</td>
<td>1499</td>
</tr>
<tr>
<td>FL_A</td>
<td>34</td>
<td>0.05</td>
<td>1322</td>
</tr>
<tr>
<td>FL_A_SA</td>
<td>33</td>
<td>0.05</td>
<td>1323</td>
</tr>
<tr>
<td>SA</td>
<td>10</td>
<td>0.36</td>
<td>2869</td>
</tr>
<tr>
<td>DB</td>
<td>33</td>
<td>0.07</td>
<td>1396</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>0.35</td>
<td>2813</td>
</tr>
</tbody>
</table>

- FL\_L: the Fischetti and Luzzi strategy, ordering the pieces by non-increasing length
- FL\_A: the Fischetti and Luzzi strategy, ordering the pieces by non-increasing area
- FL\_A\_SA: the Fischetti and Luzzi strategy, ordering the pieces by non-increasing area and the variables of each NFP by non-increasing area of the corresponding slice
- SA: ordering the variables by non-increasing area of the corresponding slice
- DB: dynamic branching
- BC: branching on constraints

The results in Table 2 show that the Fischetti and Luzzi strategy works very well, but only if the pieces have been previously ordered by length or, even better, by area. Doing that, the first pieces to be separated are the largest ones and lower bounds increase sharply. The table also shows that adding the ordering of variables by slice area to the previous strategy neither harms nor improves the algorithm and it is very poor when used alone. Dynamic branching works quite well. Its slightly worse results are due to the fact that it needs to read the solution at each node, which slows down the search and much fewer nodes are explored. Nevertheless, the strategy seems promising for a more complex algorithm in which the solution at each node has to be read, for instance in a Branch and Cut procedure in which the separation algorithms run at each node require the current solution. Branching on constraints performs quite poorly. It seems clear that only fixing a variable to 1, separating at least two pieces, has a strong effect on the current solution. When branching on constraints, this effect is missing and the results are much worse.

5.4 Lower bounds

Table 3 shows the relative performance of the three lower bounds described in Section 4.2, expressed as the average percentage distance from each bound to the optimal or best known solution for each instance in the test set. The 1-CBP problem associated with each instance has been solved using CPLEX 12.1, with a time limit of 900 seconds. The quality of the area bound is, obviously, much higher than the bound of the longest piece, specially for large instances. The bound based on the 1-CBP problem is much better than the area bound, but its computational cost is sometimes very high and increases with the problem size. Moreover, all these bounds do not evolve during the search because the integer variables of the formulation do not fix the absolute position of the pieces, but only their relative position. Therefore, we decided not to spend time at the beginning of the algorithm calculating bounds and only the longest piece bound, included in the formulation, is used.
5.5 The effect of updating bounds and fixing variables

Tables 4 and 5 show the effect on the performance of the algorithm of updating bounds and fixing variables as described in Sections 4.2 and 4.3. Table 4 focuses on the 34 instances solved to optimality and therefore the information of interest is the number of nodes and the running times. Table 5 contains the relevant information for the 16 instances that could not be solved to optimality within the limit of 3600 seconds, the GAP between the lower and the upper bounds and the value of the lower bound. In each table, we compare the results obtained by the Initial strategy (column 2), without updating bounds and fixing variables, with several strategies developed for implementing these procedures. In Section 4.2 two methods for updating variables were developed. Method 1, using the non-overlapping constraints, is exact but much slower. Method 2, based on the extended slices, is slightly inexact but faster. A preliminary computational comparison between them clearly showed that the second method performed much better for the same time limit. Therefore, Method 2 is used. The first natural strategy was to update bounds at each node in which a variable had been fixed to 1 and then to try to fix variables for all the variables not yet fixed in the problem. The results of this Strategy 1 appear in column 3. Using these procedures for all the variables in all the nodes in which a variable has been fixed to 1 has a positive effect in reducing the number of nodes to be explored, but the required computational effort slows down the algorithm. In Table 4 we observe a significant reduction in the number of nodes, but the running time is more than doubled. In Table 5, with a time limit, the results are worse than the initial ones in terms of GAP and lower bound. Therefore, it seemed necessary to modify the strategy to reduce the computational effort of these procedures. Columns 4 show the results of Strategy 2, when not all the variables but only those strictly positive in the solution are considered for fixing. The results improve those of the previous strategy in terms of computing time, though the reduction of nodes is not so sharp, but they are still worse than the initial ones. A further way of reducing the computational burden is not using the procedures at every node in which one variable is fixed to 1, but only in some nodes, allowing the solution to be more profoundly changed before using them again. That can be done in two ways: calling the procedures after a given number of nodes, for instance every 5, 10 or 25 nodes, or calling them when, in the branch to which the node belongs, a given number of variables has been fixed to 1 from the last call, for instance 3 or 5 variables. Both alternatives have been tested for the values mentioned. The best results are obtained for 10 nodes (Strategy 3) and for 3 variables (Strategy 4) and they appear in columns 5 and 6. Strategy 3 seems to be the best alternative. For the instances solved, there is a small reduction in the number of nodes and a slight increase in the running time. Therefore, its results are very similar to those of the Initial strategy. For the unsolved instances, the results are slightly better, decreasing the GAP and increasing the lower bound. In summary, updating bounds and fixing variables to 0 has a positive effect only if the computational effort involved is carefully balanced with their advantages in terms of reducing the size of the problem to be solved in successor nodes. According to our results, these procedures should be used every 10 nodes in which a variable has been fixed to 1 and only for variables which are strictly positive.
Table 5: Comparing strategies for updating bounds and fixing variables: Unsolved instances

<table>
<thead>
<tr>
<th></th>
<th>Initial</th>
<th>Strategy 1</th>
<th>Strategy 2</th>
<th>Strategy 3</th>
<th>Strategy 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average GAP (%)</td>
<td>15.9</td>
<td>19.3</td>
<td>17.5</td>
<td>15.4</td>
<td>19.0</td>
</tr>
<tr>
<td>Average LB</td>
<td>25.14</td>
<td>24.42</td>
<td>24.60</td>
<td>25.18</td>
<td>24.48</td>
</tr>
</tbody>
</table>

5.6 Results of the complete algorithm

Summarizing the results obtained in previous subsections, our algorithm uses formulation $HS_2$, with the branching strategy $FL_A$, no lower bounds, except the longest piece bound, and a strategy for updating bounds and fixing variables in which the procedures are called every 10 nodes in which a variable has been fixed to 1, and only strictly positive variables in the current solution are considered for fixing. The complete results of the final version of the exact algorithm appear in Tables 6 and 7, separating its behavior for instances solved to optimality within the initial time limit of 3600 seconds from those not solved in that time. In Table 6, for each instance solved to optimality we show the optimal solution, number of nodes in the tree and running time. In Table 7, we allow the algorithm to run much longer in order to study its evolution for harder problems and show the lower and upper bounds obtained at each milestone (1 hour, 2 hours, 5 hours, 10 hours). The last two columns show the total running time, if the instance has been solved to optimality, and the optimal solution if it is known. If this is not the case, the value corresponds to the best known solution and is marked with an asterisk.

The results of these two tables indicate that our branch and bound procedure is able to solve optimally all the instances with up to 10 pieces, most of those with 12 pieces and some of those with 14 pieces. Even for the solved instances, there are large differences in terms of the number of nodes in the search tree and running times. For example, instances $threep3$ and $threep3w9$ have the same pieces and only differ in the strip width, 7 and 9 respectively, but this small increase results in a very large increase in the solution time. Comparing sets $J_1$ and $J_2$, we can observe that instances derived from $J_1$ are much easier than those derived from $J_2$. Pieces in $J_1$ are more regular and fit together more nicely, while pieces in $J_2$ are more irregular and there are always large amounts of waste between them (see Annex).

Table 7 shows that long runs for instances $J_2$ with 12 pieces can obtain if not optimal, then at least solutions which are very close to optimality, while for instances of the same set with 14 instances even for long runs the gaps between lower and upper bounds do not close.

In summary, even for the moderate number of pieces of the instances tested, our integer formulation, based on assigning variables to regions derived from the edges of the non-fit-polygons, involves a large set of binary variables. Good branching strategies and reduction procedures, plus the power of the latest version of CPLEX, are not enough to speed the search process and ensure an optimal solution for medium size problems. The optimal solutions for the tested instances appear in the Annex. When optimality has not been reached, the figure indicates that it corresponds to an upper bound. These instances will be available at the ESICUP web page and therefore all researchers on nesting problems could use them to assess the behavior of their exact or heuristic procedures.

6 Conclusions

In this paper we have developed a systematic study of the nesting problem. We have reviewed the existing integer linear formulations and we have proposed a new formulation based on the ideas of Fischetti and Luzzi (2008). We have also studied several branching strategies, lower bounds and procedures for fixing variables. All these contributions have been tested on a large set of instances, some of them taken from literature and some others which were randomly generated. The computational study shows that the proposed formulation is more efficient than previous alternatives, that the branching strategy is critical for the performance of the algorithm and that the reduction procedures are only useful if their
Table 6: Results of the Branch and Bound algorithm: Instances solved in less than 3600 seconds

<table>
<thead>
<tr>
<th>Instance</th>
<th>Pieces</th>
<th>Nodes</th>
<th>Time</th>
<th>Instance</th>
<th>Pieces</th>
<th>Nodes</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>three</td>
<td>3</td>
<td>0</td>
<td>0.76</td>
<td>J1,10,20,0</td>
<td>10</td>
<td>8917</td>
<td>10.2</td>
</tr>
<tr>
<td>shapes_4</td>
<td>4</td>
<td>10</td>
<td>0.06</td>
<td>J1,10,20,1</td>
<td>10</td>
<td>10627</td>
<td>8.3</td>
</tr>
<tr>
<td>fu_5</td>
<td>5</td>
<td>511</td>
<td>0.67</td>
<td>J1,10,20,2</td>
<td>10</td>
<td>12361</td>
<td>8.7</td>
</tr>
<tr>
<td>glass1</td>
<td>5</td>
<td>0</td>
<td>0.02</td>
<td>J1,10,20,3</td>
<td>10</td>
<td>957992</td>
<td>469.4</td>
</tr>
<tr>
<td>fu_6</td>
<td>6</td>
<td>153</td>
<td>0.16</td>
<td>J1,10,20,4</td>
<td>10</td>
<td>117337</td>
<td>63.5</td>
</tr>
<tr>
<td>three2</td>
<td>6</td>
<td>6536</td>
<td>1.14</td>
<td>J2,10,35,0</td>
<td>10</td>
<td>1691310</td>
<td>630.6</td>
</tr>
<tr>
<td>shapes</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>J1,10,20,1</td>
<td>10</td>
<td>10627</td>
<td>8.3</td>
</tr>
<tr>
<td>fu_7</td>
<td>7</td>
<td>157</td>
<td>0.56</td>
<td>J2,10,35,2</td>
<td>10</td>
<td>269290</td>
<td>101.3</td>
</tr>
<tr>
<td>glass2</td>
<td>7</td>
<td>2278</td>
<td>1.95</td>
<td>J2,10,35,3</td>
<td>10</td>
<td>2725640</td>
<td>736.5</td>
</tr>
<tr>
<td>fu_8</td>
<td>8</td>
<td>277</td>
<td>0.84</td>
<td>J2,10,35,4</td>
<td>10</td>
<td>1070385</td>
<td>281.5</td>
</tr>
<tr>
<td>shapes_8</td>
<td>8</td>
<td>11969</td>
<td>4.76</td>
<td>J1,12,20,0</td>
<td>12</td>
<td>67833</td>
<td>29.9</td>
</tr>
<tr>
<td>fu_9</td>
<td>9</td>
<td>203094</td>
<td>37.58</td>
<td>J1,12,20,1</td>
<td>12</td>
<td>106897</td>
<td>46.6</td>
</tr>
<tr>
<td>threep2</td>
<td>9</td>
<td>3056101</td>
<td>1107.1</td>
<td>J1,12,20,2</td>
<td>12</td>
<td>24819</td>
<td>15.1</td>
</tr>
<tr>
<td>three2w9</td>
<td>9</td>
<td>11803440</td>
<td>3365.6</td>
<td>J1,12,20,3</td>
<td>12</td>
<td>21502</td>
<td>12.9</td>
</tr>
<tr>
<td>glass3</td>
<td>9</td>
<td>11803440</td>
<td>3365.6</td>
<td>J1,12,20,4</td>
<td>12</td>
<td>564821</td>
<td>282.5</td>
</tr>
<tr>
<td>fu_10</td>
<td>10</td>
<td>1133842</td>
<td>272.5</td>
<td>J1,14,20,0</td>
<td>12</td>
<td>1162529</td>
<td>601.8</td>
</tr>
<tr>
<td>dighe2</td>
<td>10</td>
<td>3835</td>
<td>2.5</td>
<td>J1,14,20,3</td>
<td>12</td>
<td>219984</td>
<td>120.6</td>
</tr>
</tbody>
</table>

Table 7: Results of the Branch and Bound algorithm: Instances not solved in 3600 seconds

<table>
<thead>
<tr>
<th>Instance</th>
<th>Pieces</th>
<th>1 hour</th>
<th>2 hours</th>
<th>5 hours</th>
<th>10 hours</th>
<th>Time</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2-12-35-0</td>
<td>12</td>
<td>24.3</td>
<td>25.4</td>
<td>26.2</td>
<td>26.2</td>
<td>12803</td>
<td>26.21</td>
</tr>
<tr>
<td>J2-12-35-1</td>
<td>12</td>
<td>22.0</td>
<td>22.5</td>
<td>23.1</td>
<td>24.4</td>
<td>24.0</td>
<td>24.4</td>
</tr>
<tr>
<td>J2-12-35-2</td>
<td>12</td>
<td>20.0</td>
<td>20.0</td>
<td>20.0</td>
<td>20.4</td>
<td>22.0</td>
<td>21.50</td>
</tr>
<tr>
<td>J2-12-35-3</td>
<td>12</td>
<td>20.0</td>
<td>22.8</td>
<td>20.0</td>
<td>22.0</td>
<td>21.0</td>
<td>21.7</td>
</tr>
<tr>
<td>J2-12-35-4</td>
<td>12</td>
<td>22.0</td>
<td>24.0</td>
<td>22.7</td>
<td>23.8</td>
<td>22.9</td>
<td>23.8</td>
</tr>
<tr>
<td>fu</td>
<td>12</td>
<td>32.2</td>
<td>33.5</td>
<td>33.1</td>
<td>33.1</td>
<td>5844</td>
<td>33.1</td>
</tr>
<tr>
<td>J1-14-20-1</td>
<td>14</td>
<td>11.0</td>
<td>11.7</td>
<td>11.3</td>
<td>11.3</td>
<td>6004</td>
<td>11.3</td>
</tr>
<tr>
<td>J1-14-20-2</td>
<td>14</td>
<td>13.0</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
<td>14996</td>
<td>14.0</td>
</tr>
<tr>
<td>J1-14-20-4</td>
<td>14</td>
<td>13.8</td>
<td>14.0</td>
<td>14.0</td>
<td>14.0</td>
<td>3984</td>
<td>14.0</td>
</tr>
<tr>
<td>J2-14-35-0</td>
<td>14</td>
<td>23.5</td>
<td>31.3</td>
<td>24.0</td>
<td>30.0</td>
<td>24.4</td>
<td>28.0</td>
</tr>
<tr>
<td>J2-14-35-1</td>
<td>14</td>
<td>22.0</td>
<td>30.0</td>
<td>22.0</td>
<td>30.0</td>
<td>23.5</td>
<td>28.00*</td>
</tr>
<tr>
<td>J2-14-35-2</td>
<td>14</td>
<td>20.0</td>
<td>27.0</td>
<td>20.0</td>
<td>27.0</td>
<td>20.0</td>
<td>24.75*</td>
</tr>
<tr>
<td>J2-14-35-3</td>
<td>14</td>
<td>20.3</td>
<td>26.0</td>
<td>21.6</td>
<td>26.0</td>
<td>22.0</td>
<td>25.00*</td>
</tr>
<tr>
<td>J2-14-35-4</td>
<td>14</td>
<td>20.0</td>
<td>26.0</td>
<td>20.7</td>
<td>26.0</td>
<td>21.3</td>
<td>24.00*</td>
</tr>
<tr>
<td>poly1a</td>
<td>15</td>
<td>13.0</td>
<td>17.0</td>
<td>13.0</td>
<td>16.2</td>
<td>13.0</td>
<td>15.07*</td>
</tr>
<tr>
<td>dighe1ok</td>
<td>16</td>
<td>95.0</td>
<td>137.4</td>
<td>98.2</td>
<td>137.3</td>
<td>100.0</td>
<td>11109</td>
</tr>
</tbody>
</table>
computational effort is carefully balanced with their effect on producing smaller search trees. The results also show that the proposed algorithm can only solve instances of a moderate size.

There are two alternative ways of extending this work. On the one hand, the formulation can be enhanced by identifying valid inequalities and developing a branch and cut procedure. On the other hand, the proposed model and algorithm can be used to develop a matheuristic algorithm, that is an algorithm that combines heuristic and exact procedures.

Acknowledgements

Special thanks to Miguel Gomes and Jose Fernando Oliveira for providing us with the NFP data of test problems, their codes for the formulation and the heuristic algorithm.

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

References


Annex

three: \( L = 6 \)
threep2: \( L = 9.33 \)
threep2e9: \( L = 8 \)
threep3: \( L = 13.53 \)
threep3e9: \( L = 11 \)

shapes4: \( L = 24 \)
shapes8: \( L = 26 \)
fu5: \( L = 17.89 \)
fu6: \( L = 23 \)
fu7: \( L = 24 \)

fu8: \( L = 24 \)
fu9: \( L = 25 \)
fu10: \( L = 28.68 \)
fu11: \( L = 33.13 \)
poly1: \( L_{a,b} = 15.13 \)
glass1: \( L = 45 \)
glass2: \( L = 45 \)
glass3: \( L = 100 \)
dighe2: \( L = 100 \)
dighe1ok: \( L = 100 \)

J1-10-20-0: \( L = 18 \)
J1-10-20-1: \( L = 17 \)
J1-10-20-2: \( L = 20 \)

J1-10-20-3: \( L = 20.75 \)
J1-10-20-4: \( L = 12.5 \)