Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints

Wei Han

College of Information and Engineering, Nanjing University of Finance and Economics, Nanjing, China, 210046

Julia A. Bennell (corresponding author)

School of Management, University of Southampton, Southampton, SO17 1BJ, U.K., J.A.Bennell@soton.ac.uk, +442380595671

Xiaozhou Zhao

School of Management, University of Southampton, Southampton, U.K.

Xiang Song

School of Mathematics, University of Portsmouth, Portsmouth, U.K.

Abstract

The paper examines a new problem in the irregular packing literature that has many applications in industry: two-dimensional irregular (convex) bin packing with guillotine constraints. Due to the cutting process of certain materials, cuts are restricted to extend from one edge of the stock-sheet to another, called guillotine cutting. This constraint is common place in glass cutting and is an important constraint in two-dimensional cutting and packing problems. In the literature, various exact and approximate algorithms exist for finding the two dimensional cutting patterns that satisfy the guillotine cutting constraint. However, to the best of our knowledge, all of the algorithms are designed for solving rectangular cutting where cuts are orthogonal with the edges of the stock-sheet. In order to satisfy the guillotine cutting constraint using these approaches, when the pieces are non-rectangular, practitioners implement a two stage approach. First, pieces are enclosed within rectangle shapes and then the rectangles are packed. Clearly, imposing this condition is likely to lead to additional waste. This paper aims to generate guillotine-cutting layouts of irregular shapes using a number of strategies. The investigation compares three two-stage approaches: one approximates pieces by rectangles, the other two approximate pairs of pieces by rectangles using a cluster heuristic or phi-functions for optimal clustering. All three approaches use a competitive algorithm

for rectangle bin packing with guillotine constraints. Further, we design and implement a one-stage approach using an adaptive forest search algorithm. Experimental results show the one-stage strategy produces good solutions in less time over the two-stage approach.
Keywords: heuristic, cutting and packing, forest search, bin packing, irregular, phi-functions.

---

1. Introduction

There exist in the literature a high volume and variety of investigations into two-dimensional (2D) cutting and packing problems, which reflects the large application scope, such as ship building, shoe manufacturing, garment manufacturing and tool manufacturing, and a range of materials, for example glass, metal, wood, and textiles. Within these publications, a popular focus of research is generating cutting patterns that satisfying guillotine cutting constraints. These constrain any cut to begin at one edge of the stock sheet and continue in a straight line to another edge of the stock sheet. To the best of our knowledge, all of the algorithms are designed for solving 2D rectangular shape cutting problems where all cuts are orthogonal to the edges of the stock-sheet. Guillotine cutting with irregular pieces has not been tackled directly. In this problem, guillotine cuts are not constrained to be orthogonal to the rectangular stock sheet edges, and pieces can be continuously rotated.

An example of the irregular shape bin packing problem with guillotine constraints arises from the glass cutting industry and in particular the manufacture of conservatories (glass houses). Although many of the pieces are rectangular, there is a substantial number of irregular pieces. These are convex polygons with up to five sides in general, and occasionally more. It is common for conservatories to be a bespoke design (usually based on a standard style) to fit the specific building, hence, glass is cut to order. To satisfy the guillotine cutting constraints in practice, items of irregular shapes are individually, or in pairs, enclosed within rectangles and these rectangles are then arranged into a cutting pattern. This adds a restriction that is not present in practice, which is likely to create patterns with more waste than necessary.

In this paper, we implement four pattern generation strategies with the objective

of minimizing the number of bins required to pack all items. Primarily, we aim to investigate the benefit of generating cutting patterns which satisfy the guillotine cutting constraint by implementing the cuts directly on the irregular shapes instead of on rectangle enclosures. This one-stage approach is based on an efficient forest search algorithm. The forest constructs multiple layouts in parallel according to a dynamic measure of the quality of the partial layout. In order to benchmark our approach we implement a state of the art rectangle guillotine cutting algorithm of Charalambous and Fleszar [6] and generate solutions by approximating each piece by its minimum enclosing rectangle. Further, we attempt to improve on the two stage approach used in practice by using phi-functions to cluster all pairs of pieces in their minimum rectangle enclosure and use a greedy heuristic to select a subset to pack, again using the approach of Charalambous and Fleszar [6].

The contributions of this paper are many. We have brought a new problem to the literature that is found in practice. As a result, there is significant scope for further research. We have designed an efficient search heuristic using a dynamic solution evaluation function. The approach can handle continuous rotation of the pieces, multiple bins, and guillotine cuts. Irregular shape packing usually constrains the number of orientations of the pieces, approaches generally only pack a single strip and to our knowledge formulations have never included guillotine constraints. Further, the paper describes a second approach, based on industry practice, but using state of the art techniques to solve the problem by first optimizing the rectangle enclosure of pairs of pieces using phi-functions and then packing using a rectangle guillotine packing approach. This in itself is new to the literature. Finally, we have also introduced new benchmark data sets for this problem.

In the next section (section 2), we give a more detailed description of the problem. In section 3 we review some related literature on guillotine bin packing. Section 3 explains the one-stage approach, including some notation and definitions to describe the important characteristics of the problem, and details of a core function, best match, that forms the basis of the algorithm. It also includes a description of the forest search algorithm. In section 5, we describe the two-stage approach based on the work of Charalambous and Fleszar [6], minimum rectangle clustering based on phi-functions and our greedy selection. Section 6 contains the computational study and the discussion

of the results. Finally, in Section 7 we present the conclusions.

2. Problem Description

The problem objective is to cut all demand pieces from the minimum number of stock sheets possible, hence it is an input minimization problem. There are sufficient standard size rectangular stock-sheets available to meet demand, where the stock sheet has length L and width W. The demand set D contains N irregular shaped pieces, where each piece is considered to be unique and the demand of each piece is one. According to the typology proposed by Wäscher et al. [16] this is a single bin size bin packing problem (SBSBPP).

Further refinements to the problem type are that all pieces are convex, and usually irregular. Pieces can be rotated continuously i.e. there are no fixed rotation angles. Further, the stock sheet can be rotated. In principle this is taken care of by rotating the pieces. However, in Charalambous and Fleszar [6] the orientation of a non-square stock sheet is important. Only guillotine cuts are allowed. A guillotine cut is a single straight line cut that begins at an edge of the stock-sheet and ends at another edge. Unlike the vast majority of the literature, the cutting line is not constrained to be parallel to an edge of the stock-sheet. Often when considering guillotine constraints, pieces must be cut free from the stock sheet with a maximum number of cuts, which is typically three. In this problem there are no limits on the number cuts.

3. Literature Review

There are three key components of the problem under consideration: bin packing, guillotine cuts, and irregular shapes. To our knowledge there are no papers that tackle these three together. In addition, irregular shape packing literature is almost exclusively strip packing (single infinite length stock sheet) with a finite fixed set of rotation angles. Those who have tackled multiple stock sheet problems reduce the problem to a one-dimensional cutting stock problem using pre-defined pattern layouts, for example, Degraeve and Vandebroek [8]. A key challenge in packing irregular shapes is handling complex geometry, particularly when pieces contain concavities. In this paper all pieces are convex. Instead, the key challenge arises in modelling efficiently continuous rotation

of the pieces, which is not commonly dealt with in the literature. For a discussion of techniques for handling the geometry in irregular shape packing see Bennell and Oliveira [4]. Aside from the geometry, solution approaches to irregular packing are almost all heuristic and can be divided into those that build up to a final solution through sequentially adding to partial solutions, and those that work with complete solutions and search by making small changes to the incumbent solution. The latter approach can be subdivided into representing the solution by a sequence, or packing order, that is decoded by a construction heuristic, or by the co-ordinate positions of the pieces in the layout. For a review of solution approaches to the irregular packing problem see Bennell and Oliveira [3].

The rectangle bin packing problem with guillotine constraints has the most similarities to the problem we are tacking in this paper. Lodi et al. [12] survey two dimensional rectangle bin packing, including algorithms that handle guillotine constraints. They describe the one- and two-phase approach, where both consist of packing pieces onto shelves along the width of the bin. The former directly packs pieces into the bin, where as the latter optimizes the packing of the shelves into the bins by modelling as a one dimensional bin packing problem. Lodi et al. [10] creates the shelves by solving a series of 0-1 knapsack problems improving on the performance of the finite first fit and finite best strip heuristics of Berkey and Wang [5]. More recent construction heuristics are not constrained to creating shelves. Charalambous and Fleszar [6] start by generating simple patterns, initially across the width of the bin, and subsequently within free rectangle areas. Pieces may shift horizontally or vertically in order to maximize the size of the free rectangle, while maintaining the guillotine constraint. Fleszar [9] propose a constructive heuristic where the insertion decision is made by first-fit, best-fit or critical-fit criteria. Patterns are generated using a tree structure where nodes determine the cuts and the leaf nodes are pieces. Further improvement is made by a justification heuristic. Polyakovsky and M'Hallah [13] modify the well know bottom left construction heuristic to meet guillotine constraints. After placing each piece, they apply both horizontal and vertical guillotine cuts and select the one that gives the largest rectangle area available for packing. Pieces are assigned to bins using an agent-based algorithm. Pieces may be agent-initiators attracting individual-agents (pieces) to their group in order to maximize the fitness of the group. Individual-agents compete to join groups to maximize their

purpose parameters. These groups are assigned to the same bin and arranged using the guillotine bottom left heuristic. Lodi et al. [11] uses tabu search to assign pieces to bins. Initially, one piece is packed in each bin. The heuristic attempts to empty weak bins by assigning pieces to sub-instances that include the pieces from k bins. Instead of assigning pieces to bins and then packing, Alvelos et al. [1] define a sequence for packing the pieces while keeping a list of candidate locations for the next piece. The solution is improved using variable neighborhood descent where moves are made within the packing sequence. Although none of these papers directly apply to the problem addressed in this paper, we have made use of some core knowledge. We adopt methodologies for efficiently processing the geometry of irregular convex shapes with free rotation using some classic concepts and phi-functions. We follow the common theme of construction methodologies for generating patterns while meeting guillotine constraints, and for the two-stage strategies we directly use the approach of Charalambous and Fleszar [6].

4. One-stage approach

Our one-stage solution approach is a constructive heuristic where the algorithm begins with generating efficient clusters of polygons, and then uses these clusters to construct the bin packing solution. In the following sections we first define the process and criteria for clustering two polygons, this is called a match. A match is accepted as a node in the search forest if the utilization ratio meets or exceeds a certain acceptance threshold θ. The forest is populated by matching polygons with polygons, polygons with clusters, and clusters with clusters. The latter leading to the term forest search. These clusters are called blocks. The forest is complete once there are no further matches for the blocks, this may be as a result of the boundary constraint of the stock sheet or there are no matches that meet the threshold. The bin packing step sequentially selects the block with the greatest summed area of polygons, while removing any other blocks from the forest that contain common polygons. If no further blocks can be placed in a bin and the utilization ratio is below a given threshold θ_u, the forest is recreated with a lower acceptance threshold. Eventually the threshold is set to zero so all polygons are packed.

4.1. Best match of two convex polygons

In this section we describe a core function of the overall methodology, the best match of two convex polygons. For the purposes of this section, we describe the approach in the context of the original convex polygons. Later we will generalize the procedure and associated definitions for clusters of polygons.

4.1.1. Notation and definitions

A convex polygon, P, can be expressed by a set of vertices $(p_1, p_2, ..., p_n)$, where n is the number of vertices of the convex polygon, and each vertex p_i is defined by cartesian co-ordinates (x_i, y_i) where $i =, 1, ..., n$. p_1 is set as the origin of P. The ith edge can be expressed by $e_i = (p_i, p_{i+1})$ where $i = 1, ..., n-1$, and the nth edge is $e_n = (p_n, p_1)$. P is convex if all vertices lie on, or to one side of, the infinite line concurrent with e_i, for each e_i. Let $INT(P)$ be the interior of P and $FR(P)$ be the edges, or frontier, of P. Let set $D = \{P_i \mid i = 1, ..., N\}$ be customer demand, where N is the total number of pieces ordered. P_i is called a basic polygon and has a demand of one.

Clearly because of the guillotine constraint, only convex shapes can be cut. When combining two basic items, the resulting union may not be convex. Hence, two useful convex approximations of the union are the convex hull and enclosing rectangle. Given a point set S, let $O(S)$ be the convex hull of S and $R(S)$ be the minimum area enclosing rectangle of S. For convex polygon P, $O(P) = P$.

4.1.2. Transformation and overlap

A polygon can be transformed by three operations: reflection (mirror), translation and rotation. Let f be a transformation of polygon P, which can be expressed by a four element group $\{m, x, y, t\}$. $m \in \{0, 1\}$ is a reflection transformation, where the values 1 and 0 represent reflection and no reflection respectively. Note that you only need to reflect in one axis and all other reflections arise from rotating the reflected polygon. x, y represent the translation distance along the x-axis and y-axis. $t \in (-\pi, \pi]$ is the rotation angle of the convex polygon around the origin p_1. Obviously, the transformation space, Ψ, is the composition of reflection, translation and rotation. We also call $f_i(P_i)$ the transform of P_i.

Since it is possible to change the position, orientation and reflection of a polygon, it is important to define the combinations of transformations of multiple polygons that are

feasible with respect to mutual overlap. Hence, the set of non-overlapping transformations is defined in equation 1. A non-overlapping configuration of two polygons is called a *match*.

$$f(P_i, P_j) = f_i(P_i) \bigcup f_j(P_j) \quad \text{and} \quad INT(f_i(P_i)) \bigcap INT(f_j(P_j)) = \emptyset \tag{1}$$

4.1.3. Definition of best match

In order to construct the cutting pattern, we must decide where to place each polygon relative to the other polygons. This amounts to choosing a match between two polygons. Given we wish to minimize waste, we only consider positions where the two polygons touch. A match that gives the minimum convex hull would arguably be a good choice. Given the stock sheets are rectangular, then the minimum area enclosing rectangle would also be a sensible measure and favor configurations that fit the boundary of the placement area. Hence, we want to evaluate both these contributions to waste. Given polygons $P_1, P_2 \in D$, the convex hull and rectangle enclosure of the match $f(P_1, P_2)$ are denoted as $O(f(P_1, P_2))$ and $R(f(P_1, P_2))$ respectively, see figure 1. For consistency with other measures of solution quality, we define the utilization of the convex hull and rectangle enclosure of $f(P_1, P_2)$ using equations 2 and 3, where $Area(.)$ is the area of the polygon.

$$U_{cov}^{f(P_1,P_2)} = \frac{Area(P_1) + Area(P_2)}{Area(O(f(P_1, P_2)))} \tag{2}$$

$$U_{rec}^{f(P_1,P_2)} = \frac{Area(O(f(P_1, P_2)))}{Area(R(f(P_1, P_2)))} \tag{3}$$

Figure 1: The waste contributions of the convex hull and enclosing rectangle.

Equation 2 measures the ratio of piece area and convex hull area, if high then the match is tight. Equation 3 measures the ratio of convex hull area and rectangle enclosure area, if high then the match generates a rectangle shape, which fits our global objective. Since both attributes are desirable, we define a weighted sum of these measures of

utilization. Given the weight $w \in [0, 1]$, the weighted utilization ratio of the match $f(P_1, P_2)$ is given in equation 4.

$$U_w^{f(P_1,P_2)} = wU_{rec}^{f(P_1,P_2)} + (1 - w)U_{cov}^{f(P_1,P_2)} \tag{4}$$

It is unlikely that there is one ideal value of w across all data instance. Note that during the earlier stage of the search, tight combinations of polygons is desirable with little concern for creating rectangular shaped clusters. A heavier weight on the convex hull ratio would achieve this. While in the later stage of the search, achieving a rectangular cluster in order to not incur large amounts of waste between the convex hull of the cluster and the edges of the stock sheet is more important. A heavier weight on the rectangular enclosure would encourage this sort of configuration. In our experiments we evaluate a number of different fixed and dynamic weighting strategies.

4.1.4. Heuristic method to find the best match

The heuristic procedure for finding the best match of two polygons uses three main operations: Mirror, Attach and Slide, these are defined as follows:

$Mirror(P_l, m_l)$, $l = 1, 2$ and $m_l = 0, 1$. If $m_l = 1$ reflect P_l

$Attach(P_1, P_2, i, j)$. Let P_1 be the fixed polygon with counter clockwise direction and P_2 be the sliding polygon with clockwise direction. The attach operation between P_1 and P_2 is as follows: move polygon P_2 so that the j-th convex vertex on P_2 coincides with the i-th convex vertex on P_1. Rotate P_2 so that the j-th edge of polygon P_2 coincides with the i-th edge of polygon P_1. Recall that the k-th edge of a polygon is between vertices $(k, k + 1)$. Since P_1 and P_2 have opposite orientation and are both convex, the attach procedure will not result in the polygons overlapping.

$Slide(P_1, P_2, i, j, d)$. After the attach operation, slide the j-th point on P_2 along the i-th edge of polygon P_1. Let d be the slide distance, then $d \leq max\{0, dis(e_i) - dis(e_j)\}$, where $dis(e_j)$ is the length of edge e_j. Each call of $slide(P_1, P_2, i, j, d)$ will slide polygon P_2 an additional distance ϵ. As long as we consider all combinations of $Mirror(P_l, m_l)$, then all incremental slide points are captured. An illustration of an attached pair and the slide operation is given in figure 2, where three candidate positions are shown including the first and last positions.

Figure 2: Attach and slide operators.

Since only the relative positions of the two polygons are of interest, we can find all matches by rotating and translating just one of the polygons. Before sliding along a certain edge attachment, we test the potential of that match on the given edge combination, called the *match degree*, $md(i, j)$. Only if the match degree is greater than a threshold parameter, θ_d, will the algorithm proceed to the attach and slide operation. The match degree of the attachment operation $Attach(P_1, P_2, i, j)$ is defined in equation 5.

$$md(i, j) = 1 - \frac{\mid dis(e_i) - dis(e_j) \mid}{max\{dis(e_i), dis(e_j)\}} \tag{5}$$

The match degree is high for edges that have similar lengths and low for edges that have very different lengths. Note that if the match degree does not exceed θ_d then the edge combination is rejected. If all edge combinations are rejected then the two polygons are not matched. The best match procedure is given in algorithm 1.

4.2. Forest search

In this section we describe the definitions, functions and procedures to generate the forest search. Some of the notation and definitions are analogous to those described in the previous section. A key progression is that the operations are performed with clusters of polygons rather than the original basic polygons.

4.2.1. Notation and definitions

Let set T be the subset of the demand set D, so $T = \{P_1, P_2, ..., P_t\} \subseteq D$. Let $f(T) = f_1(P_1) \bigcup f_2(P_2) \bigcup ... \bigcup f_t(P_t)$ be a transformation of set T, where $f_i(P_i), i = 1, ..., t$, is a transformation of polygon P_i. All the transformations on T form the *transformation space*, denoted by Ψ_T. The convex hull of $f(T)$ is $O(f(T))$. A uniform transformation of T is when $f_i = f_j$, for all $i, j \in [1, t]$.

4.2.2. Feasible transformations

In order for the transformation of set T to be feasible, $f(T)$ must satisfy two conditions: no pair of polygons may overlap and every polygon can be removed from the stock sheet using guillotine cuts.

Algorithm 1 Best match

Input: Polygon $P_1, P_2, \theta_d, w, \epsilon$

Initialize $MaxU = 0$, $match = 0$

for each pair of vertices i on P_1 and j on P_2 **do**

  if $md(i,j) > \theta_d$ **then**

    $match = 1$

    Create copies of polygons $P_1' \leftarrow P_1, P_2' \leftarrow P_2$

    for $m_1 = 0, 1$ **do**

      $Mirror(P_1', m_1)$.

      for $m_2 = 0, 1$ **do**

        $Mirror(P_2', m_2)$.

        for $d = 0, max\{0, dis(e_i) - dis(e_j)\}$, step ϵ **do**

          $Attach(P_1', P_2', i, j)$

          $Slide(P_1', P_2', i, j, d)$

          if $U_w^{f(P_1', P_2')} > MaxU$ **then**

            $MaxU \leftarrow U_w^{f(P_1', P_2')}$, $best(m_1) \leftarrow m_1$, $best(m_2) \leftarrow m_2$, $best(i) \leftarrow i$, $best(j) \leftarrow j$, $best(d) \leftarrow d$

          end if

        end for

      end for

    end for

  end if

end for

if $match = 0$ **then**

  Return $P_1 \bigcup P_2 = \emptyset$

end if

$Mirror(P_1, best(m_1)), Mirror(P_2, best(m_1))$

$Attach(P_1, P_2, best(i), best(j)), Slide(P_1, P_2, best(i), best(j), best(d))$

Return $P_1 \bigcup P_2$

Proposition 1: Algorithm 1 produces non-overlapping configurations of pieces.

Proof: The attach and slide operators only generate non-overlapping configurations of P_1 and P_2. The convex hull of the best match from algorithm 1 is $O(P_1 \bigcup f(P_2))$. Let $P_1^* = O(P_1 \bigcup f(P_2))$ and P_2^* be the next polygon to be matched, then the matching of the three polygons will meet the no-overlap constraint. This is also true if P_2^* is the convex hull of a cluster of polygons. Provided algorithm 1 is matching convex polygons, or convex hulls of clusters of polygons, then the no-overlap constraint holds.

Proposition 2: Algorithm 1 produces guillotine-able configurations of pieces.

Proof: Let P_1 and P_2 be convex polygons and $O(P_1 \bigcup f(P_2))$ be the convex hull of the optimal match, where the match arises from edge e_i from P_1 and e_j from P_2 coinciding. Let $\vec{o}$ be the infinite line coinciding with e_i and e_j, then $\vec{o} \supset FR(P_1)$ and $\vec{o} \supset FR(P_2)$. Since P_1 and P_2 are convex then $\vec{o} \bigcap INT(P_i) = \emptyset$ for $i = 1, 2$ then we know that $\vec{o}$ is a feasible guillotine cut for P_1 and P_2 . Guillotine cuts are defined in reverse order, i.e. the first match defines the last cut. In general, $O(f(T))$ is called guillotine-able if one of the following two conditions is satisfied:

1. T only contains two convex basic polygons P_1 and P_2.
2. There exists one guillotine segment in $O(f(T))$, which divide T into two subsets T_1 and T_2, and T_1 and T_2 are guillotine-able.

A non-overlapping guillotine-able transformation is called a feasible transformation.

4.2.3. Blocks

Blocks, B_i, populate the forest. In order for subsets of polygons to appear in the forest, they must be part of a valid block. A valid block is the convex hull of a feasible transformation of T with a utilization ratio of at least θ, where θ is the acceptance threshold for blocks to appear in the forest. It can be defined by equation 6

$$B_i = \{O(f(T_i)) \mid U_w^{f(T_i)} \geq \theta, f \text{is a feasible transformation}\} \tag{6}$$

Equation 6 is a necessary condition to define a block but it does not fully describe its composition. A block may be made up of several groups of pieces that have been matched and approximated by their convex hull. Figure 3 shows a small portion of the forest and illustrates the creation and composition of blocks through the levels of the

search. At the top level the forest contains all basic items and each basic item is a block. The second level contains pairs of basic items in the configuration that gives the best U_w, these are the level two blocks. Level three adds a basic item to the level two blocks. Level four contains a level three block combined with a basic item, but could also contain two level two blocks. Level five is empty in this example, but could contain combinations from level four and one or level two and three. Finally level six contains all basic items made up from a block from levels two and four.

Figure 3: Example of a portion of the search forest.

In general, at the first level of the forest, $m = 1$, a block is a basic polygon. Applying algorithm 1 directly to the basic polygons generates all candidate feasible transformations for $m = 2$. The convex hull of each feasible transformation becomes a block in the second level if the weighted utilization ratio is at least θ. Note that the level refers to the number of basic polygons in the subset. Hence, $m = 3$ would match a block from level one and level two, using algorithm 1 and so on. Hence, the convex hull of a feasible transform of subset T is a block, B, if one of the following two conditions is satisfied.

1. B contains only one element.
2. B can be divided into two subsets B_1, B_2, and the weighted utilization of B is greater than θ, and B_1, B_2 are valid blocks.

In order to control the size of the forest, we only accept a new block in the forest if the weighted utilization ratio meets, or exceeds, an acceptance threshold θ. The utilization ratios and weighted utilization are calculated using equations 2, 3 and 4, where P_1 is replaced by B_1 and P_2 is replaced by B_2.

As discussed earlier, the weighted utilization ratio represents the two aspirations of tight packing and an overall rectangular layout to fit the stock sheet area. Initially the tightness of the packing is more important. As the layout grows closer to the stock sheet size, the rectangular shape is more important. As a result, we define a number of alternative weighting schemes, both dynamic and fixed.

There are three fixed weighting schemes that set $w = \{0, 0.5, 1.0\}$ for the entire search process. The dynamic weighting schemes increases the value of w for each generation,

m, where $m = 1, ..., G$, and G is the maximum number of generations. We use the following S shape function to manage the transition of the weights.

$$w_m = 1 - \frac{1}{1 + e^{\frac{2m-G}{2K}}} \tag{7}$$

K controls the linearity of the function. When $K = 1$, the range of the exponential component is from almost zero to very large, hence the graph of w_m is a sharp S shape function from zero to one. When K is large w_m is approximately linear. Also note that at half the maximum generations $w_m = 0.5$. In our experiments we use $K = \{3, 5, 15\}$ as illustrated in figure 4.

4.2.4. Forest construction

Figure 4: Plot of the dynamic weighting scheme.

The generation of the forest naturally extends from the definition of a block, and in particular the concept of a block as a tree, where all the trees make up the forest. Note that branches of blocks may be shared. Equation 8 provides a mathematical description of each level of the forest. $g_m^{\theta,w}$ where m is the level, θ is the acceptance threshold and w is the weight applied to the utilization ratio. Level 1 is simply the basic polygons in the demand set D. Subsequent levels are θ acceptable matches of blocks from the previous level.

$$g_m^{\theta,w} = \begin{cases} P_i, i \in D & m = 1 \\ \bigcup_{i+j=m} \{f(B_i, B_j) \mid B_i \in g_i^{\theta,w}, B_j \in g_j^{\theta,w}, U_w^{f_w(B_i,B_j)} \leq \theta, T_i \bigcap T_j = \emptyset\} & m > 1 \end{cases} \tag{8}$$

In addition to the acceptance threshold on weighted utilization, there are two further constraints on the acceptability of a block. First, only one of each piece type can appear across all the patterns. Before matching blocks, the procedure performs a conflict check for common pieces. Hence, a match between B_i and B_j is made only if $T_i \bigcap T_j = \emptyset$. Second, each new block must not violate the dimensions of the stock sheet. Hence, we need to check if the length and width of the block is larger than those of the rectangle stock sheet. Note that the entire block can be freely rotated and a block may exceed

the boundaries of the stock sheet in one orientation and not in another. Since the minimum area enclosing rectangle will have an edge collinear to the edge of the block (Toussaint [15], then the worst case number of tests equals the number of edges of the block. However, since we only need to satisfy this constraint, the number of tests may be many fewer. Let $l(R(B)), w(R(B))$ be the length and width of the candidate rectangle enclosure of block B, where the rectangle has at least one edge collinear with the block, then $l(R(B)) \leq L, w(R(B)) \leq W$ must hold for one of the candidates. Algorithm 2 details the procedure to generate a forest.

Algorithm 2 Forest Construction

Input: G, D, K, θ
Initialize $g_m^\theta \leftarrow \emptyset$ for all m. Set $m = 1$ and calculate w_m according to K and G
$g_1^\theta \leftarrow D$
for $L = 2, ..., m$ **do**
  for $i = 1, ..., \lfloor \frac{L}{2} \rfloor$ **do**
    For each block $B_x \in g_L^\theta$ combined with each block $B_y \in g_{L-i}^\theta$
    if $T_x \bigcap T_y = \emptyset$ **then**
      construct $f(B_x, B_y)$ using algorithm 1
      if $U_{w_L}^{f(B_x,B_y)} > \theta$ **then**
        $g_L^\theta \leftarrow g_L^\theta \bigcup f(B_x, B_y)$
      end if
    end if
  end for
end for

4.3. Bin packing

The final step of the approach is to pack the blocks that populate the forest into the bins. All blocks will satisfy the acceptance threshold, hence in this step we seek to place blocks on stock sheets as efficiently as possible. Clearly, packing blocks into bins will generate more waste between the edges of the stock sheet and the blocks, and between adjacent blocks. A utilization threshold, θ_u, determines whether a bin packing pattern is acceptable. Once the newly generated patterns are no longer acceptable, the approach reduces both the acceptance threshold (θ) and θ_u and generates a new forest

with the remaining unpacked pieces, repeating the bin packing procedure with the lower utilization threshold. Eventually, both thresholds are set to zero to ensure all pieces are packed. The procedure has a number of operations as follows:

Recursive fill (RF): select the block with the largest area that will fit into a given size stock sheet/partial stock sheet, place the block at the top left corner. Note that a block can come from any part of the forest. Mark that block, and any other block containing common pieces, as used.

Single bin (SB): take a new stock sheet and call RF. Horizontal and vertical guillotine cuts divide the unpacked areas into S1 and S2. Figure 5 shows the two possible ways of generating these partial stock sheets, we generate both. Call RF for each partial stock sheet and select the best. Continue this process until no further blocks can be packed.

Bin packing (BP): generate the forest given the input data, acceptance threshold and weighting scheme. θ_u initially is set to 0.8. Repeatedly call SB, and accept a pattern if the stock sheet utilization is greater than θ_u. Otherwise reduce θ to 0.9 and θ_u to 0.7, generate a new forest with the remaining pieces and call SB as before. The third and final forest generation sets θ to 0.8 and θ_u to zero.

Note that initially we expect to fill a stock sheet with a single block, but it is highly unlikely that all stock sheets can be filled this way. In order to control computation time, the procedure generates at most three forests at reducing values of θ. After the third forest, set $\theta_u = 0$ so all pieces are packed, potentially individually as the forest may not accept any matches. Since θ reduces the size of the forest, it should not be too small, however, the decrease must be sufficient to generate useful size blocks. Algorithm 3 gives a summary of the one stage procedure.

Figure 5: The two arrangement of cuts to generate partial stock sheets.

Algorithm 3 One-stage procedure

1: *Input:* $G, D, K, W, L, \theta, \theta_d, \theta_u, \epsilon, k = 1$
2: Generate forest using algorithm 2
3: Pack single bin, BIN_k
4: **until** no further blocks can be packed, call recursive fill
5: **if** stock sheet utilization $\geq \theta_u$ **then**
6: go to 3
7: **end if**
8: **if** remaining blocks contain unused pieces **then**
9: Reduce θ_u and θ and go to 2
10: **end if**
11: Return $\text{BIN}_i, i = 1, ..., k$

5. Two-stage approach

In order to benchmark our approach we implement a two-stage procedure. The first stage encloses individual or pairs of pieces into rectangles, the second stage packs the rectangles. For the former we investigate two approaches. One clusters the pieces using algorithm 1, described earlier. The other uses a state of the art convex polygon clustering approach using phi-functions presented by Scheithauer et al. [14]. See Bennell et al. [2] for a discussion of phi-functions. For the latter we use the recently published guillotine bin packing approach of Charalambous and Fleszar [6].

5.1. Rectangle bin packing with guillotine cuts

The approach is directly taken from Charalambous and Fleszar [6] and therefore only briefly described here.

The fundamental building block of the approach is a simple pattern generator that arranges a subset of pieces side by side. Items that are available to be packed are sorted in non-increasing order of the weighted sum of their normalized height and area and then patterns are generated using the well-known first fit rule. Rotation is taken care of by including two copies of each piece in each orientation, if one copy is used the other becomes unavailable. They generate a number of alterative simple patterns by varying the weights, and identify those that satisfying a sufficiency criterion, which reduces the

greediness of the approach. From the identified patterns, they select the pattern with the maximum total area of items, if no patterns satisfy the sufficiency criteria, they select the pattern that violates it the least. Clearly the simple pattern must fit within the available rectangle. Initially this rectangle is the size of the bin, but subsequent calls to the generator will be for smaller empty rectangle areas remaining in the bin.

The procedure generates a simple pattern and places it in the bottom left corner of the bin, this is the committed pattern. Then it identifies multiple empty overlapping rectangles that are all above the simple pattern, removing any rectangle areas that are too small for any available item. The simple pattern generator fills each rectangle and selects the best when combined with the current committed pattern. Following sets of overlapping empty rectangles may appear above, below or to the left of the last committed pattern and are filled in that order. While maintaining guillotine cuts, these rectangles are expanded to the maximum size by shifting sets of pieces in the pattern vertically or horizontally. Once no further pieces can be added to the pattern, the procedure begins again on the next bin until all items are packed. If the stock sheet is not square, the complete procedure is repeated with the stock sheet rotated by 90 degrees. The authors suggest a further improvement to the constructive heuristic, which involves varying the strength of the sufficiency criteria using a bias.

5.2. Clustering approach

The basic items are approximated by their enclosing rectangle individually and in pairs. The minimum rectangle enclosure of a single piece is straightforward to find given it will have one edge colinear with the edge of the piece (Toussaint [15]). The maximum number of edges in our data sets is five, hence complete enumeration is quick and simple. The minimum area enclosing rectangle for a pair of pieces, where the pieces can be freely rotated is non-trivial. For one variant we cluster two pieces using algorithm 1. For the other variant, we use the phi-function implementation of Scheithauer et al. [14].

The phi-function procedure is complex and only briefly outlined here. Given two convex polygons, find the phi-function for the pair of polygons with free rotation and the phi function for each polygon with free rotation and the complement of a rectangle with variable length and width. See Chernov et al. [7] for details of this procedure. These phi-functions are deconstructed into phi-trees where the terminal nodes corre-

spond to systems of nonlinear inequalities, each focusing on a subset of configurations of the polygons. Each set of inequalities are solved to minimize the size of the enclosing rectangle using IPOPT and the best is selected. Since there are multiple local optima for each set of inequalities, we define many starting solutions. Note that if the polygons have fixed orientation, the procedure defines a linear set of inequalities that can be solved to optimality.

5.3. Greedy selection

The clustering approach generates a complete set of candidate rectangles that include multiple copies of each piece. In order to determine the set of rectangles to use as input to the guillotine packing algorithm, we apply a greedy selection. This sorts all the enclosing rectangle clusters, individual and pairs, from minimum to maximum waste, where waste is defined by equation 9.

$$W_k = Area(R(f(P_1, P_2))) - (Area(P_1) + Area(P_2)) \tag{9}$$

In the case of an individual piece, P_2 is an empty set. The heuristic selects the first rectangle on the sorted list, removes any rectangles that contain common piece(s), selects the next available rectangle on the sorted list, and so on until the list is empty. At this point all basic items will appear once in the rectangle clusters.

6. Experiments and results

In this section we provide details of the results for both the one-stage and two-stage approach. For the one-stage approaches, we investigate two initial settings for the acceptance threshold (θ). The two-stage approach includes three sets of experiments: two variants that allow pieces to be clustered in pairs and one variant that only enclose individual pieces in rectangles. Experiments were programmed using C++ and Java and run on PC with 2.4GHz and 2G memory.

6.1. Data

Table 1 provides details of the test data to be packed on stock sheets of size 3210 by 2250. Sets coded J are taken from real industrial data provided by a company specializing in glass cutting for conservatories. Sets coded H are generated using the

Table 1: Test datasets

Dataset	Ave. no. edges	Stdev. edges	Ave. area	Stdev. area	Irregular degree
J40	3.56	0.741	1070889	864460	0.2741
J50	3.70	0.647	1104653	825371	0.3416
J60	3.73	0.607	1041775	791634	0.2986
J70	3.77	0.569	1018279	782675	0.2578
H80	3.67	0.508	727813	622035	0.2457
H100	3.83	0.493	968581	739522	0.2520
H120	3.61	0.562	819777	732018	0.3142
H149	3.82	0.695	932110	813401	0.2667

properties of the industrial data. The number indicates the number of pieces in each data set. Recall each piece is considered unique. The table details the average number of edges, standard deviation of edges, the average area and the standard deviation of the area. The final column indicates the degree of irregularity of the data set found by equation 10. These data sets are available on the European Working Group in Cutting and Packing (ESICUP) website.

$$\text{Irregular degree} = \frac{1}{n}\sum_{P\in D}(1 - \frac{Area(P)}{R(P)}) \tag{10}$$

6.2. Results

The tables 2 and 3 detail the results for the two approaches. The main body of the tables include the following information for each data set and variant: total number of bins to pack all pieces (N), stock sheet utilization (U), and the fractional number of bins used (F). The utilization is calculated by equation 11.

$$U = \frac{\sum_{i=1}^{n} Area(P_i)}{((N-1)\times L \times W) + R^*} \tag{11}$$

Where R^* is the rectangle area of the stock sheet used once the reusable residual has been removed by either a complete horizontal or vertical guillotine cut, depending on which gives the largest reusable rectangle piece. This measure of utilization is helpful in

differentiating the quality of competing methods when they produce solutions with the same number of bins. The fractional number of bins is calculated as $(N-1)$ plus the proportion of the final bin used once the reusable residual is removed.

For table 2, the results refer to two starting acceptance thresholds θ and for each there are six weighting schemes for U_w. The first three are constant where $w = 0$ will focus solely on the convex enclosure and $w = 1$ will focus solely on the rectangle enclosure. The former will report lower waste for the same match, so more solutions will be accepted under this measure than the rectangle enclosure measures. This gives greater opportunity to find blocks but will result in longer run times (see table 4). $w = 0.5$ gives equal weight to each. The second three schemes change the weights as the search moves through the levels of the forest, starting with a focus on convex enclosure, corresponding to small w, and shifting the focus to rectangle enclosure later in the search, corresponding to large w. The dynamic weights are linear, $K = 3$, which has the steepest sigmoid curve, and $K = 5$, which is less steep. These will accept fewer blocks than $w = 0$, but theoretically the accepted blocks will be more suitable for the final bin packing. This is born out in the results where the better results arise from the dynamic weighting schemes in general. Further, the S shaped weighting schemes consistently out perform the linear weighting scheme, where the less steep function found with $K = 5$ does better for a lower initial θ and the steepest function, $K = 3$, does better for the higher θ.

Table 3 retains the results for $\theta = 0.94$ and $K = 5$, and $\theta = 0.97$ and $K = 3$, and compares them with the two-stage approach where rectangles are clustered individually (single) and in pairs using algorithm 1 (Alg 1) and phi-functions (Phi fn). Clearly, clustering individually consistently produces inferior solutions than all the other approaches. Clustering in pairs using phi-functions performs better than algorithm 1 for all the data sets. The benefit of phi-functions reduces as the data sets grow in size with 4.8% improvement in utilization for small data sets and 1.6 % for the largest. Clustering in pairs (using phi-functions) and the one-stage approach do similarly well, with one-stage doing better on five instances and two-stage doing better on three. The one-stage strategy packs the beginning stock-sheets very well but has a weak tail, where as two-stage will perform more consistently throughout. A key drawback of the two-stage approach using phi-functions is the computational time. The pairing process takes just over an hour for

Table 2: Results of the one-step approach

		$\theta = 0.94$						$\theta = 0.97$					
		$w = 0$	$w = 0.5$	$w = 1$	linear	$K = 3$	$K = 5$	$w = 0$	$w = 0.5$	$w = 1$	linear	$K = 3$	$K = 5$
	N	9	8	8	8	8	8	8	8	8	8	8	9
J40	U	0.686	0.816	0.801	0.781	0.821	0.821	0.819	0.787	0.791	0.804	0.83	0.727
	F	8.85	7.44	7.58	7.77	7.39	7.39	7.41	7.72	7.68	7.55	7.32	8.35
	N	10	9	10	9	9	9	10	9	10	9	9	10
J50	U	0.797	0.873	0.776	0.902	0.913	0.894	0.776	0.893	0.782	0.885	0.904	0.808
	F	9.58	8.75	9.85	8.47	8.37	8.55	9.85	8.56	9.78	8.63	8.45	9.45
	N	12	11	11	11	11	11	11	11	12	12	11	11
J60	U	0.735	0.828	0.798	0.812	0.82	0.845	0.807	0.814	0.752	0.761	0.823	0.802
	F	11.77	10.44	10.84	10.65	10.54	10.23	10.72	10.62	11.5	11.36	10.51	10.78
	N	14	13	14	14	13	13	13	13	14	14	13	14
J70	U	0.743	0.778	0.737	0.737	0.778	0.785	0.782	0.784	0.721	0.729	0.791	0.746
	F	13.47	12.86	13.58	13.58	12.87	12.75	12.8	12.76	13.88	13.74	12.65	13.41
	N	11	10	10	10	10	10	10	10	10	11	10	11
H80	U	0.785	0.863	0.843	0.846	0.835	0.881	0.839	0.865	0.853	0.807	0.863	0.799
	F	10.38	9.45	9.67	9.63	9.76	9.25	9.71	9.42	9.56	10.1	9.45	10.2
	N	18	17	18	17	17	17	17	17	17	17	17	18
H100	U	0.768	0.805	0.765	0.818	0.814	0.821	0.801	0.809	0.806	0.801	0.819	0.777
	F	17.44	16.64	17.52	16.38	16.45	16.31	16.72	16.56	16.62	16.72	16.35	17.24
	N	18	17	17	17	17	17	18	17	19	17	17	18
H120	U	0.777	0.834	0.82	0.827	0.834	0.844	0.79	0.834	0.752	0.791	0.827	0.799
	F	17.65	16.44	16.73	16.58	16.44	16.25	17.36	16.44	18.24	17.33	16.58	17.17
	N	24	23	24	23	23	23	23	23	23	23	23	23
H149	U	0.822	0.843	0.819	0.847	0.86	0.86	0.855	0.845	0.85	0.851	0.857	0.842
	F	23.36	22.77	23.46	22.68	22.33	22.33	22.47	22.72	22.6	22.58	22.41	22.8

the smallest data set (40 pieces) and over eighteen hours for the largest (149 pieces), although the packing algorithm of Charalambous and Fleszar [6] takes only one or two seconds for all data sets. Pairing using algorithm 1 also has negligible computational times (less than a second for all data sets). The computational times for the one-stage approach are in table 4. The weighting scheme that emphasizes convex hull will accept more matches and lead to more branches, and as a result take longer to run. Also a lower initial θ will accept more matches. A lower θ will provide more opportunity to find better solutions, but the benefit of this is not clear cut in these results. Clearly there are many parameters that can be varied in the one-stage approach, in particular the acceptance thresholds at various points in the algorithm which control the relationship between run time and scope of the search. For $\theta = 0.94$, the number of nodes is just over 10000 for the largest data set and the average is approximately 5000. Increasing θ to 0.97 reduces the average to around 1000.

7. Conclusions

The paper addresses the irregular shape guillotine bin packing problem, which is found in the glass cutting industry, specifically for this paper, in the manufacture of conservatories. To the authors' best knowledge, this problem has not been addressed in the literature before. We propose a forest tree search algorithm to solve this problem approximately, which is novel in the literature. First, we develop a procedure to find the best match of any two given convex shapes, which is evaluated using a newly derived function that includes two measures: how tight the packing is and how well it approximates to the rectangular stock sheet. The emphasis between these measures is dynamic through the search. Secondly, we construct search forest by selecting only those blocks with the utilization ratio function larger than a given value. Since this is a new problem in the literature, we can not benchmark our results against previous work. Instead, we implement a second approach, called two-stage. This clusters all possible individuals and pairs into an enclosing rectangle, greedily selects a subset of rectangle enclosures so that all pieces are represented once, then generates the bin packing layout using a recently published guillotine cutting heuristic. The forest search and two-stage, using phi-functions for pairing, perform similarly well, but the forest search is significantly faster. A fast two-stage approach uses an alternative heuristic for pairing and

Table 3: Results of one-step and two-step approach

		Two-step			One-step	
			Alg 1	Phi fn	$\theta = 0.94$	$\theta = 0.97$
		Single	Pair	Pair	$K = 5$	$K = 3$
	N	11	9	8	8	8
J40	U	0.593	0.718	0.787	0.821	0.83
	F	10.26	8.47	7.72	7.39	7.32
	N	13	11	10	9	9
J50	U	0.608	0.706	0.792	0.894	0.904
	F	12.57	10.84	9.66	8.55	8.45
	N	14	12	11	11	11
J60	U	0.628	0.746	0.794	0.845	0.823
	F	13.78	11.59	10.89	10.23	10.51
	N	15	13	12	13	13
J70	U	0.676	0.777	0.825	0.785	0.791
	F	14.59	12.70	11.95	12.75	12.65
	N	12	10	10	10	10
H80	U	0.687	0.823	0.8476	0.881	0.863
	F	11.88	9.92	9.629	9.25	9.45
	N	20	17	17	17	17
H100	U	0.679	0.803	0.819	0.821	0.819
	F	19.76	16.70	16.39	16.31	16.35
	N	21	17	17	17	17
H120	U	0.673	0.819	0.851	0.844	0.827
	F	20.42	16.76	16.14	16.25	16.58
	N	28	23	23	23	23
H149	U	0.704	0.847	0.863	0.86	0.857
	F	27.33	22.69	22.29	22.33	22.41

Table 4: Run times (sec) for one-step approach

	$\theta = 0.94$						$\theta = 0.97$					
	$w = 0$	$w = 0.5$	$w = 1$	linear	$K = 3$	$K = 5$	$w = 0$	$w = 0.5$	$w = 1$	linear	$K = 3$	$K = 5$
J40	124	83	40	71	47	110	88	66	35	64	47	170
J50	171	110	80	95	51	130	119	87	78	82	74	176
J60	189	134	82	114	73	170	125	97	69	88	60	187
J70	214	152	119	148	103	200	176	132	91	111	98	245
H80	587	318	283	341	252	405	433	216	185	187	187	427
H100	818	512	460	446	394	700	585	339	275	281	201	549
H120	845	607	540	593	418	714	591	486	458	324	247	668
H149	1120	677	620	614	647	947	676	496	484	383	389	723

produces results with between 1% and 5% reduction in stock sheet utilization. We have also introduced new benchmark data sets for this problem.

There is significant scope for more research on this problem, given its relevance and lack of attention by researchers. Two suggestions for investigation that build on this research are: to improve the quality of packing at the tail end of the pattern construction for the one-stage approach, and investigating a larger clusters for the two-stage approach.

Acknowledgement

Prof Romanova, Prof Stoyan and Prof Pankratov for generously supporting this project by allowing us to use their phi-function implementation.

References

[1] Alvelos, F., T. Chan, P. Vilaca, E. Silva, J.M. Valério de Carvalho. 2009. Sequence based heuristics for two-dimensional bin packing problems. *Engineering Optimization* **41** 773-791.

[2] Bennell, J., G. Scheithauer, Yu. Stoyan, T. Romanova. 2010. Tools of mathematical modeling of arbitrary object packing problems *Annals of OR* **179** 343-368.

[3] Bennell, J.A., J.F. Oliveira. 2009. A tutorial in irregular shaped packing problems. *Journal of Operational Research Society* **60** s93-s105

[4] Bennell, J.A., J.F. Oliveira. 2008. The geometry of nesting problems: A tutorial. *European Journal of Operations Research* **184** 397-415.

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

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

[7] Chernov, N., Yu. Stoyan. T. Romanova. 2010. Mathematical model and efficient algorithms for object packing problem *Computational Geometry* **43** 535-553.

[8] Degraeve, Z., M. Vandebroek. 1998. A Mixed Integer Programming Model for Solving a Layout Problem in the Fashion Industry *Management Science* **44** 301-310.

[9] Fleszar, K. 2013. Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts *Computers and OR* **40** 463-474.

[10] Lodi, A., S. Martello, D. Vigo. 1999. Approximation algorithms for the oriented two-dimensional bin packing problem *European Journal of Operational Research* **112** 158-166.

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

[12] Lodi, A., S. Martello, M. Monaci. 2002. Two-dimensional packing proglems: A survey *European Journal of Operational Research* **141** 241-252.

[13] Polyakovsky, S., R. M'Hallah. 2009. An agent-based approach to the two-dimensional guillotine bin packing problem *European Journal of Operational Research* **192** 767-781.

[14] Scheithauer, G., Y. Stoyan, J. Bennell, T. Romanova, A. Pankratov. 2012. Containment of a pair of rotating objects within a container of minimal area or perimeter.*Discussion paper* Preprint MATH-NM-02-2012, TU Dresden, 28p.

[15] Toussaint, T.S. 1983. Solving geometric problems with rotating calipers *Proceedings of the IEEE MELECON* Athens, **1** A10.2/1-4.

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