Efficient Local Search Heuristics for Packing Irregular Shapes in Two-Dimensional Heterogeneous Bins

Ranga P. Abeysooriya[1], Julia A. Bennell[1], and Antonio Martinez-Sykora[1]

Business School, University of Southampton, SO17 1BJ, Southampton, UK

Abstract. In this paper we proposed a local search heuristic and a genetic algorithm to solve the two-dimensional irregular multiple bin-size bin packing problem. The problem consists of placing a set of pieces represented as 2D polygons in rectangular bins with different dimensions such that the total area of bins used is minimized. Most packing algorithms available in the literature for 2D irregular bin packing consider single size bins only. However, for many industries the material can be supplied in a number of standard size sheets, for example, metal, foam, plastic and timber sheets. For this problem, the cut plans must decide the set of standard size stock sheets as well as which pieces to cut from each bin and how to arrange them in order to minimise waste material. Moreover, the literature constrains the orientation of pieces to a single or finite set of angles. This is often an artificial constraint that makes the solution space easier to navigate. In this paper we do not restrict the orientation of the pieces. We show that the local search heuristic and the genetic algorithm can address all of these decisions and obtain good solutions, with the local search performing better. We also discuss the affect of different groups of stock sheet sizes.

Keywords: Irregular shapes, Multiple bin size bin packing, Jostle Algorithm

1 Introduction

The two dimensional bin packing problem consists of placing a set of pieces, usually represented as rectangles or polygons, in one or several stock sheets (bins) in such a way the total waste generated is minimized. This problem arises in several industries where metal, foam, wood, plastic, paper or leather need to be cut. Depending on the industrial application this problem has several variants described by the properties of both the pieces to be placed and the bins. In this paper we address the problem where the pieces to be cut are irregular and may have concavities. Pieces can be placed in any orientation within rectangular bins that have a variety of different dimensions (heterogeneous bin sizes). We assume that there are a limited number of bin types and there are enough bins of each type to place all the demanded pieces in any set of bins. In this paper we denote this problem as the two-dimensional irregular shape multiple bin size bin packing problem (2D-IMBSBPP) and propose a local search heuristic, called jostle, and a genetic algorithm to find efficient solutions.

Most of the publications in the literature that consider heterogeneous bins are focused on packing rectangular pieces. Pisinger and Sigurad [19] consider a variable bin cost when solving this problem. They propose a MILP model which is then solved by column generation. The model is intractable for large instances and difficult to solve even for small instances. Ortmann et al. [18] propose a two-phase approach. During the first phase they use first-fit decreasing with the largest bins first. A second phase repacks bins into smaller bins. The two phase algorithm is implemented with different bin sizes and aims to minimise the total area of the occupied bins. Wei et al. [21] propose a tabu search that uses a sequential packing heuristic, a local search, and a post-improvement procedure to reduce the total area of used bins in a feasible solution. Alvarez-Valdes et al. [2] implement meta-heuristic algorithms with variable bin costs which are not proportional to the size of the bin. The objective is to minimize the cost of the occupied bins, and the authors employ a greedy randomized adaptive search procedure with path re-linking strategies to combine the best solutions obtained in the iterative process. In order to compare their results with Ortmann et al. [18]'s, they modify the objective function to maximize the overall utilization of the occupied bins.

The vast majority of research publications considering the packing of 2D irregular shapes address the 2D strip packing problem, also known as the nesting problem. Instead of placing pieces into bins the aim is to place all the pieces into a strip with fixed width and infinite length in such a way the total required length is minimized. However, recent publications of packing algorithms for irregular pieces consider the bin packing problem with homogeneous bins, see [14], [20], [16] and [1]. However, in many situations bins are readily available to purchase as rectangular sheets in different standard sizes. In these industries companies usually handle several bin sizes in order to satisfy customer demand. The aim of the companies is not only to reduce the waste generated in the cutting process, but also charge a competitive price to the customers, which usually depends on the area of material needed to meet the demand. There are several publications considering heterogeneous irregular bins, see [4] and [5], who solve an applied problem which arises in the leather industry. In these publications the pieces are approximated by grid squares or pixels rather than polygons. While this simplifies the geometry, solutions suffer from inaccuracy in shape representation.

It is important to highlight that in this paper we use the direct representations of the pieces as polygons and do not restrict the rotation of pieces to a predefined set of angles. Han et al. [13] and Martinez-Sykora et al. [15] considered free rotation for the 2D irregular bin packing problem with guillotine cuts, solving an application derived from the glass industry. More recently, Martinez-Sykora et al. [16] and Abeysooriya et al. [1] considered free rotation for the problem with homogeneous bins and also report results with restricted and fixed rotations. For comparison purposes, we also address the problem where a finite set of rotations are allowed.

The main contribution of this paper is to efficiently solve the 2D-IMBSBPP considering continuous rotation of pieces by using an adaptation of the jostle procedure. Jostle was first proposed by Dowsland et al. [10] for the strip packing problem and then used in [1] in bin packing problems with homogeneous bins. One of the most important properties of the jostle procedure is that it explores efficient solutions with a relatively low computational effort, compared with other available algorithms. The jostle procedure works over the sequence of pieces that represents the order pieces are inserted in the layout. The constructive heuristic iterates between packing from one end of the strip and then the other end taking the sequence from the last iteration. The results presented in this paper show a considerable improvement when using a heterogeneous set of bin sizes, which demonstrates the efficiency of the algorithms presented.

The paper is organized as follows. In Section 2 we describe the problem and we discuss the measure of performance used to evaluate the quality of feasible solutions. Section 3 explains the jostle algorithms and the genetic algorithm to solve 2D-IMBSBPP. Computational tests are presented in Section 4. Finally, in Section 5 we present the conclusions of the paper.

2 Problem Description

Let N' be the number of rectangular bin types (stock sheets) and let L_k and W_k be, respectively, the length and the width of bin type $k \in \{1, \ldots, N'\}$. Let $P = \{p_1, \ldots, p_n\}$ be the set of pieces to be cut from the bins, where n is the number of pieces. We assume that all the pieces may have a different shape.

A solution $s = \{B_k^s | \ k = 1, \ldots, N'\}$ is represented by a set of bins of each type used in the solution, where each single bin $b_{jk}^s(P_{jk}, O_{jk}, X_{jk}, Y_{jk}) \in B_k^s$ is determined by the subset of pieces placed in the bin (P_{jk}), the set of orientations used for each piece (O_{jk}) and the X and Y coordinates of the reference point of each piece. A solution s is a feasible solution if all the pieces are placed, i.e, $\bigcup_{k=1}^{N'} \bigcup_{j| \ b_{jk} \in B_k^s} \overline{P}_{jk} = P$, there is no overlapping between each pair of pieces placed in the same bin and no piece exceeds the bins dimensions.

Initially we consider an unlimited number of bins of each type k. We denote by N_k^s the number of occupied bins of type k in solution s.

The position of the pieces into the bins is given by the coordinates of the reference point, which we assume it is the bottom-left corner point of the enclosing rectangle whose edges are orthogonal to the edges of the bin.

The aim is to minimize the waste generated when placing all the demand pieces. However, this measure could lead to many ties. In order to guide the search, we propose a second measure, which is the standard deviation of absolute bin waste, where a larger standard deviation indicates the potential to empty poorly packed bins.

2.1 Evaluation function of a solution

We use two measurements to evaluate the quality of a complete solution s.

1) The overall utilization U_s is defined as;

$$U_s = \frac{\sum_{i=1}^{n} Area(p_i)}{\sum_{k=1}^{N'} W_k L_k N_k^s}$$

where $Area(p_i)$ denote the area of the i^{th} piece. Since $\sum_{i=1}^{n} Area(p_i)$ is constant then maximizing the utilization is equivalent to minimize to total area of bins used in s. In this paper the aim is to maximize U_s.

2) The standard deviation of absolute bin waste of the occupied bins (σ_s). This measurement is used during the algorithm to break ties when two solutions have the same overall utilization. A low σ_s value shares the waste among the bins with low variance and balances individual bin utility. A higher σ_s leads to a higher variation in bin space utilization. We encourage this type of solution during the search process of the algorithm to move the few pieces packed inside a large bin (i.e, a lower utilized bin) to another occupied bin.

3 Packing procedure

In this section we introduce two heuristic algorithms to solve 2D-IMBSBPP. The first algorithm is an iterated jostle approach with random assignment of bins (IJRAB), and the second is a hybrid genetic algorithm with iterated jostle (HGAIJ). Jostle iteratively applies a fast constructive procedure and searches over the sequence of pieces. Jostle was first used to solve strip packing problems in [10]. In [1], this idea was extended to the bin packing problem by placing bins consecutively simulating a strip, and adding the constraint that no piece can be placed across the boundary of two bins. However, with heterogeneous bin sizes the available width depends on the bin type and, therefore, may change from one part of the strip to another.

Most of the state of the art algorithms to solve the strip packing problem with irregular pieces works with infeasible solutions. The main idea, first proposed in [6], is to fix the strip length, randomly place the pieces within the strip and then solve the overlapping problem. Once a feasible solution is found, the strip length is reduced and the search starts again. While effective at finding good solutions, it is slow. The heterogeneous bin packing problem, requires finding feasible solutions for different bin combinations. Hence, fast constructive approaches like jostle that work with feasible solutions are a more reasonable option.

3.1 Iterated Jostling approach with Random Assignment of Bins (IJRAB)

IJRAB is an iterative procedure that uses a constructive algorithm (CA) within a local search. The fast constructive procedure allows us to build a feasible solution given a permutation of bins and pieces. The local search works over the sequence of pieces and bins.

Layout construction The constructive algorithm (CA) is based on placing the pieces sequentially, according to a given placement rule. We first select a set of bins which are placed concurrently in a given order, in such a way the bottom edge of a bins are aligned, as is depicted in Figure 1. Given the different bin dimensions, the strip width is set to be the same as the widest bin. Note that there will be areas of the strip where pieces cannot be placed, shown by shaded areas in Figure 1. Finally, the strategy to place the pieces is inspired by the TOPOS algorithm proposed in [17] and improved by Bennell and Song [9]. Therefore, for a given order of bins and a given order of pieces with a given orientation we use this fast CA to build a solution.

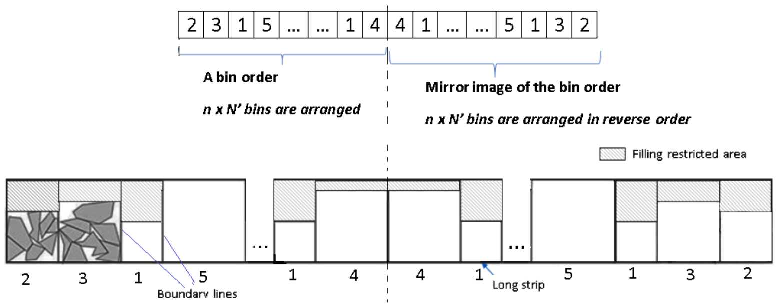


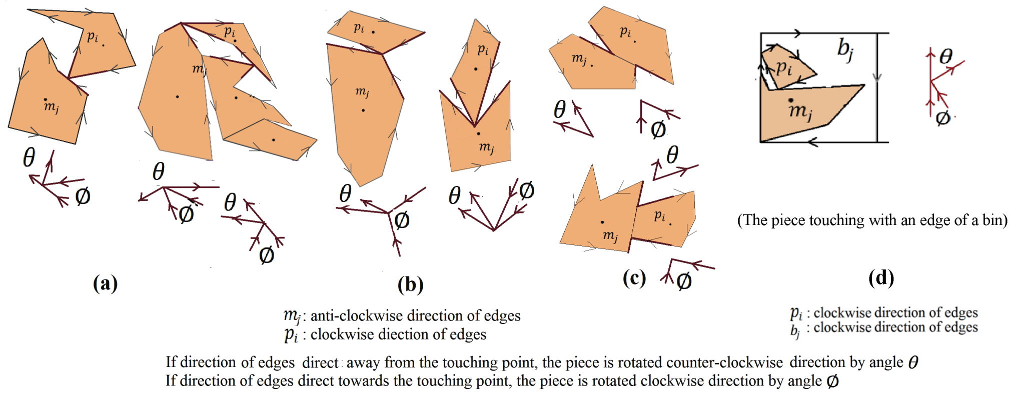
Fig. 1. Packing Layout

In the following description of the CA we assume we have a given ordered combination of bin types and a given permutation of all the pieces.

First we setup the strip of bins. Let $\tau^{(t)} \in \mathbb{N}^{n\times N'}$ be a vector with values corresponding to the bin types used in the solution in the given order. This vector represents $n \times N'$ bins in order to guarantee that all the pieces can be placed. Since jostle packs from both ends of the strip alternately, we arrange the $n \times N'$ bins on the strip and arrange another $n \times N'$ bins following the mirror order as illustrated in Figure 1.

Then, for each piece, we determine the placement position and orientation. Assuming a given orientation, the CA generates non-overlapping placement positions following the improved TOPOS approach that was adapted for bin packing by Abeysooriya et al. [1]. This approach uses the no-fit polygon (NFP) and inner-fit polygon (IFP), see [8], to identify all feasible touching positions between pieces. New pieces are inserted in the layout in a touching position with pieces already placed. Once a position and orientation is selected, the new piece is merged with the already placed pieces. Any space between pieces that could be used to place another piece is a hole and is recorded into a list of holes. At each piece insertion, the partial solution is represented by a merged polygon, for each occupied bin, and a list of holes. When placing the next piece, we first try the holes, sorted by non-increasing area. If the piece does not fit in any of the holes, then we try placing the piece on the boundary of the merged polygons starting with the first bin. If this fails, the new piece is placed in a new bin.

The above assumes a fixed orientation, whereas we are permitted to use any orientation. The placement angle of a piece is determined by an angle tuning strategy. First we identify the candidate placements by finding the best placement position for each of the pre-assigned angles, which are $o_i = 0, 90, 180, 270$. Then we pick the position and orientation angle from these candidates according to the *Maximum Utilization (MU)* placement policy described below. Note that up to this point, this is also the method CA applies if the rotation of pieces is restricted to a predefined set of angles. When the rotation is not restricted, we try alternative angles by rotating the piece so that one of its edges are concurrent to the edge of the merged piece. These are determined by the touching points between the new piece and the merged polygon or edges of the bin. Figure 2, illustrates where new angles θ and φ are obtained and highlights all the possible edge-vertex, vertex-vertex and edge-edge combinations which can occur. In each case, piece p is rotated in a counter-clockwise direction by angle θ and in a clockwise direction by angle ϕ. If none of these new angles provide a better placement position then the algorithm takes the best corresponding predefined angle position and angle as the placement of the piece. If the next piece touches the boundary of a bin, then the same procedure is followed considering the angles created by considering the edges of the bin.



m_j: anti-clockwise direction of edges
p_i : clockwise diection of edges

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 $\emptyset$

Fig. 2. Angle Tuning

The algorithm uses the *Maximum Utilization (MU)* placement policy as proposed in [1] since their results show it is more efficient than the bottom-left and minimum length placement rules. The MU rule places the next piece in the position that maximises the area utilisation of the convex hull of the layout in the first bin the piece fits. Let m_l be the merged polygon of the placed pieces in the l^{th} bin in the order. For each feasible position, the area utilisation is calculated as $(CHull(m_l) + Area(p_i))/CHull(m_l + p_i)$, where $CHull(m_l)$ denotes the convex hull area of the placed pieces m_l, $Area(p_i)$ denotes area of the new piece and $CHull(m_l + p_i)$ denotes area of the convex hull of both m_l and p_i once placed in a feasible position. The placement position corresponds to the maximum area utilisation is selected as the placement position for the new piece.

Solution improvement phase - Jostle The improvement mechanism works over the sequence of bins and pieces. IJRAB starts with a solution generated by the CA using an arbitrary order of pieces and an arbitrary order of bins and packs the pieces from the left end of the bin strip towards the right along the strip. Given this solution, all the pieces are re-ordered according to the right-most x-coordinates position of the pieces 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. The idea is to shake pieces from left to right and right to left along the bin order, so that each jostle iteration generates a new sequence of the pieces. This approach has been proven to be more efficient than a multi-start approach randomizing over the sequence of pieces (see [1]).

Abeysooriya et al. [1] points out that jostle gets stuck in local optima and suggests an iterated jostle approach where they apply a kick, analogous to iterated local search. We design two types of kick, which are applied after a predefined number of jostle cycles with no improvement. The first kick is called *piece kick*, applied to the current locally optimal solution, where a random piece is removed and reinserted in a random position in the sequence (used in [7] and [1]). The second kick is the *Bin kick*, applied to the best solution found so far, where a random bin is selected from the occupied bins of this solution and is replaced with a random bin selected from the other bin types. The corresponding change is applied to the mirror bin position of the bin order as well, so that the same bin configuration is retained at both ends of the strip. We denote K_p as the number of jostle cycles with no improvement before performing a *piece kick*, and K_b as the number of piece kicks performed with no improvement before performing a *bin kick*.

Algorithm 1 provides the steps of the IJRAB algorithm. We use $P^{(L,t)}$ to denote the piece order used at the t^{th} iteration of the algorithm, which packs pieces from left to right. Similarly, $P^{(R,t)}$ denotes the piece order, which packs pieces from right to left at the t^{th} iteration. We use $\tau^{(L,t)}$ to denote bin order at t^{th} iteration where the bin arrangement is considered from left to right when placing pieces. The bin arrangement $\tau^{(R,t)}$ is the mirror arrangement of $\tau^{(L,t)}$. As explained before, a certain bin order is considered at a time. Depending on the best solution found so far (see Algorithm 1), the leading bin order τ^* is updated.

The algorithm terminates after a given maximum computation time. At each local optima leading to a bin kick, we apply post processing to improve the best solution found within each bin configuration (see line 25). We propose the following post processing strategies.

– Strategy 1 (S1): Attempt to repack pieces in the least utilized bin into the smallest possible bin.

Algorithm 1: IJRAB

```
1  Set number of iterations $t = 1$;
2  Set $P^{(L,t)}$ as a random permutation of the pieces;
3  Set $\tau^{(L,t)}$ as a random order of bins;
4  Initialize best utilization and best standard deviation $U^* = 0$, $\sigma^* = 0$;
5  Initialize counters for piece kicks and bin kicks $q_p = 0$; $q_b = 0$;
6  while termination condition is not met do
7      Generate solution layout, $s_L$, from $P^{(L,t)}$ and $\tau^{(L,t)}$;
8      Evaluate $U_{s_L}$ and $\sigma_{s_L}$;
9      Derive $P^{(R,t)}$ from the solution;
10     Generate solution layout, $s_R$, from $P^{(R,t)}$ and $\tau^{(R,t)}$;
11     Evaluate $U_{s_R}$ and $\sigma_{s_R}$;
12     Set $s$ the best solution between $s_L$ and $s_R$ taking into account utilization
         ($U$) and breaking ties with the standard deviation ($\sigma$);
13     if $U^s > U^*$ OR ($U^s = U^*$ AND $\sigma^s > \sigma^*$) then
14         Set $U^* = U^s$, $\sigma^* = \sigma^s$; Reset $q_p = 0$;
15     else
16         $q_p = q_p + 1$;
17     end
18     if $q_b < K_b$ then
19         if $q_p > K_p$ then
20             Apply piece kick. Change the position of one piece in the current
                 solution piece sequence;
21             $q_b = q_b + 1$;
22             Reset $q_p = 0$;
23         end
24     else
25         Apply post processing to the best solution;
26         Apply bin kick. Return to the best solution found so far and change one
             bin type;
27         Reset $q_b = 0$, $q_p = 0$;
28     end
29     $t = t + 1$;
30 end
```

- Strategy 2 (S2): Attempt to repack pieces in each of occupied bins into the smallest possible bin.

For the experimental investigation we compare the effectiveness of each post processing strategy based on the solution quality and computational time. Specifically we run the following three variants:

- *IJRAB*: Implement Algorithm 1 with no post processing.
- *IJRAB-A2*: Implement Algorithm 1 applying S1 for the best solution found at each bin configuration.
- *IJRAB-A3*: Implement Algorithm 1 applying S2 for the best solution found at each bin configuration.

3.2 GA/Jostle approach (HGAIJ)

Our second computational method combines a Genetic Algorithm (GA) and the Iterated Jostle (IJ). GAs have been successfully implemented for the single bin size bin packing problem with rectangular pieces (SBSBPP) in [12], multiple bin size bin packing problem (MBSBPP) with rectangular pieces in [3], and irregular pieces packing problems with irregular bins in [4].

A solution is encoded as a chromosome by a permutation of bins followed by a permutation of pieces. This is decoded using the CA to produce the packing layout and evaluate the fitness, U_s. The coding structure of HGAIJ is illustrated in Figure 3 along with the decoded solution, which packs 16 pieces into four types (sizes) of input bins. In order to guarantee all the pieces can be packed, the length of bin permutation is $n \times N^{'}$. Note that when we use the jostle operation the strip is twice as long to include the mirror bin order.

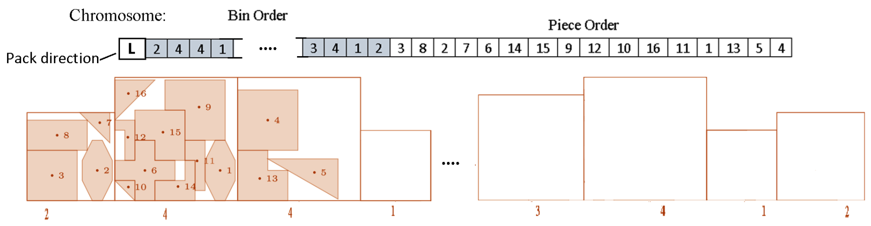


Fig. 3. Representation of solutions

The initial population contains S solutions generated by applying the CA to S random permutations of bins and pieces. Each piece permutation contains all the pieces to be packed. For each generation, we execute the main loop in Algorithm 2. A *pool* of solutions is populated at each iteration of the main-loop. This contains solutions of the current population (parents) as well as the new solutions (offspring) created by the crossover and mutation operators. The selection mechanism selects S solutions from the pool to enter the next generation.

Crossover: Crossover creates two offspring from two parent chromosomes using two type of crossover operator. We use the *uniform crossover*, see [11] for the bin part of the chromosome. The operator first constructs a random binary mask. Using the mask, the first child inherits the genes (in the bin order) of the first parent if there is a "1"in the mask and from the second parent if there is a "0"in the mask. The second child is formed in a similar way by reversing the role of mask. Step 1 in figure 4 shows an example of the crossover operator for the bin part of the chromosome.

The second crossover operator changes the piece permutation. Given a randomly selected point in the permutation the genes before this point are copied from the first parent to the first offspring as illustrated in step 2 of figure 4. The second parent's piece permutation is then scanned and the missing genes in the

Algorithm 2: General structure of HGAIJ

```
1  Initialization;
2  Main loop: Generations;
3  $GenN = 1$;
4  while $GenN < MaxGen$ do
5      Populate the pool;
6         - crossover;
7         - mutation;
8         - generate offspring solutions using the constructive algorithm $CA$;
9         - improve offspring solutions by jostling with piece kicks;
10     Sort the pool according to solution quality;
11     Select solutions for nest population;
12     $GenN = GenN + 1$;
13 end
```

offspring are inserted in the order they appear in the second parent. The same procedure is used to generate the second offspring, where each parent has the opposite role. All parents are selected without replacement for crossover generating an equal number of offspring to parents.

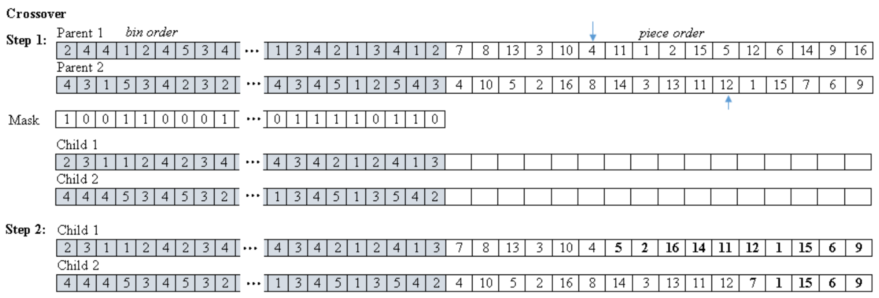


Fig. 4. Crossover operation

Mutation: The purpose of mutation is to provide greater diversity within the population and inhibit premature convergence. Each offspring will be mutated with probability P_{mu}. If an offspring is selected for mutation, then randomly select two points in the part of the bin permutation that contains pieces and reverse the order of bins between these two points. Note that this mutation generates a major change in the corresponding solution as bin spaces of the layout change dramatically.

Improving offspring solutions by jostling with piece kicks: In this step, the child chromosomes are improved with the help of the iterated jostle procedure (with piece kicks) discussed in Section 3.1. Initially, we applied IJ to all offspring, however the computation time was too long. In our final experiments we only apply IJ to the fittest solution.

Selection for the next population: The next population is selected from the pool that currently contains both the parent and offspring solutions. The selection strategy aims to ensure both quality and diversity of the next population using *elitist* and *tournament* selection methods. Solution quality is measured by utilisation, U_s. For diversity we define the distance ($dist$) of a solution, x, from the best solution in the population, y, where the best solution has the largest U_s:

$$dist(x) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_{Nx} - y_{Nx}|$$

where x_i denotes the area of the i^{th} bin located in the bin permutation of the chromosome for solution x, y_i denotes the area of the i^{th} bin located in the bin permutation of the chromosome with best U. In the case that chromosomes x and y have a different number of used bins ($N_x \neq N_y$), then the distance is calculated up to $\min\{N_x, N_y\}$.

The selection process sorts the chromosomes in the pool in descending order of their U_s value. We compare the bin permutation of chromosomes that have the same U_s value. Those with identical occupied bin permutations are compared by their $dist$ value. Since we wish to maintain diversity in the population, we retain only one of these chromosomes, keeping the one with the largest value of $dist$. Given the sorted list of offspring and parents, the elitist selection scheme copies γ of the best chromosomes in pool to the new population. The population is made up to S by tournament selection, where it randomly selects two chromosomes and selects the best by U_s breaking ties by σ_s.

Based on different computational test we performed we found the following best set of parameters. For the GA, S is set to 16, $\gamma = 2$ and $P_{mu} = 0.025$. For IJ within the GA, $K_p = 4$ and the search terminates after 12 cycles. Similarly, for IJRAB approach we found that $K_p = 4$ and $K_b = 3$ produce better results.

4 Computational Experiments

The algorithms described above were coded in Visual C++ 2012 as a sequential program. All experiments were carried out using an Intel 2.60 GHz processor and 4GB RAM. The algorithms do not contain parallel processing and therefore can be run using a single processor.

The data instances are taken from the benchmark nesting instances published on ESICUP (EURO Special Interest Group on Cutting and Packing) website. We used 14 irregular shape instances representing both convex and non-convex polygons. Since these are defined for strip packing problems, we define a set of bin sizes. In this case, for each instance, we identify parameter d_{max} as the maximum length or width among the pieces in their initial orientation. Using d_{max} we generate nine different bin sizes and define four subsets of bin sizes;

Table 1. Instances

Instance	No.of Pieces	Instance	No.of Pieces
Shapes2	28	2×Jakobs2	50
3×Dighe1	48	Poly3a	45
3×Dighe2	30	Poly3b	45
Poly4a	60	Poly4b	60
3×Fu	36	Poly5a	75
2×Han	46	Poly5b	75
2×Jakobs1	50	Shapes	43

Table 2. Bin Configurations

Configuration	No. of Bin types	Input bin sizes	Bin type ID
SB	5	$0.5d_{max}$, $0.75d_{max}$, $1.0d_{max}$, $1.25d_{max}$,$1.5d_{max}$	1,2,3,4,5
MB	5	$1.0d_{max}$, $1.25d_{max}$, $1.5d_{max}$, $1.75d_{max}$,$2.0d_{max}$	3,4,5,6,7
LB	5	$1.5d_{max}$, $1.75d_{max}$, $2.0d_{max}$, $2.25d_{max}$,$2.5d_{max}$	5,6,7,8,9
Mix	9	$0.5d_{max}$, $0.75d_{max}$, $1.0d_{max}$, $1.25d_{max}$,$1.5d_{max}$	1,2,3,4,5
		$1.75d_{max}$, $2.0d_{max}$, $2.25d_{max}$, $2.5d_{max}$	6,7,8,9

small (SB), medium (MB), large (LB) and mixed, which contains all nine bin sizes, as illustrated in tables 1 and 2.

We test IJRAB and HGAIJ using the 14 data instances combined with each set of bin sizes and run our experiments for variants where we restrict the allowed rotation angles (RR) and also allow unrestricted rotation (UR) of the pieces. For RR the permitted angles of orientation are $0, 90, 180, 270$ degrees. Since there are random elements in the algorithm, we run each algorithm test for 10 trials and report the average. Previous research into the irregular bin packing problem have only used homogeneous bins, hence there are no benchmark results. In order to validate our search mechanism, we compare with the best result from repeated runs of the constructive algorithm with random permutations of bins and pieces (CA (Rnd)). The termination condition for all algorithms is 800 seconds.

In Table 3 we compare the different implementations of IJRAB for RR and UR. For each bin configuration report the average U_s, percentage improvement over IJRAB-A1, and the average number of jostle cycles performed in the 800 seconds (Avg. cycles). We can observe that *IJRAB-A3* performs better, although the difference between *IJRAB-A2* and *IJRAB-A3* for Mix, LB and MB bin size configurations is small. For SB, since bins will contain very few pieces the bin allocation is more critical, hence there being greater value in trying to reduce the size of all bins.

In Table 4 we compare the two algorithms, IJRAB-A3 and HGAIJ along with CA (Rnd). CA (Rnd) performs least well, as expected. Although the performance gap is less than 5% showing that the CA is optimising well. IJRAB performs better than HGAIJ. This may be because a change in the bin or piece permutation can lead to a significant change in the decoded solution, hence the crossover operators are not able to retain good parts of the solution.

Table 3. Performance comparison of IJRAB algorithms for different bin configurations

Bin Config.		Restricted Rotation of pieces			Unrestricted Rotation of pieces		
		IJRAB-A1	IJRAB-A2	IJRAB-A3	IJRAB-A1	IJRAB-A2	IJRAB-A3
Mix	Avg. U	0.692	**0.700**	**0.703**	0.699	**0.706**	**0.708**
	Impv %	0%	**1.16%**	**1.59%**	0%	**1.00%**	**1.29%**
	Avg. cycles	*1202.2*	*1156.5*	*1123.4*	*802.0*	*764.9*	*744.1*
LB	Avg. U	0.667	**0.687**	**0.689**	0.680	**0.695**	**0.696**
	Impv %	0%	**3.00%**	**3.30%**	0%	**2.21%**	**2.35%**
	Avg. cycles	*1117.3*	*1072.4*	*1041.7*	*717.4*	*695.6*	*660.7*
MB	Avg. U	0.671	**0.684**	**0.686**	0.684	**0.692**	**0.693**
	Impv %	0%	**1.94%**	**2.24%**	0%	**1.17%**	**1.32%**
	Avg. cycles	*1275.9*	*1215.8*	*1176.4*	*814.7*	*789.8*	*776.7*
SB	Avg. U	0.656	**0.662**	**0.677**	0.668	**0.675**	**0.685**
	Impv %	0%	**0.91%**	**3.20%**	0%	**1.05%**	**2.54%**
	Avg. cycles	*1584.2*	*1506.6*	*1365.2*	*956.9*	*909.1*	*862.1*

Table 4. Performance comparison of IJRAB and HGAIJ methods

Bins Config.	Restricted Rot. *Avg. U*			Unrestricted Rot. *Avg. U*		
	CA (Rand)	IJRAB	HGAIJ	CA(Rand)	IJRAB	HGAIJ
Mix.	0.678	**0.703**	0.693	0.682	**0.708**	0.699
Impr. %	-	3.69%	2.21%	-	3.81%	2.49%
LB	0.665	**0.689**	0.686	0.675	**0.696**	0.690
Impr. %	-	3.61%	3.16%	-	3.11%	2.22%
MB	0.663	**0.686**	0.681	0.671	**0.693**	0.687
Impr. %	-	3.47%	2.71%	-	3.28%	2.38%
SB	0.650	**0.677**	0.659	0.657	**0.685**	0.666
Impr. %	-	4.15%	1.38%	-	4.26%	1.37%

5 Concluding Remarks and future work

In this paper, we develop solutions methods for the two dimensional bin packing problem with irregular pieces and multiple bins sizes, also known as 2D-IMBSBPP, where the objective is to maximize the overall utilisation of bins. This problem is practical in several material cutting industries yet new to the research literature.

We demonstrate that the iterated jostle algorithm outperforms an adapted genetic algorithm and simple repeated random construction. Our results also show that a greater selection of bins provides more efficient packing and, when restricting the subset of bins, large bins are more useful. Moreover, for smaller bins, the bin selection is more important than with larger bins.

Our algorithms support the objectives of efficient material use and material procurement decisions. Being able to solve bin packing over heterogeneous bins means that retaining and re-using residual material becomes possible within the cutting planning system. We notes that other costs in addition to material waste affect production cost. Our method is also robust to implement different objective functions such as including set-up and inventory costs.

References

[1] Abeysooriya, R. P., Bennell, J. A., and Martinez-Sykora, A. (2015). Jostle heuristic for 2D-Irregular shaped Packing Problems with Free Rotation. *27th European Conference on Operational Research*.

[2] Alvarez-Valdes, R., Parreño, F., and Tamarit, J. M. (2013). A GRASP/Path Relinking algorithm for two- and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40:3081–3090.

[3] Babu, A. R. and Babu, N. R. (1999). Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. *International Journal of Production Research*, 37(7):1625–1643.

[4] Babu, A. R. and Babu, N. R. (2001). A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. *Computer-Aided Design*, 33(12):879–891.

[5] Baldacci, R., Boschetti, M. A., Ganovelli, M., and Maniezzo, V. (2014). Algorithms for nesting with defects. *Discrete Applied Mathematics*, 163, Part 1:17–33.

[6] Bennell, J. A. and Dowsland, K. A. (2001). Hybridising Tabu Search with Optimisation Techniques for Irregular Stock Cutting. *Management Science*, 47(8):1160–1172.

[7] Bennell, J. A. and Oliveira, J. F. (2009). A Tutorial in Irregular Shape Packing Problems. *The Journal of the Operational Research Society*, 60:s93–s105.

[8] Bennell, J. A. and Song, X. (2008). A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. *Computers & Operations Research*, 35(1):267–281.

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

[10] Dowsland, K. A., Dowsland, W. B., and Bennell, J. A. (1998). Jostling for position: local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6):647–658.

[11] Falkenauer, E. (1999). The worth of the uniform [uniform crossover]. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, page 782 Vol. 1.

[12] Gonçalves, J. F. and Resende, M. G. C. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, 145(2):500–510.

[13] Han, W., Bennell, J. A., Zhao, X., and Song, X. (2013). Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints. *European Journal of Operational Research*, 230(3):495–504.

[14] Lopez-Camacho, E., Ochoa, G., Terashima-Marin, H., and Burke, E. K. (2013). An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, 206(1):241–264.

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

[16] Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J. A., Ruiz, R., and Tamarit, J. M. (2017). Matheuristics for the irregular bin packing problem with free rotations. *European Journal of Operational Research*, 258(2):440–455.

[17] Oliveira, J. F., Gomes, A. M., and Ferreira, J. S. (2000). TOPOS – A new constructive algorithm for nesting problems. *OR-Spektrum*, 22(2):263–284.

[18] Ortmann, F. G., Ntene, N., and van Vuuren, J. H. (2010). New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research*, 203(2):306–315.

[19] Pisinger, D. and Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167.

[20] Song, X. and Bennell, J. A. (2014). Column generation and sequential heuristic procedure for solving an irregular shape cutting stock problem. *Journal of the Operational Research Society*, 65(7):1037–1052.

[21] Wei, L., Oon, W., Zhu, W., and Lim, A. (2013). A goal-driven approach to the 2D bin packing and variable-sized bin packing problems. *European Journal of Operational Research*, 224:110–121.