International Journal of Industrial Engineering Computations 13 (2022) 491–506

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Optimization of two-dimensional irregular bin packing problem considering slit distance and free rotation of pieces

Zi Wang[a,c], Daofang Chang[b,c*] and Xingyu Man[a,c]

[a]Logistics Science and Engineering Research Institute, Shanghai Maritime University, Shanghai 200120, China
[b]School of Logistics Engineering, Shanghai Maritime University, Shanghai 200120, China
[c]Qingdao Institute, Shanghai Maritime University, Qingdao 266011, China

C H R O N I C L E

Article history:
Received July 7 2022
Received in Revised Format July 31 2022
Accepted August 3 2022
Available online August, 3 2022

Keywords:
2DIBPP
Slit distance
Free rotation
Equidistant edge expansion approach
Overlap minimization method
LS algorithm

A B S T R A C T

In this paper, we present a two-dimensional irregular bin packing problem (2DIBPP) that takes into account the slit distance and allows the pieces to rotate freely. The target is to arrange a specified collection of pieces with irregular shapes into a minimal number of bins. Firstly, we develop a mathematical model for the 2DIBPP that considers slit distance and free rotation of the pieces, and an equidistant edge expansion approach is then proposed to handle the slit distance. Secondly, a two-stage method is implemented to get a finite collection of promising rotation angles, effectively decreasing the search neighbourhood. Thirdly, we decompose the 2DIBPP into two sub-problems: piece assignment and packing. The Partial Bin Packing (PBP) strategy is employed in the allocation stage, and we adopt an overlap minimization method to pack the pieces into an individual bin. Finally, we use a local search (LS) algorithm to advance the quality of the solutions by adjusting the piece assignment across bins. Experimental evidence exhibits that our approach is competitive in most instances of the literature, with four better results in five benchmark instances.

1. Introduction

Steel production causes about 6% of manufactured CO_2 emissions yearly, and one ton of steel production emits about 1.8 tons of CO_2 (Quader et al., 2016). Improving material utilisation can help reduce steel production and CO_2 emission, which in turn do a favour to combat global climate change. Additionally, recently there has been a high level of steel waste and low material utilisation in manufacturing, which is detrimental to the sustainable development of enterprises and the environment.

Manufacturing enterprises use various pieces in production, mainly produced through the packing and cutting process. As is well known, piece packing and cutting problems are widespread in industries such as shipbuilding, textiles and glass, where packing is the basis for cutting, and the generation of highly utilisable packing solutions is the key to material saving. Therefore, it is of great importance to research the piece packing problem. The two-dimensional irregular piece packing problem can be divided into the strip packing problem (2DISPP) (Umetani & Murakami, 2022; Elkeran, 2013; Pinheiro et al., 2016) and the bin packing problem (2DIBPP) (Martinez-Sykora et al., 2017; Abeysooriya et al., 2018; Zhang et al., 2022; Liu et al., 2020). Although the packing problem has received considerable attention, most studies have been carried out on the 2DISPP. This problem aims to pack pieces within a strip stock sheet accompanying infinite length and fixed width to obtain the shortest packing length. During the packing process, the following two constraints should be met together: (1) The pieces do not overlap; (2) The pieces cannot exceed the contour of the stock sheet. Given this problem, scholars have made numerous research achievements. Umetani and Murakami (2022) proposed a dual scanline representation to solve the 2DISPP of rasterized patterns and developed coordinate descent heuristics for the raster model. An optimization method incorporating the cuckoo search and guided local search was proposed by Elkeran (2013). A pairwise clustering approach was introduced

* Corresponding author
E-mail: dfchang@shmtu.edu.cn (D. Chang)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2022.8.001

to group identical polygons together. Pinheiro et al. (2016) proposed a random key genetic algorithm (RKGA) combined with the Bottom-Left (BL) (Jakobs, 1996) algorithm to solve the packing problem. Meanwhile, a compression algorithm running in RKGA was proposed to improve the local solution quality. In addition, some scholars have attempted to use mathematical programming methods to solve the 2DISPP. These models were composed of mixed integer programming (MIP) models with linear objective functions and mixed integer constraints (Alvarez-Valdes et al., 2013; Cherri et al., 2016; Leao et al., 2016; Rodrigues & Toledo, 2017; Bennell et al., 2018) and non-linear programming models with non-linear objective functions(Cherri et al., 2016; Leao et al., 2020). However, the limitation of these models was obvious, as these models could only solve small-scale piece packing problems and were not able to help with a larger number of pieces. Moreover, solving the programming models was costly in terms of time. For more detailed information, please look up the literature (Leao et al., 2020), which describes various programming models for solving the packing problem.

As to the 2DIBPP, pieces need to be packed into several bins of fixed length and width, satisfying the two constraints like 2DISPP, aiming to obtain the minimum number of bins and the optimal solution. This problem is common in industrial production and has applications in several enterprises. Bennell et al. (2018) proposed a beam search algorithm and successfully applied it to multi-bin and single-bin size instances. Still, the limitation was evident because it could only handle convex pieces and was solely adapted to the guillotine cutting process. However, this study did not allow the pieces to rotate. Some studies allowed pieces to rotate by a limited number of special angles (Zhang et al., 2022; Liu et al., 2020), such as 90º, 180º and 270º. Zhang et al. (2022) introduced a waste least first decreasing (WLFD) scheme to allot pieces and used a greedy method to exchange pieces between two bins. In addition, a hot start along with an iterative doubling search strategy was submitted to accelerate the packing. A heuristic algorithm was used by Liu et al. (2020) to solve the 2DIBPP, in which a First Fit Decreasing (FFD) strategy was utilised to distribute pieces to bins. As far as we know, up to now, only two studies (Martinez-Sykora et al., 2017; Abeysooriya et al., 2018) allowed pieces to rotate freely. Martinez-Sykora et al. (2017) compared five methods of assigning pieces and used the MIP model to pack the pieces into bins. Abeysooriya et al. (2018) considered piece allocation and packing together and introduced the Jostle strategy to the 2DIBPP for the first time. What is more, a diversification mechanism was introduced to improve the Jostle method's performance.

The above literature has achieved many achievements based on heuristic algorithms and mathematical programming models. However, most of these have been done based on arranging pieces as closely as possible to produce a solution without considering actual manufacturing parameters. When we study the packing problem, it is valuable to consider the actual manufacturing parameters rather than only the geometry of pieces (Anand & Babu, 2015). In the piece cutting process, as the cutting machine's cutter has a definite width, to escape damaging the pieces, a specified distance needs to be reserved between pieces and between pieces and the edges of the plate, which is referred to as slit distance in this paper. All the studies of this paper are based on considering slit distance.

This paper studies the 2DIBPP that considers slit distance and allows the pieces to rotate freely. As far as we know, this is the first research that considers the two elements simultaneously. We have addressed and solved a common and pragmatic problem for many industries. Specifically, we have carried out the following work. Firstly, we propose a mathematical model considering the two elements, and an equidistant edge expanding method is introduced to deal with the slit distance. Secondly, we decompose the 2DIBPP into two sub-problems: piece allocation and piece packing. In the piece allocation stage, Partial Bin Packing (PBP) method is used, and the superiority of this method is proved by comparison with the Direct Constructive Heuristic (DCH) strategy. In the piece packing stage, we use the BL and overlap minimization algorithms. Thirdly, a two-stage method is proposed to obtain a finite set of promising rotation angles, effectively narrowing the search neighbourhood. By comparing the results obtained by free rotation and limited angles strategy, the effectiveness of the former is demonstrated. Finally, we use a local search algorithm to progress the final solution quality by adjusting piece allocation across bins.

The paper's structure is established along these lines. We first execute a detailed problem description and define some notations in Section 2. In section 3, we introduce some essential geometric pre-processing methods and tools. After that, we solve the piece assignment and packing problems in Section 4 and Section 5, respectively. Then, we introduce a local search strategy in Section 6. In Section 7, we analyse the experimental design and computational results. Finally, conclusions are drawn in Section 8.

2. Problem description

Following the terminological conventions of previous studies, we will refer to the plates for packing pieces as bins in the following, and the pieces are represented by polygons. Before formally describing the 2DIBPP, some applicable definitions are firstly given.

Definition 1 (translational operator $\oplus$). Given a polygon p_i and a translation vector $v_t = \left(v_{tx}, v_{ty}\right)$, the operator $\oplus$ describes the translation process of the polygon around the vector. For point p_0 on polygon p_i, the translational operator $\oplus$ is defined as: $p_i \oplus v_t = \left\{\left(p_{0x} + v_{tx}, p_{0y} + v_{ty}\right) \middle| p_0 \in p_i\right\}$.

Definition 2 (rotation operator $p_i\left(\theta_i\right)$). Let the set of allowable rotation angles for piece p_i be ϑ_i, then the rotation angle set for n polygons is $O=\{\vartheta_1,\vartheta_2,...,\vartheta_n\}, i=1,...,n$. Let the coordinates of point p_0 on polygon p_i, with the origin $(0,0)$ as the reference point, rotated by angle ϑ_i be $p_0^{\theta_i}=\left(p_{0x}^{\theta_i}, p_{0y}^{\theta_i}\right)$, in which rotation angle $\theta_i \in \vartheta_i$ and $\theta_i \in [0,2\pi]$, then we can get the Eq. (1).

$$\begin{bmatrix} p_{0x}^{\theta_i} \\ p_{0y}^{\theta_i} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} p_{0x} \\ p_{0y} \end{bmatrix} = \begin{bmatrix} p_{0x}\cos\theta_i - p_{0y}\sin\theta_i \\ p_{0x}\sin\theta_i + p_{0y}\cos\theta_i \end{bmatrix} \tag{1}$$

Then the rotation operator $p_i(\theta_i)$ is defined as: $p_i(\theta_i)=\left\{\left(p_{0x}\cos\theta_i - p_{0y}\sin\theta_i, p_{0x}\sin\theta_i + p_{0y}\cos\theta_i\right) \middle| p_0 \in p\right\}$.

In the 2DIBPP, there are totally n pieces to pack and the piece set is $P=\{p_1,p_2,...,p_n\}, i=1,...,n$. For any piece $p_i \in P$, denote its area as s_i. The length of the bins utilised is L and the width is W. The bin number is big enough to carry all of the pieces. The purpose is to pack these pieces into bins to find the minimum bin number, denoted as N. The bin set is denoted as $B=\{b_1,b_2,...,b_N\}, j=1,...,N$. In addition, let the quantity of pieces packed into the j_{th} bin be n^j. Let the reference point of piece p_i be r_{p_i}, then the reference point set of n pieces is $RP=\{r_{p_1},r_{p_2},...,r_{p_n}\}, i=1,...,n$. The piece reference point is designated as the lower-left vertex of the external envelope orthogon when a piece is rotated by an θ_i angle, as is shown in Fig. 3(a).

As mentioned, piece packing is the basis for cutting operations. According to the requirements of the cutting process, a defined slit distance should be reserved between different pieces and between the pieces and the boundaries of the bin. Let the slit distance between any pieces p_a and p_b be $d_1 = dist(p_a,p_b)$. Let $\partial rect(W,L)$ denotes the edge of the bin and $intrect(W,L)$ represents the interior of the bin, and let the distance between piece p_a and the edge of the bin be $d_2 = dist(p_a,\partial rect(W,L))$. The method of judging the distance between two pieces and between pieces and the bin is described in Section 3.3.

Based on the above definitions, the piece p_a whose reference point r_{p_a} is placed at v_{t_a} with a θ_a rotation angle can be expressed as $p_a^{v_{t_a}}(\theta_a) := p_a^{\theta_a} \oplus v_{t_a}$. Thus, for any given piece set, the solution to the 2DIBPP consists of four elements: the bin amount required to pack every piece, the type along with number of pieces allocated to each bin, the rotation angle, and the placement position of each piece's reference point.

Based on the above contents, we then defined the problem as follows.

$$\min N \tag{2}$$

subject to

$$dist\left(p_a^{\theta_a} \oplus v_{t_a}, p_b^{\theta_b} \oplus v_{t_b}\right) \geq d_1,\ 1 \leq a \leq b \leq n \tag{3}$$

$$dist\left(p_a^{\theta_a} \oplus v_{t_a}, \partial rect(W,L)\right) \geq d_2,\ 1 \leq a \leq n \tag{4}$$

$$p_a^{\theta_a} \oplus v_{t_a} \subseteq intrect(W,L),\ 1 \leq a \leq n \tag{5}$$

$$\theta_a \in \vartheta_a \ and\ \theta_a \in [0,2\pi],\ 1 \leq a \leq n \tag{6}$$

$$r_{p_a} \in \mathbb{R}^2,\ 1 \leq a \leq n \tag{7}$$

$$N \in Z+ \tag{8}$$

As can be expected, it is possible to get solutions with the same quantity of bins. Considering the reuse of the residuals, we cut the least utilised bins horizontally or vertically to disconnect the unused portion of the bins for future use. Eqs. (9)-(11) are introduced to select a better solution from results with an equal amount of bins, looking up the research of Lopez-Camacho et al. (2013) and Han et al. (2013).

$$U_j = \frac{\sum_{j_m=1}^{n^j} s_{j_m}}{L \times W} \tag{9}$$

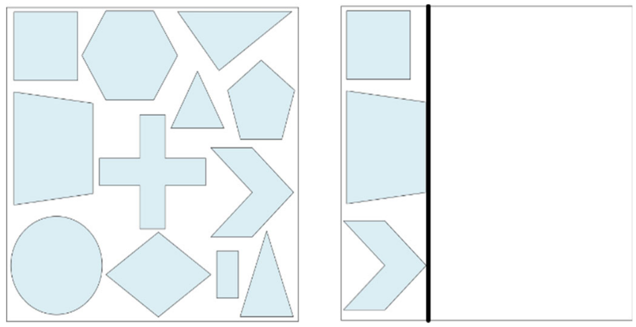
$$F = \frac{\sum_{j=1}^{N} U_j^2}{N} \tag{10}$$

$$K = N - 1 + P^* \tag{11}$$

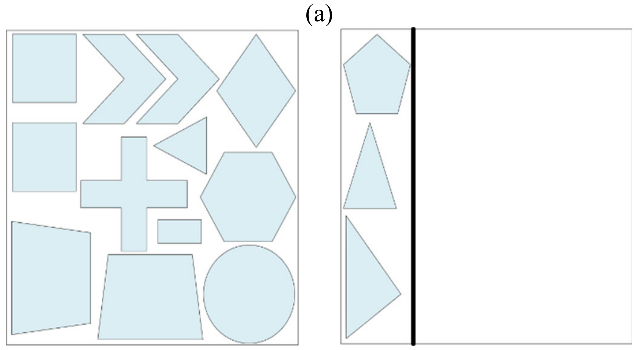
U_j represents the utilisation rate of the j_{th} bin, s_{j_m} is the area of the m_{th} piece in the j_{th} bin. Let the m_{th} piece in the j_{th} bin have t vertices, namely $p_{j_m}^1(x_1,y_1)$, $p_{j_m}^2(x_2,y_2)$, … , $p_{j_m}^t(x_t,y_t)$, so that the formula for s_{j_m} is shown in Eq. (12).

$$s_{j_m} = \frac{1}{2}\left(\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & y_2 \\ x_3 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_{t-1} & y_{t-1} \\ x_t & y_t \end{vmatrix} + \begin{vmatrix} x_t & y_t \\ x_1 & y_1 \end{vmatrix}\right) \tag{12}$$

P^* is the percentage of utilisation corresponding to the lowest utilised bin after the bin has been disconnected vertically or horizontally. Clearly, in results with an equal quantity of bins, from the point of material reuse, we want to keep the solution with F as large as possible and K as small as possible, i.e., a solution with the smaller P^* will be selected. It is clear that the solution corresponding to Fig. 1(b) has a smaller P^* than Fig. 1(a), so we tend to retain the result of Fig. 1(b).



(a)



(b)

Fig. 1. Diagrammatic drawing of the preferred solution selection process

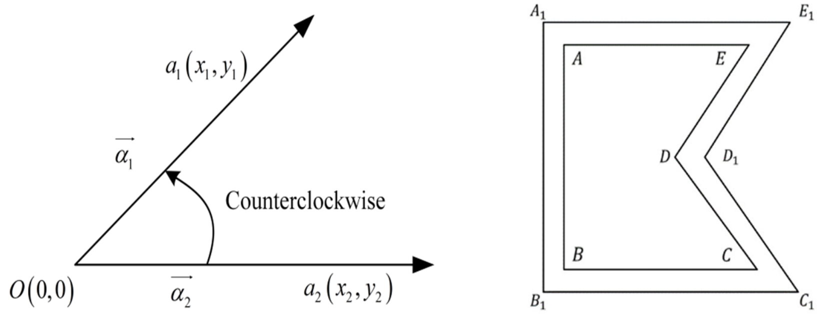
3. Geometric preprocessing

3.1. Equidistant edge expansion caused by considering slit distance

During the practical cutting process, slit distances d_1 and d_2 should be reserved. According to the relevant industrial production guidelines, the relationship between d_1 and d_2 could satisfy the following expression: $d_2 = 1/2\, d_1$. In the packing process, edges of irregular pieces are expanded by $1/2\, d_1$ outwards in translation. The expanded piece is extended so that each set of adjacent edges intersects at a point, completing the equidistant edge expansion of the piece.

As to the equidistant edge expansion of a piece, Hansen & Arbab (1992) proposed a proven method called the pair-wise offset method. The method is generally divided into three stages. First, offset each side of the polygon in the equidistant direction, check each equidistant line's self-intersection, and finally remove the invalid intersection loops. This method can satisfy the general equidistant offset problem.

In the process of piece equidistant offset, the subsequent cutting process determines that the distance of outward expansion of pieces is small. Therefore, an offset algorithm is proposed for the equidistant edge expansion of pieces. The coordinate values of each vertex of the pieces are known, and the coordinate values after the offset of each side can be found using the principle of geometric calculation. In this process, the vector cross product in geometry is used, which can be used to judge the relative position of two line segments. We can see from Fig. 2(a) that the cross product of vector $\overrightarrow{\alpha_1}$ and $\overrightarrow{\alpha_2}$ is $\overrightarrow{\alpha_1} \times \overrightarrow{\alpha_2} = (x_1 y_2 - x_2 y_1) = -\overrightarrow{\alpha_2} \times \overrightarrow{\alpha_1}$. If $\overrightarrow{\alpha_1} \times \overrightarrow{\alpha_2} < 0$, $\overrightarrow{\alpha_1}$ lies in the counterclockwise direction of $\overrightarrow{\alpha_2}$, as is shown in Fig. 2(a). If $\overrightarrow{\alpha_1} \times \overrightarrow{\alpha_2} > 0$, vector $\overrightarrow{\alpha_1}$ lies in the clockwise direction of $\overrightarrow{\alpha_2}$. If $\overrightarrow{\alpha_1} \times \overrightarrow{\alpha_2} = 0$, the vectors $\overrightarrow{\alpha_1}$ and $\overrightarrow{\alpha_2}$ are co-linear.



(a) (b)

Fig. 2. (a) Schematic of vector cross product, **(b)** Schematic of equidistant offset of piece *ABCDE*

When the equidistant offset of a piece is carried out, firstly, all the vertex coordinates of the piece will be traversed, and then each pair of adjacent edges will be regarded as a vector, and the vector cross product will be performed. At the same time, the bisector vector of the angle between adjacent edges should be calculated. After the trigonometric operation of the angle and the cross product sign judgement of the adjacent edge vectors, the coordinates of the offset point of each vertex can be gotten. Fig. 2(b) shows the results of piece $ABCDE$ after the equidistant offset operation. Algorithm 1 shows the pseudocode of the equidistant offset method.

Algorithm 1: The equidistant offset method of pieces

Input: Coordinates of each vertex of pieces before processing
Output: Coordinates of each vertex of pieces before processing

for each point v in piece p_a
  Calculate the vectors p_1, p_2 *of the adjacent edges where point v is located*
  Calculate the cross product of the adjacent edge vectors, denoted as φ
  Calculate the unit vector of the angle bisector of the angle between adjacent side vectors, denoted as α
  Calculate the sine value of the half angle
 Find the offset extension of the angle bisector
 if $\varphi > 0$
   Vector p_1 *lies in the clockwise direction of* p_2
   Find the coordinates of the offset point of the point v
 else
   Vector p_1 *lies in the counterclockwise direction of* p_2
   Find the coordinates of the offset point of the point v

3.2. The generation of NFP and IFP

In this paper, non-fit polygon (NFP) (Burke et al., 2006) as well as inner fit polygon (IFP) (Sato et al., 2019) is adopted to ensure no overlap between pieces and that the pieces will not extend beyond the edges of a bin. For arbitrary pieces p_a and p_b , let their reference points be r_{p_a} and r_{p_b} respectively. The $NFP_{p_a p_b}$ represents a region where p_b will overlap with p_a if r_{p_b} lies inside it. The process of generating $NFP_{p_a p_b}$ is described as follows: fix p_a and slide p_b on the outer contour of p_a. During the sliding process, p_b and p_a keep in contact but do not overlap, and the direction of p_b should not be changed. The final graph generated by the trajectory of r_{p_b} is defined as $NFP_{p_a p_b}$. Introducing the $NFP_{p_a p_b}$ is vital in that the positional relationship between two pieces is converted into the positional relationship between a point and a piece, dramatically reducing the computation and analysis difficulty. The positional relationship between r_{p_b} and $NFP_{p_a p_b}$ corresponding to the positional relationship between p_b and p_a is described as follows.

(1) If and only if $r_{p_b} \in intNFP_{p_a p_b}$, p_b overlaps p_a.
(2) If and only if $r_{p_b} \in \partial NFP_{p_a p_b}$, p_b and p_a contact but do not overlap.
(3) If and only if $r_{p_b} \notin intNFP_{p_a p_b}$ and $r_{p_b} \notin \partial NFP_{p_a p_b}$, p_b is separated from p_a.

To make the best use of space during the packing process, we hope that p_b and p_a are arranged tangentially after the equidistant offset. Therefore, points on the outline of $NFP_{p_a p_b}$ are regarded as the ideal placement positions for r_{p_b} .

Similarly, given a bin b and a piece p_b , NFP_{bpb} , also known as IFP_{bpb} , can be developed by translating p_b one turn throughout the interior boundary of b . Using the IFP_{bpb} , the relative position of the pieces to the bin's edges can be determined, thus ensuring that each piece is packed strictly within the bin. The generation processes of $NFP_{p_a p_b}$ and IFP_{bpb} are presented in Fig. 3(b) and Fig. 3(c), and the detailed facts can be observed in Bennell and Oliveira (2009).

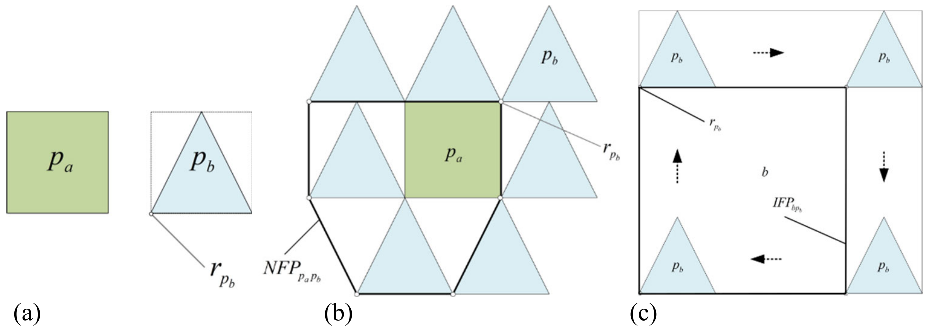


Fig. 3. (a) Piece p_a and Piece p_b , **(b)** The generation process of $NFP_{p_a p_b}$, **(c)** The generation process of IFP_{bpb}

3.3. Penetration depth and penetration vector

In order to measure the overlap created during the piece packing process, the penetration depth and penetration vector (Zhang et al., 2022; Leung et al., 2012) are introduced in this paper. When pieces p_a and p_b intersect, we illustrate the penetration depth $PD(p_a, p_b)$ as the minimal distance travelled to dissociate p_a and p_b .

Definition 3 (penetration depth and penetration vector). The penetration depth between two intersecting pieces p_a and p_b can be defined as $PD(p_a, p_b) = min\{\| v \| \mid p_a \cap (p_b \oplus v) = \varnothing\}$, and the corresponding vector v is called penetration vector. The symbol $\| v \|$ represents the 2-norm of the penetration vector.

The $PD(p_a, p_b)$ can be obtained from $NFP_{p_a p_b}$. If the reference point r_{p_b} of p_b is in the interior of $NFP_{p_a p_b}$, $PD(p_a, p_b)$ is equal to the minimum distance from r_{p_b} to $NFP_{p_a p_b}$. According to Definition 3, if two pieces do not intersect, the value of penetration depth is 0. Similarly, $PD(p_a, p_b)$ denotes the shortest moving distance that brings p_b completely into bin b , so we can determine $PD(p_a, p_b)$ by calculating the shortest distance between reference point r_{p_b} and the boundary of bin b .

This research utilises the square of the penetration distance for measuring overlap. Let the set of pieces packed into bin b_j be P_j , the set of piece reference point placement positions be V_j and the set of piece rotation angles be R_j . We denote $h_{ab}(P_j, V_i, R_i)$ as the overlap between p_a and p_b , which can be denoted using the penetration depth as $h_{ab}(P_j, V_j, R_j, b_j) = PD^2(p_a^{\theta_a} \oplus v_{t_a}, p_b^{\theta_b} \oplus v_{t_b})$. In the same way, the overlap between p_a and b_j can be described as $k_a(P_j, V_j, R_j, b_j) = PD^2(b_j, p_a^{\theta_a} \oplus v_{t_a})$. Then the expression of the overlap is shown in Eq. (13).

$$Overlap(P_j, V_j, R_j, b_j) = \sum_{1 \le a < b \le n^j} h_{ab}(P_j, V_j, R_j, b_j) + \sum_{1 \le a \le n^j} k_a(P_j, V_j, R_j, b_j) \tag{13}$$

4. Assignment strategy

In studying the 2DIBPP, such an approach is often adopted: packing pieces into a bin until it is full and cannot contain more pieces, then closing it and opening a new one to carry on the packing process (Parreno et al., 2010). In this paper, we first assign pieces to bins and then pack the pieces into every single bin. That is, the assignment and packing are independent of each other, and the assignment process is executed before packing. As the assignment is mainly based on piece area, and pieces vary in shape, especially when concave surfaces are present, inevitably, one or more pieces cannot be packed into the specified bin. Given this situation, we also design corresponding modified measures.

The general framework of our method is adapted from Martinez-Sykora et al. (2017).

4.1. 1DBPP

Firstly, each piece is approximately represented by its area s_i, then the First Fit Decreasing (FFD) (Kang & Park, 2003) algorithm is used to obtain the maximal bin number needed to pack every piece, designated as N_{max}. The subsequent binary variables are defined: $e_j, \forall j = 1, 2, ..., N_{max}$. If bin b_j is used in the packing process, the value of e_j is 1, otherwise 0; $w_{ij}, \forall i = 1, 2, ..., n, \forall j = 1, 2, ..., N_{max}$. If piece p_i is packed into b_j, the value of w_{ij} is one, otherwise zero. Then the integer programming (IP) model for distributing pieces to different bins can be expressed as 1DBPP, described as follows.

$$\min \sum_{j=1}^{N_{max}} e_j \tag{14}$$

subject to

$$\sum_{i=1}^{n} s_i w_{ij} \leq LW \qquad 1 \leq j \leq N_{max} \tag{15}$$

$$w_{ij} \leq e_j \qquad 1 \leq i \leq n, 1 \leq j \leq N_{max} \tag{16}$$

$$\sum_{j=1}^{N_{max}} w_{ij} = 1 \qquad 1 \leq i \leq n \tag{17}$$

$$e_j \in \{0,1\}, \ w_{ij} \in \{0,1\} \qquad 1 \leq i \leq n, 1 \leq j \leq N_{max} \tag{18}$$

The formula (14) minimizes the bin volume used to pack all the pieces. Constraint (15) ensures that the pieces' total area packed into a bin will not exceed the bin's space. Constraint (16) ensures that if a piece is packed into a specified bin, that bin will appear in the eventual solution. Constraint (17) makes sure that every piece will be packed into a bin, i.e., all pieces will be involved in the packing process. Moreover, constraint (18) ensures that e_j and w_{ij} are binary variables. It is worth noting that this solution would be optimal if we find an optimal solution to this IP model and successfully pack the pieces allocated to each bin into it.

4.2. Piece assignment strategy

In this section, we use the PBP (Martinez-Sykora et al., 2017) strategy to assign pieces bins, which allocates pieces to bins through working out an IP model of the 1DBBP.

In the assignment process, the greedy method tends to assign smaller pieces first, which allows for a larger number of pieces to be assigned to a bin but leaves larger pieces behind, ultimately resulting in an overall poor solution and secondary allocation. As a result, the PBP averts this circumstance by concentrating on allocating pieces to only one bin and employs an objective function which tends to assign bigger pieces first, as is shown in formula (19). s_{max} is the piece area with the largest area in the piece set P. w_i is a two-valued variable that equals one when p_i is allocated to the bin and zero otherwise. In fact, piece allocation is a knapsack problem, aiming to maximize the piece area assigned to a bin.

$$\max \sum_{i=1}^{n} \left(\frac{s_i}{s_{max}} \right)^2 w_i \tag{19}$$

$$\sum_{i=1}^{n} s_i w_i \leq LW \qquad 1 \leq i \leq n \tag{20}$$

$$w_i \in \{0,1\} \qquad 1 \leq i \leq n \tag{21}$$

This section also briefly describes another direct construction heuristic (DCH) strategy. Unlike PBP, DCH does not pre-assign pieces but solves piece allocation in the packing process. DCH directly packs pieces into bins in a given sorting order according to these pieces' area, using the packing approach in Section 5. The steps of the DCH are described below.

(1) Arrange all pieces in a non-increasing sequence of area, and open a bin.
(2) For each piece p_i, pack it into the open bin using the packing algorithms. If a bin is filled, close it and open a new one.
(3) Repeat procedure (2) until all pieces are arranged into the bins.

5. Packing algorithms

5.1. Getting a definite collection of promising rotation angles

In this paper, we consider two cases regarding the rotation of pieces. Firstly, we let the pieces rotate by some limited rotation angles, denoted as ϑ^{limit}, and the detailed information of these angles can be seen in Table1 from Martinez-Sykora et al. (2017). Next, we consider the case where the pieces can rotate freely.

In the first case, we generate a feasible placement position set using the packing algorithms and record the best results for each rotation angle set.

There are many possible rotation angles in the free rotation case, and only a subset can be assessed. Based on the mechanism introduced by Abeysooriya et al. (2018), we propose a two-stage method to reduce infinite rotation angles to a finite collection, whose principle is to identify a promising angle subset in accord with the piece distribution in the present partial solution.

In the first stage, pack the first piece p_a. Place the longest side of the piece parallel to the bin's edge. Choose a random type of discharge if there is more than one longest edge.

In the second stage, pack the other pieces. We start this stage by packing piece p_b in a momentary position and angle using the method in the first case, ensuring that p_b touches the boundary of the pieces placed earlier and the best rotation angle is selected. As p_b touches the boundary of other pieces, two new orientation angles will be defined by each touching point or edge.

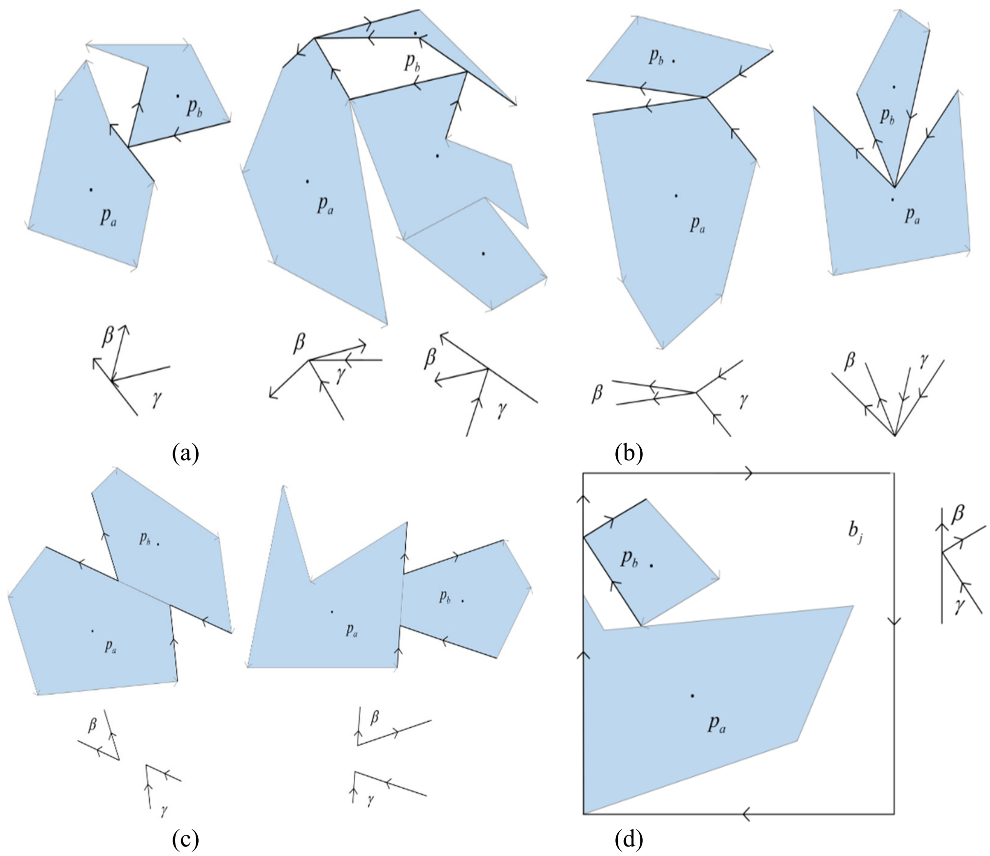


Fig. 4. (a) An edge-vertex combination, **(b)** A vertex-vertex combination, **(c)** An edge-edge combination, **(d)** Angles generated by the bin's edge

An edge-vertex combination, a vertex-vertex combination and an edge-edge combination can be seen in Fig. 4. Piece p_a consists of counter-clockwise direction of edges, while p_b is composed of clockwise direction of edges. In each case, p_b is rotated by an angle β in the counter-clockwise direction and an angle γ in the clockwise direction. In other words, if the direction of edges directs away from the point or edge of contact, the piece is rotated counter-clockwise by an angle of β. If the direction of edges directs towards the point or edge of contact, the piece is rotated clockwise by an angle γ. Two independent touching points and consequent four new orientations can be seen in the second example of Fig. 4(a). The above constitute new candidate rotation angles, and according to the placement program, the best placement angles and positions for pieces of these angles are obtained. These are compared with the short-lived placement positions and rotation angles. If new angles fail to provide a superior position, the packing algorithms take the momentary position and rotation angle as the final result. If p_b contacts the bin's edge, the angles generated by the bin's edge will also be considered, as displayed in Fig. 4(d).

Through this method, a definite collection of promising rotation angles ϑ^{free} is obtained, narrowing the scope of the search neighbourhood.

5.2. Placement of a single bin

This paper uses the overlap minimization method (Elkeran, 2013; Zhang et al., 2022; Liu et al., 2020; Leung et al., 2012) to pack assigned pieces into bins, which is widely used in solving the 2DISPP.

5.2.1. Initial solution generation using BL algorithm

We first use the BL algorithm in Algorithm 3 to produce an original layout for every piece. Algorithm 3 attempts to place the allocated pieces into one of the bins, and the candidate placement positions of the pieces are obtained using NFP. When the BL algorithm is called, the length of the bin is regarded as infinite, so it is possible to produce an initial layout where pieces are discharged beyond the boundary of the bin. Under this condition, we consider that the pieces overlap the bin. We use *Overlap* to measure the overlap. To achieve a reasonable layout, we first randomly select two pieces from the pieces packed in the bin, then switch their positions by invoking the *Swap* algorithm and minimize the *Overlap* using the *Separation* algorithm. If and only if *Overlap* equals zero, the solution will be accepted. At this time, a constant *True* will be returned, otherwise *False*. Additionally, the iterations of random search are controlled by *number*. Algorithm 2 describes the overlap minimization method in detail.

Algorithm 2: Pack the pieces in the piece set P_j assigned to bin b_j to obtain the final position V_j and rotation angle R_j of the pieces

---

Input: The assigned piece set P_j and the bin b_j

Output: The final position V_j and rotation angle R_j

$(V_j, R_j) = BL(P_j, b_j)$
if $Overlap(P_j, V_j, R_j, b_j) == 0$
  return True
for $num = 1$ *to* $number$ *do*
  Randomly select p_a *and* p_b *from* P_j
  $(V_j^{'}, R_j^{'}) = Swap(p_a, p_b, P_j, V_j, R_j, b_j)$
  $(V_j^{''}, R_j^{''}) = Separate(P_j, V_j^{'}, R_j^{'}, b_j)$
  if $Overlap(P_j, V_j^{''}, R_j^{''}, b_j) < Overlap(P_j, V_i, R_i, b_j)$
    $(V_j, R_j) = (V_j^{''}, R_j^{''})$
  if $Overlap(P_j, V_j, R_j, b_j) == 0$
    return True
return False

Algorithm 3 details the process of generating initial positions and angles for the pieces using the BL algorithm. Before this, the pieces are first arranged in a non-increasing order of area to obtain a corresponding order and then placed in the bin one by one in this order. For each piece in the piece set P_j, the set Q_1 of vertices and intersections of the $NFP_{p_{j_m}p_k}$ is used as candidate placement positions, and the leftmost bottom point with no overlap (discounting the overlap between the piece and the right edge of the bin) is chosen for each piece. θ_{j_m} symbolizes the rotation angle of the m_{th} piece from b_j.

Algorithm 3: Generate the initial positions of the pieces in the piece set P_j in the bin b_j

---

Input: The assigned piece set P_j and the bin b_j

Output: The original position V_j and rotation angle R_j

for $j_m = 1$ *to* n^j *do*
  for each $\theta_{j_m} \in \vartheta_{j_m}^{limit}$ *or* $\theta_{j_m} \in \vartheta_{j_m}^{free}$ *do*
    for $k = 1$ *to* $j_m - 1$ *do*
      Get the vertices and intersection $(denote\ Q_1)$ *of* $NFP_{p_{j_m}p_k}$
  $(V_j, R_j) = BL(Q, P_j, b_j)$
  return (V_j, R_j)

5.2.2. Swap algorithm

The initial solution generated by the BL algorithm has a non-zero overlap in many cases, which requires using the *Swap* algorithm (Leung et al., 2012) to exchange the positions of two pieces. Meanwhile, the *Separation* algorithm is used to minimize the *Overlap*. Pieces p_a and p_b are randomly selected from P_j, and the *Swap* algorithm attempts to swap their positions and return the new position of each piece in P_j. This is done by first removing p_a from b_j by setting its position to $(+\infty,+\infty)$, then moving p_b to the original position of p_a with the smallest overlap, and then reinserting p_a into b_j with the smallest overlap. The *Swap* algorithm is shown in Algorithm 4.

Algorithm 4: Randomly swap p_a and p_b in b_j

Input: Pieces p_a and p_b from P_j, bin b_j
Output: The position V_j and rotation angle R_j after the swap operation

1 $\left(V_j,R_j\right)|_{j=a}=\left(r_{p_a},\theta_a\right)=\left(+\infty,+\infty\right)$
2 $\left(V_j,R_j\right)=Move\left(p_b,P_j,V_j,R_j,b_j\right)$
3 $\left(V_j,R_j\right)=Move\left(p_a,P_j,V_j,R_j,b_j\right)$
4 *return* $\left(V_j,R_j\right)$

The method *Move* (Leung et al., 2012) is called in Algorithm 3, which moves p_a and p_b from their original positions to new ones with less overlap. Taking the second line in algorithm 3 as an example, in order to avoid a situation where piece p_b has not been moved, *Overlap* needs to be recalculated, as shown in Eq. (22).

$$Overlap(p_b,P_j,V_j,R_j,b_j)=Overlap(P_j,V_j,R_j,b_j)+Overlap\left(p_b,p_b^{'}\right) \tag{22}$$

In equation (22), $p_b^{'}$ is defined as the piece after p_b has been moved, and $Overlap\left(p_b,p_b^{'}\right)$ represents the overlap between p_b and $p_b^{'}$. Adding this term makes sure that p_b will not stay at its original position, thus ensuring its effective movement. For each rotation of p_b, the set of vertices and midpoints on the edges of $NFP_{p_a p_b}$ are considered as candidate positions. The position and angle with the least overlap is chosen as the ultimate position of p_b. Algorithm 5 shows the *Move* procedure.

Algorithm 5: Move piece p_b in bin b_j to minimize overlap

Input: Piece p_b to be moved in piece set P_j, bin b_j
Output: The position V_j and rotation angle R_j of the pieces after the movement of p_b

1 $minOverlap=+\infty$
2 *for each* $\theta_b\in\vartheta_b^{limit}$ *or* $\theta_b\in\vartheta_b^{free}$ *do*
3 *for* $j_m=1$ *to* n^j *do*
4 *Get the vertices and midpoint edges of* (*denote* Q_2) *of* $NFP_{p_b p_{j_m}}$
5 *Get the vertices and midpoint of edges* (*add to* Q_2) *of* $IFP_{p_b b_j}$
6 *for each point* $q\in Q$ *do*
7 $\left(V_j^{'},R_j^{'}\right)=\left(V_j,R_j\right)$
8 $\left(V_j,R_j\right)|_{j=a}=\left(r_{p_b},\theta_b\right)=\left(q,\theta_b\right)$
9 *if* $Overlap(p_b,P_j,V_j,R_j,b_j)$ < $minOverlap$
10 $minOverlap=Overlap(p_b,P_j,V_j,R_j,b_j)$
11 *else*
12 $\left(V_j,R_j\right)=\left(V_j^{'},R_j^{'}\right)$
13 *return* $\left(V_j,R_j\right)$

5.2.3. Separation algorithm

This part employs the *Separation* algorithm to detach the overlapping pieces after the *Swap* operation, which is proposed by Leung et al. (2012). By fixing the rotation R_j in Eq. (13), we can obtain Eq. (23), in which we omit P_j and b_j for the sake of expression.

$$Overlap\left(V_j\right)=\sum_{1\le a<b\le n^j} h_{ab}\left(V_j\right)+\sum_{1\le a\le n^j} k_a\left(V_j\right) \tag{23}$$

An unconstrained non-linear programming model is utilised to deal with this problem. The following are the detailed steps.

(1) Express the $h_{ab}\left(V_j\right)$ and $k_a\left(V_j\right)$ in terms of penetration depth as follows.

$$\begin{aligned} h_{ab}\left(V_j\right) &= \left(PD\left(p_a \oplus v_{t_a}, p_b \oplus v_{t_b}\right)\right)^2, 1\le a<b\le n^j \\ k_a\left(V_j\right) &= \left(PD\left(p_a \oplus v_{t_a}, b_i\right)\right)^2, 1\le a\le n^j \end{aligned} \tag{24}$$

(2) The penetration vector v is used to replace the penetration depth in step 1 and is denoted as $\nabla h_{ab}\left(V_j\right)=\left(\nabla_1 h_{ab}\left(V_j\right),\nabla_2 h_{ab}\left(V_j\right),...,\nabla_{n^j} h_{ab}\left(V_j\right)\right)$, in which ∇ is the gradient symbol and $\nabla_t=\left(\partial/\partial V_{x_t},\partial/\partial V_{y_t}\right),1\le t<n^j$. Thus, $h_{ab}\left(V_j\right)$ and its gradient can be expressed in terms of $\nabla f_{ab}\left(V_j\right)$, $1\le a\le b\le n^j$ as follows.

$$\begin{aligned} h_{ab}\left(V_j\right) &= \| v \|^2 \\ \nabla_a h_{ab}\left(V_j\right) &= -\nabla_b f_{ab}\left(V_j\right) = 2v \\ \nabla_t h_{ab}\left(V_j\right) &= 0,\ t\in\left\{1,...,n_i\right\}\setminus\left\{m,n\right\} \end{aligned} \tag{25}$$

Similarly, $k_a\left(V_j\right)$ and its gradient can be expressed as follows.

$$\begin{aligned} k_a\left(V_j\right) &= \| v \|^2 \\ \nabla_a g_a\left(V_j\right) &= 2v \\ \nabla_t k_a\left(V_j\right) &= 0,\ t\in\left\{1,...,n^j\right\}\setminus\left\{m\right\} \\ \nabla k_a\left(V_j\right) &= \left(\nabla_1 k_a\left(V_j\right),\nabla_2 k_a\left(V_j\right),...,\nabla_{n^j} k_a\left(V_j\right)\right) \\ \nabla_t &= \left(\partial/\partial V_{x_t},\partial/\partial V_{y_t}\right),1\le t<n^j \end{aligned} \tag{26}$$

(3) The limited memory BFGS (L-BFGS) (Boggs & Byrd, 2019) method is invoked in steps 1 and 2 to solve the unrestricted nonlinear programming problem (23) to get the minimum value.

6. Local search algorithm for solution improvement

In this section, we use a local search (LS) algorithm to better the final solution, modified from LS2 by Martinez-Sykora et al. (2017). Combining Eq. (9) to Eq. (11), we can see that the F value increases when either of the two cases occurs. First, the number of bins used is reduced. Second, the number of bins remains unchanged, but the utilisation of one of the bins increases. In both cases, the new solution is accepted.

Given a solution containing N bins, sort these bins in a non-decreasing sequence of use ratio, i.e., $U_1\le U_2\le\cdots\le U_N$. In the LS algorithm, we first select the least utilised bin b_c and then try to move pieces from b_c into a collection of bins $B'=\left\{b_{d_1},b_{d_2},...,b_{d_z}\right\},z<N,B'\subseteq B\setminus\left\{b_c\right\}$ rather than just one bin, which distinguishes this algorithm from LS1 (Martinez-Sykora et al., 2017). This method, therefore, increases the possibility of moving pieces from b_c to other bins.

The steps of the algorithm are as follows. Firstly, the pieces in b_c are sorted in a decreasing sequence of area. To each bin in B', the pieces are disposed in the non-decreasing order of area. For each bin b'_{d_k}, piece p_{d_k} is removed in the order in which they have been sorted, so that an empty space is created, and each piece in b_c is then considered to be moved to this empty area. Starting from the first piece, the pieces in b_c are moved in turn to b'_{d_k} using the packing algorithms. The other pieces in b_c are then tried in turn. If no bin in B' can hold a particular piece, it is temporarily stored in the set $Array_2$. Once all the pieces in b_c have been considered, we consider moving the pieces removed from each bin back to their original place. Again, if the movement is unsuccessful, the piece is also temporarily stored in $Array_2$.

In this case, we start to check if any bin in B' has increased in utilisation. If none of these bins has been utilised more, the move is invalid. If $Array_2$ is empty, one bin has been reduced, and the new solution is accepted. Otherwise, the pieces in $Array_2$ are loaded into a new bin, and the above process is repeated, i.e., one piece is loaded into one bin at a time in the non-increasing sequence of its area. If a bin cannot maintain all the pieces from $Array_2$, the new solution requires one more bin than the original one and is discarded. Meanwhile, a $False$ is returned. If a bin can hold all the pieces from $Array_2$ and the final F value increases, the new solution is accepted, and a $True$ is returned. Algorithm 6 shows the procedure of the LS algorithm.

Algorithm 6: Using a LS strategy to better the solution

Input: Bins $b_c, b_{d_1}, b_{d_2}, ..., b_{d_z}$ ($U_c \le U_{d_1} \le U_{d_2} \le ... \le U_{d_z}$) and the pieces packed inside

Output: z or $z+1$ new bins that pack the same piece set with a larger value of F

$Array = b_c, b'_{d_1} = b_{d_1}, b'_{d_2} = b_{d_2}, ..., b'_{d_r} = b_{d_r}, b_c = \varnothing$
$improved = True$
while improved do
  $improved = False$
  for $k = 1$ *to* z *do*
    for each piece p_{d_k} *in* b'_{d_k}
      $s_{out} = s_{p_{d_k}}, s_{in} = 0$, *and remove* p_{d_k} *from* b'_{d_k}
      $Array_2 = Array$, *add* p_{d_k} *to the end of Array*
      for each piece p_a *in* $Array_2$ *do*
        Use the packing algorithm to see whether p_a *can be placed into* b'_{d_k}
        if p_a *can be placed*
          $s_{in} = s_{in} + s_{p_a}$, *remove* p_a *from* $Array_2$ *and update* b'_{d_k}
        else
          Study the next piece
      if $s_{in} > s_{out}$
        if $Array_2 == \varnothing$
          return $b'_{d_1}, b'_{d_2}, ..., b'_{d_z}$ (*a sulution with one bin less have been found*)
        else
          $Array = Array_2$, *sort the array by non-increasing area,* $b_{d_k} = b'_{d_k}$, $improved = True$
      else
        $b_{j_k} = b'_{j_k}$
  if the number of pieces in Array has decreased
    Use the packing algorithm to pack pieces from Array into b_c, *and construct a new bin* b'_c
    if all pieces can be packed
      return True, return $b_c, b'_{d_1}, b'_{d_2}, ..., b'_{d_z}$
    else
      return False
  else
    return False

By analysing the above process, we can get that the LS algorithm starts from the least utilised bin and considers each bin in turn. Whenever we find an improvement, we start with the new solution and re-examine all the bins until we finally get a better result.

7. Interpretation of experimental results

This part exhibits the experimental process and analyses the data. The algorithms in this paper are programmed in Python, with a 2.90 GHz processor and 8 GB of RAM to implement them.

7.1. Data

We have used two benchmark instances, the nesting instances that are widely used in the 2DISPP, and the instances offered by López-Camacho et al. (2013). They are available from the ESICUP (EURO Special Interest Group on Cutting and Packing) website https://www.euro-online.org/websites/esicup/.

There are 23 datasets in the nesting instances, and the jigsaw puzzle instances consist of two subsets, JP1 and JP2. Subset JP1 is made up of 540 instances, divided into 18 classes with 30 problems, and all the pieces are convex. JP2 has a collection of 480 instances (18 classes with 30 problems), and there are convex and non-convex pieces. In addition, the number of pieces in each class is different. In the jigsaw puzzle instances, the bins' length and width are set to 1000 fixedly. In contrast, the sizes of the nesting instance bins are needed to be defined.

Referring to the paper of Martinez-Sykora et al. (2017), for each instance, three kinds of square bins are defined depending on the maximal dimension of the enclosing rectangle of the pieces in their initial orientation, represented as $d_{\max}$. In addition, pieces and bins are dimensioned in uniform units.

(1) Nest-SB (small bins). $W = L = 1.1\, d_{\max}$.
(2) Nest-MB (medium bins). $W = L = 1.5\, d_{\max}$.
(3) Nest-LB (large bins). $W = L = 2\, d_{\max}$.

Due to the three kinds of bins, 69 instances of bin packing are produced from the 23 nesting instances. Table 1 from Martinez-Sykora et al. (2017) exhibits the detailed information of these instances, including the piece number, the dimension of three bins and the limited rotation angles ϑ^{limit} for each instance.

7.2. Experimental results

7.2.1. Comparison between PBP and DCH

This section compares the results obtained by the PBP and DCH algorithms in Table 1. We present the average bin amount (N), the mean value of F and K , and the mean execution time T with the unit in second.

Table 1
Comparison between the PBP and DCH

Subsets	DCH				PBP			
	N	*F*	*K*	*T*	*N*	*F*	*K*	*T*
JP1	7.811	0.662	7.361	74	7.789	0.671	7.342	76
JP2	7.406	0.674	6.997	93	7.364	0.689	6.963	66
Nest-SB	9.352	0.389	8.899	99	9.340	0.391	8.872	96
Nest-MB	4.981	0.404	4.676	145	4.981	0.409	4.654	126
Nest-LB	2.994	0.376	2.587	327	2.994	0.387	2.583	288
Average	6.509	0.501	6.104	147.6	6.494	0.509	6.083	130.4

As far as indicator F is concerned, all the results obtained by the PBP algorithm are larger than those obtained by the DCH, although the gap in some instances is not very obvious. For N and K , PBP also gets better results, except for a few N values that agree with the results obtained by the DCH. As for the computation time T , compared with the DCH, most of which the PBP takes is shorter, except for the JP1 instance. The PBP outperforms the DCH in all parameters regarding the average value. PBP implements piece assignment by solving the 1DBPP one bin at a time and reduces the method's greediness by modifying the objective function. DCH does not pre-assign pieces and only places the pieces in the first suitable bin, which results in more effort to find a bin to arrange a piece, thus boosting the computing time. As a result, we come to the conclusion that it is beneficial to allocate pieces to bins before packing. In the process of actual industrial production, we can also consider using this strategy to improve the utilisation rate of plates. Because the experimental results have proved that the PBP is superior to the DCH, it is used in our subsequent studies and analyses.

7.2.2. Comparison between limited rotation angles and free rotation

Through the approach in Section 5.1, we get a finite set of promising rotation angles for each piece. To evaluate the effect of allowing pieces to rotate freely, the pieces are firstly rotated at limited angles of $90°$, $180°$ and $270°$, and the results are compared with those obtained by free rotation. Table 2 displays the results. The nesting instances are used in the comparison, as they provide angles at which each piece set can be rotated. In the *Limited angles* column, the angles allowed for each piece set are exhibited in Table 1 from Martinez-Sykora et al. (2017). The results obtained by free rotation are displayed in the *free rotation* column. The superior results are displayed in bold.

Table 2
Analysis of Nest-MB instances at different rotation angles

Instances	Limited angles		Free rotation		Instances	Limited angles		Free rotation	
	N	F	N	F		N	F	N	F
albano	3	0.467	3	**0.492**	poly5a	8	0.420	8	**0.426**
dighe1	2	**0.354**	2	0.271	poly2b	4	0.371	4	**0.378**
dighe2	1	**0.823**	2	0.222	poly3b	5	0.416	5	**0.422**
fu	4	0.421	4	0.421	poly4b	6	0.445	6	**0.451**
han	3	0.362	3	**0.366**	poly5b	7	0.459	7	**0.464**
jakobs1	4	**0.510**	4	0.482	shapes0	8	0.502	8	**0.546**
jakobs2	4	**0.383**	4	0.358	shapes1	7	0.255	6	**0.368**
mao	3	0.250	3	**0.280**	shapes2	10	**0.358**	10	0.350
poly1a	2	0.291	2	**0.300**	shirts	7	0.264	6	**0.368**
poly2a	4	0.323	4	**0.325**	swim	5	0.377	5	0.377
poly3a	5	0.399	5	**0.401**	trousers	3	0.535	3	**0.559**
poly4a	7	0.377	7	**0.381**	Average	4.87	0.407	4.826	0.392

As can be observed from Table 2, out of the 23 instances, better results are obtained for 16 sets in the *free rotation* column. The F values for the *fu* and *swim* instances are the same for both conditions. These demonstrate that free rotation can improve bin utilisation by giving pieces more viable placement options. Because of the specific nature of the production process, companies such as garments have strict requirements on piece placement angles. While for shipyards and other enterprises, because the primary purpose of piece packing is to improve the plate utilisation rate, there is no special requirement for the placement angles. We can conclude that when there is no practical reason to set a particular rotation angle, the free rotation will eliminate any deviation in creating the piece set to optimize the packing results. It is worth paying special attention to *dighe1* and *dighe2*. The F value obtained in the *Limited angles* column is much higher than that of the *free rotation* column. According to Table 1 from Martinez-Sykora et al. (2017), the two instances have a permissible rotation angle of $0°$, i.e., the pieces are allowed to be packed in their original orientation, which will develop the perfect or near-perfect solution. Furthermore, to avoid repetitive work, only medium-sized bins are selected by us, as this is sufficient to illustrate the superiority of the free rotation strategy.

7.2.3. *Comparison between PBP and LS-PBP*

In the following, we will compare the results of the two cases: only using the PBP and introducing the LS strategy depicted in Section 6. $F_{Inc}(\%)$ represents the percentage of improvement in indicator F for LS-PBP compared with PBP.

Comparing the results in Table 3, it is clear that the LS strategy effectively improves the results. Inevitably, the computational time increases exponentially due to the additional computational effort associated with introducing this strategy. We note that processing the Nest-SB and Nest-MB datasets takes much longer than processing the Nest-LB, and this is because for the same set, the larger the bin size, the fewer bins are required, so there are fewer opportunities to exchange pieces across bins, and thus the processing time will be relatively less. In addition, we can also observe that for $F_{Inc}(\%)$, the percentage of Nest-SB and Nest-MB increase are much higher than that of Nest-LB, which means the smaller the bin size, the better the effect of crossing bin to distribute pieces. Because the smaller-size bin can accommodate fewer pieces, the pieces' bin-cross movement will significantly change the final results.

Table 3
Comparison between PBP and LS-PBP.

SubSet	PBP				LS-PBP				
	N	F	K	T	N	F	K	T	$F_{Inc}(\%)$
JP1	7.789	0.671	7.342	76	7.788	0.705	7.321	182	5.07
JP2	7.364	0.689	6.963	66	7.382	0.719	6.947	196	4.35
Nest-SB	9.340	0.391	8.872	96	9.232	0.417	8.852	1366	6.65
Nest-MB	4.981	0.409	4.654	126	4.959	0.431	4.627	3279	5.38
Nest-LB	2.994	0.387	2.583	288	2.975	0.397	2.567	875	2.58
Average	6.494	0.509	6.083	130.4	6.467	0.534	6.063	1180	4.81

7.2.4. *Comparison with other literature*

Our findings are compared with those of Martinez-Sykora et al. (2017), Abeysooriya et al. (2018), Liu et al. (2020) and Zhang et al. (2022), which are recent developments in literature. Only the data of LS2-PBP, LJS1-MU and LJS3-MU, LS-4R and LocalSearch-WLFD are included in Table 4 because they were the best results in separate studies. As the above algorithms were written in different languages on different platforms, the experimental results are not comparable in terms of time. In addition, since the values of N and K were not explicitly listed in other literature, we only compare the F value obtained by different algorithms.

Table 4
Comparison with other literature about F .

Subset	LS2-PBP	LJS1-MU	LJS3-MU	LS-4R	LocalSearch -WLFD	LS-PBP (slit)	LS-PBP (no slit)
JP1	0.723	0.691	0.732	0.704	0.744	0.705	**0.746**
JP2	0.729	0.701	0.747	0.701	**0.777**	0.719	0.768
Nest-SB	0.432	0.393	0.403	0.442	0.445	0.417	**0.448**
Nest-MB	0.452	0.418	0.409	0.445	**0.482**	0.431	**0.482**
Nest -LB	0.403	0.425	0.410	0.438	0.443	0.397	**0.446**

As can be observed, the LS-PBP (slit) column represents the F value obtained when the cut distance is considered, and the results obtained by us are better than those of some of the above literature, but this advantage is not prominent. This is because, in our study, the pieces are not allowed to be packed as closely as possible as in all other papers. Instead, slit distance is reserved. Although the slit distance is small in relation to piece and bin dimensions, the effect of slit distance on the utilisation of the individual bin and the F value cannot be ignored when abundant pieces are packed. We also add the column LS-PBP (no slit), which describes the F value obtained when the slit distance is not taken into account, so we can compare the results with other papers under the same standard. In this case, the comparison results are more convincing. Our method achieves the optimal results in JP1, nest-SB, nest-MB as well as Nest-LB, and the gap with the optimal results in JP2 is very small. Thus, we draw the conclusion that our algorithms are somewhat competitive.

8. Conclusions

In this paper, we investigate a 2DIBPP that considers slit distance and allows pieces to rotate freely. Although this problem often arises in industrial production, there is no good approach to solve it. Better results are obtained using our methods, and practical industrial production could benefit from them. Specifically, we carry out the following work.

Firstly, we propose a mathematical model that considers the above two elements, and an equidistant edge expansion algorithm is introduced to handle the slit distance. Secondly, the 2DIBPP is divided into two sub-problems: piece allocation and packing. We also describe a DCH algorithm that handles the two sub-problems simultaneously and demonstrates the superiority of the PBP through comparative experiments. We then investigate the effect of free rotation versus limited angles on the nesting instances. Except for a few piece datasets, better solutions are found by allowing pieces to rotate, as the free rotation strategy eliminates any deviations in the creation of the piece dataset and thus optimises the packing results. In the packing process, a two-stage method is presented to obtain promising rotation angles for pieces, which effectively decreases the search neighbourhood and betters the results. Finally, we use the LS algorithm to improve the solution by adjusting piece allocation across bins. The data show that this strategy's introduction effectively enhances the packing results.

For future work, we plan to apply the methods in this paper to bins with irregular shapes. We also plan to investigate the problem of packing irregular pieces into bins with some defective regions.

Acknowledgments

This research was supported by the National Key Research and Development Plan of China (2019YFB1704403).

References

Abeysooriya, R. P., Bennell, J. A., & Martinez-Sykora, A. (2018). Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation. *International Journal of Production Economics*, *195*, 12-26.

Alvarez-Valdes, R., Martinez, A., & Tamarit, J. M. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, *145*(2), 463-477.

Anand, K. V., & Babu, A. R. (2015). Heuristic and genetic approach for nesting of two-dimensional rectangular shaped parts with common cutting edge concept for laser cutting and profile blanking processes. *Computers & Industrial Engineering*, *80*, 111-124.

Bennell, J. A., Cabo, M., & Martinez-Sykora, A. (2018). A beam search approach to solve the convex irregular bin packing problem with guillotine cuts. *European Journal of Operational Research*, *270*(1), 89-102.

Bennell, J. A., & Oliveira, J. F. (2009). A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, *60*(sup1), S93-S105.

Boggs, P. T., & Byrd, R. H. (2019). Adaptive, limited-memory BFGS algorithms for unconstrained optimization. *SIAM Journal on Optimization*, *29*(2), 1282-1299.

Burke, E., Hellier, R., Kendall, G., & Whitwell, G. (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, *54*(3), 587-601.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M., Oliveira, J. F., & Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, *253*(3), 570-583.

Elkeran, A. (2013). A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, *231*(3), 757-769.

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

Hansen, A., & Arbab, F. (1992). An algorithm for generating NC tool paths for arbitrarily shaped pockets with islands. *ACM Transactions on Graphics*, *11*(2), 152-182.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, *88*(1), 165-181.

Kang, J., & Park, S. (2003). Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, *147*(2), 365-372.

Leao, A. A., Toledo, F. M., Oliveira, J. F., & Carravilla, M. A. (2016). A semi-continuous MIP model for the irregular strip packing problem. *International Journal of Production Research*, *54*(3), 712-721.

Leao, A. A., Toledo, F. M., Oliveira, J. F., Carravilla, M. A., & Alvarez-Valdés, R. (2020). Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, *282*(3), 803-822.

Leung, S. C., Lin, Y., & Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, *39*(3), 678-686.

Liu, Q., Zeng, J., Zhang, H., & Wei, L. (2020, September). A heuristic for the two-dimensional irregular bin packing problem with limited rotations. *In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (*pp. 268-279*). Springer, Cham.

López-Camacho, E., Ochoa, G., Terashima-Marín, H., & Burke, E. K. (2013). An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research*, *206*(1), 241-264.

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

Parreño, F., Alvarez-Valdés, R., Oliveira, J. F., & Tamarit, J. M. (2010). A hybrid GRASP/VND algorithm for two-and three-dimensional bin packing. *Annals of Operations Research*, *179*(1), 203-220.

Pinheiro, P. R., Amaro Júnior, B., & Saraiva, R. D. (2016). A random-key genetic algorithm for solving the nesting problem. *International Journal of Computer Integrated Manufacturing*, *29*(11), 1159-1165.

Quader, M. A., Ahmed, S., Dawal, S. Z., & Nukman, Y. (2016). Present needs, recent progress and future trends of energy-efficient Ultra-Low Carbon Dioxide (CO_2) Steelmaking (ULCOS) program. *Renewable and Sustainable Energy Reviews*, *55*, 537-549.

Rodrigues, M. O., & Toledo, F. M. (2017). A clique covering MIP model for the irregular strip packing problem. *Computers & Operations Research*, *87*, 221-234.

Sato, A. K., Martins, T. C., Gomes, A. M., & Tsuzuki, M. S. G. (2019). Raster penetration map applied to the irregular packing problem. *European Journal of Operational Research*, *279*(2), 657-671.

Umetani, S., & Murakami, S. (2022). Coordinate descent heuristics for the irregular strip packing problem of rasterized shapes. *European Journal of Operational Research*.

Zhang, H., Liu, Q., Wei, L., Zeng, J., Leng, J., & Yan, D. (2022). An iteratively doubling local search for the two-dimensional irregular bin packing problem with limited rotations. *Computers & Operations Research*, *137*, 105550.