



Deposited via The University of Leeds.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/134688/>

Version: Accepted Version

Article:

Abeysooriya, RP, Bennell, JA and Martinez-Sykora, A (2018) Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation. *International Journal of Production Economics*, 195. pp. 12-26. ISSN: 0925-5273

<https://doi.org/10.1016/j.ijpe.2017.09.014>

© 2017 Elsevier B.V. This is an author produced version of a paper published in *International Journal of Production Economics*. Uploaded in accordance with the publisher's self-archiving policy. This manuscript version is made available under the Creative Commons CC-BY-NC-ND 4.0 license <http://creativecommons.org/licenses/by-nc-nd/4.0/>

Reuse

This article is distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) licence. This licence only allows you to download this work and share it with others as long as you credit the authors, but you can't change the article in any way or use it commercially. More information and the full terms of the licence here: <https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation

Ranga P. Abeysooriya, Julia A. Bennell, Antonio Martinez-Sykora

Southampton Business School, University of Southampton, UK

Abstract

The paper investigates the two-dimensional irregular packing problem with multiple homogeneous bins (2DIBPP). The literature on irregular shaped packing problems is dominated by the single stock sheet strip packing problem. However, in reality manufacturers are cutting orders over a multi-stock sheets. Despite its greater relevance, there are only a few papers that tackle this problem in the literature. A multi-stock sheet problem has two decision components; the allocation of pieces to stock sheets and the layout design for each stock sheet. In this paper, we propose a heuristic method that addresses both the allocation and placement problems together based on the Jostle algorithm. Jostle was first applied to strip packing. In order to apply Jostle to the bin packing problem we modify the placement heuristic. In addition we improve the search capability by introducing a diversification mechanism into the approach. Furthermore, the paper presents alternative strategies for handling rotation of pieces, which includes a restricted set of angles and unrestricted rotation. Very few authors permit unrestricted rotation of pieces, despite this being a feature of many problems where the material is homogeneous. Finally, we investigate alternative placement criteria and show that

the most commonly applied bottom left criteria does not perform as well as other options. The paper evaluates performance of each algorithm using different sets of instances considering convex and non-convex shapes. Findings of this study reveal that the proposed algorithms can be applied to different variants of the problem and generate significantly better results.

Keywords:

Cutting and packing, heuristics, Irregular shapes, bin packing

1. Introduction

The paper addresses the Two-Dimensional Bin Packing Problem where the pieces may be any simple polygon; we denote it as the 2DIBPP. The problem arises in many industries where there is a requirement for irregular shape pieces to be cut from multiple fixed dimension stock sheets such as metal, wood, paper and plastic. Despite the many real-life applications, the problem has received relatively little attention in the literature when compared to the rectangle bin packing problem or the irregular shape strip packing problem. Moreover, most data instances for irregular packing problems restrict the orientation of the pieces to one, two or four angles of orientation. However, many types of material are homogeneous and pieces can be cut in any orientation. Specifically, the paper describes a heuristic solution method for the 2DIBPP where the stock sheets are homogeneous and a feasible solution is found when all the pieces are packed within the bins with no overlap between pieces. The objective is to minimize the total number of bins needed to cut all of the pieces.

Our approach draws on the literature for irregular strip packing and ap-

plies these concepts to bin packing. Further the paper describes new features that significantly enhance the performance of the algorithm and make it more generally applicable. The components of our approach include: a single-pass constructive algorithm (CA) that constructs the bins, placement decisions that allow either a finite set of rotation angles or free rotation of the pieces, and an improvement heuristic. The CA is based on TOPOS, which was originally proposed by [Oliveira et al. \(2000\)](#) and later improved by [Bennell and Song \(2010\)](#). Since the problem definition allows pieces to take any angle of orientation, we develop placement rules that permit pieces to rotate either by a pre-defined finite set of orientations or any angle in the range of 0 to 360 degrees. Finally, the improvement heuristic uses the principles of the Jostle approach proposed by [Dowland et al. \(1998\)](#) for strip packing. The combination of Jostle and the CA means that the approach solves both the placement and allocation problems together rather than considering them as two independent problems. Further improvements are made to the Jostle heuristic to enhance its ability to search the solution space including a kick to diversify away from local optima and a merge and split mechanism to retain profitable combinations of pieces.

The contribution of this paper is as follows. We are one of the few papers to consider multiple bins for packing irregular shapes, and the first to include the free rotation of the pieces. Hence, the contribution is to propose and solve a more general and more realistic problem for many industries. Second, we introduce a new method for identifying promising rotations for the pieces in the case when any angle is permitted. This is important because free rotation implies an infinite neighbourhood, which can be sampled from

but not exhaustively search. The final contribution is the development of an efficient heuristic adapted from the Jostle approach that is competitive across all nesting benchmark data sets. In addition, we introduce a diversification mechanism from local search to improve the performance of the Jostle approach.

The paper is organised as follows. Section 2 provides an overview of the published work on irregular packing with multiple stock sheets. In Section 3, we define the problem while introducing some notations used in the paper. Section 4 introduces our packing procedure. This includes the proposed CA and describes how we handle finite rotation, free rotation and hole-filling. Section 5 describes the implementation of the Jostle heuristic for the bin packing problem and considers its performance as a local search procedure. In the same section, we propose an iterated Jostle approach and describe three variants. We present the experimental design and the computational results in Section 6. This includes results of experiments we use to improve the performance of the algorithms and the results we achieved for existing benchmark problems. Section 7, discusses our conclusions from the research.

2. Literature Review

In this section we provide a brief overview of the literature. Note that there is very little literature on this specific problem, while there is extensive literature on the rectangle bin packing problem and the irregular strip packing problem. Here we select the papers most relevant to our study.

Irregular shape packing problems hold additional challenges associated with dealing with complex geometry that can make solution generation com-

putationally time consuming. Only recently the research field has sufficiently matured, along with greater computer processing power, to make considering multiple stock sheets a fruitful proposition over solving each stock sheet separately. Hence, literature explicitly covering the 2DIBPP is limited. [Lopez-Camacho et al. \(2014\)](#), [Lopez-Camacho et al. \(2013a\)](#), [Lopez-Camacho et al. \(2013b\)](#) and [Terashima-Marin et al. \(2008\)](#) all describe a hyper-heuristic approach based on a genetic algorithm for the 2DBPP for regular and irregular shapes. They describe several selection heuristics to solve the allocation of pieces to bins including variants of first fit, best fit, filler and Djang & Finch (DJD). For the latter, the position of the pieces in the allocated bin is determined by the Bottom-Left (BL) algorithm developed by [Jakobs \(1996\)](#) or an improved bottom left (BLLT) version proposed by [Liu and Teng \(1999\)](#) or a variant of the constructive approach presented by [Hifi and M'Hallah \(2003\)](#). [Song and Bennell \(2014\)](#) solve the irregular shape cutting stock problem. Cutting stock problems are slightly different to bin packing problems. While both pack multiple stock sheets, the cutting stock problem has many copies of each piece, hence the piece set is weakly heterogeneous. As a result, multiple copies of a particular layout can be cut to meet demand. They propose a column generation procedure, where the master problem determines how many copies of each layout, and the sub-problem is solved heuristically to generate a bin layout. They investigated three solution approaches; column generation, adapted column generation that permits fewer patterns, and a sequential heuristic procedure to compare the quality of packing solutions.

These papers, as with the majority of the existing literature on irregular packing, have achieved efficient solutions with restricted rotation of the ir-

regular pieces e.g. [Bennell and Song \(2010\)](#), [Burke et al. \(2007\)](#), while only a few papers ([Martinez-Sykora et al., 2015](#); [Rocha, 2014](#); [Han et al., 2013](#); [Stoyan et al., 2012](#)) have considered unrestricted rotation of irregular pieces. To our knowledge, [Han et al. \(2013\)](#) and [Martinez-Sykora et al. \(2015\)](#) are the only papers that considered unrestricted rotation with multiple identical stock sheets. Both these papers consider an application of cutting glass where pieces are convex and require guillotine cuts. The guillotine cuts may be at any angle relative to the edge of the stock sheet. This constraint has a dominating influence over the algorithm design and strongly influence the rotation angle of the pieces.

The main body of literature on packing irregular shapes addresses the strip packing problem, known as nesting problems. [Bennell and Oliveira \(2009\)](#) provide a tutorial in nesting problems and describe the different types of solution strategies adopted by researchers. A review of these approaches is beyond the scope of the paper. Here we highlight the work of [Oliveira et al. \(2000\)](#) and [Dowsland et al. \(1998\)](#) as these form part of our approach. [Oliveira et al. \(2000\)](#) TOPOS approach constructs a layout from a given permutation of pieces. Each piece added to the partial solution is placed according to a range of criteria. A key innovation in this paper is that the partial solution does not have a fixed origin on the stock sheet, hence pieces could be added at any point on the perimeter of the partial layout provided it does not exceed the width of the stock sheet. They were an early adopter of the nofit polygon and used this to identify a finite set of placement positions that optimised their criteria. [Dowsland et al. \(1998\)](#) also use the nofit polygon to identify feasible positions in their Jostle algorithm. Essentially

the algorithm involves repeated applications of a CA. A key innovation is using the co-ordinates of the current layout to determine the next packing sequence of pieces starting with the last piece. This simulates the jostling of pieces from left to right and back again.

3. Problem Description

In this section we give a more formal description of the problem.

Let $P = \{p_1, \dots, p_n\}$ be a set of n polygons to be packed into identical rectangular bins. We denote the bins as rectangular objects with fixed width W and fixed length L . A feasible solution of the 2DIBPP is described by a set of bins $B = \{b_1, \dots, b_N\}$ where N is the number of bins required to pack the given n irregular polygons. We denote $P_j \subseteq P$ as the set of packed pieces in each bin b_j and n_j denotes the number of placed pieces in bin b_j . Each packed piece p_i , has an orientation angle $o_i \in [0, 360^\circ)$ and a position in its allocated bin (x_i, y_i) , which is the position of the piece origin. We consider the origin of each piece as the bottom-left corner point of the enclosing rectangle for a given angle of rotation and the bottom left corner of the first bin is the bin origin.

The objective is to minimize the number of bins utilized, N . However, this measure is not very sensitive and many competing solutions will have the same value of N for a given problem instance. As a result, we use the following two measurements to evaluate the bin packing solutions.

1. Fractional number of bins $K = N - 1 + R^*$ where R^* is the proportion of the least utilised bin used to pack pieces after applying either a vertical

or horizontal cut. This assumes that offcuts can be kept and used later.

See [Han et al. \(2013\)](#).

2. $F = \frac{\sum_{j=1}^N U_j^2}{N}$ where U_j is the utilization of each bin b_j , proposed by [Lopez-Camacho et al. \(2013a\)](#). The utilization of each bin U_j is defined as the total area of pieces inside the bin b_j divided by the area of the bin (LW).

Both measures reward solutions that highly utilise bins while trying to empty one weakest bin, which supports the search in reducing the number of bins.

4. Packing procedure

In this section we describe all the components of our approach that support the construction of a single layout. Our packing procedure is a CA within which we need to identify the candidate placement positions and the orientation of the piece. We also discuss the selection criteria that will chose between placement candidates.

A solution method of 2DIBPP addresses two main decision problems: assigning pieces to bins and placing those pieces in the bin. The majority of the cutting and packing literature that considers 2D irregular pieces focuses on strip packing where there is a single stock sheet. In this case the main decision is where to place the piece on the strip with the aim of minimising length. Our approach follows the strip packing approach while imposing the division between bins therefore making the bin allocation implicit within the placement decisions. The proposed method always leads to a feasible placement of the next piece by addressing the allocation and placement decisions

together.

Our approach seeks to simulate the jostling of pieces. [Dowland et al. \(1998\)](#) used this principle to reduce the packing length of the layout for the irregular shape strip packing problem by moving the pieces from one end of the strip to the other as in a shaking motion. They called the algorithm Jostle and suggest it was inspired by the natural human behaviour to shake a container to encourage objects to pack together as tightly as possible. The basic idea is to begin with an initial random ordering of the pieces and apply a constructive heuristic following a placement policy of left most position packing from the outside edges of the strip towards the centre. Once all the pieces are packed, Jostle proceeds to take the pieces in order of largest to smallest x-co-ordinate and pack at the opposite end the stock sheet using a right most policy and filling holes where possible. The next iteration takes the packing order from smallest to largest x-coordinate and again packs with a left most policy, and so on simulating the jostling of pieces from one end of the stock sheet to the other. Note that packing from left to right and then right to left is called a Jostle cycle. At the time of publishing, the approach generated the best results on the benchmark data sets. Since this work, researchers have developed more effective methods for handling the geometry and utilise more powerful computers. In this paper, we apply the principles of Jostle to the homogeneous multiple bin packing problem.

In order to adapt Jostle to the bin packing problem, we first define a very long packing strip of width W and length $n \times L$. This strip represents n bins joined together. We then define bin spaces by assigning a set of barrier lines at fixed intervals of length L along the strip. The bottom left corner of bin

$i \in \{1, \dots, n\}$ is placed at $(L \times (i - 1), 0)$. We can directly apply Jostle to the strip with the additional constraint that pieces may not be placed across a barrier. Along with the usual containment and overlap constraints, this ensures pieces are always placed entirely within the bins.

[Dowland et al. \(1998\)](#) imposed a grid of feasible placement positions over the strip, which formed a search grid for finding the position of the next piece to be packed. Restricting the placement positions by a grid over constrains the solution space and good solutions can be lost. More recent work in irregular packing has developed effective placement policies based on the nofit polygon (e.g. [Bennell and Song \(2010\)](#)). As a result we design several new placement policies and incorporate them within Jostle.

4.1. Construction algorithm

In this section we introduce the single-pass CA, which is made up of the packing order of pieces and the placement policy. The packing order is a randomly generated sequence for the first layout construction and subsequently is decided by Jostle. The placement policy sets the criteria to compare the candidate placement positions and orientations of the next piece to be placed, where any candidate position must be feasible with respect to overlap and containment in a bin.

Any solution approach to a packing problem involving irregular shapes requires computational tools for handling the geometry. Specifically, search heuristics need an efficient test for identifying feasible placement positions of pieces. Our tool set is based on the nofit polygon (NFP) and inner fit polygon (IFP) obtained using the approach of [Bennell and Song \(2008\)](#). Since pieces are permitted to be placed in any orientation, there are infinitely

many possible NFPs, hence it is necessary to generate these online as they are needed. In order to place a piece, the algorithm must generate multiple NFPs, one for each of the placed piece with the next piece to be placed. Our investigation shows that it is significantly more efficient to sequentially merge all the placed pieces into a single polygon and then generate a single, albeit large and complex, NFP. This is the approach of [Oliveira et al. \(2000\)](#), called TOPOS, which was later improved by [Bennell and Song \(2010\)](#).

The proposed constructive algorithm represents the existing partial solution as an array of polygons m_j where each m_j is the polygon generated by merging the set of pieces packed within the region b_j (equivalent to a bin). Unlike [Oliveira et al. \(2000\)](#), we fix the position of the partial solution within the bin. As in [Bennell and Song \(2010\)](#), the algorithm retains any enclosed gaps between the pieces when a piece is merged with the existing partial solution. These gaps form holes in the merged polygon, which are made available for placement of subsequent pieces provided the area of the hole is greater than the area of smallest unpacked piece. Let $H_k : h_1, \dots, h_q$ be the set of holes generated after placing the k th piece. [Figure 4.1](#), provides an example of two regions, which represent bins and contain a number of packed pieces. The groups of packed pieces are merged into two polygons m_1 and m_2 , which contain usable holes h_1 and h_2 . Note, holes that are too small are ignored. The spaces outside of m_j are denoted by S_j .

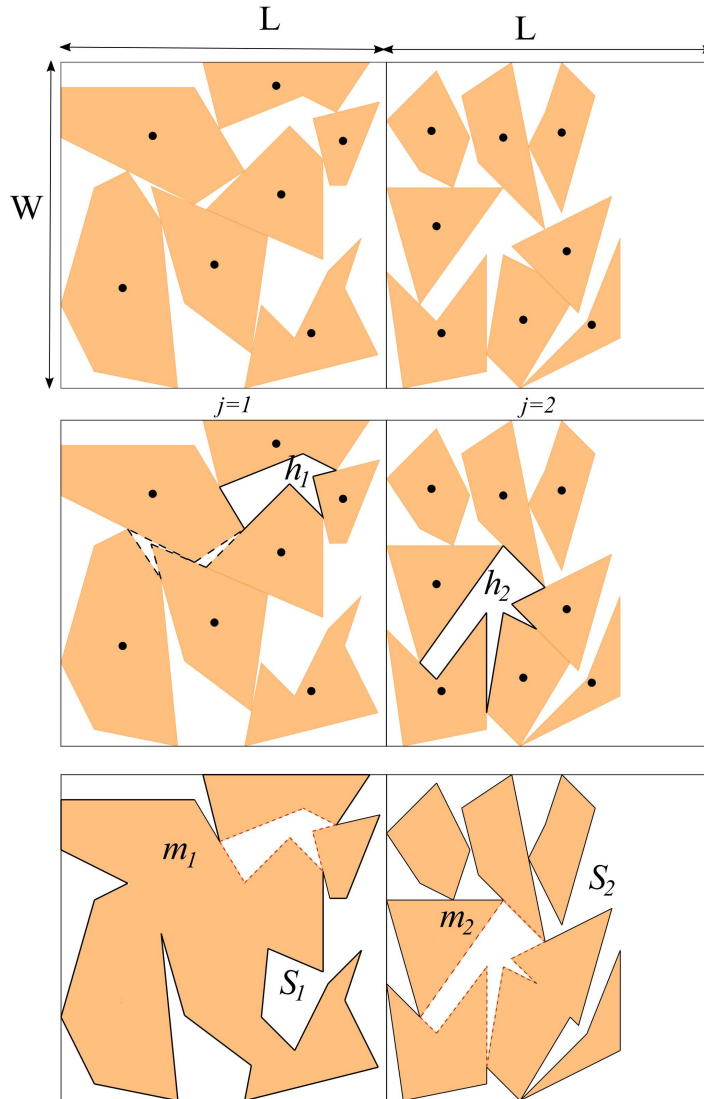


Figure 4.1: Partial solution

4.1.1. Generating candidate placement positions

In this section we describe our approach for identifying the feasible candidate positions for the next piece, p_i , including its orientation. p_i may be

placed within a hole, in a space or in a new bin region at the end of the layout. Below we describe the procedure for determining whether p_i will fit in a particular hole or space in a partially packed bin and the range of orientations.

Each hole h_l is represented as a simple polygon. If the area of h_l is less than the area of p_i then p_i will not fit into h_l . Otherwise, the procedure generates the IFP between h_l and p_i ($IFP_{h_l p_i}$). The interior and boundary of the IFP represents all the relative positions of the two component polygons so that the second subscript polygon, p_i , fits inside the first subscript polygon, h_l . See Figure 4.2. The boundary of the IFP contains all positions where the boundaries of the polygons touch and this forms the set of feasible placement positions within the hole. If the IFP does not exist then p_i will not fit into h_l .

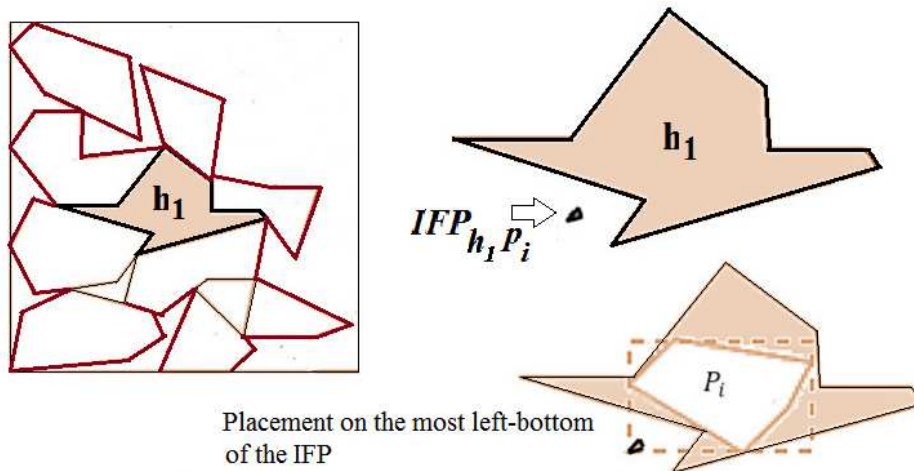


Figure 4.2: Hole-filling

In order to find the feasible placement positions within a bin region that contains polygon m_j , we generate the NFP between the external boundary

of m_j and p_i , $NFP_{m_j p_i}$ and the IFP between the bin region b_j and p_i , $IFP_{b_j p_i}$. The exterior and boundary of the NFP represents all relative positions of the two component polygons where they do not overlap. The intersection of the $IFP_{b_j p_i}$ and the complement of $NFP_{m_j p_i}$ provides all the positions of p_i such that it is contained within b_j and not overlapping m_j , assuming b_j and m_j have a fixed position. We only consider the boundary segment of the $NFP_{m_j p_i}$ that intersects with $IFP_{b_j p_i}$ as the feasible set, which is all the feasible touching positions between m_j and p_i . See Figure 4.3 for an example of this evaluation.

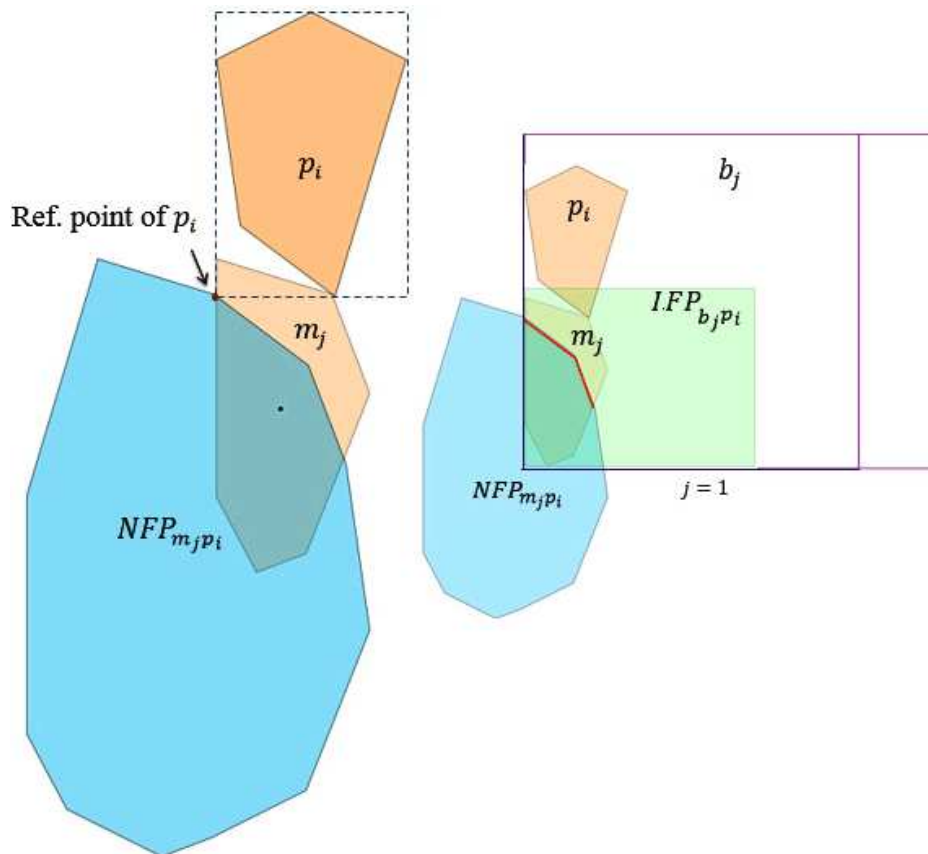


Figure 4.3: Feasible placement for a new piece

4.1.2. Placement orientation of a piece

The above discussion assumes p_i is in a given orientation. In this paper, we consider two scenarios with respect to rotating pieces. First, we allow pieces to rotate by a pre-assigned set of angles, which we define as the finite rotation approach. This is the most common case in the literature where the rotation angles of pieces are restricted to $\alpha_i = 0^\circ, 180^\circ$ or $\alpha = 0^\circ, 90^\circ, 180^\circ, 270^\circ$. Second we consider rotating pieces freely.

In the case of finite rotation angles, we simply consider p_i in each of the four orientations. For each orientation we generate the set of feasible placement positions, as described in 4.1.1, and select the best according to the placement policy criteria, described in the next section.

For free rotation there are an infinite set of possible orientations, where only a subset of these can be evaluated. In order to reduce these to a finite set of candidate orientations, we devise a mechanism to identify a subset of promising angles according to the arrangement of the pieces in the current partial solution. We describe the approach assuming p_i can be placed in a region that contains a merged polygon m_j .

Our approach begins by placing p_i on the packing layout in a temporary position and orientation using the finite rotation placement approach, where the best orientation is selected. As discussed in the previous section, the position will be touching the boundary of m_j . Each touching point or edge defines two new angles of orientation. These are illustrated in figure 4.4.

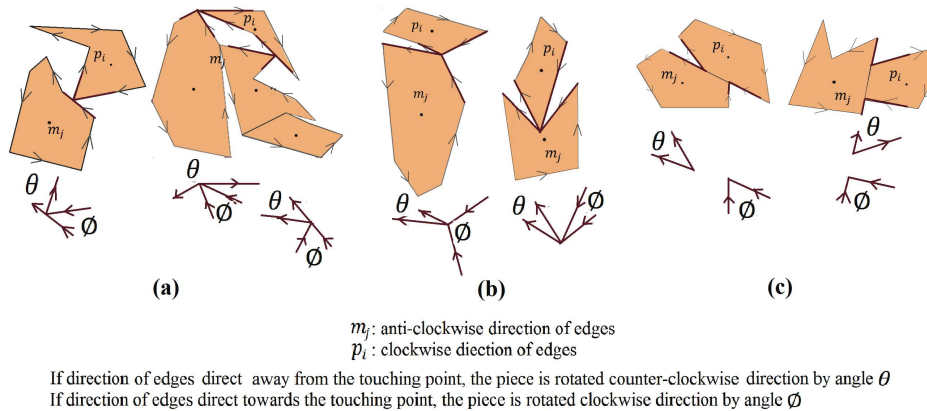
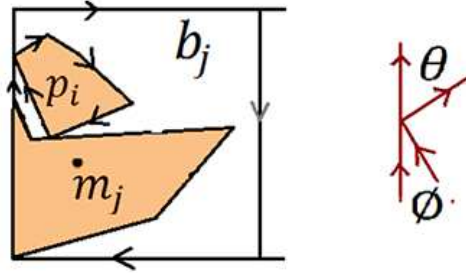


Figure 4.4: Tuning the orientation angle

Figure 4.4(a) shows an edge-vertex combination, figure 4.4(b) shows a vertex-vertex combination and figure 4.4(c) shows an edge-edge combination. In each case, piece p is rotated in a counter-clockwise direction by angle θ and in a clockwise direction by angle ϕ . The second example in figure 4.4(a), shows two separate points of contact and as a result there are four new orientations. These form the new candidate orientations and following the placement procedure, we find the best placement position for the pieces in these orientations. These are compared with the temporary position and orientation, if none of these new angles provide a better placement position then the algorithm takes the temporary position and orientation as the placement of the piece. If the next piece touches the boundary of a bin, then we also consider the angles created by the edges of the bin (see Figure 4.5).



p_i : clockwise direction of edges
 b_j : clockwise direction of edges

If direction of edges direct away from the touching point,
the piece is rotated counter-clockwise direction by angle θ

If direction of edges direct towards the touching point,
the piece is rotated clockwise direction by angle ϕ

Figure 4.5: Angle tuning at the edges of bins

4.2. Placement policy

The above procedures provide a set of feasible placement positions and orientations for piece p_i . The placement policy sets the criteria that determines which position is chosen. Note that for some of the placement rules, we can often avoid the computational cost of assessing all positions by controlling the order in which they are assessed. In the following we describe the placement rules in the context of packing from the left hand end of the strip extending towards the right. However, Jostle also packs from right to left, in which case simply swap left and right in the following explanation.

We evaluate three well known placement rules; bottom left placement, minimum length placement and maximum utilization placement, to determine the position of each piece. However, these rules are dominated by first trying to fill holes in the layout.

The first step when placing p_i is to sort all the holes in H_k in order of left most x-coordinate. These holes are evaluated in that order using the hole filling procedure. If p_i fits in a given hole, it is placed and no further holes are considered. If p_i does not fit within any hole, the algorithm attempts to find a feasible placement position in each of the bin regions b_j starting with the left most. Using the procedure described in Section 4.1 for finding placement positions and orientation in a partially packed bin, the candidate positions are found for each bin region b_j in order. The placement policies are as follows:

- Bottom-left (BL) prioritizes placing the reference point of the piece in the leftmost feasible position on the stock sheet breaking ties by selecting the bottom-most of the left most positions. This is a well-known single pass placement rule applied in many cutting and packing papers (Dowland et al., 1998; Blazewicz et al., 1993; Albano and Sapuppo, 1980). By applying this rule when hole filling fails, once a feasible position is found in region b_j the search can stop. If more than one orientation of p_i gives the same BL position, select the orientation that minimises the width of the bounding box of the piece. If there is still a tie, select the orientation randomly. Apply the same tie breaker rules for placing p_i in a new bin.
- Minimum Length (ML) placement rule selects the position that minimize the length from the origin of the bin to the right-most x-coordinate of p_i . The piece is orientated to minimise the right-most x-coordinate. In case of ties, we select the bottom-left placement position.

- Maximum Utilization (MU) placement rule selects the position that provides the maximum area utilisation in the earliest bin. We assume the occupied area of a region b_j is the area of the convex hull of m_j . Let the area of the convex hull of m_j be $\text{CHarea}(m_j)$, the area of p_i be $\text{area}(p_i)$ and the area of the convex hull of the new merged polygon after p_i is placed in a certain position be $\text{CHarea}(m_j + p_i)$. Then the utilisation of the placement position (Utp) is calculated as follows.

$$Utp = \frac{\text{CHarea}(m_j) + \text{area}(p_i)}{\text{CHarea}(m_j + p_i)}$$

Given the set of feasible placement positions on the boundary of the NFP, we select each feasible vertex of the NFP as a candidate placement position. This is illustrated in Figure 4.6. Placement positions are evaluated bin by bin starting at the left most bin. If a feasible placement is found in a bin then no further bins are considered. Ties are broken by selecting the bottom left position and minimum length orientation.

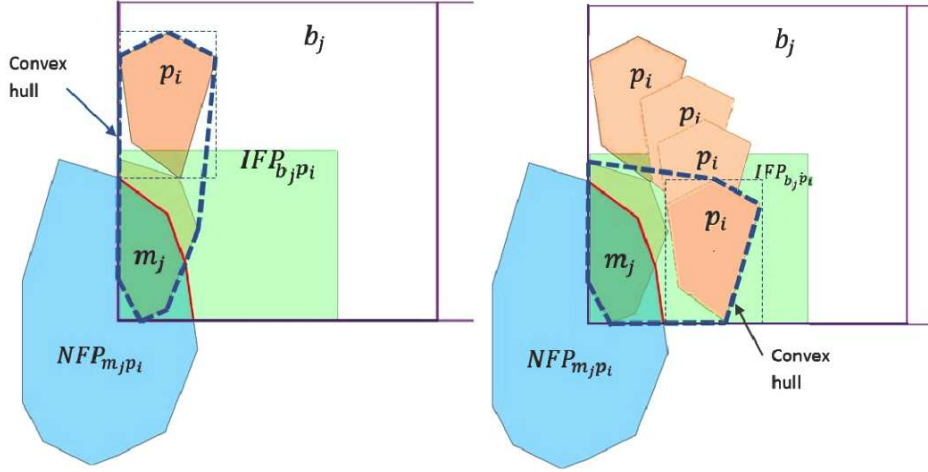


Figure 4.6: Placement of piece on different vertices along the feasible placement segment

Algorithm 1 denotes the constructive algorithm discuss in this paper.

5. Solution Improvement

In this section we describe the search mechanism for finding improving solutions. This includes four variants of the Jostle heuristic proposed by [Dowland et al. \(1998\)](#). The first modifies Jostle to work for bin packing problems, the second and third are designed to include mechanism for avoiding becoming trapped in local optima, and the fourth introduces a merge and split mechanism to retain profitable combinations of pieces.

5.1. Jostle heuristic

In this section, we discuss the design of the Jostle heuristic to address the 2DIBPP. As explained in Section 3, the Jostle heuristic determines the piece sequence that feeds into each iteration of the constructive algorithm. The first application of the constructive algorithm uses an arbitrary order of pieces

Algorithm 1: Constructive Algorithm

```
Input :  $P_0$ , Placement rule
Output:  $P^*$ : A feasible packing of all pieces in  $N$  Bins
1 Set  $\alpha = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ ,  $N = 1$ ;
2 for  $i = 1 : n$  do
3   for  $l = 1, 2, \dots, q$  do
4     if  $Area(p_i) \leq Area(h_l)$  then
5       Find feasible placement in hole  $h_l$  by rotating the existing  $p_i$  by each  $\alpha$  angle;
6       if there exist any feasible points for placement then
7         Place  $p_i$  with corresponding  $\alpha$  following the Placement rule;
8         Update  $H_{i-1} \rightarrow H_i$  and update  $q$ ;
9         Sort all the holes in  $H_i$  in order of left most x-coordinate;
10        End For Loop  $l$ ;
11      else
12        move to the next hole in the list  $H_{i-1}$ ;
13      end
14    else
15      move to the next hole in the list  $H_{i-1}$ ;
16    end
17  end
18  if No feasible placement by Hole-filling then
19    for  $j = 1, 2, \dots$  do
20      if  $Area(p_i) \leq Area(b_j) - Area(m_j)$  then
21        Find feasible placement path by rotating the existing  $p_i$  by each  $\alpha$  angle;
22        Find the best orientation angle;
23        Derive new angles by angle tuning mechanism;
24        Find feasible placement path by rotating  $p_i$  by each new angle;
25        if there exist any feasible points found for placement by  $\alpha$  angles or tuning angles then
26          Evaluate best placement position and angle for  $p_i$  following the
27            Placement rule;
28          Merge  $p_i$  to  $m_j$ ;
29          Capture any hole created ( $\geq$  smallest piece area);
30          Update  $H_{i-1} \rightarrow H_i$  and update  $q$ ;
31          If there are any new hole, sort holes  $H_i$  by left most x-coordinate;
32          if  $(j > N)$   $N = j$ ;
33          Break the For loop  $j$ ;
34        else
35          move to the next bin  $j$ ;
36        end
37      else
38        move to the next bin  $j$ ;
39      end
40    end
41  end
```

and packs the pieces from left to right along the strip. After generating a complete solution, all the pieces are re-ordered according to the piece with the right-most x-coordinates continuing to the left-most. Following this order,

pieces are packed starting at the right-most position of the strip building the packing layout from right to left. Jostle continues to oscillate from packing left to right then right to left until the termination criterion is met.

Jostle is analogous to a local search where the search occurs over the sequence of the pieces and each Jostle iteration generates a neighbour sequence. The constructive algorithm takes a sequence of pieces and creates a layout. The layout is converted back to a sequence according to the x-coordinate order. Hence, the new layout defines a new sequence of pieces, this is a neighbour solution based on the arrangement of pieces of the previous packing layout. Each iteration generates only one neighbour, which is always accepted. The optimisation arises from the neighbour generation. Instead of applying a fixed rule to the sequence i.e. swap or insert, the constructive algorithm changes the sequence by optimising the packing arising from the current piece sequence. Each complete solution is evaluated according to K and F defined in Section 3. At the end of the search, the best solution according to these measures is kept.

Since the constructive algorithm includes filling of holes and multiple alternative orientation, the piece sequence and placement positions change between iterations. However, our experiments show that the search can converge and get stuck in local optima. In order to overcome this we incorporate the diversification ideas from iterated local search to escape from local optima. The new variant is called Iterated Jostle.

5.2. Iterated Jostling

The Iterated Jostle algorithm combines the basic Jostle procedure and a kick intended to move the solution to a new area of the solution space and

escape local optima. This feature is motivated by our observation that the Jostle heuristic becomes trapped in local optima after running several Jostle cycles. This is more commonly found for data sets that are small and have similar size pieces. We perform a kick from the best local optimum found so far and consider that we are at a local optima after c_{max} Jostle cycles with no improvement. We investigate two kick strategies, described below. Note that in cutting and packing, a small change in the permutations can lead to a significant change in the layout of the pieces once it has been decoded by the constructive algorithm. In the case of the first kick strategy, we check that the move does not produce an identical permutation, which can occur when there are multiple pieces of the same type.

1-piece insert move: We perform a single insert move as follows:

1. Select two random positions along the permutation of pieces,
2. Select the piece that corresponds to the highest position selected in step 1,
3. Remove the selected piece from the permutation and insert it in front of the piece in the other position selected in step 1,
4. Order the remaining pieces following the existing permutation.

Bin-swapping move: This strategy changes the order of pieces by effectively swapping two bins as follows:

1. Select the bin with the lowest utilization and let sub-sequence A be the pieces in that bin in x-co-ordinate order,
2. Select another bin randomly and let sub-sequence B be the pieces in that bin in x-coordinate order,

3. Swap sub-sequence A and sub-sequence B in the permutation,
4. Obtain the new permutation of pieces.

Algorithm 2 describes the iterated Jostle procedure (IJS). Let t be the Jostle cycle, P_t^L be the permutation of pieces to be packed from left to right in cycle t , P_t^R be the permutation of pieces to be packed from right to left in cycle t and P_0 be the initial random permutation. P^* stores the permutation of pieces from the solution with smallest objective function value found so far, K^* , and D^* be the direction of packing. The variable c counts the number of consecutive Jostle iterations with no improvement in solution quality (K). The termination criteria is the maximum total number of Jostle cycles that the algorithm will run, T_{max} .

5.3. Iterated Jostling with join and release of pieces

This final version of Iterated Jostle draws on the ideas of Adamowicz and Albano (1972), which was more recently applied by Elkeran (2013). They combine pieces that fit together well reducing the overall number of pieces. In our approach we allow two pieces to be combined into one merged piece for some of the Jostle iterations. The algorithm scans the arrangement of pieces in each bin and attempts to join adjacent pieces in their current position. If the combined piece is convex with no holes then they are merged for the following Jostle iterations. Each scan only permits pairwise merging. However, scans in later iterations may join a piece to a merged piece resulting in three or more original pieces joining together. Pieces are joined after each Jostle iteration until a kick is performed, at which time they are separated again. The type of kick alternates between 1-piece insert and bin-swapping.

The following steps outline the IJS with join and release procedure.

Algorithm 2: IJS

Input : P_0 , Placement rule, kick type, c_{max} , T_{max}

Output: P^* , K^* , $D^* = L(left) \text{ or } R(right)$

```
1 Set  $t = 1$ ;  
2  $P^* = P_t^L = P_0$ ;  
3 Set  $D^* = L$ ;  
4 Set  $K^* =$  a suitably large value;  
5 Initialize  $c = 0$ ;  
6 while  $t \leq T_{max}$  do  
7   Generate solution layout from  $P_t^L$ ;  
8   Evaluate the solution and find  $K_t^L$ ;  
9   Derive  $P_t^R$  from solution;  
10  Generate solution layout from  $P_t^R$ ;  
11  Evaluate the solution and find  $K_t^R$ ;  
12  if  $K_t^L < K_t^R$  then  
13    if  $K_t^L < K^*$  then  
14      Set  $K^* = K_t^L$ ,  $P^* = P_t^L$ ,  $D^* = L$ ;  
15      Reset  $c = 0$ ;  
16    else  
17       $c = c + 1$ ;  
18    end  
19  else  
20    if  $K_t^R < K^*$  then  
21      Set  $K^* = K_t^R$ ,  $P^* = P_t^R$ ,  $D^* = R$ ;  
22      Reset  $c = 0$ ;  
23    else  
24       $c = c + 1$ ;  
25    end  
26  end  
27  if  $c > c_{max}$  then  
28    Apply kick to  $P^*$  and obtain  $P_t^{D^*}$ ;  
29    Generate solution layout from  $P_t^{D^*}$ ;  
30    Derive  $P_t^L$  from solution;  
31  end  
32   $t = t + 1$ ;  
33 end  
34 return  $P^*$ ,  $K^*$ ,  $D^*$ ;
```

1. From a given initial permutation, apply Jostle until the kick criteria is met.
2. Apply a kick alternating between bin-swap and 1-piece insert,
3. Jostle for one iteration
4. Execute pairwise joining of pieces
5. If stop criteria met, then stop else
6. If kick criteria met, release all joined pieces and go to step 2, else
7. go to step 3

6. Experimental Design and Results

In this section we provide details of our experiments and a discussion of the results. All the algorithms are coded in C++11 and use version 1.58.0 of the Boost computational geometry library to implement some of the geometric operations. The tests are performed by 2.6 GHz processor and 4 GB of RAM to execute the program.

6.1. Data instances

Our experiments utilise two sets of benchmark instances: the nesting instances used by a wide variety of authors for the irregular strip packing problem and instances proposed by Lopez-Camacho et al. (2013) that are jigsaw puzzle instances as they have known optimal solutions of 100% utilisation of the stock sheet. Both are available from the ESICUP (EURO Special Interest Group on Cutting and Packing) web site <http://paginas.fe.up.pt/~esicup/datasets>.

The nesting instances contain 23 irregular strip packing data sets and contain both convex and non-convex pieces. As these are strip packing instances we need to define bin sizes for each data set. For each instance we define three different bin sizes based on the maximum dimension of the enclosing rectangles of each piece in their original orientation denoted as d_{max} . The bin sizes are: small bin ($1.1d_{max}$ for length and width), medium bin ($1.5d_{max}$ for length and width) and large bin ($2d_{max}$ for length and width). These bin sizes were proposed by [Martinez-Sykora et al. \(2017\)](#).

The jigsaw puzzle (JP) instances contain two sets, which we denote as JP1 and JP2. JP1 has a collection of 540 instances in which all the pieces are convex. These instances are divided in 18 classes with 30 problems and each class carries a different number of pieces. The second set, JP2, has 480 instances (16 classes with 30 problems in each class) in which some of the pieces have concavities.

6.2. Experimental results

Our solution strategy includes a number of alternative approaches in terms of the placement policy in the constructive heuristic, the permitted orientations and the kick strategies in Jostle. [Table 6.1](#) and [Table 6.2](#) provide a summary and the notation for the alternative approaches. Any variant that includes a random input is repeated for 10 trials and we report the average across the trials. We conduct the following set of computational experiments and compare the performance of each approach.

Table 6.1: Notation of Algorithms: placement rule and rotation

Notation	Placement Rule	Piece orientation
BL4	Bottom-Left	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
BL ∞	Bottom-Left	Unrestricted
ML4	Minimum-Length	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
ML ∞	Minimum-Length	Unrestricted
MU4	Maximum Utilization	Restricted $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2})$
MU ∞	Maximum Utilization	Unrestricted

Table 6.2: Notation of Algorithms: algorithm and kick type

Notation	Description	Kick-move strategy
JS	Jostling	-
IJS1	Iterated Jostling	1-piece insert move
IJS2	Iterated Jostling	Bin swapping move
IJS3	Iterated Jostling	1-piece insert move , Bin swapping move , Join and Release of pieces

6.2.1. Test 01: Jostling search vs. random order of pieces

These experiments compare the performance of the two rotation criteria, finite set of angles and free rotation, and the alternative placement policies, $BL4$, $BL\infty$, $ML4$, $ML\infty$, $MU4$, and $MU\infty$, in terms of quality of packing and computational time. These experiments use the nesting instances. A CA trial is made up of 500 random permutations of pieces, each packed using a constructive heuristic. The best layout is recorded from each trial. In addition, we evaluate the Jostle heuristic and compare with packing random permutations of pieces using the CA. These experiments intend to investigate whether the Jostle heuristic provides a good search mechanism that improves on random trials. Each trial of the basic Jostle heuristic executes 250 Jostle cycles (which generates 500 layouts). Table 6.3 details the results and considers measure K and F . Note we wish to minimise K and maximise F .

Table 6.3: Performance of construction algorithm for different rotation criteria

	Random order						Jostling					
	Small bins		Medium bins		Large bins		Small bins		Medium bins		Large bins	
	K	F	K	F	K	F	K	F	K	F	K	F
<i>BLA</i>	9.91	0.379	4.792	0.389	2.601	0.395	9.61	0.387	4.712	0.392	2.602	0.398
<i>BL∞</i>	9.73	0.383	4.632	0.394	2.555	0.392	9.48	0.389	4.612	0.397	2.487	0.399
<i>MLA</i>	9.94	0.378	4.794	0.390	2.613	0.396	9.52	0.382	4.703	0.394	2.551	0.404
<i>ML∞</i>	9.76	0.384	4.612	0.393	2.551	0.392	9.42	0.389	4.511	0.398	2.464	0.407
<i>MU4</i>	9.89	0.383	4.944	0.385	2.592	0.391	9.65	0.387	4.786	0.402	2.561	0.401
<i>MU∞</i>	9.62	0.385	4.614	0.393	2.524	0.392	9.47	0.402	4.573	0.408	2.470	0.413

The experimental results in Table 6.3 show that allowing free rotation of pieces can significantly improve the solution quality. Considering a pairwise comparison of four angles of rotation and free rotation for each placement rule (BL, ML, MU), 16 of 18 comparisons produced better solutions using free rotation for both K and F , and the remaining two comparisons showed free rotation is better for measure K but not for F . We also compared the random permutations to applying the CA to an ordered permutation by decreasing area. On average the random permutations are 3.6% better by the F value, but take significantly longer in computation time.

Considering the performance of Jostle against the randomly generated permutations, the experimental results shown in Table 6.3 reveal that the Jostle heuristic performs better for all pairwise comparisons between placement rules and rotation criteria. The best solutions are produced when pieces are allowed to rotate freely. Since there is no significant difference between the computational time in both cases, we can also conclude that the time consumption of the Jostle heuristic is negligible and time is mainly spent on the execution of the CA. The results are aggregating over many problem instances. A detailed look at the instances reveals that often the number of bins are reduced and in other cases the piece are packed more tightly across

the same number of bins. We also observed that Jostle struggles to diversify from the first local optima, so the improvements are local with respect to the initial solution. With this in mind, we conclude that Jostle can perform as an optimisation procedure with more intelligence than a random search.

The final observation from these results concerns the placement rule. For random permutations BL performs the best once on measure F , ML performs the best twice and MU performs the best three times. For Jostle the result is more consistent where ML performs the best for each measure of K and MU performs the best for each measure of F . This result is intuitive since K aims to minimise length to provide a greater residual in the last bin, while F is based on utilisation. It is interesting to note the BL is consistently adopted in the literature, but in our investigation it is consistently inferior.

6.2.2. Test 02: IJS vs JS

These experiments compare the performance of the four Jostle strategies; JS, IJS1, IJS2 and IJS3, where the latter three strategies use a kick. Table 6.2 provides details of these strategies. Ten random permutations are used as initial permutations for each variant and each trial runs for 500 Jostle cycles. After initial experiments with a range of values for c_{max} , we set $c_{max} = 4, 6, 6$ Jostle cycles for IJS1, IJS2 and IJS3 respectively.

The alternative strategies arose through the observation that JS tends to get stuck in local optima. This occurs more frequently with small data sets and with shapes that are less irregular. As an example, Figure 6.1(a) illustrates solution F values received at each iteration for a certain trial for Fu data instance using JS. In this case the Jostle heuristic has taken 9-12 cycles to reach a better F value (i.e. $F = 0.488$). When Jostling continues,

the solution rarely reaches $F = 0.488$ and the best value is recorded at 143rd cycle. After 176th cycle, the Jostling has reached a steady state generating the same set of solutions repetitively. Figure 6.1(b) demonstrates the benefit of using a kick. Here IJS1 is run on the same data. Introducing a kick has avoided the repetitive cycle and produced better solutions (i.e. $F > 0.499$) within fewer iterations.

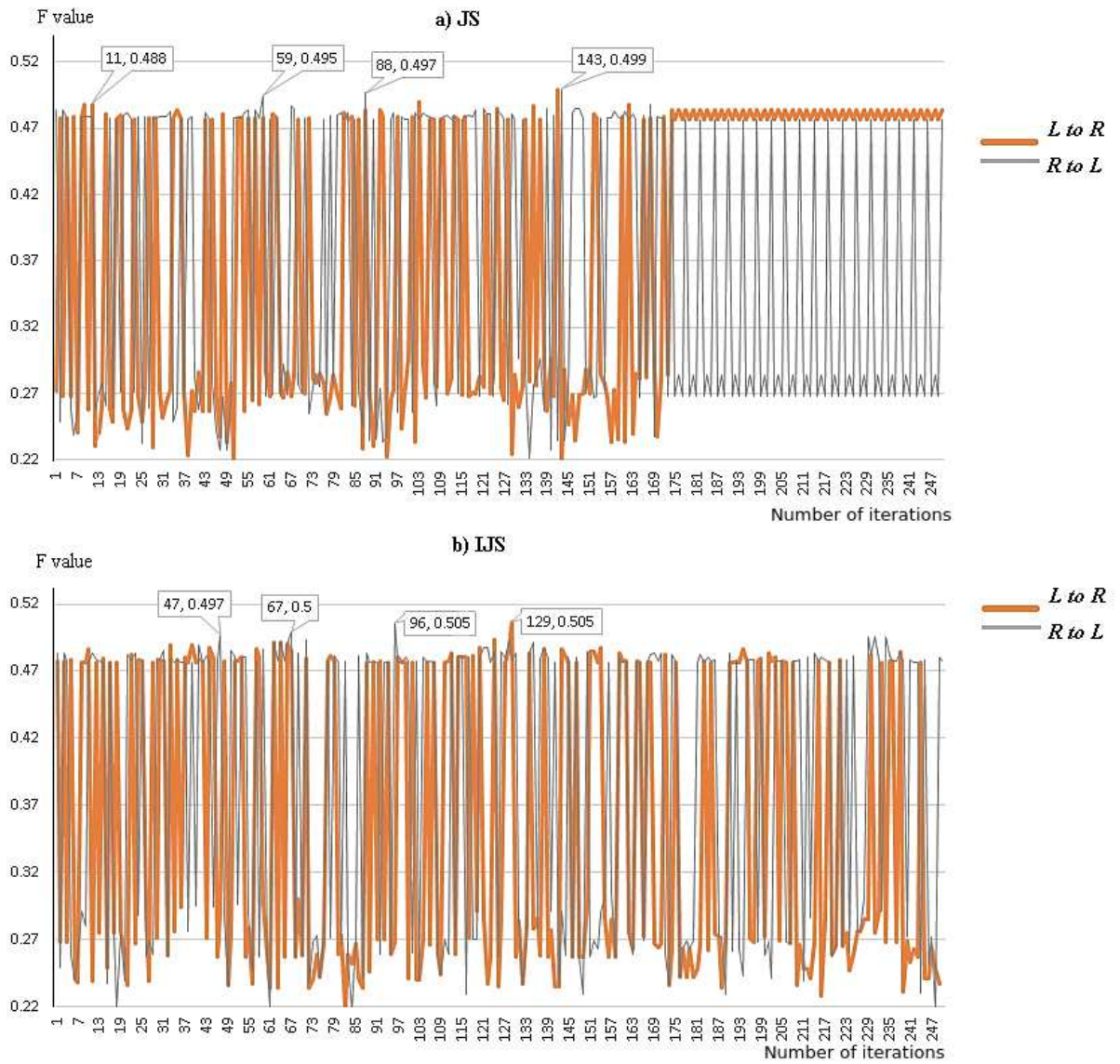


Figure 6.1: JS and LJS1 iterations

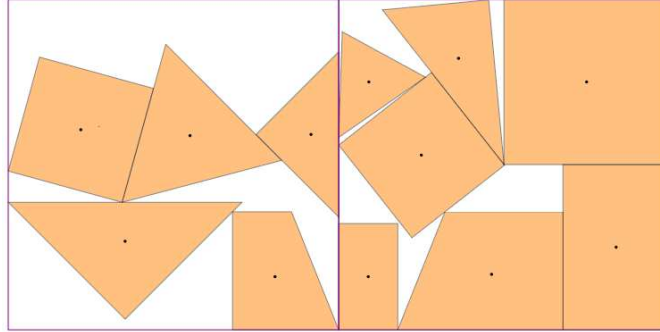


Figure 6.2: Fu instance packed in the largest bin type ($2d_{max}$) using IJS1- MU_{∞} method, $F = 0.509$, run time is 8 seconds for 500 iterations

Table 6.4: Performance results JS, IJS1, IJS2 and IJS3 for nesting data instances

		JS		IJS1		IJS2		IJS3	
		F	T (sec)	F	T (sec)	F	T (sec)	F	T (sec)
Small bin	BL4	0.384	304.4	0.394	322.4	0.388	310.7	0.394	319.4
	BL ∞	0.389	502.7	0.408	528.5	0.394	512.1	0.404	534.8
	ML4	0.386	278.6	0.393	313.4	0.390	285.8	0.397	320.5
	ML ∞	0.389	509.1	0.419	526.6	0.392	533.4	0.402	575.3
	MU4	0.388	281.4	0.393	294.0	0.389	286.4	0.403	335.0
	MU ∞	0.407	504.9	0.437	518.6	0.427	534.4	0.433	540.5
Avg.		0.391		0.408		0.397		0.406	
Medium bin	BL4	0.392	287.5	0.403	298.4	0.409	298.1	0.407	315.9
	BL ∞	0.398	484.7	0.409	506.1	0.403	496.3	0.404	505.4
	ML4	0.394	269.5	0.409	301.8	0.400	281.2	0.402	306.2
	ML ∞	0.398	476.6	0.422	515.9	0.418	488.7	0.414	530.6
	MU4	0.404	267.4	0.418	281.2	0.410	272.0	0.409	306.8
	MU ∞	0.408	487.9	0.436	499.0	0.415	503.4	0.429	510.1
Avg.		0.399		0.416		0.409		0.410	
Large bin	BL4	0.398	248.9	0.403	266.5	0.402	255.6	0.410	280.7
	BL ∞	0.400	421.6	0.404	450.1	0.402	430.7	0.415	485.3
	ML4	0.404	239.4	0.422	259.7	0.400	249.4	0.410	272.6
	ML ∞	0.408	426.3	0.425	451.3	0.409	441.6	0.415	477.6
	MU4	0.402	240.4	0.425	253.9	0.404	247.1	0.410	268.9
	MU ∞	0.414	426.8	0.431	453.9	0.413	436.8	0.407	480.7
Avg.		0.404		0.418		0.405		0.411	

Table 6.5: Performance results of IJS1, IJS2 and IJS3 vs JS for JP1 and JP2 instances

F values		JS		IJS1		IJS2		IJS3	
		F	T (sec)	F	T (sec)	F	T (sec)	F	T (sec)
JP1	BL4	0.589	76.5	0.652	81.2	0.648	76.3	0.700	76.2
	BL ∞	0.527	148.5	0.564	161.4	0.579	167.4	0.594	153.8
	ML4	0.508	72.7	0.625	79.6	0.647	80.8	0.693	75.8
	ML ∞	0.522	150.3	0.634	161.9	0.564	152.0	0.590	162.4
	MU4	0.566	72.1	0.691	78.1	0.678	79.6	0.732	74.9
	MU ∞	0.552	159.7	0.677	163.8	0.639	166.7	0.644	161.2
Avg.		0.544		0.641		0.626		0.659	
JP2	BL4	0.572	56.3	0.652	59.3	0.682	62.3	0.725	63.8
	BL ∞	0.506	98.4	0.522	101.2	0.532	102.4	0.547	103.3
	ML4	0.549	55.7	0.677	61.3	0.697	61.2	0.714	61.5
	ML ∞	0.505	98.1	0.632	103.4	0.607	104.6	0.516	100.1
	MU4	0.617	54.9	0.701	62.1	0.680	59.2	0.747	58.3
	MU ∞	0.584	97.8	0.655	107.8	0.626	112.5	0.613	101.5
Avg.		0.556		0.640		0.637		0.644	

In Table 6.4 we present the average results obtained by JS and the three IJS procedures for the 23 nesting instances. For all placement criteria and angles of rotation, the three IJS algorithms produce better results when compared with JS. For nesting data instances, IJS1 provides the best results for all bin sizes when averaging over all placement/rotation variants and for each individual result apart from medium bins BL4 where IJS2 is best. Focusing just on the IJS algorithms, there are 18 variants (3 placement rules x 2 angles of rotation x 3 IJS algorithms). Out of those 18 methods, IJS1-MU ∞ produces the best results irrespective of the bin sizes.

In Table 6.5 we present the average results obtained by JS and the three IJS procedures for the 540 JP1 and 480 JP2 instances. Again the IJS approaches are clearly better than the values of JS. On average IJS3 performed better for both JP1 and JP2 instances. However, in this case IJS3-MU4 has produced the best results. For this data the optimal solution can only be found at the zero rotation angle, hence it is intuitive that restricting the

rotation is beneficial.

6.2.3. Test 03: Comparing with benchmark results in literature

In this case we compare our results with the work of [Lopez-Camacho et al. \(2013a\)](#) and [Martinez-Sykora et al. \(2017\)](#). Since these algorithms are run on different computing platforms, it is not possible to run for the same amount of time. Hence we retain our termination criteria of 500 Jostle cycles and report the average result and computational time for the ten random trials.

Referring to the previous tables [6.4](#) and [6.5](#) we can compare these results with the results from the literature. [Martinez-Sykora et al. \(2017\)](#) report a best value of $F = 0.723$ with computational time 161 seconds for JP1 and $F = 0.729$ with computational time 123 seconds for JP2. In both cases our algorithm (IJS3-MU4) provides better results, specifically $F = 0.732$ with computational time 74.9 seconds for JP1 and $F = 0.747$ with computational time 58.3 seconds for JP2. Also our approach has achieved better results for 11 out of 18 groups of instances for JP1. In the case of JP2 instances our method has achieved better results for 11 out of 16 groups of instances. A detailed description of these results are provided in table [A.1](#) and [A.2](#). The only other existing results for this data is by [Lopez-Camacho et al. \(2013a\)](#), who only report results for JP1. They achieve $F = 0.690$ with computational time 50 seconds. We revised the termination of our algorithm to 50 seconds and achieved the improved result of $F = 0.703$.

For the nesting instances we consider the average over all the instances for each bin size. The only benchmark results are from [Martinez-Sykora et al. \(2017\)](#). For small bins they achieve $F = 0.434$ with computation time of 1721

seconds, for medium bins they achieve $F = 0.452$ with computation time of 3172 seconds and for large bins they achieve $F = 0.403$ with computation time of 762 seconds. IJS1-MU ∞ performs better for small and large bins and takes less time for all three bin types. The overall results of [Martinez-Sykora et al. \(2017\)](#) demonstrate that their algorithm has lower performance when average number of pieces per bin is higher. However in our case, we achieve significantly better results for large bins in most of our approaches; i.e. IJS1, IJS2 and IJS3. Even for small and medium sized bins, we achieved better or close results with no significant drop in average F values, with less computational time than [Martinez-Sykora et al. \(2017\)](#). Table [A.3](#), [A.4](#) and [A.5](#) present the results we achieved for each data instance using the IJS1 approach.

7. Concluding Remarks

In this paper we address a new solution method to solve the two dimensional irregular shapes bin packing problem which has received little attention in literature despite clear industry relevance. We modify the existing use of the Jostle procedure of [Dowland et al. \(1998\)](#) applied to strip packing and convert the heuristic for the homogeneous bin packing problem. Further we introduced a diversification strategy into Jostle that significantly improved the results. The proposed heuristic supports the use of restricted rotation and unrestricted rotation of pieces.

The findings of the study are as follows. Jostle is a promising packing heuristic that can be applied to bin packing problems. It is demonstrated to be a low cost search strategy that self governs the neighbourhood structure

that modifies the solution. The designed algorithms have been shown to improve on the results in the literature with lower computation time and does not require the use of expensive Mixed Integer Program software, making it attractive to small companies.

We have introduced a new way of dealing with free rotation of pieces. We have demonstrated the value of free rotation for nesting instances and justified the results of restricted rotation for the jigsaw puzzle instances.

We have designed and tested a number of placement policies and shown that despite the extensive use of bottom-left (BL) this does not produce the best results when compared with minimizing length or maximising utilisation.

References

- M. Adamowicz and A. Albano. A two-stage solution of the cutting stock problem. *Information Processing*, 71:1086–1091, 1972.
- A. Albano and G. Sapuppo. Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods. *IEEE Transactions on Systems, Man and Cybernetics*, 10(5):242–248, 1980. ISSN 0018-9472.
- J. A. Bennell and J. F. Oliveira. A Tutorial in Irregular shape Packing Problems. *The Journal of the Operational Research Society*, 60:s93–s105, 2009. ISSN 0160-5682.
- J. A. Bennell and X. Song. A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computers & Operations Research*, 35(1):267–281, 2008. ISSN 0305-0548.
- J. A. Bennell and X. Song. A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 16(2):167–188, 2010. ISSN 1381-1231, 1572-9397.

- J. Blazewicz, P. Hawryluk, and R. Walkowiak. Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, 41(4):313–325, 1993. ISSN 0254-5330, 1572-9338.
- E. K. Burke, R. S. R. Hellier, G. Kendall, and G. Whitwell. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179(1):27–49, 2007. ISSN 0377-2217.
- K. A. Dowsland, W. B. Dowsland, and J. A. Bennell. Jostling for position: local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6):647–658, 1998. ISSN 0160-5682.
- Ahmed Elkeran. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, 231(3):757–769, 2013. 00020.
- W. Han, J. A. Bennell, X. Zhao, and X. Song. Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints. *European Journal of Operational Research*, 230(3):495–504, 2013. ISSN 0377-2217.
- M. Hifi and R. M’Hallah. A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes. *International Transactions in Operational Research*, 10(3):195–216, 2003. ISSN 1475-3995.
- S. Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165–181, 1996. ISSN 0377-2217.
- D. Liu and H. Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles1. *European Journal of Operational Research*, 112(2):413–420, 1999. ISSN 0377-2217.
- 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(1):241–264, 2013a. ISSN 0254-5330.
- E. Lopez-Camacho, H. Terashima-Marin, G. Ochoa, and S. E. Conant-Pablos. Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(1):488–499, 2013b. ISSN 0925-5273.

- E. Lopez-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889, 2014. ISSN 0957-4174.
- A. Martinez-Sykora, R. Alvarez-Valdes, J. A. Bennell, R. Ruiz, and J. M. Tamarit. Matheuristics for the irregular bin packing problem with free rotations. *European Journal of Operational Research*, 258(2):440–455, 2017. ISSN 0377-2217.
- A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, and J. M. Tamarit. Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts. *Omega*, 52:15–32, 2015. ISSN 0305-0483.
- J. F. Oliveira, A. M. Gomes, and J. S. Ferreira. TOPOS A new constructive algorithm for nesting problems. *OR-Spektrum*, 22(2):263–284, 2000. ISSN 0171-6468, 1436-6304.
- P. Rocha. Circle Covering Representation for Nesting Problems with Continuous Rotations. In Boje Edward, editor, *Proceedings of 19th World Congress The International Federation of Automatic Control*, pages 5235–5240, 2014.
- 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*, 65(7):1037–1052, 2014.
- Y. G. Stoyan, M. V. Zlotnik, and A. M. Chugay. Solving an optimization packing problem of circles and non-convex polygons with rotations into a multiply connected region. *Journal of the Operational Research Society*, 63(3):379–391, 2012. ISSN 0160-5682.
- 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(1):369–392, 2008. ISSN 0254-5330, 1572-9338.

Appendix A.

Table A.1: Results of the IJS1, IJS2 and IJS3 algorithm applied for JP1 instances

BVL: Best values of [Lopez-Camacho et al. \(2013a\)](#), BVM: Best values of [Martinez-Sykora et al. \(2017\)](#)

	BVL(F)	BVM(F)	IJS1 (F values)						IJS2 (F values)						IJS3 (F values)					
			BL4	BL ∞	ML4	ML ∞	MU4	MU ∞	BL4	BL ∞	ML4	ML ∞	MU4	MU ∞	BL4	BL ∞	ML4	ML ∞	MU4	MU ∞
A	0.605	0.614	0.609	0.583	0.606	0.607	0.614	0.606	0.604	0.575	0.606	0.582	0.614	0.601	0.620	0.575	0.624	0.595	0.632	0.597
B	0.929	0.878	0.770	0.675	0.744	0.795	0.890	0.801	0.675	0.709	0.745	0.712	0.830	0.796	0.827	0.739	0.817	0.724	0.880	0.812
C	0.763	0.736	0.708	0.584	0.645	0.649	0.732	0.725	0.695	0.605	0.696	0.576	0.760	0.664	0.772	0.614	0.763	0.623	0.797	0.701
D	0.579	0.591	0.593	0.576	0.595	0.587	0.593	0.582	0.587	0.573	0.58	0.567	0.594	0.587	0.603	0.576	0.602	0.584	0.610	0.576
E	0.412	0.529	0.451	0.381	0.447	0.486	0.478	0.455	0.453	0.405	0.425	0.396	0.465	0.467	0.496	0.427	0.485	0.434	0.540	0.442
F	0.496	0.564	0.517	0.483	0.481	0.524	0.532	0.520	0.509	0.486	0.498	0.481	0.513	0.507	0.538	0.493	0.533	0.501	0.548	0.498
G	0.814	0.809	0.731	0.671	0.711	0.742	0.804	0.807	0.722	0.665	0.721	0.671	0.751	0.760	0.788	0.681	0.781	0.697	0.819	0.762
H	0.928	0.868	0.759	0.683	0.743	0.771	0.877	0.870	0.765	0.696	0.768	0.707	0.808	0.793	0.836	0.719	0.822	0.716	0.883	0.812
I	0.627	0.652	0.643	0.629	0.644	0.644	0.648	0.641	0.634	0.653	0.636	0.636	0.651	0.640	0.679	0.673	0.681	0.644	0.697	0.653
J	0.665	0.701	0.672	0.656	0.670	0.665	0.672	0.668	0.667	0.644	0.667	0.653	0.675	0.668	0.690	0.652	0.689	0.665	0.698	0.661
K	0.718	0.763	0.704	0.609	0.662	0.683	0.76	0.762	0.711	0.62	0.664	0.601	0.726	0.685	0.752	0.631	0.744	0.647	0.768	0.682
L	0.512	0.619	0.550	0.409	0.539	0.583	0.596	0.588	0.587	0.446	0.592	0.425	0.586	0.552	0.618	0.477	0.621	0.489	0.630	0.551
M	0.589	0.655	0.599	0.506	0.575	0.581	0.627	0.617	0.609	0.506	0.598	0.479	0.622	0.559	0.676	0.522	0.669	0.508	0.723	0.568
N	0.503	0.668	0.534	0.493	0.519	0.497	0.534	0.509	0.517	0.494	0.501	0.492	0.519	0.525	0.543	0.497	0.531	0.496	0.549	0.495
O	0.823	0.848	0.741	0.519	0.648	0.575	0.845	0.721	0.764	0.509	0.801	0.444	0.841	0.659	0.849	0.519	0.838	0.444	0.937	0.724
P	0.678	0.852	0.638	0.445	0.605	0.639	0.704	0.683	0.661	0.542	0.654	0.487	0.677	0.615	0.696	0.559	0.689	0.511	0.717	0.628
Q	1	0.965	0.807	0.639	0.723	0.689	0.805	0.868	0.783	0.676	0.792	0.653	0.842	0.727	0.865	0.699	0.858	0.699	0.956	0.729
R	0.771	0.767	0.711	0.609	0.687	0.697	0.733	0.754	0.721	0.623	0.697	0.588	0.733	0.689	0.758	0.646	0.735	0.645	0.786	0.694
Av.	0.690	0.723	0.652	0.564	0.625	0.634	0.691	0.677	0.648	0.579	0.647	0.564	0.678	0.639	0.700	0.594	0.693	0.590	0.732	0.644

Table A.2: Results of the IJS1, IJS2 and IJS3 algorithms applied for JP2 instances

BVM: Best values of [Martinez-Sykora et al. \(2017\)](#)

	BVM(F)	IJS1						IJS2						IJS3					
		BL4	BL ∞	ML4	ML ∞	Ut4	Ut ∞	BL4	BL ∞	ML4	ML ∞	Ut4	Ut ∞	BL4	BL ∞	ML4	ML ∞	Ut4	Ut ∞
A	0.680	0.587	0.546	0.584	0.574	0.600	0.582	0.588	0.552	0.594	0.579	0.578	0.571	0.609	0.569	0.602	0.562	0.604	0.569
B	0.760	0.715	0.635	0.737	0.753	0.743	0.755	0.748	0.645	0.743	0.684	0.729	0.701	0.768	0.656	0.768	0.694	0.804	0.701
C	0.691	0.613	0.553	0.629	0.752	0.636	0.673	0.624	0.567	0.636	0.598	0.624	0.697	0.680	0.574	0.644	0.542	0.692	0.594
F	0.551	0.495	0.477	0.503	0.529	0.511	0.529	0.495	0.473	0.512	0.489	0.503	0.493	0.533	0.489	0.518	0.486	0.525	0.489
H	0.828	0.709	0.651	0.788	0.702	0.792	0.743	0.741	0.661	0.789	0.745	0.744	0.762	0.792	0.673	0.804	0.652	0.809	0.719
I	0.576	0.546	0.418	0.541	0.532	0.539	0.532	0.536	0.422	0.547	0.497	0.547	0.511	0.582	0.449	0.581	0.435	0.588	0.521
M	0.681	0.569	0.485	0.578	0.579	0.581	0.583	0.576	0.496	0.581	0.517	0.539	0.531	0.617	0.508	0.596	0.368	0.627	0.543
O	0.788	0.708	0.469	0.769	0.651	0.803	0.672	0.761	0.474	0.800	0.636	0.781	0.672	0.823	0.484	0.790	0.699	0.826	0.624
S	0.537	0.522	0.378	0.539	0.526	0.623	0.531	0.621	0.386	0.623	0.512	0.602	0.518	0.621	0.412	0.591	0.308	0.642	0.474
T	0.921	0.829	0.538	0.875	0.483	0.908	0.883	0.865	0.55	0.901	0.532	0.874	0.747	0.907	0.563	0.906	0.435	0.959	0.747
U	0.764	0.651	0.417	0.73	0.678	0.728	0.678	0.711	0.432	0.735	0.712	0.728	0.674	0.777	0.443	0.772	0.303	0.823	0.596
V	0.989	0.775	0.462	0.803	0.593	0.922	0.593	0.811	0.482	0.838	0.589	0.894	0.548	0.915	0.503	0.878	0.582	0.993	0.645
W	0.669	0.696	0.647	0.689	0.648	0.705	0.693	0.735	0.665	0.704	0.666	0.678	0.641	0.782	0.678	0.787	0.677	0.843	0.643
X	0.632	0.588	0.496	0.589	0.603	0.599	0.604	0.599	0.51	0.599	0.512	0.573	0.519	0.632	0.538	0.641	0.497	0.649	0.581
Y	0.726	0.657	0.584	0.672	0.714	0.706	0.714	0.683	0.596	0.721	0.648	0.688	0.698	0.729	0.599	0.727	0.577	0.735	0.633
Z	0.893	0.779	0.590	0.811	0.801	0.826	0.715	0.814	0.603	0.826	0.803	0.802	0.729	0.829	0.615	0.818	0.446	0.839	0.728
Av.	0.729	0.652	0.522	0.677	0.632	0.701	0.655	0.682	0.532	0.697	0.607	0.680	0.626	0.725	0.547	0.714	0.516	0.747	0.613

Table A.3: Results of the IJS1 algorithm for each nesting instances with $2d_{max}$ Large bins
 Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL ∞		ML4		ML ∞		MU4		MU ∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.383	77.8	0.395	162.2	0.402	74.8	0.387	154.2	0.404	83.2	0.388	163.3
Shape2	0.437	56.2	0.424	107.5	0.446	51.2	0.447	103.7	0.440	60.7	0.451	110.3
Trousers	0.451	294.7	0.414	683.5	0.459	314.6	0.44	718.8	0.462	358.3	0.462	754.3
Shape0	0.294	245.9	0.294	504.2	0.294	248.4	0.294	521.7	0.294	263.7	0.294	548.9
Shape1	0.304	416.5	0.306	740.4	0.304	408.3	0.308	751.4	0.304	408.3	0.318	748.4
Shirts	0.647	710.9	0.651	1221.4	0.649	626.8	0.652	1143.2	0.646	618.3	0.652	1118.4
Dighe2	0.260	14.4	0.260	35.2	0.260	17.0	0.260	37.6	0.260	16.5	0.260	34.6
Dighe1	0.329	17.9	0.329	33.6	0.329	18.2	0.329	34.6	0.329	17.6	0.329	35.1
Fu	0.487	4.9	0.502	6.3	0.502	5.0	0.505	7.0	0.503	4.4	0.505	8.2
Han	0.311	83.6	0.314	172.6	0.316	86.8	0.356	176.4	0.357	79.2	0.412	144.3
Jakobs1	0.617	52.4	0.609	92.4	0.613	50.9	0.616	98.3	0.617	48.9	0.616	92.4
Jakobs2	0.455	46.0	0.452	89.8	0.45	45.8	0.454	92.2	0.456	48.4	0.448	96.8
Mao	0.256	113.4	0.256	191.3	0.61	119.2	0.61	188.3	0.61	106.7	0.610	176.7
Poly1a	0.368	22.5	0.368	41.7	0.368	23.2	0.368	43.5	0.368	24.6	0.368	42.0
Poly2a	0.376	79.5	0.384	126.3	0.382	72.4	0.384	131.7	0.381	74.3	0.386	128.8
Poly3a	0.395	149.3	0.389	247.0	0.387	131.4	0.416	258.9	0.392	133.6	0.416	246.8
Poly4a	0.399	243.3	0.416	412.5	0.394	238.6	0.407	426.4	0.399	254.9	0.418	420.7
Poly5a	0.406	311.1	0.409	526.5	0.399	336.9	0.408	556.9	0.405	328.5	0.408	589.5
Poly2b	0.443	81.8	0.455	131.0	0.454	76.6	0.45	132.9	0.454	79.4	0.459	126.3
Poly3b	0.387	171.6	0.401	252.9	0.403	177.4	0.399	249.9	0.414	156.8	0.418	239.7
Poly4b	0.402	295.8	0.405	491.2	0.397	286.8	0.401	484.7	0.393	276.1	0.406	465.7
Poly5b	0.468	433.1	0.471	652.9	0.492	428.9	0.471	671.6	0.471	408.5	0.473	701.6
Swim	0.398	2206.5	0.393	3430.0	0.404	2134.8	0.404	3396.6	0.416	1988.1	0.408	3447.3
Avg.	0.403	266.5	0.404	450.1	0.422	259.7	0.425	451.3	0.425	253.9	0.431	453.9

Table A.4: Results of the IJS1 algorithm for each nesting instances with $1.5d_{max}$ Medium bins
 Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL ∞		ML4		ML ∞		MU4		MU ∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.495	79.2	0.504	0.504	0.506	84.5	0.523	180.5	0.525	87.3	0.532	178.0
Shape2	0.289	78.1	0.331	0.331	0.315	56.9	0.335	122.3	0.338	60.7	0.339	123.6
Trousers	0.557	374.6	0.562	0.567	0.564	349.2	0.565	783.5	0.590	379.8	0.574	814.6
Shape0	0.268	302.5	0.268	0.268	0.268	275.7	0.268	584.3	0.268	295.3	0.268	592.8
Shape1	0.270	552.6	0.369	0.369	0.270	494.0	0.376	766.5	0.275	494.0	0.369	823.2
Shirts	0.518	740.8	0.526	0.546	0.634	727.1	0.557	1337.6	0.665	680.1	0.664	1230.2
Dighe2	0.322	15.4	0.277	0.277	0.302	20.9	0.317	45.9	0.281	18.8	0.322	41.5
Dighe1	0.350	18.1	0.325	0.325	0.352	19.4	0.354	39.5	0.314	20.1	0.365	39.0
Fu	0.410	5.6	0.421	0.421	0.410	5.3	0.432	8.2	0.427	4.9	0.431	9.2
Han	0.387	94.6	0.376	0.376	0.384	89.4	0.405	215.2	0.402	84.8	0.402	161.6
Jakobs1	0.570	59.1	0.519	0.519	0.558	59.1	0.541	115.0	0.565	56.2	0.579	109.0
Jakobs2	0.408	49.0	0.401	0.401	0.398	55.8	0.417	93.1	0.413	57.1	0.438	96.8
Mao	0.494	119.1	0.493	0.493	0.496	119.2	0.494	201.5	0.495	124.8	0.501	192.6
Poly1a	0.320	23.0	0.322	0.329	0.335	24.1	0.322	46.2	0.320	26.8	0.349	46.2
Poly2a	0.361	96.9	0.358	0.358	0.353	78.9	0.368	131.7	0.346	80.2	0.366	150.7
Poly3a	0.424	164.3	0.431	0.431	0.427	142.0	0.435	310.7	0.425	147.0	0.442	273.9
Poly4a	0.398	282.2	0.412	0.412	0.392	243.3	0.433	464.7	0.401	295.7	0.437	441.8
Poly5a	0.405	345.4	0.439	0.459	0.394	397.5	0.457	640.5	0.458	335.0	0.465	613.0
Poly2b	0.378	86.7	0.381	0.381	0.381	78.2	0.389	154.2	0.396	93.7	0.402	154.1
Poly3b	0.434	185.4	0.432	0.432	0.433	195.2	0.431	249.9	0.428	175.6	0.462	292.4
Poly4b	0.461	343.2	0.465	0.465	0.453	349.9	0.465	533.2	0.401	289.9	0.483	568.2
Poly5b	0.415	441.7	0.439	0.479	0.438	450.3	0.478	698.4	0.478	412.6	0.479	834.9
Swim	0.339	2405.1	0.351	0.351	0.342	2625.8	0.351	4143.9	0.402	2246.6	0.368	3688.6
Avg.	0.403	298.4	0.409	0.413	0.409	301.8	0.422	515.9	0.418	281.2	0.436	499.0

Table A.5: Results of the IJS1 algorithm for each nesting instances with $1.1d_{max}$ small bins

Values include the average of F values of 10 trials, t : average time for 500 iterations

	BL4		BL ∞		ML4		ML ∞		MU4		MU ∞	
	F	t	F	t	F	t	F	t	F	t	F	t
Albano	0.597	80.8	0.590	168.4	0.591	89.6	0.590	180.5	0.594	96.0	0.605	185.1
Shape2	0.300	85.1	0.300	121.9	0.293	58.0	0.300	134.5	0.300	61.9	0.300	133.5
Trousers	0.679	408.3	0.678	840.8	0.684	363.2	0.681	807.0	0.681	387.4	0.684	839.0
Shape0	0.244	296.4	0.244	560.2	0.244	295.0	0.244	654.5	0.244	292.4	0.244	616.5
Shape1	0.246	569.2	0.327	957.4	0.282	479.2	0.317	766.5	0.282	563.1	0.320	922.0
Shirts	0.601	829.7	0.590	1448.3	0.473	763.4	0.611	1377.7	0.465	680.1	0.611	1279.4
Dighe2	0.336	17.2	0.373	40.3	0.341	23.6	0.381	44.5	0.373	18.2	0.394	46.1
Dighe1	0.421	19.3	0.428	39.4	0.424	19.4	0.428	41.9	0.422	21.3	0.427	41.7
Fu	0.358	6.1	0.351	6.9	0.352	5.3	0.358	9.0	0.352	5.1	0.448	9.1
Han	0.437	105.9	0.430	206.9	0.447	99.3	0.445	241.0	0.461	90.7	0.446	168.0
Jakobs1	0.358	59.1	0.429	114.4	0.368	57.9	0.433	111.5	0.358	56.2	0.433	118.8
Jakobs2	0.365	48.5	0.415	102.8	0.401	58.1	0.415	97.8	0.401	58.8	0.418	101.7
Mao	0.437	120.3	0.450	229.1	0.447	131.1	0.450	205.5	0.449	138.5	0.472	190.6
Poly1a	0.300	25.9	0.402	44.2	0.310	27.3	0.451	49.4	0.309	29.5	0.454	47.6
Poly2a	0.349	104.7	0.353	131.4	0.352	78.9	0.363	134.3	0.367	80.2	0.452	150.7
Poly3a	0.384	185.6	0.368	320.1	0.373	154.7	0.382	326.3	0.367	148.5	0.451	304.1
Poly4a	0.402	285.0	0.349	489.7	0.384	267.7	0.351	520.5	0.383	337.1	0.409	437.3
Poly5a	0.382	335.0	0.364	578.5	0.367	437.3	0.399	704.5	0.358	355.1	0.413	692.7
Poly2b	0.371	95.4	0.412	145.7	0.37	80.5	0.412	161.9	0.339	100.3	0.419	154.1
Poly3b	0.373	207.6	0.384	292.0	0.381	218.6	0.452	247.4	0.378	172.1	0.456	304.1
Poly4b	0.406	332.9	0.407	551.5	0.407	356.9	0.418	549.2	0.405	292.8	0.428	562.5
Poly5b	0.397	503.6	0.422	766.4	0.419	490.8	0.422	726.4	0.415	416.8	0.435	935.1
Swim	0.330	2693.7	0.325	3999.4	0.326	2652.1	0.338	4019.5	0.338	2358.9	0.337	3688.6
Avg.	0.394	322.4	0.408	528.5	0.393	313.4	0.419	526.6	0.393	294.0	0.437	518.6