A Matheuristic for the Irregular Bin Packing with free rotation.

A. Martinez-Sykora[a], R. Alvarez-Valdes[b], J. Bennell[a], R. Ruiz[c], J.M. Tamarit[b]

a University of Southampton, Southampton, UK
J.A.Bennell@soton.ac.uk; A.Martinez-Sykora@soton.ac.uk

b Dept. of Statistics and Operations Research, University of Valencia, Doctor Moliner 50, 46100 Burjassot, Spain
ramon.alvarez@uv.es; jose.tamarit@uv.es

c Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain.
rruiz@eio.upv.es

Abstract

We present a constructive algorithm able to solve a wide variety of variants of the two-dimensional irregular bin packing problem (2DIBPP). The aim of the 2DIBPP is to pack a set of irregular pieces, which may have concavities, into stock sheets (bins) with fixed dimensions in such a way that the utilization is maximized. This problem is inspired by a real application from a ceramic company in Spain. In addition, this problem arises in other industries such as the garment industry or ship building. The constructive procedure presented in this paper allows both free orientation for the pieces, as in the case of the ceramic industry, or a finite set of orientations as in the case of the garment industry. The algorithms presented in this paper are the first to consider free rotation and one of the few to address the bin packing problem with irregular pieces. We propose several Integer Programming models to determine the association between pieces and bins and then we use a Mixed Integer Programming model for placing the pieces into the bins. The computational results show that the algorithm obtains high quality results in sets of instances with different properties. We have used both real data and the available data in the literature of 2D irregular strip packing and bin packing problems.

Keywords: Cutting and packing; Two-Dimensional irregular bin packing; Integer Programming.

1 Introduction

A Spanish company in the ceramic tile sector, Butech Building Technology, by PORCELANOSA Group, has developed a new solid surface which, due to its properties of strength, durability, and color stability, can be used for ventilated facades, among many other applications. The material, whose commercial name is Krion®, is produced in plates of 366.6 by 75 centimeters. Particular to this product is the possibility to cut it into pieces of any polygonal shape, convex or not, following the design developed by the architects. This is a significant departure from the classical tiles used in ventilated facades, which are always of rectangular shape. The problem is, then, to determine how the demanded pieces are cut from the plates so as to minimize the number of required plates. As this new material is quite expensive and the number of pieces in a project can be very large, reducing the number of plates to a minimum can produce substantial savings. At the time undertaking this research, the company manually designed the cutting patterns, where each order could take up to four person weeks.

The above cutting problem is a two-dimensional bin packing problem with irregular (non-convex) pieces (2DIBPP). According to the typology proposed by Wäscher et al. (2007), the 2DIBPP problem can be classified as a two-dimensional Irregular Single Bin Size Bin Packing Problem (SBSBPP) where pieces to be cut from the bins (plates) are simple polygons. There are no restrictions on the angle of rotation, making the problem more difficult to solve efficiently. Further, the pieces cannot be reflected because each side of the plate has a different texture and only one side can face outwards. The objective is to minimize the total number of bins needed to cut all the pieces.

The most studied packing problem involving irregular shapes that can be found in the literature is the two-dimensional strip irregular packing problem, also known as the Nesting Problem (see Bennell and Oliveira (2009) for a survey). There are few publications considering the bin packing problem with irregular pieces and, to our knowledge, there is no publication allowing items to be continuously rotated. Terashima-Marin et al. (2010) propose a hyper-heuristic algorithm which combines several placement heuristics. They combine the Bottom-Left algorithm developed by Jakobs (1996) and an improved version proposed by Liu and H.Teng (1999). For both

procedures Terashima-Marin et al. (2010) add the possibility of rotating the pieces at specific angles. Furthermore, they include different modifications of the constructive approach presented by Hifi and M'Hallah (2002). The hyper-heuristic is embedded into a Genetic Algorithm and it is tested basically on jigsaw puzzle instances with convex pieces. Lopez-Camacho et al. (2013) propose an adaptation of the Djang and Finch heuristic (DJD), proposed by Ross et al. (2002) for the one-dimension bin packing problems, to the two-dimensional irregular bin packing problems. Despite the fact the algorithm proposed by Lopez-Camacho et al. (2013) consider non-convex pieces, in the computational experiments they only report results involving pieces with convex shapes.

Another interesting application of bin packing problems with irregular pieces arises in the glass industry. Since the cutting process forces the use of guillotine cuts, the pieces have to be convex, but they are not restricted to take rectangular shapes. Bennell et al. (2012) propose two constructive heuristics and report results on some real data. More recently, Martinez-Sykora et al. (2015) present several MIP-based constructive procedures in which they improve the best known solutions in all the instances and, furthermore, they obtain competitive results on rectangular bin packing problems. Another study that considers multiple bins and irregular pieces is that of Song and Bennell (2013), who propose a column generation procedure for solving the irregular shape cutting stock problem. The authors use the beam search algorithm proposed in Bennell and Song (2010) for generating the patterns and study three solution approaches: column generation, adapted column generation and a sequential heuristic procedure. In the computational experiments the authors build eleven instances obtained from well-known 2D irregular strip packing instances by fixing the length and multiplying the demand of each shape by 100. Therefore, despite the fact the instances are non-convex and in some of the instances they have quite complex shapes, the problem is a cutting stock problem, which is slightly different than the two-dimensional bin packing problem considered in our study.

This paper describes a constructive algorithm which makes decisions iteratively. The distinctive feature of the algorithm is that some of the decisions concerning the positions of the pieces are flexible and can be modified during the constructive procedure. The algorithm is divided into two phases. The first phase determines the subset of pieces assigned to the next bin. We have developed and tested several strategies, ranging from simple priority rules based on the dimensions of the pieces, to solving variants of the one-dimensional Bin Packing Problem. In a second phase, the pieces of the selected subset are taken one at a time and attempted to be cut from the new bin. This phase consists of two steps. First, we determine a set of promising rotations for the piece and then, for each rotation, we determine if the piece fits into the bin by solving a Mixed Integer Programming (MIP) model.

The paper is organised as follows: in Section 2 we define the problem and some notation. In Section 3 and Section 4 we give, respectively, the mathematical formulation to be solved in order to determine the assignment of pieces to bins and the placement of items into each bin. Section 5 contains two local search procedures designed to improve the solutions obtained. In Section 6 we describe the implementation of the constructive algorithm, including how the rotations of the items are calculated, and in Section 7 we present the computational results in different subsections: the instances obtained from the real data, the existing instances available in the literature, and some complex instances we have built from the instances available for the Nesting Problem.

2 Problem description and algorithm outline

We denote by $P = \{1, \ldots, n\}$ the set of n pieces to be placed into identical rectangular bins. The number of available bins can be considered large enough to pack all the items and the objective is to minimize the total number of bins needed to cut all the pieces in P. The width and length of the bins are denoted by W and L respectively. The pieces can be rotated continuously. We denote by $IBPP(P, L, W)$ the problem which minimizes the number of bins needed to pack all the pieces in P into bins whose length and width are L and W, respectively.

A solution of $IBPP(P, L, W)$ is given by a set of bins $B = \{b_1, \ldots, b_N\}$. Each bin $b_i(P_i, O_i, X_i, Y_i)$, $\forall i = 1, \ldots, N$, is composed of a set of pieces $P_i \subseteq P$, a vector $O_i = \{o_1, \ldots, o_{n_i}\}$, defining the orientation of the pieces, where $n_i = |P_i|$, $o_k \in [0, 2\pi[$, $k = 1, \ldots, n_i$, and two vectors X_i and Y_i which are the X-coordinate and Y-coordinate of the reference point of each piece. We define the reference point of the piece $i \in P$ in the orientation o_i as the bottom left corner of the enclosing rectangle (see Figure 1). In addition, we consider that the bottom-left corner of each bin is placed at (0,0), so that the coordinates of the reference points are always positive. Note that, since all the pieces have to be placed, then $P = \cup_{i=1}^{N} P_i$.

Despite the fact that the objective of minimizing the total number of bins is quite straightforward, we are going to consider a secondary objective in order to compare solutions that use the same number of bins. The idea is to apply a vertical or a horizontal cut in order to separate the non-utilized part of the bin for future use. The cut is

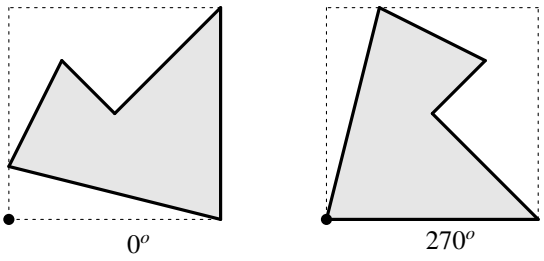

Figure 1: *Reference point of a piece in two different orientations*

applied to the least utilized bin only. We define the fractional number of bins as follows:

$$K = N - 1 + R^* \tag{1}$$

where R^* is the proportion of utilized area of the whole bin after applying the vertical and horizontal cuts. Figure 2 shows two different solution of instance *Han* (in class Nest-MB), taken from Nesting Problems. Though in both solutions $N = 3$, using the secondary objective solution (b) is better because the fractional number of bins after saving the un-utilized part of the last bin is $K_b = 2.20$, which is less than for solution (a), $K_a = 2.35$. In these two cases the cuts are vertical rather than horizontal because this gave the lower fraction bin.

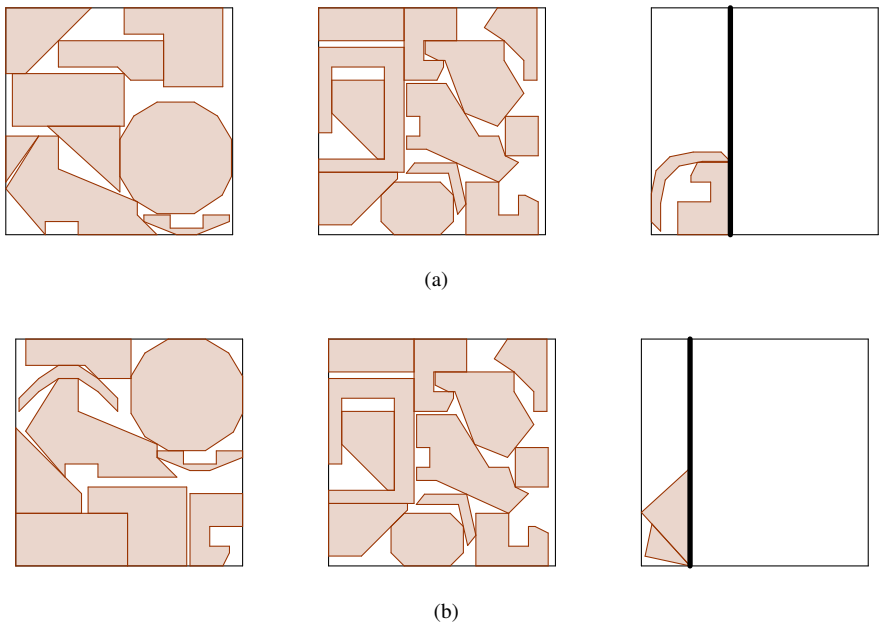

Figure 2: *Two solutions of instance han (Nest-MB) obtained by CA-PBP-20 and CA-PBP-20-LS2. a) $K = 2.35$; (b) $K = 2.20$.*

Lopez-Camacho et al. (2013) propose an alternative measure of performance based on the percentage usage of each bin as follows:

$$F = \frac{\sum_{i=1}^{N} U_i^2}{N} \tag{2}$$

where U_i is the utilization ratio of each bin $i \in 1 \ldots, N$. The idea is to maximise F. This measure also avoids the ties among different solutions with the same number of bins, but F and U are not equivalent. While improving the value of U will improve the value of F, a solution with a better F does not necessarily lead to a better U. In our case, we consider that minimizing the usage U is more relevant because the aim is to keep as much raw material as possible for future use. However, we employ this measure of performance for comparing with the results presented in Lopez-Camacho et al. (2013).

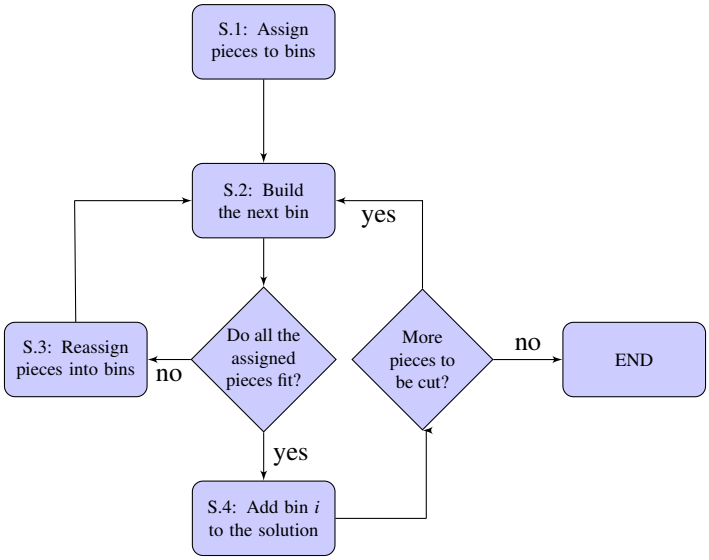

Figure 3: *Scheme of the constructive procedure*

The constructive procedure is organized as shown in Figure 3. In S.1, we decide which pieces are assigned to each bin by using a variety of strategies including solving an IP formulation of the one dimensional bin packing problems (1DBPP). Given the assignment, S.2 attempts to place the pieces into the next bin by solving a *MIP* model for the insertion of each piece previously ordered by non increasing area. In Section 4 we describe the MIP models to obtain the coordinates of the pieces and the procedure to obtain promising rotations. If a feasible solution for the bin is found, e.g, all the pieces fit into the bin with no overlap, then we add the bin to the solution (S.4) and we build the next bin (S.2) unless all pieces are packed. If all the MIP models built for placing a piece into the bin are infeasible, we assume that the piece does not fit into the bin. Therefore, we go to step (S.3) for reassigning the pieces into the bins.

3 Assignment models

In this section we describe a number of approaches for assigning pieces to bins. This includes a two-phase exact procedure and then some relaxation strategies leading to different heuristic procedures.

Let $\overline{N}$ be an upper bound on the total number of bins required to pack all the pieces, obtained by using the First Fit Decreasing (FFD) algorithm (Johnson et al. (1974)). We define the following binary variables: $s_i, \forall i = 1, \dots \overline{N}$, which take value 1 if bin b_i is used in the solution and 0 otherwise, and z_{ij}, $i = 1, \dots, \overline{N}$, $j = 1, \dots, n$, which take value 1 if piece j is assigned to bin i and 0 otherwise. An integer programming model for assigning pieces to bins can be formulated as a 1DBPP as follows:

$$Min. \ \sum_{i=1}^{\overline{N}} s_i, \tag{3}$$

$$\sum_{j=1}^{n} a_j z_{ij} \leq LW, \qquad 1 \leq i \leq \overline{N}, \tag{4}$$

$$\sum_{i=1}^{\overline{N}} z_{ij} = 1, \qquad 1 \leq j \leq n, \tag{5}$$

$$z_{ij} \leq s_i, \qquad 1 \leq j \leq n, \ 1 \leq i \leq \overline{N}, \tag{6}$$

$$s_i \in \{0,1\}, \ z_{ij} \in \{0,1\} \qquad 1 \leq j \leq n, 1 \leq i \leq \overline{N}. \tag{7}$$

The objective function minimizes the total number of bins used in the solution. Constraints (4) ensure that the total area of the pieces assigned to bin b_i does not exceed the area of the bin. Equalities (5) force each piece to be assigned to one bin. Inequalities (6) guarantee that if any piece is placed into a given bin, then the corresponding bin is used in the solution. Finally, constraints (7) force the variables to be binary.

Note that, if this IP model is solved to optimality and we pack all the pieces assigned to each bin successfully, the solution would be optimal. This is highly unlikely because area is a poor approximation of a 2D shape,

particularly when they are irregular. It is for the purpose of assignment, each piece is approximated by its area, a_i, and the problem becomes a 1DBPP. For the cases when we do not find a feasible solution for packing the subset of pieces $Q \subseteq P$ in a bin, an obvious solution to this problem is to add the following constraints

$$\sum_{j \in Q} z_{ij} \leq |Q| - 1, \ 1 \leq i \leq \overline{N}. \tag{8}$$

to the IP model to ensure that not all the pieces in Q will be assigned to the same bin, and solve it again to obtain a new assignment of the pieces. However, we do not follow this strategy because it leads to high computational effort. Note that, while this procedure could be applied until the packing problem in each bin is solved successfully, the number of inequalities added to the IP could be very large leading to high computational times. For this reason we propose several simplifications of this procedure. The first relaxation could be solving the packing problem associated to each bin in a heuristic way. That would reduce the computational time, although the solution obtained would not necessarily be optimal if the numbers of required bins is increased. Nevertheless, even with this simplification the iterative procedure could take very long for large instances. Other strategies, with more adjusted computing time, are described below. In all of them the pieces assigned to a bin are ordered by non-increasing area and the packing procedure takes one piece at a time and tries to place it in the partially filled bin.

3.1 Bin Packing with Greedy Decisions (BPGD)

This strategy is based on the observation that the packing procedure usually fails having packed most of the assigned pieces and attempting to insert the few remaining pieces. If we solve the assignment problem again, the effort in packing the placed pieces would be wasted. The *BPGD* strategy is based on solving the model described above at most once per bin. When trying to pack the pieces assigned to a bin, there are two possible outcomes:

a) The packing algorithm finds a feasible placement for all the pieces. Move to S4 (see scheme in Figure 3).

b) The packing algorithm fails when placing a given piece. In this case, instead of only packing the rest of the assigned pieces. We try to fill up the bin considering all unpacked pieces initially assigned to other bins. We take all the pieces not yet placed, except the one for which the packing failed, order them by non-increasing area and try to place them one at a time in the partially packed bin. When all the pieces have been tried, the bin is closed and we solve the assignment model again for the unpacked pieces.

The IP model has to be solved each time case (b) occurs, because the set of remaining pieces has been modified: one or more pieces assigned to the bin are still unpacked and possibly some pieces assigned to the next bins have already been packed. Note that each time the IP model is solved at least one bin has been closed and the number of the remaining pieces has been reduced. Therefore $\overline{N}$ is updated and the number of binary variables is reduced.

3.2 Partial Bin Packing (PBP)

This strategy is based on the idea that there is no need to assign all the pieces to the bins if we are almost sure that sometime the packing will fail and the assignment will have to be recalculated. Therefore, we focus on the assignment of the pieces to just one bin. Then, the binary variables in that case, q_j, $j = 1 \ldots, n$, take the value 1 when piece j is assigned to the bin and 0 otherwise. The IP model can be formulated as follows:

$$Max. \ \sum_{j=1}^{n} \left(\frac{a_j}{a_{max}}\right)^{\kappa} q_j, \tag{9}$$

$$\sum_{j=1}^{n} a_j q_j \leq LW, \qquad 1 \leq j \leq n, \tag{10}$$

$$q_j \in \{0, 1\}, \qquad 1 \leq j \leq n. \tag{11}$$

where a_{max} is the area of the biggest piece and $\kappa \in [1, \infty)$. When $\kappa = 1$, this objective function maximizes the assigned area of the bin. The problem with $\kappa = 1$ is that it is possible that solutions where only small pieces are assigned to the bin if together they attain the best utilization. This could produce bad global results because the biggest pieces would be assigned to latter bins and it is more difficult to combine big pieces in a bin. Then, packing the big pieces as soon as possible seems to be a good strategy in the constructive procedure. On the other hand, if κ takes a large value, only the biggest pieces will be placed into the bin. We consider $\kappa = 2$, as an acceptable balance between the utilization of the bin and the assignment of the biggest pieces.

If one of the assigned pieces, j, does not fit into the bin, we solve a similar model in order to reassign the pieces to the current bin, but forcing the pieces already placed by the packing procedure to be assigned to the bin and not including piece j.

3.3 First Fit Algorithm (FF)

An alternative to solving IP models to optimality described above is the use of a known heuristic algorithm such as the First Fit Algorithm (FF) (Johnson et al. (1974)). The FF takes an ordered list of pieces and assigns them iteratively to the bins. In order to assign one piece it checks if it fits in any of the bins in the order they were created until a feasible assignment is found. If the piece does not fit into any existing bin, a new bin is created and the piece is assigned to this new bin. This algorithm is very fast and depends critically on the initial ordering of the pieces. Therefore, in order to obtain a good assignment we run it many times (for instance, 5000), starting from random permutations of the pieces, and keep the solution with the minimum number of bins. Ties are broken by the minimum used area of the least utilized bin.

Each time the packing fails we have a set of bins in which the packing has been successfully applied, one bin b_i in which a piece j cannot be placed, and the rest of bins where the packing has not been applied yet.

All the successfully packed bins are kept and their pieces removed from consideration. The pieces in the partially packed bin, b_i, that where successfully placed remain assigned to bin b_i. All remaining pieces are reassigned to bins using FF with the constraint that piece j cannot be assigned to b_i again.

All the pieces already placed in the bins are eliminated and the remaining pieces are assigned into new bins by applying the FF algorithm again. Note that there is one bin partially built, b_i, which corresponds to the last bin where the packing failed to add piece j. Then, we do not allow piece j to be assigned to bin b_i because we know that this piece will not fit into that bin.

3.4 Hybrid Bin Packing (HBP)

In order to combine the quality obtained by an exact procedure as the PBP and the efficiency of the FF we study a hybrid approach. The idea is to apply first the PBP with a time limit. If optimality is proved within this limit, we use the assignment given by PBP in the packing procedure. If the time limit is reached and the solver used (CPLEX) is not able to prove optimality, FF is run several times and the best solution obtained by both procedures is used. The time limit is denoted by $TiLim$, and the number of runs of the FF is denoted by $nruns$. We will consider $TiLim = 45$ seconds and $nruns = 1000$. In the previous strategies, the $TiLim$ used to solve the IP models is 50 seconds and $nrun$ in the FF is 5000. Therefore, we reduce the computational time of the PBP strategy slightly in order to run 1000 iterations of the FF algorithm and pick the best solution.

3.5 Two Phases Strategy (TPS)

This strategy is based on rearranging the pieces in a solution obtained by the BPGD algorithm. While the solution obtained by the BPGD procedure provides the optimal number of bins required to pack all the pieces the solution does not consider how small and large pieces are distributed. In generall it is better to have large pieces in the first bins and use small pieces to fill the gaps. Therefore, we propose another IP model to reassign pieces into bins. Let z_{ij}, $i = 1, \ldots, N'$, $j = 1, \ldots, n$ be the binary variables which take value 1 if piece j is associated with bin i. Note that in that case the total number of bins N' is given by the solution of BPGD algorithm.

$$Min. \ \sum_{i=1}^{N} \sum_{j=1}^{n} \left(\frac{a_j}{a_{max}}\right)^2 \frac{2(N'-i)}{N'^2+N'} \ z_{ij}, \tag{12}$$

$$\sum_{j=1}^{n} a_j z_{ij} \leq LW, \qquad 1 \leq i \leq N', \tag{13}$$

$$\sum_{i=1}^{N'} z_{ij} = 1, \qquad 1 \leq j \leq n, \tag{14}$$

$$z_{ij} \in \{0, 1\}, \qquad 1 \leq i \leq N', \ 1 \leq j \leq n. \tag{15}$$

The objective function (12) forces the assignment of large pieces into the first bins while minimising the total number of bins is minimized. Note that $\left(\frac{a_j}{a_{max}}\right)^2$ takes a greater value when p_j has more area and $\frac{2(N'-i)}{N'^2+N'}$ takes a greater value for the first bins. The constraints (13) and (14) ensure, respectively, that the pieces associated to one bin does not exceed the area of the bin and that each piece is assigned to one bin.

4 Packing model

In this section we describe the model that determines the exact location and orientation of the pieces in a bin. We assume that a set of pieces $P_b \subseteq P$, ordered by non-increasing area, has been assigned to the current bin, b. The MIP model places one piece in the bin in order to minimise the weighted rectangle of the partial solution. Although, the pieces can be rotated continuously, but the MIP we use to pack the pieces requires that each piece has a fixed orientation. As a result, the MIP model is run several times for each piece solving a model for each orientation we want to consider. This idea was first proposed in Martinez-Sykora et al. (2015).

In order to obtain promising rotations for placing the next piece, we consider the edges of the pieces already placed in the bin and the edges of the bin. We calculate the angles of rotation in such a way that one edge of the piece being placed matches with either one edge of a piece already placed or one edge of the bin. If we obtain more than three different rotations, we sort the angles by the following criteria (GR):

- Non-increasing number of matches between the edges of the polygon obtained by applying the given rotation to the piece, and the edges of all the pieces already placed in the bin and the edges of the bin.
- Ties are broken by the total length of the edges of all the matchings.

If the number of rotations is not specified, we consider the first three rotations for the insertion of each piece. If in a given instance, only some fixed orientations are allowed, we use these orientations.

In what follows we give a description of the MIP model. For the sake of clarity, we first describe the model for the insertion of the first piece, then the second piece and finally the general case. The placement model for the first piece has a linear programming formulation, in which the variables x_1 and y_1 correspond to the coordinates of the reference point of piece 1:

$$Min.\ \omega L_u + (1-\omega)W_u, \tag{16}$$

$$L_u \leq L, \tag{17}$$

$$W_u \leq W, \tag{18}$$

$$x_1 \leq L_u, \tag{19}$$

$$y_1 \leq W_u. \tag{20}$$

L_u and W_u denote the used length and width of the bin when placing the first piece and ω is a coefficient which depends on the dimensions of the bin. In order to balance the used length and width when adding pieces we fix $\omega = \frac{1}{(W/L)+1}$. Therefore, with this objective function the pieces will be placed in such a way that the dimensions of the enclosing rectangle are proportional to the dimensions of the bin. Constraints (17) and (18) ensure that the used length and width do not exceed the dimensions of the bin and constraints (19) and (20) force piece 1 to be inside the enclosing rectangle (L_u, W_u).

We solve this linear programming formulation once for each different orientations we consider for the piece. In this step we determine and fix the orientation of the piece for which the objective function is minimized. In Figure 4, the first two rotations are considered because two edges of the piece are parallel to the edges of the bin. The values of x_1 and y_1 given by the LP solution are not fixed in the following models but the orientation is. The third rotation, which happens to be the best, is considered because the largest edge of the piece is parallel to one edge of the bin.

When inserting the second and later pieces, the main difficulty is to ensure that they do not overlap. We use the *MIP* model, proposed by Alvarez-Valdes et al. (2013), in which non-overlapping constraints are obtained using *No-Fit Polygons (NFP)*. The *NFP* is a polygon derived from every pair of pieces in such a way that its interior represents all the overlapping positions between the pieces, and its boundary represents all the touching positions. Figure 5 gives an example of the NFP, corresponding to pieces 1 and 2 (NFP_{12}), where the orientation of piece 1 is fixed in the previous step and piece 2 was the same shape as piece 1 and is rotated 90^0. Deriving the NFP is not the topic of this paper, see Bennell and Oliveira (2008) for a review of procedures. Given the correct placement of piece 1, the reference point of piece 2 must be placed outside NFP_{12} to avoid overlapping. Alvarez-Valdes et al. (2013), following the idea proposed by Fischetti and Luzzi (2008), who divide the outer region into horizontal slices and define a binary variable for each slice so that it takes value 1 if the reference point is in this slice and 0 otherwise.

In the example in Figure 5 we need 13 binary variables. The notation we use to write the formulation is a three-index notation in which the two first indexes represent the pieces, and the third index enumerates the slices (for the sake of clarity in Figure 5 we show only the third index). In general, if pieces have complex shapes, the set

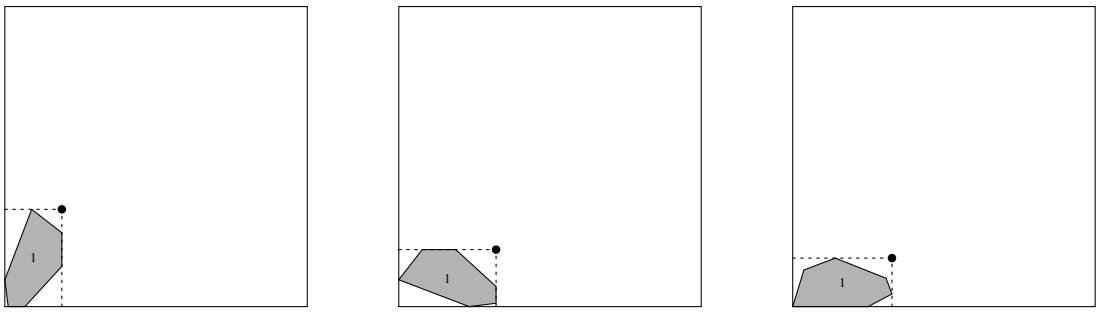

Figure 4: *Three rotations for the first piece (instance poly3a-x2). Rotation* 3 *is selected and fixed*

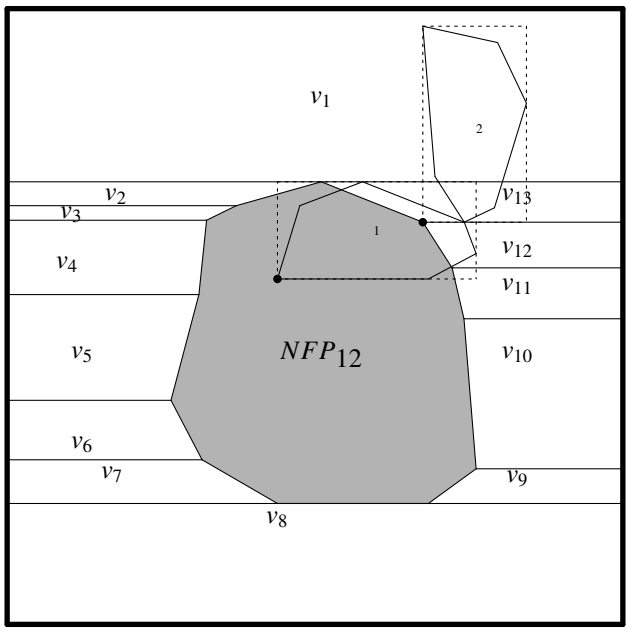

Figure 5: NFP_{12}

of points not belonging to the NFP may have a structure more complex than that of the Figure 5. In these cases, before defining the slices and their associated variables, a preprocess is required and additional binary variables are associated with the subsets of the region not included in the slices (see Alvarez-Valdes et al. (2013) for a complete discussion).

Let m_{12} be the total number of binary variables, i.e, the number of slices we derived from NFP_{12} ($m_{12} = 13$ in Figure 5). We denote by t_{12i} the total number of inequalities required to describe slice S^i associated with v_{12i}, $i = 1, \ldots, m_{12}$. Despite the fact that in Figure 5 $t_{12i} = 4$, $\forall i = 1, \ldots, m_{12}$, it might be possible that t_{12i} takes different values if the NFP_{12} has a more complex shape. In what follows we describe first the non-overlapping constraints and then the containment constraints.

If, for some slice k of the NFP_{12}, $v_{12k} = 1$, the constraints defined by the sides of the slice:

$$\alpha_{12}^{kf}(x_2 - x_1) + \beta_{12}^{kf}(y_2 - y_1) \leq \delta_{12}^{kf} \qquad f = 1, ..., t_{12k} \tag{21}$$

where α_{12}^{kf} and β_{12}^{kf} are the coefficients defining the slope of the inequality and δ_{12}^{kf} the intercept, must be satisfied. However, if $v_{12k} = 0$, these constraints must be relaxed. This can be done by using big-M constants:

$$\alpha_{12}^{kf}(x_2 - x_1) + \beta_{12}^{kf}(y_2 - y_1) \leq \delta_{12}^{kf} + (1 - v_{12k})M \qquad f = 1, ..., t_{12k} \tag{22}$$

Fischetti and Luzzi (2008) describe a procedure for building these constraints avoiding the big-M constants. Since exactly one slice is used in any feasible solution, this equality holds:

$$\sum_{i=1}^{m_{12}} v_{12i} = 1 \tag{23}$$

Then a tighter constant could be written for each slice, eliminating the big-M and including on the right hand side of the inequality all the binary variables, defined by NFP_{12}, multiplied by a given constant. These constants are obtained by solving the following problem:

$$\delta_{12}^{kfh} = max_{p_2 \in S^h}\ \alpha_{12}^{kf}(x_2 - x_1) + \beta_{12}^{kf}(y_2 - y_1) \tag{24}$$

which corresponds to the maximum value of the left hand side when piece 2 lies on slice h, $h \in \{1, \ldots, m_{12}\}$.

Then, the non-overlapping inequalities can be written as follows:

$$\alpha_{12}^{kf}(x_2 - x_1) + \beta_{12}^{kf}(y_2 - y_1) \leq \sum_{h=1}^{m_{12}} \delta_{12}^{kfh} v_{12h} \qquad k = 1, ..., m_{12}, f = 1, ..., t_{12k} \tag{25}$$

The containment constraints ensure that each piece does not exceed the limits of the bin (as with inequalities (19) and (20) in the formulation for one piece). Alvarez-Valdes et al. (2013) describe a lifting procedure for these inequalities, adding some of the binary variables of the corresponding NFP, associated with slices on which one piece protrudes either horizontally or vertically from the other. Let us denote by $\overline{y}_{12k}$ and $\underline{y}_{12k}$ ($\overline{y}_{12k}$ and $\underline{x}_{12k}$) the maximum and minimum value of $y_2 - y_1$ ($x_2 - x_1$), respectively, when slice k is used. Let $\underline{X}^{12}$ and $\underline{Y}^{12}$ be the minimum X-coordinate and Y-coordinate value of NFP_{12} respectively. We define the following classification of the binary variables:

- $U_{12} := \{v_{12k} \mid \underline{y}_{12k} \geq 0\}$, the set of binary variables whose associated slices do not allow piece 2 to protrude from below piece 1. In Figure 5, $U_{12} = \{v_1, v_2, v_3, v_{12}\}$.
- $R_{12} := \{v_{12k} \mid \underline{x}_{12k} \geq 0\}$, the set of binary variables whose associated slices do not allow piece 2 to protrude from the left of piece 1. In Figure 5, $R_{12} = \{v_9, v_{10}, v_{11}, v_{12}\}$.
- $LS_{12} := \{v_{12k} \mid \lambda_{12k} > 0\}$ where $\lambda_{12k} = l_1 - (\overline{x}_{12k} - \underline{X}^{12})$ and l_1 is the length of piece 1 in the current orientation. Then, LS_{12} is the set of binary variables whose associated slices force piece 1 to protrude from the right of piece 2. In Figure 5, $LS_{12} = \{v_2, v_3, v_4, v_5, v_6, v_7\}$.
- $DS_{12} := \{v_{12k} \mid \mu_{12k} > 0\}$ where $\mu_{12k} = w_1 - (\overline{y}_{12k} - \underline{Y}^{12})$ and w_1 denotes the width of piece 1. Then, DS_{12} is the set of binary variables whose associated slices force piece 1 to protrude from above piece 2. In Figure 5, $DS_{12} = \{v_7, v_8, v_9\}$, variable v_6 is not in DS_{12} because $\mu_{126} = 0$.

The containment constraints for piece 1 are:

$$\textstyle\sum_{k \in R_{12}} \underline{x}_{12k} v_{12k} \leq x_1 \leq L_u - l_1 - \sum_{k \in LS_{12}} \lambda_{12k} v_{12k} \tag{26}$$

$$\textstyle\sum_{k \in U_{12}} \underline{y}_{12k} v_{12k} \leq y_1 \leq W_u - w_1 - \sum_{k \in DS_{12}} \mu_{12k} v_{12k} \tag{27}$$

Similarly we can build the containment constraints for piece 2. In Figure 5, $U_{21} = \{v_5, v_6, v_7, v_8, v_9, v_{10}\}$, $R_{21} = \{v_2, v_3, v_4, v_5, v_6, v_7\}$, $LS_{21} = \{v_9, v_{10}, v_{11}, v_{12}\}$, and $DS_{21} = \{v_1, v_2, v_3, v_4, v_{11}, v_{12}\}$. The containment constraints for piece 2 can be written as follows:

$$\textstyle\sum_{k \in R_{21}} \lambda_{21k} v_{12k} \leq x_2 \leq L_u - l_2 - \sum_{k \in LS_{21}} \underline{x}_{12k} v_{12k} \tag{28}$$

$$\textstyle\sum_{k \in U_{21}} \overline{y}_{12k} v_{12k} \leq y_1 \leq W_u - w_2 - \sum_{k \in DS_{21}} \mu_{21k} v_{12k} \tag{29}$$

where $\lambda_{21k} = l_2 - (\overline{X}^{12} - \underline{x}_{12k})$ and $\mu_{21k} = w_2 - (\overline{Y}^{12} - \underline{y}_{12k})$.

The MIP formulation we consider to place pieces 1 and 2 has as objective function expression(16), subject to:

- Constraints (17) and (18) to ensure that L_u and W_u do not exceed the limits of the bin.
- Lifted containment constraints (26) and (27) for piece 1.

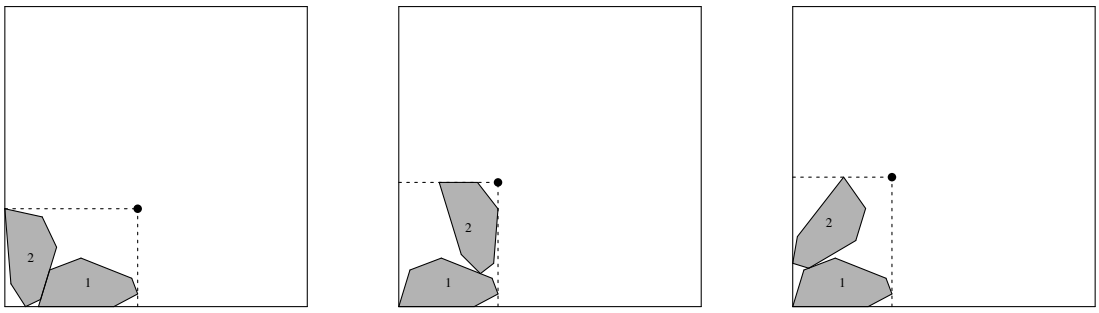

Figure 6: *Three rotations for the second piece. Rotation* 3 *is selected and fixed*

- Lifted containment constraints (28) and (29) for piece 2.
- Non-overlapping constraints (23) and (21).

This MIP model is solved three times since we consider three orientations as illustrated in Figure 6. Each time we solve the model for a given rotation and we obtain an improved feasible solution, the value of its objective function is used as an upper bound for the following rotations. Therefore, only models producing improved solutions are solved to optimality. If the orientation being tested cannot produce a better solution, the model is quickly identified as unfeasible by the solve, speeding up the process. In the three solutions depicted in Figure 6 the orientation of piece 1 is fixed, as a result of the previous step, but is position is determined by the solution of the current model.

There are 13 binary variables in each one of the models solved for the insertion of the second piece. Since this number increases exponentially with the number of pieces already placed into the bin, it maybe necessary to fix the relative position between the pieces already placed to reduce this number. If the computational time needed to solve the MIP model to optimality exceeds a given threshold, θ, then we will fix the binary variables to the values obtained in the previous model. Otherwise, we allow more flexibility by keeping the binary variables free.

For the general case, where the next piece to be inserted is piece j, $j-1$ pieces have been inserted and now have fixed orientation. The model for the insertion of j can be written as follows:

$$Min.\ \omega L_u + (1-\omega)W_u, \tag{30}$$

$$L_u \leq L, \tag{31}$$

$$W_u \leq W, \tag{32}$$

$$\sum_{k\in R} \underline{x}_{j_1 j_2 k} v_{j_1 j_2 k} \leq x_{j_1} \leq L_u - l_{j_1} - \sum_{k\in LS_{j_1 j_2}} \lambda_{j_1 j_2 k} v_{12k}, \qquad j_1, j_2 \in \{1,\ldots,j\} \tag{33}$$

$$\sum_{k\in U_{j_1 j_2}} \underline{y}_{j_1 j_2 k} v_{j_1 j_2 k} \leq y_{j_1} \leq W_u - w_{j_1} - \sum_{k\in DS_{j_1 j_2}} \mu_{j_1 j_2 k} v_{j_1 j_2 k}, \qquad j_1, j_2 \in \{1,\ldots,j\} \tag{34}$$

$$\alpha^{kf}_{j_1 j_2}(x_{j_2} - x_{j_1}) + \beta^{kf}_{j_1 j_2}(y_{j_2} - y_{j_1}) \leq \sum_{h=1}^{m_{j_1 j_2}} \delta^{kfh}_{j_1 j_2} v_{j_1 j_2 h}, \qquad 1 \leq j_1 < j_2 \leq j, k = 1,...,m_{j_1 j_2}, f = 1,\ldots,t_{j_1 j_2 k} \tag{35}$$

$$\sum_{i=1}^{m_{j_1 j_2}} v_{12i} = 1, \qquad 1 \leq j_1 < j_2 \leq i \tag{36}$$

$$v_{j_1 j_2 k} \in \{0,1\}, \qquad j_1, j_2 \in \{1,\ldots,j\} \tag{37}$$

$$x_l,\ y_l \geq 0, \qquad 1 \leq l \leq j \tag{38}$$

Constraints (31) and (32) ensure that L_u and W_u do not exceed the dimensions of the bin. The containment constraints (33) and (34) are required for all the pair of pieces taking into account the order ($j(j-1)$ pairs) and the non-overlapping constraints (35) and (36) are required for all the NFP ($\frac{j(j-1)}{2}$ pairs).

In Figure 7 we show the insertion of piece 15. In this example the previous MIP model was difficult to solve to optimality (the computation time was greater that the threshold θ). Therefore, the relative position of the pieces already placed is fixed while piece 15 is being inserted. We can see that we obtain different solutions with the same objective function value. In this case, the algorithm would solve the first MIP model to optimality (Rotation 1) obtaining a solution. This objective function value will be used in the next two MIPs as upper bound. As a result, the second and third model will be quickly identified as infeasible. Their solutions appear in Figure 7 only for illustrative purposes.

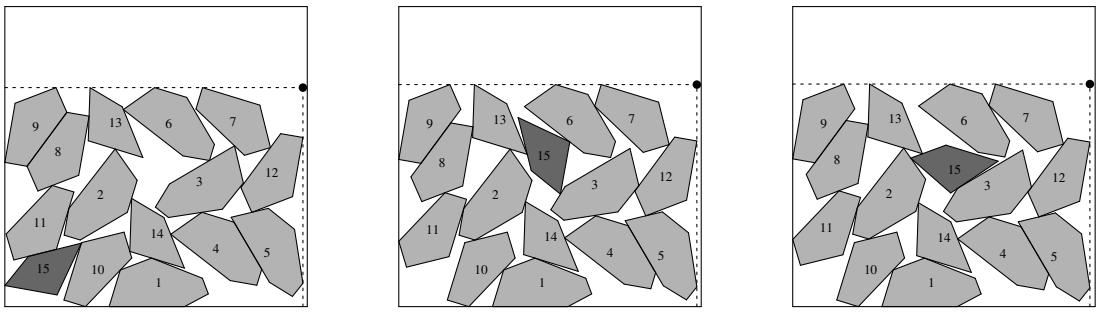

Figure 7: *Three rotations for the 15th piece. Rotation* 3 *is selected and fixed*

5 Local search

In this section we present two different strategies for designing a local search procedure in which we consider that a solution has been improved if its coefficient F is increased.

In this section we present two alternative local search procedures. Both use F, defined in (2), as the objective function and both are first found improve strategies with no diversification element. Hence the procedures are hill climbing and terminate when they hit a local optimum. F is useful to guide the search due to the following properties: If a move leads to an increase in the total utilisation of the bins, then the value of F will increase (see equation (2)); if a move leads to the same total utilisation staying the same, but the utilization of one bin increasing, then the value of F will still increase. This is useful as it differentiates between solution with the same number of bins and removes large plateaux regions from the solution space.

5.1 LS1

Let s be a solution whose set of bins $B = \{b_1^s, \ldots, b_{N_s}^s\}$ is sorted by non-decreasing utilization, i.e, $U_1^s \leq U_2^s \leq \cdots \leq U_{N_s}^s$. Local search procedure (*LS1*) is based on combining pieces from a pair of bins, b_i^s and b_j^s with $i < j$, in such a way that either only one bin is needed to pack all the pieces (and the number of bins is reduced) or U_j^s is increased and consequently U_i^s is decreased, i.e. same number of bins but better F.

The way to combine two bins is described in Algorithm 1, denoted as *Mov1*. For the sake of simplicity we have omitted superscript s. The idea of this movement is to remove one piece $p_j \in b_j$ from the fullest bin and try to place into it all the pieces of b_i into b_j before trying to place again p_j. This move will produce a new bin b'_j. If it has a better utilization than b_j, we try to place the unpacked pieces into the other bin, producing a new bin b'_i. If it contains all the pieces not packed into b'_j, we have obtained an improved solution. Note that b'_i is built from scratch because the set of pieces is quite different from the set of pieces originally packed in b_i. In Algorithm 1 we include the cases where we fail to build b'_i (in this case, no movement is performed and we go on to study the following pieces in b_j) and the case in which all the pieces fit in b'_j (then only one bin is needed to pack all the pieces).

Data: b_i and b_j such that $u_i \leq u_j$
Result: $b_{i'}$ and $b_{j'}$ such that $u_{i'} \geq u_i$ and $u_{j'} \leq u_j$
$b'_i = b_i$;
$b'_j = b_j$;
for *each piece* p_j *in* b_j **do**
  Set $area_{out} = area_{p_j}$, $area_{in} = 0$, $Unplaced = \emptyset$;
  Remove p_j from b'_j;
  List: Sort pieces in b_i by non-increasing area;
  Add p_j to the end of *List*;
  for *each piece p in List* **do**
    Obtain rotations piece p described in Section 4;
    Solve the corresponding MIP models (one per rotation);
    if *piece p fits* **then**
      Update b'_j;
      $area_{in} += area_p$;
    else
      $Unplaced = Unplaced \cup \{p_j\}$;
    end
  end
  if $area_{in} > area_{out}$ **then**
    if $Unplaced \neq \emptyset$ **then**
      Sort *Unplaced* by non increasing area;
      Build b'_i trying to place all the pieces into the bin (see Section 4);
      if *All pieces fit* **then**
        return b'_i and b'_j;
      else
        Set $b'_i = b_i$ and $b'_j = b_j$;
      end
    else
      return b'_j and b'_i as an empty set (all the pieces fits only in one bin);
    end
  else
    Set $b'_i = b_i$ and $b'_j = b_j$;
  end
end

Algorithm 1: Improving the partial solution with two bins (*Mov1*)

Local search procedure *LS1* calls *Mov1* for each pair of bins starting with the bins with the lowest utilization. Each time an improvement is found we start again from the new solution studying the pairs of bins with lowest utilization which have not been considered in the previous move.

5.2 LS2

This local search aims to extend the scope of the search in *LS1*. The idea is to consider one bin, say b_i, of a solution s with N bins, and try to empty it as much as possible by either moving to or swapping pieces with a set of r bins $B' = \{b_{j_1}, \ldots, b_{j_r}\}, r < N, B' \subseteq B \setminus \{b_i\}$, where all B' have greater utilization that b_i.

Data: $b_i, b_{j_1}, \ldots, b_{j_r}$ such that $u_i \leq u_{j_1} \leq \cdots \leq u_{j_r}$
Result: r or $r+1$ new bins packing the same set of pieces with better coefficient F
Copy pieces of b_i in *List* ;
$b_i = \emptyset$;
Copy bins $b'_{j_1} = b_{j_1}, \ldots, b'_{j_r} = b_{j_r}$;
while *movements* **do**
  movements=false;
  for $k = 1$ *to* r **do**
    for *each piece* p_{j_k} *in* b'_{j_k} **do**
      Set $area_{out} = area_{p_{j_k}}$, $area_{in} = 0$;
      Remove p_{j_k} from b'_{j_k};
      Copy *List*2 = *List*;
      Add p_{j_k} to the end of *List*;
      for *each piece p in List*2 **do**
        Obtain rotations for piece p as described in Section 4;
        Solve the corresponding MIP models (one per rotation);
        if *piece p fits* **then**
          Update b'_{j_k};
          $area_{in} += area_p$;
          Remove p from *List*2;
        else
          Study next piece;
        end
      end
      if $area_{in} > area_{out}$ **then**
        if *List*2 $\neq \emptyset$ **then**
          List = *List*2 and sort the list by non-increasing area;
          $b_{j_k} = b'_{j_k}$;
          movements=true;
        else
          return $b'_{j_1}, \ldots, b'_{j_r}$ (a solution with one bin less have been found);
        end
      else
        $b'_{j_k} = b_{j_k}$;
      end
    end
  end
  if *List has been modified* **then**
    Try to place all the pieces from *List* into b_i, building a new bin b'_i;
    if *All pieces fit* **then**
      Improvement is found: return b_i and bins $b'_{j_1}, \ldots, b'_{j_r}$;
    else
      return (no improvement is found);
    end
  else
    return (no improvement is found);
  end
end

Algorithm 2: Improving the partial solution with $r+1$ bins (*Mov2*)

This movement has some similarities with *Mov1*. The main idea of both movements is that we try to increase the usage of one bin b_j by removing one of its pieces and trying to pack the pieces of a less used bin b_i. The main difference is that while in *Mov1* once we have built a new bin b'_j we try to rebuild b_i, in *Mov2* we consider $r \geq 2$ bins in such a way that the weakest bin is not rebuilt until no improvement is found in the r bins. If we fail to pack all the pieces from the list of unpacked pieces in the new bin b'_i, the whole movement fails and we return to the previous solution. Compared with *Mov1*, *Mov2* is a larger neighborhood move, which involves a greater computational effort, but it can find new improving moves.

6 Implementation

In this Section we describe different variants of the constructive procedure obtained by varying:

- The strategy to assign pieces to bins.
- The strategy to obtain the rotations.

- The threshold θ, which defines when a MIP model is difficult to solve by the current solver (*CPLEX 12.5.0.0*).

In Section 3 we present five different approaches for assigning pieces to bins, BPGD, PBP, FFD, HBP and TPS, each one based on solving a variant of the one dimensional bin packing problem. In addition we consider another greedy approach based on sorting the pieces by non-increasing area and then applying the First Fit Decreasing (FFD) algorithm (Johnson et al. (1974)). We denote this assignment procedure as *SAR*.

For instances in which the pieces can rotate freely, their orientation will be obtained by the criteria described in Section 4. We denote by GR1 the criterion in which only the best rotation is considered and by GR3 the case in which the best three rotation are studied.

For instances in the literature in which the orientations of the pieces are limited, usually 0^o and 180^o, or 0^o, 90^o, 180^o, and 270^o, instead using GR we study all the allowed orientations of each piece.

Taking into account the strategy to assign pieces to bins and the threshold θ in the packing procedure, we propose seven variants of the constructive algorithm. The algorithms are denoted $CA-I-J$, where I indicates the assignment strategy and J denotes the value of the threshold θ. Then, $I \in$ {SAR, BPGD, CBP, PBP, FFD, HBP, TPS} and J is given in seconds. In all the assignment strategies we consider $J = 20$, with a time limit of 50 seconds per MIP solved. In addition, we have considered a more ambitious approach, the constructive algorithm $CA-TPS-80$, in which the time limit is increased to 100 seconds for the MIPs considered in the packing and the IP model used in the TPS is limited to 20 seconds.

In Section 5 we have described two local search procedures. Although they can be applied to the solutions obtained by any constructive algorithm, we have tested them only in combination with the constructive algorithm $CA-PBP-20$, because it is the fastest algorithm and the quality obtained is competitive with the other variants. We denote by $CA-PBP-20-LS1$ and $CA-PBP-20-LS1$ the algorithms which include local search.

7 Computational results

The constructive algorithms proposed in the previous sections were coded in C++ and implemented using Visual Studio 2008. We use *CPLEX*, version 12.5.0.0, for solving both the Integer Programming models in the assignment strategy and the Mixed Integer Programming models in the packing strategy, and the computational tests were run on a PC with core i7 2600 processor and 4 GB memory.

The computational experiments are divided into four parts with respect to the source of the data instances. First, we compare the strategies described in Section 6 using two set of jigsaw puzzle instances, a set of 540 instances with irregular convex shapes and 480 instances with concave shapes. These instances have been proposed by Lopez-Camacho et al. (2013) and in line with this paper results are reported by using coefficient F as a measure of performance (see Section 2).

Second, we report the solutions we obtain in three real instances for the ceramic industry provided by a Spanish company located in Castellon, BUTECH-PORCELANOSA. The instances have 209, 340 and 484 pieces. Two of the instances have pieces with concavities and one instance has only convex pieces. There is no restriction on the orientation of the pieces.

The first set of instances are jigsaw puzzles, and therefore is not representative of real cases. While the second set of instances obtained from BUTECH-PORCELANOSA, arise from a real application, the number of instances is very small so it is clearly not enough to assess the performance of the algorithms. Therefore, a third set of instances is obtained from the well-known instances for strip packing Nesting problems available online at ESICUP web site (*http://paginas.fe.up.pt/ esicup/tiki-index.php*), which have a high variety of shapes (convex and non-convex). We use the pieces from these instances to create three sets of 22 instances with different bin sizes.

Finally, the fourth set of instances have been obtained from Bennell et al. (2012) for the 2-Dimensional Bin Packing Problem with Irregular convex Pieces and Guillotine Cuts (2DBPPIPGC). The authors proposed 8 instances, 4 obtained from the real industry and the other 4 randomly generated.

7.1 Jigsaw puzzle convex instances used by Lopez-Camacho et al. (2013)

These 540 instances can be found on the ESICUP web site. They are divided into 18 types, differing in the number of pieces per bin and the squareness of the pieces (see Terashima-Marin et al. (2010)). In all of them the optimal solution consists in filling up all the bins, without any waste. The rotation of the pieces is restricted, considering up to four orientations, 0^o, 90^o ,180^o, and 270^o. However, note that the known optimal solution of 100% utilization arises from the original orientation (0^o).

Table 1: *Results of the constructive algorithms on the jigsaw puzzle convex instances*

	CA-SAR-20	CA-BPGD-20	CA-PBP-20	CA-FFD-20	CA-HBP-20	CA-TPS-20	CA-TPS-80
	F	F	F	F	F	F	F
A	0.605	0.583	**0.609**	0.581	0.578	0.607	0.601
B	0.684	0.865	0.781	0.648	0.654	0.807	**0.902**
C	0.702	0.584	**0.720**	0.603	0.583	0.707	0.688
D	**0.596**	0.564	0.579	0.578	0.578	0.579	0.586
E	**0.581**	0.476	0.524	0.513	0.513	0.524	0.538
F	**0.515**	0.489	0.504	0.492	0.492	0.505	0.509
G	0.719	0.708	0.724	0.631	0.625	**0.736**	0.732
H	0.751	0.758	0.766	0.655	0.658	0.799	**0.905**
I	0.627	0.617	0.628	0.624	0.623	**0.629**	0.625
J	**0.672**	0.654	0.669	0.655	0.655	0.670	0.671
K	**0.763**	0.643	0.757	0.613	0.608	0.747	0.753
L	**0.595**	0.499	0.558	0.517	0.524	0.553	0.544
M	**0.673**	0.543	0.628	0.539	0.543	0.596	0.593
N	**0.519**	0.499	0.513	0.503	0.503	0.513	0.515
O	0.823	0.877	0.823	0.600	0.584	0.832	**0.929**
P	**0.738**	0.624	0.710	0.618	0.611	0.699	0.694
Q	0.903	0.680	**0.953**	0.680	0.679	0.911	0.919
R	**0.766**	0.655	0.754	0.643	0.627	0.742	0.741
Av.	0.680	0.629	0.678	0.594	0.591	0.675	**0.692**
Av.Time	115	102	57	61	104	288	616

Figure 8 shows the solutions of two convex instances of different types. The size of the bins is the same (scaled different in the picture). However, the shape of the pieces and the number of pieces per bin are strongly different. Both solutions are obtained by algorithm CA-PBP-20. The top solution belongs to an instance of class L and the solution depicted at the bottom is an instance of class O (in which the algorithm was able to find the optimal solution).

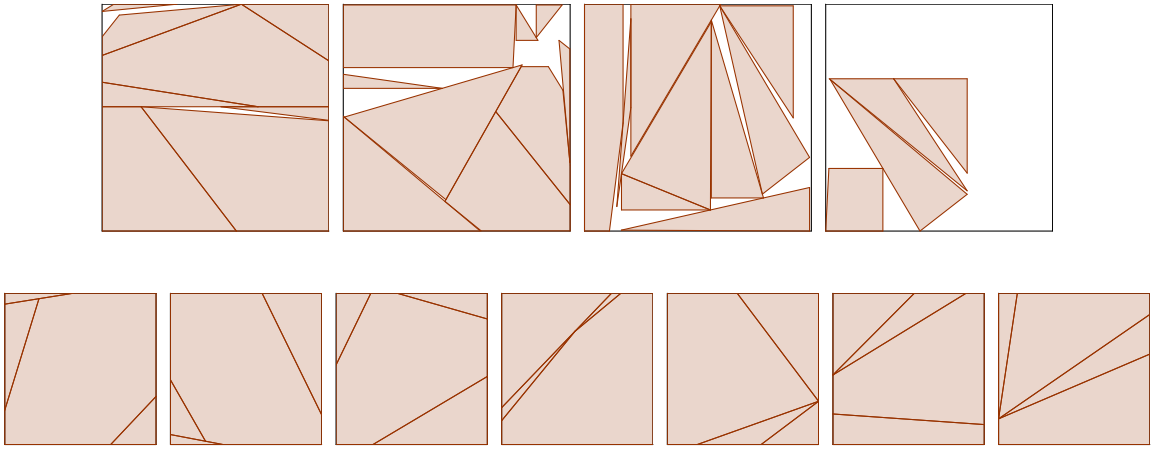

Figure 8: *Two solutions of different type instances.*

In Table 1 we present the results of the seven variants of the constructive algorithm described in the previous Section. For each variant of the constructive algorithm we report the average value of coefficient F obtained for the 30 instances of each type. In this first computational test we allow pieces to be rotated freely and therefore the algorithm would not depend on the initial angles of rotations of the pieces.

Comparing all the constructive algorithms we propose, the first algorithm CA-SAR-20, based on ordering pieces by area, obtains the best average results in 8 of the 18 types (results in bold). However, the best average results are obtained by CA-TPS-80, based on the two-phase assignment procedure, which obtains the best result only in one class, but obtains a better average in 10 of the 18 classes. Finally, we want to emphasize that algorithm CA-PBP-20, based on the partial assignment, improves the BKR average on 11 classes and requires less computational time than any of the other variants. The computational time used by the algorithms in the different classes varies considerably. This behavior is explained by the number of pieces placed in each bin. When packing few pieces in a bin, the MIP models used in the packing procedure are relatively simple. However, when this number increases, the models are more complicated to be solved and more computational time is required.

Table 2 shows the computations results obtained by the previous best three variants of the constructive al-

Table 2: *Results of the best constructive algorithms allowing four orientations.*

	CA-SAR-20	CA-PBP-20	CA-TPS-80	BKR
	F	F	F	F
A	0.615	0.604	0.599	0.605
B	0.761	0.781	0.914	0.929
C	0.731	0.711	0.691	0.763
D	0.591	0.589	0.584	0.579
E	0.515	0.508	0.520	0.412
F	0.513	0.510	0.509	0.496
G	0.716	0.724	0.737	0.814
H	0.751	0.766	0.902	0.928
I	0.630	0.624	0.626	0.627
J	0.669	0.671	0.671	0.665
K	0.764	0.750	0.746	0.718
L	0.589	0.577	0.568	0.512
M	0.658	0.614	0.598	0.589
N	0.518	0.518	0.514	0.503
O	0.826	0.831	0.914	0.823
P	0.713	0.700	0.693	0.678
Q	0.941	0.916	0.918	1.000
R	0.757	0.745	0.741	0.771
Av.	0.681	0.674	0.691	0.690
Av.Time	60	52	622	

Table 3: *Results of the local search procedures limiting the rotation to four angles and fixing the rotation.*

	CA-PBP-20	CA-PBP-20-LS1	CA-PBP-20-LS2	CA-PBP-20-LS2(No Rot)	BKR
	F	F	F	F	F
A	0.604	0.611	0.612	0.642	0.605
B	0.781	0.858	0.878	0.973	0.929
C	0.711	0.736	0.722	0.775	0.763
D	0.589	0.591	0.590	0.578	0.579
E	0.508	0.525	0.529	0.449	0.412
F	0.510	0.520	0.564	0.512	0.496
G	0.724	0.797	0.809	0.775	0.814
H	0.766	0.868	0.810	0.921	0.928
I	0.624	0.637	0.652	0.632	0.627
J	0.671	0.675	0.701	0.667	0.665
K	0.750	0.763	0.707	0.758	0.718
L	0.577	0.590	0.619	0.617	0.512
M	0.614	0.639	0.592	0.660	0.589
N	0.518	0.520	0.668	0.504	0.503
O	0.831	0.848	0.796	0.945	0.823
P	0.700	0.712	0.852	0.757	0.678
Q	0.916	0.965	0.858	0.899	1.000
R	0.745	0.757	0.762	0.776	0.771
Av.	0.674	0.701	0.707	0.713	0.690
Av.Time	52	129	161	50	
Impv.		3.75%	4.59%	5.46%	

gorithm, CA-SAR-20, CA-PBP-20, and CA-TPS-80, when the orientations are limited to four angles, 0^o, 90^o ,180^o and 270^o. The last column, BKR, shows the best result obtained by the best of the 11 algorithms presented in Lopez-Camacho et al. (2013), where the computational time ranges between 0.3 and 8.4 seconds. We can see that the quality of the results is similar to the results presented in Table 1. In this type of problems, in which the optimal solution requires that the pieces have the original orientation in order to recover the initial decomposition, the possibility of rotating the pieces is only useful when the assignment of the pieces to bins has not been correct but has just a limited, corrective effect.

Table 3 shows the benefit obtained by applying the two local search procedures, algorithms CA-PBP-20-LS1 and CA-PBP-20-LS2, defined in Section 6. These algorithms are created by applying the two local search procedures explained in Section 5 to CA-PBP-20. We can see that the improvement average is 3.76% and 4.59%, respectively, over the constructive procedure. Furthermore, we can observe that the second local search (LS2) produce better results than LS1. Column CA-PBP-20-LS2(No Rot) shows the result obtained by CA-PBP-20-LS2 in the case in which only one angle of rotation is allowed (the initial angle). In this case we obtain the optimal solution on 83 instances over the 540 against 45 solutions obtained by the same algorithm when 4 angles of rotations are allowed. Despite the fact that the problem is simplified because we fix the orientation to be the one to reproduce jigsaw puzzle and therefore we eliminate a possible source of errors, the resulting problem is by no means simple, as can be seen in the examples of Figure 9. In addition, since we try to insert the piece in each angle of rotation, when there is only one angle the computational time is reduced because we try less number of attempt when inserting the pieces.

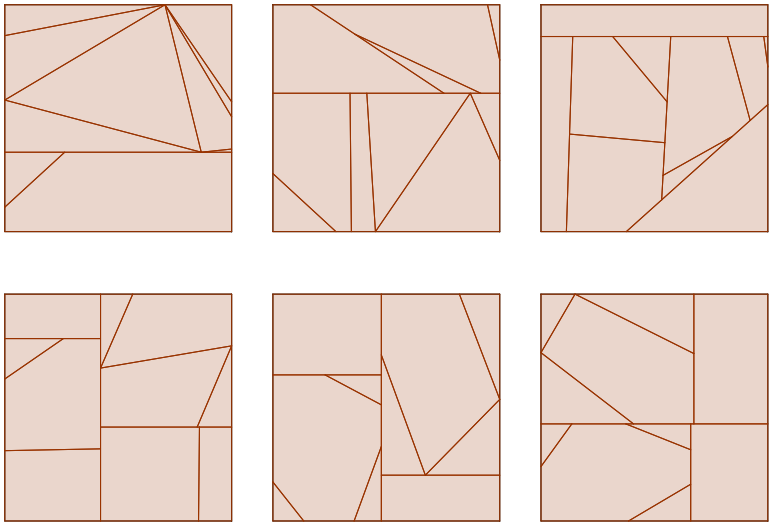

Figure 9: *Two optimal solutions obtained by CA-PBP-20-LS2(Fix Rot) (instances TA001 and TL023).*

Table 4: *Results of the best constructive algorithms applied on non-convex jigsaw puzzle instances*

	CA-SAR-20	CA-PBP-20	CA-TPS-80	CA-PBP-20-LS1	CA-PBP-20-LS2
	F	F	F	F	F
A	0.584	0.597	0.564	0.601	0.600
B	0.699	0.714	0.672	0.737	0.739
C	0.643	0.666	0.635	0.682	0.691
F	0.500	0.498	0.499	0.499	0.502
H	0.717	0.731	0.719	0.753	0.763
L	0.551	0.546	0.532	0.564	0.570
M	0.578	0.594	0.547	0.600	0.617
O	0.663	0.721	0.659	0.744	0.739
S	0.447	0.467	0.460	0.482	0.484
T	0.778	0.790	0.771	0.808	0.821
U	0.570	0.657	0.583	0.686	0.689
V	0.780	0.873	0.776	0.892	0.910
W	0.627	0.638	0.603	0.661	0.670
X	0.529	0.581	0.510	0.590	0.592
Y	0.671	0.694	0.648	0.721	0.734
Z	0.789	0.820	0.787	0.831	0.846
Av.	0.633	0.662	0.623	0.678	0.686
Av.Time	35	25	314	81	123
Imp.				2.41%	3.50%

7.2 Jigsaw puzzle non-convex instances

The 480 jigsaw puzzle non-convex instances can also be found in the ESICUP web site. The instances are divided into 16 types (more details can be found in Terashima-Marin et al. (2010)). The authors only provide the results for the set of convex instances, and to our knowledge, there is no publication reporting results for these instances.

The first three columns on Table 4 show the results obtained by the best three constructive algorithms when pieces can be rotated continuously. We can observe that constructive algorithm CA-PBP-20 gives the best results and CA-TPS-80 obtains worse results with much longer running times. The last two columns shows the improvement obtained when the two local search procedures are applied on CA-PBP-20. We can observe that the second local search produce more improvement than the second local search. In addition, the average results obtained in this set of instances are slightly worse than the results obtained in the convex instances.

7.3 BUTECH-PORCELANOSA instances

BUTECH-PORCELANOSA provided three instances with a wide variety of pieces in terms of shape and size. Pieces can be rotated continuously, while due to size, there are some pieces that can fit only with a specific angle of rotation. In this case it is necessary to change the initial orientation of the piece in order to fit them into the bins. Some of the the bins are built by placing just one piece while in some others several pieces can be packed together. The instances are denoted as BPx, where x indicates the number of pieces. The results are shown in

Table 5: *Computational results for the real data instances*

	CA-SAR-20			CA-BPGD-20			CA-PBP-20			CA-FFD-20			CA-HBP-20		
	N	U	T	N	U	T	N	U	T	N	U	T	N	U	T
BP209	**48**	0.9157	320	52	0.8454	1280	**48**	0.9157	293	51	0.8577	589	51	0.8577	1786
BP340	204	0.9354	663	214	0.8918	4049	**202**	0.9455	234	210	0.9089	568	210	0.9089	4513
BP484	**316**	0.9732	22	388	0.7900	1827	**316**	0.9749	83	325	0.9471	527	321	0.9590	5658

Table 5.

We tried to solve these instances with all the constructive algorithms described in Section 6. For these instances, the upper bound on number of bins and the number of pieces are very large. As a result, solving the corresponding IP models many times, as required by CA-TPS-20 and CA-TPS-80, is computationally prohibitive. These variants of the algorithm are therefore not included. However, algorithms CA-BPGD-20 and CA-HBP-20, which solve a similar IP formulation but a much lower number of times, produce a solution in reasonable computing time and are included. Nevertheless, the best algorithm in terms of computational time and quality of the solutions is CA-PBP-20, that always provides an utilization greater than 90% and up to 97.49%. Note that this variant only decides which pieces are selected into the current bin and does not take into account the impact on later bins.

7.4 Nesting instances

In Nesting problems the length of the strip into which the pieces are packed is not limited and the objective is to minimize the total required length needed to pack all the pieces. In order to adapt the nesting instances to our bin packing problem, we have built three sets of instances defining bin sizes as follows. Let m_d be the maximum value of either the length or the width of all the pieces in the initial rotation of a given instance. Then, by considering the original set of pieces of each nesting instance and modifying the bin size (to a fixed dimensional square) we build the following sets:

- *Nest-SB* (small bins). The bin dimensions are $W = L = 1.1m_d$
- *Nest-MB* (medium bins). The bin dimensions are $W = L = 1.5m_d$
- *Nest-LB* (large bins). The bin dimensions are $W = L = 2m_d$

Each one of these set of instances has 22 instances, corresponding to the different instances we have found for the Nesting problem. Note that there are instances such as shapes0 and shapes1 in which the set of pieces is the same and the rotation is different. As we allow free rotation of the pieces, we have built just one instance per bin size. The idea of considering three different size is that, intuitively, in the set of instances *Nest-SB* the assignment problem should be more important than in the instances of *Nest-MB* and *Nest-LB* because fewer pieces can fit into one bin and the packing problem will be easier. Conversely, in the instances of *Nest-LB*, the packing phase will be more important. Solving the packing problem more efficiently may lead to better solutions even if the assignment is not very efficient.

According to the results of previous tests, Tables 1,2, and 4, we use only the constructive algorithm CA-PBP-20 and the algorithms applying the local search procedures.

Table 6 shows the results obtained in each instance separately, and for each class of instances the total sum of the fractional number of bins (denoted by K), the average using coefficient F, the average computational time in seconds and the improvement obtained by applying the local search to the constructive algorithm.

The constructive procedure CA-PBP-20, the first column in each class, obtains similar results for the three classes in terms of average utilization F (0.419, 0.415, 0.409). However, the computational effort increases when the bin size increases (44, 93 and 195 seconds), because larger bins require solving packing problems with more pieces.

When we apply the local search procedures we observe that the constructive solutions are clearly improved. The improvement percentages are larger in small bins and decrease for larger bins. As it was expected, in the case of small bins, poor piece assignments cannot be corrected in the packing phase, because very few pieces fit into one bin. In these cases, the improvement moves, that move pieces between bins, can produce much better solutions. This effect decreases for larger bins. The packing phase, though more time-consuming, is able to accommodate adequately the pieces assigned to each bin and the effect of moving pieces between bins decreases. It can also be observed that the computational times increased very sharply when the local searches are applied. This increase is

Table 6: *Results obtained by CA-PBP-20 applying LS1 and LS2 on Nesting instances*

	Nest-SB						Nest-MB						Nest-LB					
	CA-PBP-20		LS1		LS2		CA-PBP-20		LS1		LS2		CA-PBP-20		LS1		LS2	
	K	T	K	T	K	T	K	T	K	T	K	T	K	T	K	T	K	T
albano	4.69	12	4.69	141	4.67	197	2.56	35	2.56	117	2.56	35	1.43	62	1.43	100	1.43	102
shapes2	19.73	5	19.73	17	19.73	19	9.93	7	9.93	280	9.93	55	4.8	19	4.8	102	4.4	146
trousers	4.75	171	4.79	361	4.77	553	2.56	316	2.56	704	2.56	869	1.5	541	1.5	984	1.5	989
shirts	15.77	132	15.63	747	15.63	577	7.1	216	7.1	1174	7.1	1769	4.11	480	4.11	1234	4.07	1223
shapes0-1	11.75	25	11.39	146	11.39	788	5.57	79	5.43	729	5.29	4967	2.93	183	2.93	922	2.93	1320
dighe2	2.43	1	2.39	4	2.43	4	1.29	2	1.27	18	1.31	183	0.76	15	0.75	25	0.75	26
dighe1	2.64	6	2.62	14	2.52	23	1.46	18	1.42	149	1.38	242	0.81	47	0.81	74	0.81	72
fu	7.58	1	6.7	2	7.57	2	3.57	1	3.57	3	3.57	4	1.71	5	1.71	8	1.71	10
han	4.2	8	4.2	26	4.2	33	2.35	20	2.2	227	2.2	178	1.26	29	1.26	45	1.26	49
jakobs1	7.6	4	7.6	15	7.6	14	3.33	20	3.3	236	3.24	315	1.78	40	1.78	65	1.78	83
jakobs2	6.45	8	6.42	86	6.42	1045	3.5	23	3.2	364	3.2	154	1.74	42	1.74	65	1.74	83
mao	3.95	10	3.93	464	3.84	236	2.17	18	2.17	85	2.17	86	1.13	43	1.13	58	1.13	57
poly1a	3.28	4	2.87	11	2.9	12	1.48	15	1.48	178	1.53	1161	0.8	45	0.8	75	0.8	83
poly2a	6.38	20	5.67	207	5.69	693	3.27	41	3.22	1274	3.21	4294	1.74	98	1.74	150	1.74	146
poly3a	9.38	42	8.95	196	8.36	426	4.8	106	4.56	3086	4.4	4094	2.55	200	2.48	351	2.52	385
poly4a	12.78	72	11.56	961	11.53	1182	6.25	155	5.97	4634	5.93	7053	3.41	318	3.41	583	3.35	525
poly5a	15.38	97	14.59	1745	14.74	911	7.77	270	7.74	2085	7.34	12804	4.37	450	4.3	2119	4.3	3004
poly2b	6.61	13	6.41	39	6.54	106	3.39	33	3.39	722	3.27	384	1.91	112	1.87	150	1.87	198
poly3b	8.6	31	8.49	109	8.47	278	4.52	79	4.47	1667	4.4	1941	2.53	165	2.5	321	2.5	365
poly4b	10.91	66	10.61	211	10.62	579	5.8	148	5.7	1202	5.61	2757	3.18	273	3.18	411	3.18	447
poly5b	12.92	113	12.39	1307	12.51	935	6.8	231	6.53	1853	6.63	985	3.88	511	3.76	842	3.88	1025
swimm	9.3	49	9.3	1220	8.62	2959	4.8	219	4.71	1445	4.71	3760	2.64	603	2.64	1412	2.62	1746
Total (K)	187.08		180.93		180.75		94.27		92.48		91.54		50.97		50.63		50.27	
Av. F	0.419		0.443		0.445		0.415		0.430		0.436		0.408		0.409		0.410	
Av. Time		40		365		526		93		1011		2186		195		459		549
Impr.			5.44%		5.82%				3.39%		4.90%				0.22%		0.55%	

specially large for medium-size bins. The initial solution can be improved by significant percentages, 3.37% and 3.94%, but that involves many moves of pieces and many medium-size MIP models to be solved in the packing phase. The times are not so large for small bins, because the packing problems are easier to solve, and for large bins, because fewer improving moves are identified. Besides these general considerations, based on averages, we can observe that the behavior of each algorithm varies for each instance. There are instances for which $LS1$ outperforms $LS2$ while in others $LS2$ obtains much better results. The computational times also vary widely among instances and algorithms.

7.5 Two Dimensional Bin Packing Problem with Irregular Pieces and Guillotine Cuts (2DBPPIPGC)

Bennell et al. (2012) propose two different constructive algorithms to solve the 2DBPPIPGC problem. The authors test their algorithms on 8 instances. 4 of these instances, called J40, J50, J60 and J70, are real data obtained by a company in the glass cutting industry. The remaining 4 instances, called H80, H100, H120 and H149, were generated randomly taking into account the main properties of the real data. The objective in this problem is to reduce the fractional number of bins (K).

Note that the algorithms proposed in Bennell et al. (2012) have two important differences with the algorithms we present in this paper:

- The algorithms proposed in Bennell et al. (2012) deal with guillotine cuts, i.e, all the pieces in each bin must be obtained by applying only guillotine cuts. As a result all pieces are convex.
- The mirror transformation is allowed, i.e, pieces can be reflected potentially leading to better solutions.

The guillotine cuts are limiting the solution space so the solutions obtained by algorithms designed for those problems are expected to produce lower quality solutions. However, the mirror transformation allows more, possibly better, quality solutions. As a result it is not clear how a fair comparison of these algorithms with our approach can be made.

Most recently, Martinez-Sykora et al. (2015) develop a constructive procedure that outperform the results obtained by Bennell et al. (2012) in all of these 8 instances. The authors study different variants of the constructive algorithm and show that there is an important benefit when the reflection (mirror transformation) of the pieces is allowed. Here, we are going to compare our algorithms with the version of the constructive procedure described in Martinez-Sykora et al. (2015) where reflection is not allowed; algorithm CA6. Given we cannot make a fair comparison, the purpose of these experiments is to understand to what extent the guillotine cuts constraints the solution quality on these instances.

Table 7: *Results obtained by CA-PBP-20 applying LS1 and LS2 on Nesting instances*

	CA-PBP-20		LS1		LS2		CA6	
	K	T	K	T	K	T	K	T
J40	7.48	61	7.30	900	7.40	681	7.70	41
J50	9.28	72	9.22	631	9.16	497	9.53	24
J60	10.38	131	10.21	930	10.17	1664	10.68	65
J70	11.67	158	11.54	1135	11.42	1278	12.18	44
H80	9.31	254	9.25	2195	9.27	1094	9.34	117
H100	15.39	171	15.60	2879	15.28	1728	16.16	92
H120	15.62	439	15.55	2427	15.37	3705	16.39	158
H149	21.68	671	21.51	8555	21.46	2238	22.09	193
Total	100.81		100.18		99.52		104.07	
Av. Time		245		2457		1611		92

Table 7 shows the fractional number of bins (K) obtained by constructive procedure CA-PBP-20 and both local search apporaches LS1 and LS2. The last algorithm, CA6, is the variant that does not allow reflection and includes guillotine cuts, presented in Martinez-Sykora et al. (2015). The algorithms developed in this study outperform CA6 in all the test instances. The improvements obtained (3.13%, 3.74%, and 4.37%) can be considered as an approximate measure of the effect of removing the condition of using guillotine cuts on these instances. However, these improvements involve larger computing times, especially in the case of using local search.

8 Conclusions

In this paper we have addressed a new problem for which there is no specialized algorithm. Despite the fact that the problem may arise in industries as the ship building or furniture, the main application in which we have focused our efforts is in the ceramic tile sector, from which we have tested the algorithm in several instances provided by BUTECH (PORCELANOSA group) in Spain. Due to the high cost of the material used, the main objective is to obtain high utilization solutions.

The problem is the two dimensional bin packing problem with irregular and concave pieces in which there is no limitation on the rotation of the pieces. As we stated before, this problem has clearly two components (subproblems). First, the assignment problem, in which pieces are grouped to be placed into the same bin, and secondly, the packing problem in which we need to find a feasible rotation and placement for the pieces assigned to each bin. We have designed a two-phase constructive procedure and used several integer programming models to solve the problems involved at each phase. Also, two local search procedures have been proposed, producing a clear improvement over the constructive procedures.

In addition, in order to significantly extend the computational study, we have designed the algorithm in such a way that it can be applied to the cases in which the rotation is either fixed or limited to certain angles. Therefore, we have been able to test the algorithms on other sets of existing instances, obtaining solutions which are clearly better than those reported in previous studies.

Concluding, the algorithm presented in this paper is able to solve the two dimensional bin packing problem with convex and concave pieces, fixed rotation, limited rotation, or free rotation for the pieces and any combination thereof. In the section of computational results we have used four different families of instances in which the properties of the pieces and the bins vary considerably, including real world instances. In all the families the algorithms proposed produce high quality results outperforming all competing approaches in the literature.

Acknowledgments

This study has been partially supported by the Spanish Ministry of Economy and Competitiveness, DPI2011-24977, and by the Generalitat Valenciana, PROMETEO/2013/049. The authors would like to acknowledge the contribution of the professionals at PORCELANOSA Group, in particular to Javier M. Chiva Bartoll, which have contributed to the practical application in this work in a significant way.

References

R. Alvarez-Valdes, A. Martinez-Sykora, and J.M. Tamarit. A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, 145:466–477, 2013.

J.A. Bennell and J.F. Oliveira. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184:397–415, 2008.

J.A. Bennell and J.F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:S93–S105, 2009.

J.A. Bennell and X. Song. A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 16(2):167–188, 2010.

J.A. Bennell, W. Han, X. Zhao, and X. Song. Construction heuristics for two dimensional irregular shape bin packing with guillotine constraints. *European Journal of Operations Research*, 230:495–504, 2012.

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

M. Hifi and R. M'Hallah. A best-local postiion procedure-based heuristic for two-dimensional layout problems. *Studia Informatica Universalis, International Journal on Informatics*, 2(1):33–56, 2002.

S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operations Research*, 88: 165–181, 1996.

D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garvey, and R.L. Graham. Worst-case performace bounds for simple one-dimensional packing algorithms. *SIAM Journal on Comuting*, 3:299–325, 1974.

D. Liu and H.Teng. An improced bl-algorithm for genetic algorithm of the othogonal packing of rectangle. *European Journal of Operations Research*, 112:413–419, 1999.

E. Lopez-Camacho, G. Ochoa, H. Terashima-Marin, and E.K. Burke. An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206:241–264, 2013.

A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, and J.M. Tamarit. Cosntructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts. *Omega*, 52:15–32, 2015.

P. Ross, S. Schulenburg, and J.G. MarÃn-BlÃ¡zquez. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. *Lecture notes in computer science. Conference on genetic and evolutionary computation*, pages 942–948, 2002.

X. Song and J.A. Bennell. Column generation and sequential heuristic procedure for solving an irregular shape cutting stock problem. *Journal of the Operational Research Society*, pages 1–16, 2013.

H. Terashima-Marin, P. Ross, C.J. Farias-Zarate, E. Lopez-Camacho, and M. Valenzuela-Rendon. Generalized hyper-heuristics for solving 2d regular and irregular packing problems. *Annals of Operations Research*, 179: 369–392, 2010.

G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research, 183, 109-1130*, 2007.