

GA and GWO algorithm for the special bin packing problem encountered in field of aircraft arrangement

Qiang Luo, Yunqing Rao (Ph.D)^{*}, Deng Peng

The State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China

ARTICLE INFO

Article history:

Received 2 May 2021

Received in revised form 28 October 2021

Accepted 2 November 2021

Available online 19 November 2021

Keywords:

Irregular bin packing

Heuristic algorithm

Two-dimensional spatial arrangement

Genetic algorithm

Grey wolf optimization

ABSTRACT

This paper addresses a special case of irregular bin packing problem which the irregular pieces with free rotation have to fill a large irregular stock sheet with defective regions while satisfying the special boundary constraint, i.e., the piece can protrude from the sheet so long as the key points in the piece's interior lie within the container. The objective of this problem is to maximize the number of filled pieces. To our best knowledge, the piece must be placed completely inside the sheet for all packing problem tackled by published literature. Thus, existing approaches are not good solutions to this special packing problem. To achieve the goal of automated arrangement of pieces and maximize the space utilization, the genetic algorithm and grey wolf optimization algorithm are designed to solve it. The genetic algorithm adopts the elitism strategy for maintaining the portion of the best chromosomes. A new method of updating the main controlling parameter is applied for reinforcing the exploration ability of the grey wolf optimization. These two algorithms use a vector of pieces as the solution representation, and a novel heuristic algorithm decodes it to produce a layout. The proposed heuristic algorithm divides the process of packing into two stages with the objective of satisfying constraints and achieving good space utilization of sheet. Computational experiments show that the presented methods can solve this new kind of the packing problem very well.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

The two-dimensional packing problem addressed in the most published literature is commonly encountered in sheet metal, leather, clothing and glass industries [1–3], which generally defined as finding a configuration of pieces to cut from or pack inside larger objects with the objective of minimizing the waste. This paper focuses on a new kind of irregular bin packing problem arisen in the field of two-dimensional spatial arrangement, which is NP-Complete problem [4]. The one of the real-word applications is that placing the carrier-borne aircraft in the carrier's deck with the objective of maximizing the number of aircrafts. The identical aircraft must be arranged on a platform owned irregular boundary and regions not allowed to be placed. And this arrangement mission has unconventional feature, i.e., the object can stick out from platform so long as its key points locates inside the platform. The two-dimensional spatial arrangement problem is illustrated in Fig. 1.

The packing problem involved in this study is summarily defined as a given set of irregular identical pieces with free rotation is placed on an irregular sheet with disabled region. The objective

is to maximize the number of the piece while having all pieces without overlap. The special boundary constraint of it, referred to here as “Point Constraint”, is that the piece is not required to be placed completely inside the sheet while its key points must locate inside the container. In other word, the piece's allowed to beyond the edges of the limited stock sheet but all its key points are placed inside the sheet. The piece presented by red dash in Fig. 1 does not satisfy the point constraint.

The two-dimensional strip packing problem which the rectangular sheet is open dimension and the piece is regular or irregular draws more attention than other kinds of problems [5]. There are few publications investigating bin packing problem, let alone irregular bin packing problem where the irregular stock sheet has unusable zones. Compared to the regular packing problem, the irregular two-dimensional packing problem is geometrically more complex and more difficult to implement due to the natural irregularity and non-convexity of the pieces to be placed [6]. Furthermore, the packing problem involved in this paper is harder than problems studied before because of the characters as follows: the irregular sheet has disabled regions, the piece can freely rotate and the special boundary constraint.

Based on the characteristics of this special packing problem, the genetic algorithm (GA) and grey wolf optimization (GWO)

^{*} Corresponding author.

E-mail addresses: lq_dr@qq.com (Q. Luo), ryq@hust.edu.cn (Y. Rao).

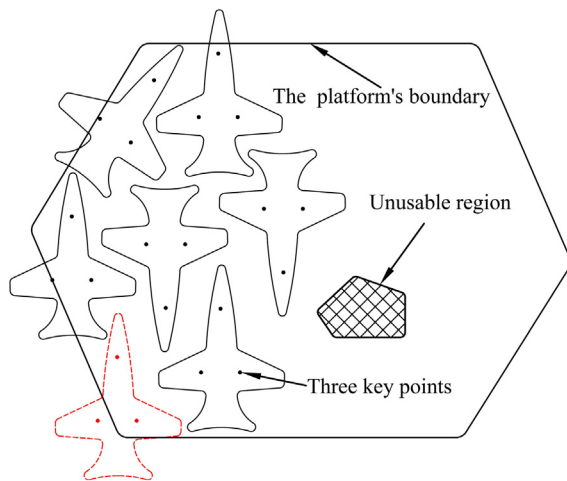


Fig. 1. The illustration of the studied problem and point constraint.

are designed to solve it. A mixed coding strategy of real number combined with integer is proposed for both metaheuristics. Besides, the elitism strategy is adopted in the GA for maintaining the portion of the best chromosomes, and a new method of updating the main controlling parameter is applied for reinforcing the exploration ability of the GWO. A robust single-pass heuristic placement algorithm (HPA), as a decoder, is devised. We creatively put forward the method that placing the piece along the boundary of sheet according to the fact that increasing the piece's region outside the sheet leads to decrease the internal zone of the sheet covered by the piece. The proposed HPA not only guarantees the key points of the piece inside the sheet but also can increase the space utilization of the sheet as much as possible.

The contribution of this paper is as follows. First, we briefly summarize the types of cutting and packing problems currently studied in academic community so that providing reference for the follow-up researchers. Second, it is the first paper to consider irregular identical item packing problem with special boundary constraint, our work enriches the field of the Cutting and Packing Problem. Third, we propose two effective and robust metaheuristic algorithms to solve this realistic problem. Last, some instances are offered and the work we have done is able to provide reference for follow-up research.

The remainder of this paper is organized as follows. Section 2 provides a brief view of related works and summarizes the types of packing problems. The problem is defined and the geometric tool used in our algorithm is introduced in the following section. Section 4 exhaustively introduces our approaches. The instances used to evaluate the quality of the proposed approaches are defined, and the results obtained by our approaches are presented and analyzed in Section 5. The final section discusses conclusions and makes some comments about future work.

2. Literature review

In this section, we will provide concise overview of the literature. To our best knowledge, there is no literature on this new kind of packing problem but there is existing lots of literature on the rectangle bin packing problem and irregular strip packing problem. Hence, the paper most relevant to our study is selected. These works are helpful to our research and provide some references and inspiration.

The well know representative of Identical Item Packing Problem (IIPP) is the Manufacturer's Pallet Loading (Packing) Problem

(MPLP), in which a single pallet must be loaded with a maximal number of identical boxes. In other word, the problem is packing maximum number of identical rectangles into a larger rectangle without overlapping. Morabito et al. [7] and Dowsland et al. [8] comprehensively studied MPLP and proposed good methods to solve it. In addition, there is a Cylinder Packing Problem studied by Correia et al. [9] and Birgin et al. [10]. It possesses an almost identical problem structure compared to the MPLP. Instead of identical rectangles, the Cylinder Packing Problem is assigning circles of the same size to a given rectangle. Birgin et al. [10] proposed a nonlinear model for this problem and extend it for packing identical circles into a circle and non-identical circles into rectangles and circles. López et al. [11] further studied the problem in which considering a variety of containers: the unit circle, unit square, rectangle, isosceles right-angled triangle and semicircle.

Irregular shape packing problems hold more additional challenges than regular shape (rectangle, circle) due to the increased complexity of the solution space and the intensively time-consuming calculation of feasibility of the solution. That could be the reason why literature explicitly covering the irregular piece and sheet with disabled region is limited. The work most relevant to our research is published by Costa et al. [12]. They hold the view that the possible approach of solving irregular IIPP where congruent copies of one single shape fills a limited rectangular sheet is the periodic placement of the shapes, in a lattice way. Experimental results show the lattice heuristic can get good solution.

Most of existing literature on packing irregular pieces tackle the strip packing problem in which a set of irregular pieces is laid out on single rectangular sheet of fixed width and variable length with the objective of minimizing the length. Besides, the piece is only allowed to rotate a finite set of angles, generally 90° and 180° . Imamichi et al. [13] proposed an iterated local search algorithm to solve it. The proposed algorithm is mainly composed of separation algorithm and an operation. The separation algorithm based on nonlinear programming is for separating overlapping polygons because overlap between pieces is permitted in their algorithm. The operation is to swap two polygons in a layout for finding their new positions in the layout with the least overlap. Later on, Stephen C.H. et al. [14] adopted a similar approach, but applying the Tabu search algorithm to avoid local minima and a compact algorithm to improve the result. Pinheiro et al. [15] presented a random-key genetic algorithm for this problem. In their algorithm, a solution is encoded as a vector consist of three components: the piece, rotation variant of each piece and the placement rule that is adopted to pack the pieces. Correspondingly, the decoding process has three phases. The similar technique has been used in our approach but it is much simple. Cherri et al. [16] adopted a robust mixed-integer linear programming model by decomposing items into convex parts, which showed slightly improved results. Rao et al. [17] proposed a search algorithm hybridized beam search with Tabu search to solve this problem. There still existing sophisticated heuristic methods, such as SA-SMT [18], ROMA [19] and μ -BRKGA [20]. Recently, there are several published works for solving this problem. Leao et al. [21] conducted literature survey and reviewed the mathematical models proposed in the literature for this irregular packing problem. And Oliveira et al. [3] investigated an variant of it which integrates the irregular strip packing and the cutting path determination problems. Hu et al. [1] proposed a greedy adaptive search algorithm for the problem encountered in the fabric industry. Guo et al. [22] adopted a fuzzy matching strategy which includes the operation of searching geometric similarity features to solve the problem. Amaro et al. [23] proposed a biased random-key genetic algorithm and used the dotted board model to present the position of the piece.

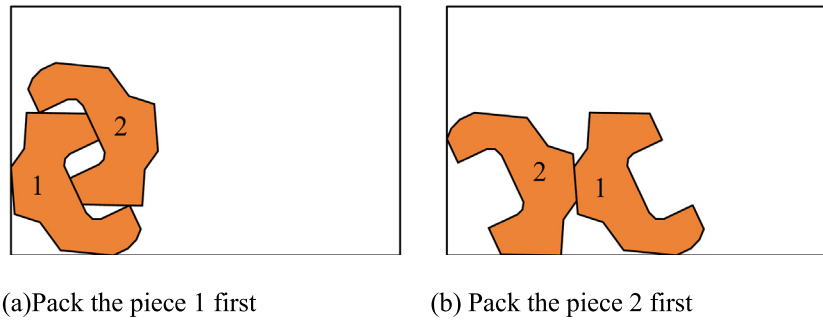


Fig. 2. The diverse layout produced by different the sequential order of identical piece.

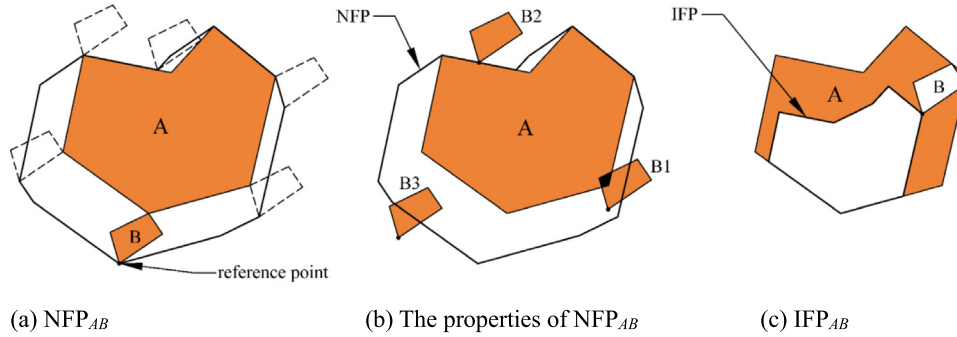


Fig. 3. The NFP and IFP generated by polygons A and B.

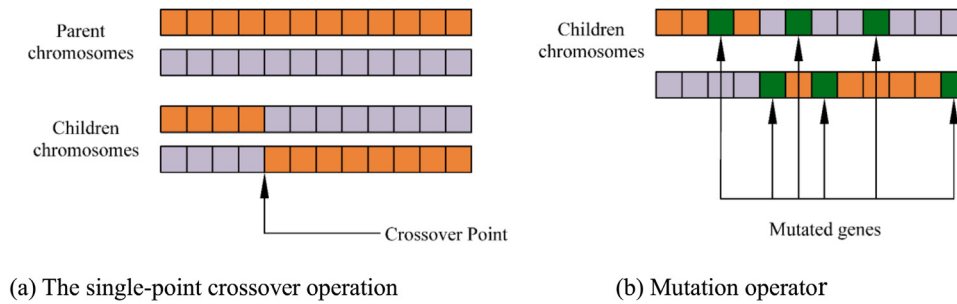


Fig. 4. The crossover and mutation operations of GA.

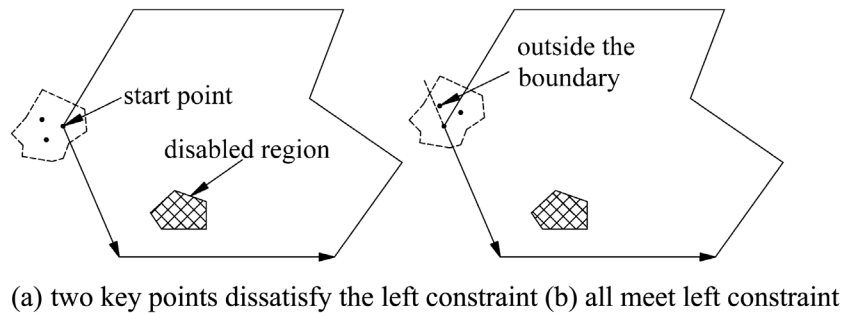


Fig. 5. The operation of moving piece.

Judging from the number of documents, the scientific community has paid more attention to the irregular strip packing problem than others, but there are still literatures deal with other kind of packing problem. López-Camacho et al. [24] examined the two-dimensional (2D) bin packing problem where the identical rectangular sheet with fixed dimensions is infinite and the piece only rotates by a finite set of angles. The type of problem of the pieces can rotate freely is studied by Sykora et al. [25] and Ranga

et al. [5]. Note that the irregular piece includes convex and non-convex regarding all above-mentioned packing problems. Bennell et al. [2], Sykora et al. [26] and Wei Han et al. [27] researched the convex irregular bin packing problem with guillotine cuts. A guillotine cut is a single straight line cut that begins at an edge of the stock-sheet and ends at another edge [27]. All pieces can freely rotate in this kind of problem. There is a little difference about packing problem studied by Bennell et al. [2] and the other two, which the sheet can be of different sizes.

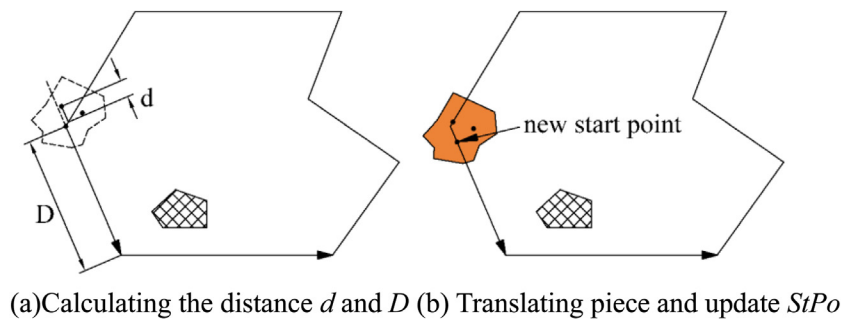


Fig. 6. The illustration of "point inclusion test".

Table 1

The types of the mainly studied two-dimensional packing problem on literatures.

Literature	Kind	Shape	Assortment	Number	Special feature
Morabito et al. [7]	Piece	Rectangle	Identical	Infinite	Rotate by 90°
Dowland et al. [8]	Sheet	Rectangle	Identical	Finite	/
Correia et al. [9]	Piece	Circle	Identical	Infinite	/
Birgin et al. [10]	Sheet	Rectangle or circle	Identical	Finite	/
López et al. [11]	Piece	Circle	Identical	Infinite	/
	Sheet	Regular	Single	Finite	/
Costa et al. [12]	Piece	Irregular	Identical	Infinite	
	Sheet	Rectangle	Single	Finite	
Hu X. et al. [1]	Piece	Irregular	Heterogeneous	Finite	Rotate by finite angles
Imam et al. [13]					
Stephen et al. [14]					
Pinheiro et al. [15]					
Cherri et al. [16]	Sheet	Rectangle	Single	Infinite	Open dimension
Rao et al. [17]					
Sato et al. [18,19]					
Júnior B et al. [20]					
Guo et al. [22]					
Amaro et al. [23]					
López et al. [24]	Piece	Irregular	Heterogeneous	Finite	Rotate by finite angles
	Sheet	Rectangle	Identical	Infinite	/
Sykora et al. [25]	Piece	Irregular	Heterogeneous	Finite	Free rotation
Ranga et al. [5]	Sheet	Rectangle	Identical	Infinite	/
Bennell et al. [2]	Piece	Convex & Irregular	Heterogeneous	Finite	Free rotation
	Sheet	Rectangle	Heterogeneous	Infinite	Guillotine cut
Sykora et al. [26]	Piece	Convex & Irregular	Heterogeneous	Finite	Free rotation
Han et al. [27]	Sheet	Rectangle	Identical	Infinite	Guillotine cut
This paper	Piece	Irregular	Identical	Infinite	Free rotation & point constraint
	Sheet	Irregular	Single	Finite	Unusable zones

To our knowledge, these above-mentioned problems are the types of two-dimensional packing problem mainly addressed by scientific community. Note that the stock sheet is homogeneous and regular, have not some defects or defining unusable zones. Table 1 summarizes the main features of the widely studied two-dimensional packing problem. Our aim is not to provide an exhaustive review of the literature but to draw on the literature to describe the various studied packing problems. It should be noted that the word "identical" implies there are many same-size sheets or pieces.

3. Problem definition and geometric tool

3.1. Problem definition

In this section we provide a formal description of the problem. A set of irregular and identical pieces, represented by simple polygon in our algorithm, is laid out on single irregular sheet possessed disabled regions and fixed dimension under two constraints. The basic constraint is to have pieces do not overlap with each other, called "overlap constraint". The second constraint is that the internal key points of the piece must locate inside the

container regardless the piece is whether placed completely on the sheet or not, called "point constraint". There are no other constraints related to the weight, density, fragility, etc.

Under the typology of cutting and packing problems presented by Wäscher et al. [28], the problem is a special two-dimensional, irregular Identical Item Packing Problem (2DIIIPP). It should be point out that considering the sequential order of piece packed into sheet is necessary even though the piece is identical. The reason is that the feasible placement zone produced by packed piece is different (shown in Fig. 2) because of the irregularity and arbitrary rotation of the piece. This is key difference between irregular IIPP and regular IIPP. Fig. 1 shows in detail the character of the problem. It also should be mentioned that the number of key points can exceed three.

The objective is to maximize the number of the packed pieces, N . However, this measuring method is not very sensitive because many competing solutions will have the same value of N for a given problem instance. Hence, we propose a new measurement to evaluate the solutions according to the fact that increasing the piece's region outside the sheet leads to decrease the internal zone of the sheet covered by the piece and result in an increase of the number of the packed piece. The fractional number of pieces

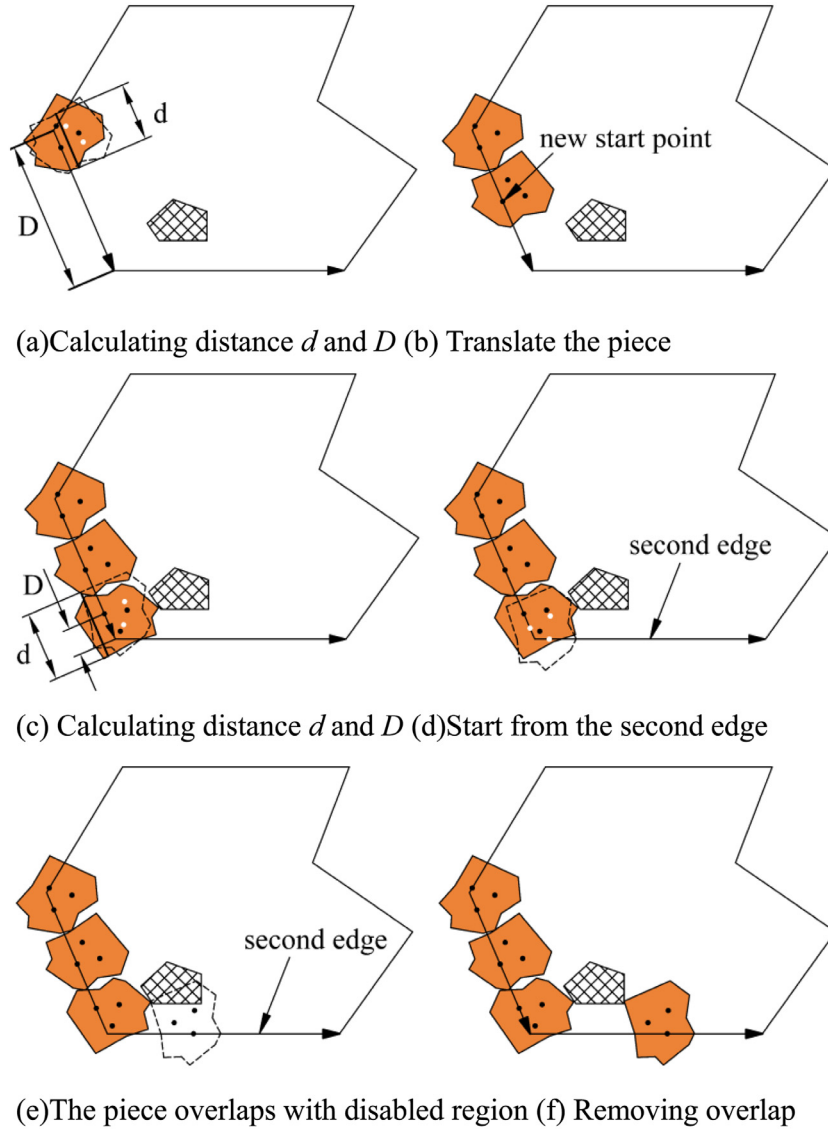


Fig. 7. The process of testing intersection and removing it.

$$fitness = N + \frac{a_1 + a_2 + \dots + a_m}{Am} \quad (3.1)$$

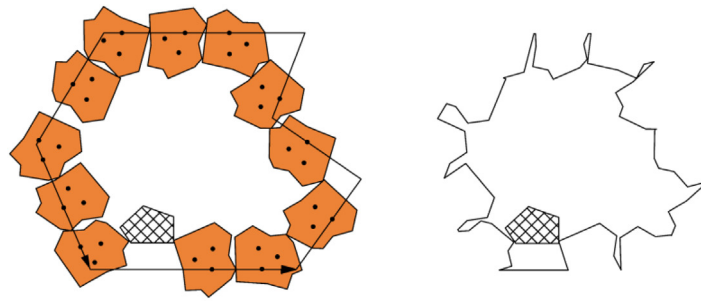
where A is the area of piece, a_i ($i = 1, 2, \dots, m$) donates the piece's area outside the boundary of sheet and m is the total number of such pieces.

3.2. Geometric tool

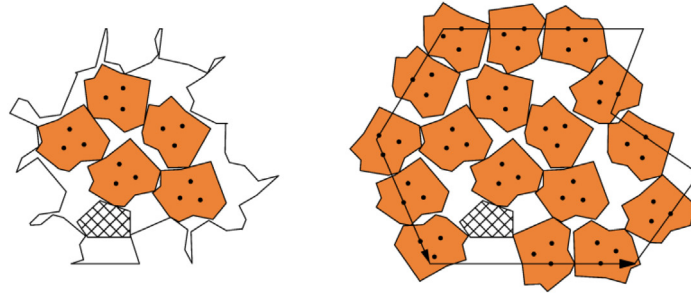
The most visible feature of irregular packing problem and the first obstacle researcher must be faced with is the geometry. This means we need to develop an approach to identify the positional relation of two pieces on the sheet, i.e. overlap, touch or separated. There existing many methods ranging from simple to complex, the direct trigonometry and no-fit polygon (NFP) are employed in this study. The former mainly includes well know tests for line intersection and point inclusion, while the NFP is another popular option for dealing with the geometry. See Bennell and Oliveira [6] for a detailed discussion of direct trigonometry and the NFP for implementing this concept in irregular packing problem.

The NFP is a simple polygon formed by two polygon A and B in such a way that A has fixed position and B slides around A meanwhile ensuring that B is always touching A and never intersects with A . Given a reference point on B (selected as the bottom vertex), the locus of the reference point forms a closed path that is the boundary of NFP_{AB} , as displayed in Fig. 3(a). The three relative positions between reference point of B and the NFP_{AB} , inside, on the boundary and outside, represents A and B overlap, A and B touch without overlap and A and B separate respectively, as shown in Fig. 3(b). Hence, the NFP reduces the computational burden of identifying if two polygons overlap to a simple point-inside test. And the calculation of the NFP can be performed in a pre-processing phase.

The inner-fit-polygon (IFP) is derived from the NFP concept and it represents the feasible set of points for placing one polygon inside the sheet. For the inner-fit-rectangle construction, B slides along the internal contour of polygon A , shown in Fig. 3(c). Apparently, if the reference point of B is placed in the exterior of the IFP_{AB} then B is not contained in A ; if the reference point of B is placed in the boundary of the IFP_{AB} then B is contained in A and touches A ; if the reference point of B is placed in the interior of the IFP_{AB} then B is contained in A but does not touch A .



(a) layout produced by first stage of HPA (b) Generating new sheet for second packing stage



(c) layout produced by second stage of HPA (d) the final layout produced by HPA

Fig. 8. The process of second stage of the HPA and the final layout.

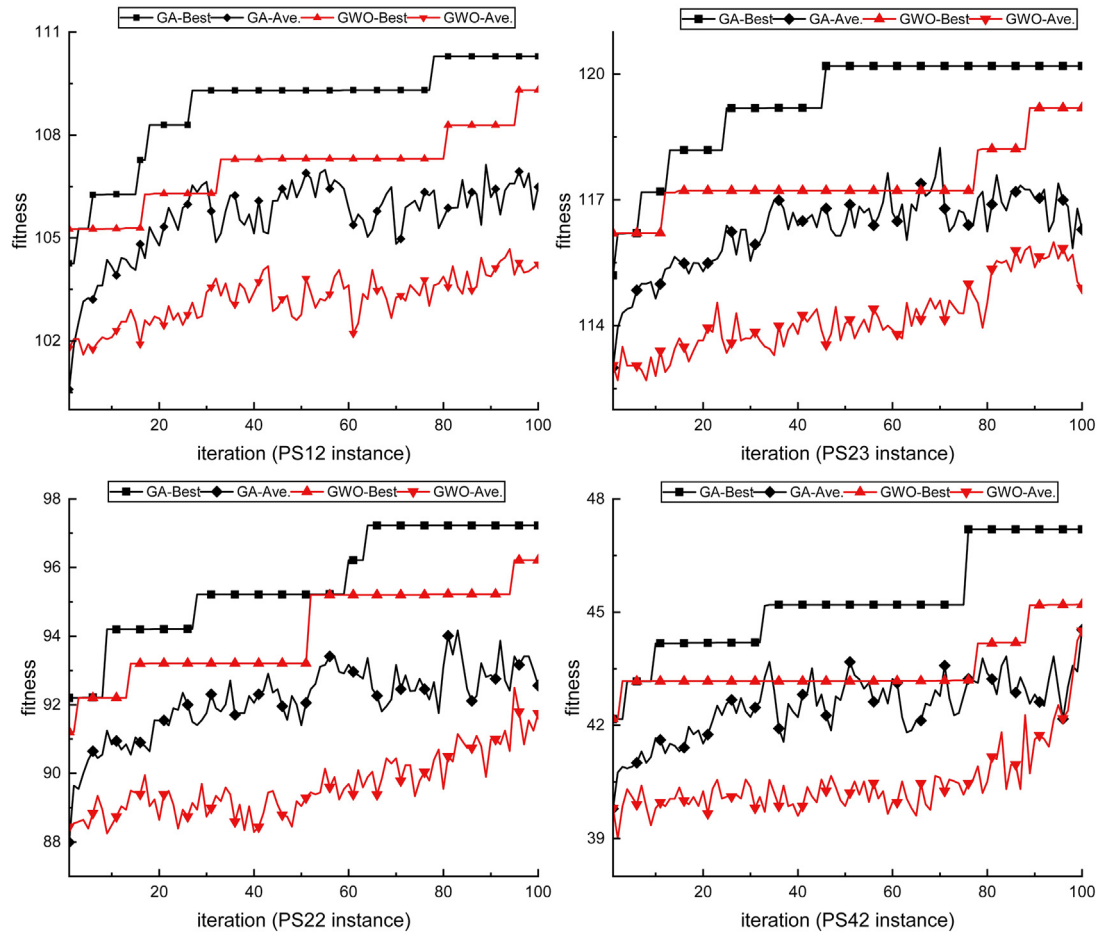


Fig. 9. The convergence curve and average fitness of population about GA and GWO in some instances.

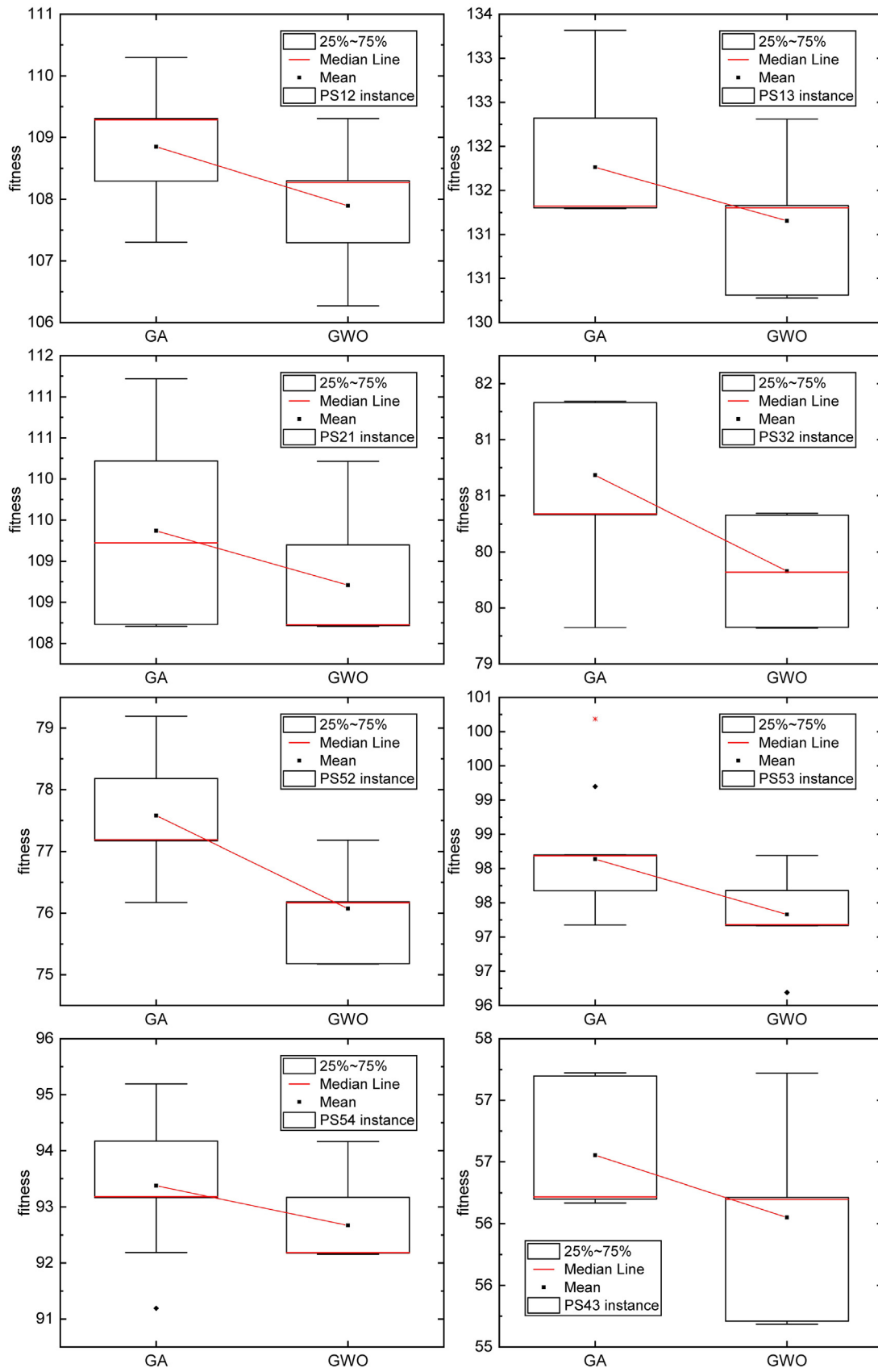


Fig. 10. Fitness box figure of some instances by proposed algorithms.

Table 2

The average computation time of the HPA algorithm with instances.

Instance	Pie. vertex	Sht. vertex	First stage(s)	Second stage(s)	Total time(s)	Ave. num.	Time(s)/Per piece
PS11	12	14	0.3548	4.6867	5.0415	115.75	0.0436
PS12	12	14	0.6536	3.6700	4.3236	101.25	0.0427
PS13	12	20	0.5124	4.9955	5.5079	125.85	0.0438
PS14	12	30	0.8541	5.2838	6.1379	119.50	0.0514
PS21	19	14	0.4865	4.1159	4.6024	103.35	0.0445
PS22	19	14	0.9627	3.5225	4.4852	88.40	0.0507
PS23	19	20	0.6715	4.4383	5.1097	113.90	0.0449
PS24	19	30	1.1651	5.2489	6.4139	107.25	0.0598
PS31	10	14	0.2635	3.1957	3.4592	85.15	0.0406
PS32	10	14	0.3912	2.5129	2.9041	74.80	0.0388
PS33	10	20	0.1767	3.2726	3.4493	91.05	0.0379
PS34	10	30	0.9686	4.2079	5.1765	87.30	0.0593

4. Approaches for the problem

In this section two approaches to solve the studied problem are elaborately explained. The first is Genetic algorithm and the other is the grey wolf optimization. Both methods represent the solution of the problem by a vector and use the same decoder. Although the studied packing problem is an NP-Complete problem, our approaches can effectively solve it and obtain good solution.

Specifically, a solution is encoded as the vector $\vec{X} = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$ composed of two parts. The first is a list where each value represents the rotation angle of piece in the interval $[0^\circ, 360^\circ]$. The m is an integer which is equal to the perimeter of sheet divided by the half of perimeter of enclosing rectangle of the piece. The second part n is an integer represented the number of pieces possibly packed on the sheet, which is equal to the area of sheet divided by the piece's area. Both m and n are greater than the actual number of pieces packed into sheet. The HPA will decode the vector, generate a layout and calculate the fitness.

4.1. Genetic algorithm and grey wolf optimization

4.1.1. Genetic algorithm

Genetic algorithm (GA) is one of the most well-regarded evolutionary algorithm in the history, which is widely used to solve various of optimization problems included the irregular packing problem [15,20,29].

The optimization process of the GA simulates the evolutionary mechanism of nature: the survival of the fittest. In nature, the organism which has higher adaptability of environment possesses higher probability of survival. This makes them to transfer their genes to the offspring, which leading to good genes become dominant in subsequent generations.

Inspired by chromosomes and genes in nature, the solution for an optimization problem corresponds to a chromosome and each gene corresponds to a decision variable of the problem. Generally, the chromosome is coded as a vector $\vec{X}_i = (x_1, x_2, \dots, x_m)$, and the population is $(\vec{X}_1, \vec{X}_2, \dots, \vec{X}_p)$.

The GA devises three operators for chromosome, i.e., selection, crossover and mutation, with the purpose of modeling the process of evolutionary. These operators are performed to improve the chromosomes in each generation until the stop criterion of the algorithm is met. The specifical method of three operators is various, see detailed introduction in the literature [30].

4.1.2. Genetic algorithm for packing problem

The pseudo code of our GA algorithm is presented in Algorithm 1. In our proposed algorithm, a chromosome is mapped as a vector $\vec{X} = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$, the meaning of this vector has been explained above.

Algorithm 1: Genetic Algorithm

```

Set parameters  $P_c, P_m, p, T$ ;
for  $i = 1$  to  $p$  //Initialize the population;
     $\vec{X}_i \leftarrow$  Randomly produce an initial solution;
     $fitness(\vec{X}_i) = HPA(\vec{X}_i)$ ; // see algorithm 4;
    add the  $\vec{X}_i$  to the pop  $P_0$ ;
end for
Int  $t = 0$ ;
while ( $t < T$ )
    // elitism strategy
    Cope  $e$  fittest individual from pop  $P_t$  to the next pop  $P_{t+1}$ 
     $P_{t+1} \leftarrow$  Select, Crossover, Mutate; // Produce  $p-e$  individuals
    for  $i = e$  to  $p$  // the set of  $p-e$  individuals in  $P_{t+1}$ ;
         $fitness(\vec{X}_i(t+1)) = HPA(\vec{X}_i(t+1))$ ; // see algorithm 4;
    end for
     $t = t+1$ ;
end while
Return the best solution;

```

The proposed algorithm starts the optimization process with a set of random chromosomes, i.e., the value of $\alpha_k (k = 1, 2, \dots, m)$ in the vector $\vec{X}_i = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$ are randomly generated. After initializing the population, the fitness of every individual is calculated by using fitness function HPA.

Before executing the three operators at each iteration, our proposed algorithm adopts the elitism strategy, i.e., a small group of e fittest individuals are copied to the next generation directly, in order to retain the best solutions and maintain the portion of the best chromosomes in the current population.

The roulette wheel is adopted as the selection operator in the algorithm [30]. The individuals with higher fitness values have a higher chance of being selected to take advantages of their good features. In the crossover operation, the single-point method is used to exchange genes between two chromosomes when a random number in the interval of $[0,1]$ less than Probability of Crossover (P_c). An example is visualized in Fig. 4(a). The chromosomes of two parent solutions are swapped before and after a single point. In the mutation operation, a random number in the interval of $[0,1]$ is generated for each gene in the new child. If this random number is less than Probability of Mutation (P_m), the gene is assigned with a random number within $[0, 360]$, see Fig. 4(b). The proposed algorithm will immediately calculate the fitness of new individuals after performing crossover and mutation operation to produce $p-e$ individuals.

4.1.3. Grey wolf optimization

The Grey Wolf Optimization (GWO) is originally proposed by Mirjalili et al. [31]. The optimization process of it simulates hierarchical leadership structure and the cooperative hunting strategy of the Grey Wolves [32].

A pack of wolves has a very strict social hierarchy, i.e., the pack includes four levels, called alpha, beta, delta, and omega in proper order. The lower level has to obey the order from the dominant wolves. In order to mimic the social hierarchy of grey wolves, the GWO considers the best, second-best, and third-best solutions as alpha, beta, and delta respectively. The rest of the candidate solutions are assumed to be omega. In addition to the social hierarchy, another intriguing social behavior of grey wolves is cooperative hunting. This behavior includes three main steps: tracking and approaching the prey, pursuing and encircling the prey, and attacking the prey.

The first step of GWO is to initialize a population of p wolves (candidate solutions), population = $(\vec{X}_1, \vec{X}_2, \dots, \vec{X}_p)$, as for the population-based optimization algorithm. Each solution is a vector of m dimensions, i.e., $\vec{X}_i = (x_1, x_2, \dots, x_m)$. To mathematically model the social hierarchy of the grey wolves, the GWO

Table 3

The maximal and minimal number of packed shapes and computational time of our proposed algorithms in each instance.

Inst.	GA			GWO		
	Number		Time	Number		Time
	Be.	Wo.	Ave.	Be.	Wo.	Ave.
PS11	123	120	196.55	122	120	195.64
PS12	110	107	140.53	109	106	139.57
PS13	133	131	206.97	132	130	205.60
PS14	126	124	208.01	126	124	208.47
PS21	111	108	179.62	110	108	176.52
PS22	97	94	129.49	96	93	129.64
PS23	120	117	187.32	119	117	186.08
PS24	115	112	188.60	113	112	191.13
PS31	91	89	141.35	90	89	122.93
PS32	81	79	88.05	80	79	98.80
PS33	96	95	162.00	96	94	161.95
PS34	92	91	163.48	92	90	179.79
PS41	53	51	113.66	53	51	114.36
PS42	47	44	83.88	45	44	85.29
PS43	57	56	123.48	57	55	124.75
PS44	55	53	123.54	54	53	128.57
PS51	91	89	140.90	90	88	141.46
PS52	79	76	102.63	78	75	104.73
PS53	100	97	145.43	98	96	147.51
PS54	95	93	150.62	94	92	151.67

determines the best, second-best, and third-best solutions as $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$ respectively based on the value of fitness of each candidate solution. The rest of the solutions is considered as $\vec{X}_\omega$.

The candidate solutions constantly update their position in an attempt to search better solution. The process of updating simulates the cooperative hunting strategy of the grey wolves by using two equations

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \quad (4.1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (4.2)$$

where t indicates the current iteration, $t+1$ is the index number of the next iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ indicates the position vector of the prey, $\vec{X}(t)$ represents a wolf (candidate solution).

The coefficient vectors $\vec{A}$ and $\vec{C}$ are calculated and updated using the following equations:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (4.3)$$

$$\vec{C} = 2 \cdot \vec{r}_2 \quad (4.4)$$

where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations and $\vec{r}_1$, $\vec{r}_2$ are random vectors in $[0,1]$.

The GWO assumed that the first three best solutions have more knowledge about the potential location of the optimum (prey). Thus, the candidate solutions ω are updated in GWO using these three solutions by the following equations

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \quad (4.5)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta) \quad (4.6)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (4.7)$$

Encircling and attacking behavior corresponding to exploration and exploitation are the main behaviors of the wolf (candidate

Table 4

The statistical results of our proposed algorithms in all instances.

Inst.	GA				GWO			
	Ave.	Med.	IQR	Std.	Ave.	Med.	IQR	Std.
PS11	122.21	122.31	0.03	0.64	121.50	121.30	1.01	0.62
PS12	108.85	109.29	1.01	0.69	107.89	108.27	1.00	0.89
PS13	131.76	131.32	1.02	0.61	131.15	131.31	1.00	0.71
PS14	125.40	125.29	0.99	0.72	125.25	125.29	0.03	0.51
PS21	109.37	109.22	1.99	0.93	108.71	108.23	0.98	0.60
PS22	95.41	95.21	1.98	1.06	94.30	94.21	0.02	0.79
PS23	118.62	118.23	1.01	0.75	118.36	118.21	0.03	0.49
PS24	113.21	113.21	0.28	0.79	112.55	112.22	0.98	0.48
PS31	90.28	90.33	0.03	0.61	89.88	90.32	1.00	0.51
PS32	80.68	80.34	1.00	0.59	79.83	79.82	1.00	0.51
PS33	95.53	95.34	0.02	0.41	95.18	95.32	0.02	0.49
PS34	92.08	92.33	0.26	0.45	91.57	91.33	0.98	0.55
PS41	52.06	52.21	0.28	0.50	51.40	51.20	0.02	0.53
PS42	44.99	45.18	1.00	0.58	44.28	44.18	0.02	0.32
PS43	56.55	56.22	0.99	0.49	56.05	56.20	1.00	0.67
PS44	54.14	54.19	0.03	0.51	53.38	53.19	0.03	0.41
PS51	89.88	90.17	1.00	0.73	89.18	89.18	0.02	0.65
PS52	77.58	77.19	1.01	0.94	76.07	76.17	1.00	0.79
PS53	98.14	98.19	0.28	0.76	97.33	97.18	0.26	0.58
PS54	93.68	93.18	1.01	0.69	92.67	92.18	0.99	0.60

solution) in the algorithm. The exploitation stage is triggered when $|\vec{A}| < 1$, and the exploration is started based on the coefficient vectors $\vec{A}$ and $\vec{C}$. These behaviors are controlled by the parameter of the algorithm and can be switched smoothly based on the adaptive values of the parameter.

4.1.4. Grey wolf optimization for packing problem

The GWO was used to solve kinds of combinatorial optimization problems, such as the scheduling of workflow by Bilal H [32], the shop scheduling problem by Zhu [33], the two-dimensional rectangular strip packing problem by Luo et al. [34] and so on. Inspired by these works, an effectively grey wolf optimization is devised to solve the studied packing problem.

Algorithm 2: Grey Wolf Optimization

Initialize the parameters: p , T , $\vec{a}$, $\vec{A}$ and $\vec{C}$;
for $i = 1$ to p // Initialize the grey wolf population;
 $\vec{X}_i \leftarrow$ Randomly produce an initial solution;
 $fitness(\vec{X}_i) = HPA(\vec{X}_i)$; // see algorithm 4;
end for
 $\vec{X}_\alpha =$ best search agent;
 $\vec{X}_\beta =$ second best search agent;
 $\vec{X}_\delta =$ third best search agent;
Int $t = 0$;
while ($t < T$)
for $i = 1$ to $p-3$
 $\vec{X}_i(t+1) \leftarrow$ Update the position $\vec{X}_i(t)$ by equation (4.7);
 $fitness(\vec{X}_i(t+1)) = HPA(\vec{X}_i(t+1))$;
end for
Update $\vec{a}$, $\vec{A}$ and $\vec{C}$;
Update $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$;
 $t = t+1$;
end while
return the $\vec{X}_\alpha$;

Algorithm 2 shows our GWO algorithm for studied packing problem. The first step of our algorithm is to randomly initialize the grey wolf population, i.e., the value of $\alpha_k(k = 1, 2, \dots, m)$ in the vector $\vec{X}_i = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$ represented for each solution are randomly generated. And then the fitness of each solution is calculated by the fitness function HPA. The first three highest

Algorithm 3: PlaceAlongBoundary ($\vec{X}_i$)

```

bool isPacked = true, isEnd = false;
int k = 1, i = 0;
int Count = 0; // record the number of packed pieces;
 $e_k \leftarrow$  Get the  $k$ 'th edge of sheet boundary; // the boundary is anti-clockwise direction
 $StPo \leftarrow$  the starting point of the edge  $e_k$ ;
while isEnd == false
    if isPacked == true // last piece is packed
         $i = i + 1$ ; //  $P_i$  represents the  $i$ 'th piece from the list  $\vec{X}_i$ 
        isPacked = true;
        Move the  $P_i$  and update  $StPo$ ; // all key points meet the left constraint
        bool isTestEnd = false, isMove = true, isFirst = true;
        while isTestEnd == false
            if isMove == true // the point inclusion test
                isMove = false;
                for j = 1 to B // B is the number of the key points of  $P_i$ 
                    if key point j isn't inside the sheet
                        Calculate the value  $d$  and  $D$ 
                        if  $d > D$  or  $d == 0$ 
                            isPacked = false;  $k = k + 1$ ;
                            if  $k > K$  // K is the number of edges of the sheet
                                isEnd = true;
                        else
                             $StPo \leftarrow$  the starting point of the edge  $e_k$ ;
                            break;
                    else
                        isMove = true;
                        Translate  $P_i$  along the edge  $e_k$  and update  $StPo$ ;
                End for
            if isPacked == false or (isMove == false and isFirst == false)
                isTestEnd = true;
            if isMove == true or isFirst == true // overlap test
                isFirst = false; isMove = false;
                for j = 1 to Q // Q is the number of all packed piece and disabled regions
                    if  $P_i$  is overlap with Polygon j
                        Calculate the value  $d$  and  $D$ 
                        if  $d > D$ 
                            isPacked = false;  $k = k + 1$ ;
                            if  $k > K$  // K is the number of edges of the sheet
                                isEnd = true;
                        else
                             $StPo \leftarrow$  the starting point of the edge  $e_k$ ;
                            break;
                    else
                        isMove = true;  $j = 0$ ;
                        Translate  $P_i$  along the edge  $e_k$  and update  $StPo$ ;
                End for
            End while // point inclusion test and overlap test are over
            if isPacked == true
                Count = Count + 1;
            if  $i == m$  && isPacked == true // m is the number of the length of  $\vec{X}_i$ 
                isEnd = true; // there is none piece left
        End while // placing along the boundary stage is end
return Count;

```

value of fitness is $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$ respectively. The rest of the candidate solutions are assumed to be $\vec{X}_\omega$.

The candidate solutions in GWO are updated according to the Eq. (4.7) and then the fitness is calculated. Note that the second part of the vector $\vec{X}_i$, i.e., the n , is not updated all the time. The next step is to update $\vec{a}$, $\vec{A}$ and $\vec{C}$, and update the three first

best solutions $\vec{X}_\alpha$, $\vec{X}_\beta$ and $\vec{X}_\delta$. It should be note that the numbers in the vector $\vec{a}$ decrease from 2 to 0 with each iteration of the GWO according to the following equation:

$$a = 2 - 2 \times \frac{t^2}{T^2} \quad (4.8)$$

where T is the maximal iteration of the algorithm.

4.2. Heuristic placement algorithm

The HPA is responsible for transforming the vector of piece into a layout by applying specific placement rules and then calculating the value of *fitness*. It packs the piece on the sheet one after another meanwhile satisfying the overlap constraint and point constraint. The HPA designed for this special packing problem is divided into two stages. The first is packing the piece along the boundary of sheet, the second stage is packing on interior of sheet. The reasons why it is designed to be like this are that maximizing the utilization of the sheet, dealing with the point constraint and meeting the requirement of free rotation of piece. A detailed explanation will be given in the following part.

4.2.1. Placing along boundary

Assuming that a solution is encoded as a vector $\vec{X} = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$, and the pieces have counter clockwise orientation, i.e., that the interior of the polygon is on the left side of the oriented edges, while the inner contour of sheet represented as disabled region has clockwise orientation. The first stage of the HPA is packing piece along the boundary of sheet according to the list of rotation angle, i.e., $(\alpha_1, \alpha_2, \dots, \alpha_m)$.

First, the HPA will rotate the piece by angle α_i and then place one of its key points on the point (donated as *StPo*). Initially, *StPo* is the starting point of certain boundary edge of the sheet. The next step is to identify whether the rest of key points is on the left side of the oriented edge, called "left constraint". If not, the HPA will put key point on the *StPo* until all key points meet this requirement. This moving operation is illustrated in Fig. 5.

Second, the "Point Inclusion" test needs to be done for all key points because the moving operation cannot strictly ensure all key points are within the sheet as shown in Fig. 5(b). The calculation of translation distance d along with the oriented edge will performed once detecting the key point outside the sheet. If d is equal to or less than the distance D between start point and end point of the oriented edge, the HPA will move the piece by the distance d to make sure this key point inside the sheet and then update the point *StPo*. If not, the next oriented edge will be selected when there is edge left, setting the *StPo* as the starting point of the edge and the packing process start from scratch; if there is none oriented left the packing process end. Fig. 6 depicts this operation. Obviously, translation distance d is much less than the distance D . Thus, the polygon is moved along the oriented edge with distance d , and all key points meet the requirement.

Third, conducting overlap test to solve the overlap problem. The current piece overlaps with the packed piece and might intersect with the disabled region. Our proposed algorithm will calculate the maximal translation distance d in order to remove the overlap, and the distance D between start point and end point of the oriented edge. if d is equal to or less than D , the piece will be translated and then update the point *StPo*; If not, the same thing will happen just as the process of point inclusion test.

The third step is depicted in Fig. 7. See Fig. 7(a), a piece represented by yellow color is packed on the sheet, the piece donated by dash is placed on the *StPo* and pass the point inclusion test. Obviously, the second one overlaps with the packed piece. The maximal translation distance d is less than D , the piece will translate and then update the point *StPo*, shown in Fig. 7(b).

However, if the distance d is larger than D , as shown in Fig. 7(c), the second oriented edge is selected and the algorithm starts with first step (showed in Fig. 7(d)). When removing the overlap between the fourth and the third piece, the fourth intersects with the disabled region (showed in Fig. 7(e)). Thus, removing overlap operation will be done again, the result is displayed in Fig. 7(f). Note that the point inclusion test and overlap test is performed repeatedly until all constraints is satisfied or selecting the next oriented edge.

The Pseudo code of first stage of the HPA is depicted in Algorithm 3. This approach is very simple and easy to implement. This method creatively deals with the point constraint while increasing the utilization of limited sheet. It is because increasing the piece's region outside the sheet leads to an increase of the number of the packed piece.

4.2.2. Placing on interior

The second stage of the HPA is that placing the piece on sheet by using the Bottom Left (BL) placement rule. See Dowsland et al. [35] for a detailed discussion of BL. The first step is to get the new boundary from the partial layout produced by the HPA's first stage. The feasible placement positions of the piece will be found using the geometric tool under each rotation angle. And then selecting the most bottom-left position from all feasible placement positions to place the piece. In this way, the requirement of free rotation of piece is perfectly satisfied. The final layout is generated by combining result of the first stage with the second stage. This is the ultimate outcome produced by our proposed heuristic algorithm. The packing process of second stage of the HPA is illustrated in Fig. 8.

In conclusion, the proposed HPA algorithm can perfectly satisfy the special point constraint and possesses the great competence of solving the studied 2DIIIP. Besides, its simplicity and good understandability greatly enhance the performability of the algorithm. The pseudo code of our HPA algorithm is presented in Algorithm 4.

Algorithm 4: HPA ($\vec{X}_i$)

```

Input: the vector  $\vec{X}_i = (\alpha_1, \alpha_2, \dots, \alpha_m, n)$ ;
 $\vec{X}_i' = (\alpha_1, \alpha_2, \dots, \alpha_m) \leftarrow$  Get the angle list from  $\vec{X}_i$ ;
int Count = PlacingAlongBoundary ( $\vec{X}_i'$ ) // algorithm 3
double Area = 0;
for  $j=1$  to Count
     $a \leftarrow$  the packed piece's area outside the boundary of sheet
    Area = Area +  $a$ ;
end for
NewSheet  $\leftarrow$  Get the boundary of partial layout;
int  $m = \mathbf{PlacingOnInterior}$  ( $n$ , NewSheet);
 $fitness(\vec{X}_i) \leftarrow$  Calculate the according to the formula (3.1);
return  $fitness(\vec{X}_i)$ ;

```

5. Computational results

5.1. Environment and instance used

The proposed algorithms were implemented as a sequential algorithm using the C++, and all experiments were conducted on a Windows computer with Intel(R) Xeon (R) CPU E5 2.4 GHz and 4 GB RAM.

Unfortunately, we cannot find any results published in literature concerning this problem. So, and for future result comparison, twenty instances were created to test the proposed algorithms. These instances were divided into five sets, and each

set includes four instances. Each piece has three key points and each sheet has two or three unusable zones. The instance named PS11 means placing piece 1 on sheet 1. Detailed data about them are in [Appendix B](#). The angle step of piece rotation in all experiments is set as 1° at the second stage of the HPA. The best layout obtained by our proposed algorithms for each instance is displayed in [Appendix A](#).

5.2. Experimental results

The computational time is one of the main evaluation criteria of the algorithm's performance within the field of nesting problem. Hence, the first primary test was to assess the performance of the HPA in computational time. It was tested with twelve created instances and run twenty times at each instance. The results are summarized in [Table 2](#). The second to fifth column in [Table 2](#) present vertices of piece, vertices of sheet, the average time of twenty runs in first and second stage, respectively.

By analyzing the data of [Table 2](#), we can reach some interesting conclusions. It only takes an average of 0.046 s to place a piece in sheet. It is beneficial to adopt high-level iterative search algorithm for obtaining better results. And it is necessary to do this because the proposed HPA is unlikely to provide a global

optimum. Under this scale problem, such a time consuming is acceptable. We note that the time increases with the number of vertices of piece and sheet. See the instance PS24, the average time of placing a piece is greater than any other instances because both the number of vertices of piece and sheet are maximal. This result matches well with the theoretical prediction.

Another main evaluation criterion of the algorithm's performance is the utilization of sheet, i.e., the number of packed pieces in this study. The population size of both GWO and GA is set 20, and the number of iterations is 100. The crossover and mutation rate in GA is 0.8 and 0.4, respectively.

[Tables 3](#) and [4](#) show the computation results and some statistical data of our proposed algorithms from twenty runs for each instance. The better result obtained is marked in boldface. By analyzing these statistical data, some conclusions are drawn.

The mean computational time of two algorithms for each instance is almost the same. The time difference between the two algorithms is within five in seventeen instances. Only three instances, i.e., PS31, PS32 and PS34, are over ten minutes, see the column "Time" in [Table 3](#). It has been proved that the computational time is determined by the decoder, i.e., the HPA algorithm.

The GA can obtain better solution than GWO in 75% instances, and the result is the same in the remaining 25% instances. The GA

Table B.1

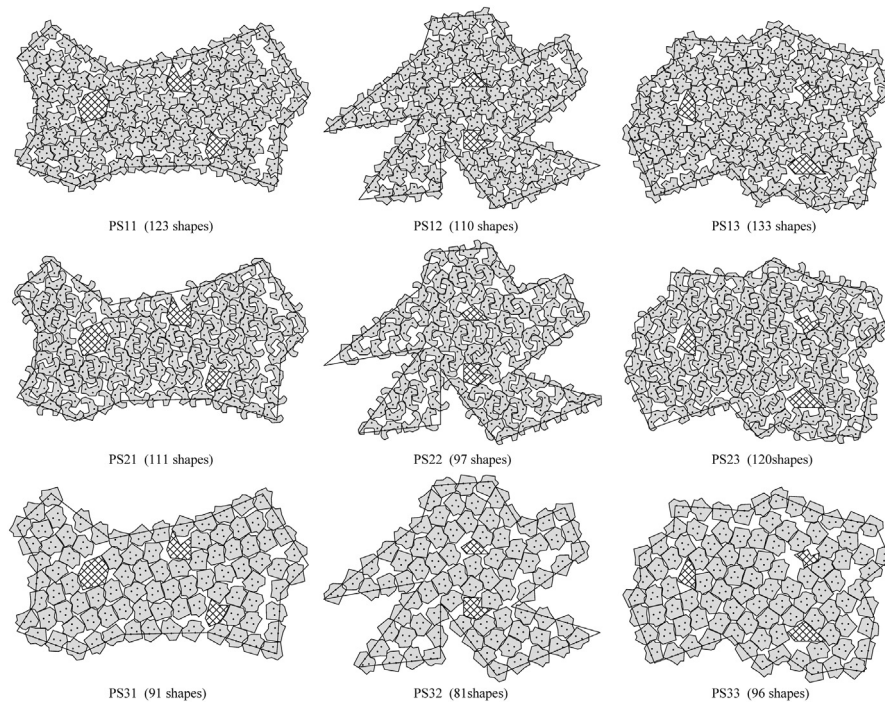
The data of piece used in instances.

Piece 1	Piece 2	Piece 3	Piece 4	Piece 5
Quantity Infinite	Quantity Infinite	Quantity Infinite	Quantity Infinite	Quantity Infinite
Number of vertices 12	Number of vertices 19	Number of vertices 10	Number of vertices 10	Number of vertices 6
Vertices (x,y)	Vertices (x,y)	Vertices (x,y)	Vertices (x,y)	Vertices (x,y)
49.36 81.14	42.74 0.83	23.52 96.87	68.96 17.96	83.54 77.04
17.00 95.20	90.18 0.00	22.02 70.25	113.36 0.00	0.00 44.60
9.96 63.60	92.08 28.21	0.00 43.10	97.65 38.44	16.54 0.00
22.92 37.97	102.56 42.90	14.12 9.81	137.92 75.23	83.54 24.84
0.00 19.41	99.67 79.70	89.70 0.00	105.09 101.87	150.53 0.00
22.29 0.00	79.71 86.05	101.65 51.63	68.96 208.87	167.07 44.60
49.36 10.68	63.38 108.04	116.19 83.64	32.82 101.87	Key Point
80.51 0.84	21.81 112.13	92.54 96.45	0.00 75.23	132.69 26.58
94.20 33.60	10.24 106.69	88.42 113.94	40.27 38.44	34.38 26.58
70.80 45.47	3.46 99.77	39.20 107.10	24.56 0.00	83.54 61.38
88.75 63.60	0.00 91.50	Key Point	Key Point	
76.46 94.63	8.80 72.83	39.58 46.76	68.96 140.68	
	31.43 83.50	75.62 46.76	34.60 61.88	
Key Point	38.10 83.47	58.97 76.19	103.32 61.88	
46.70 33.69	43.25 78.55			
32.57 66.90	60.55 41.84			
68.60 66.90	59.73 36.07			
	55.94 32.27			
	33.01 21.46			
	Key Point			
	49.53 91.18			
	83.10 61.62			
	68.14 20.95			

Table B.2

The data of sheet used in instances.

Sheet 1		Sheet 2		Sheet 3		Sheet 4	
Quantity		Quantity		Quantity		Quantity	
1		1		1		1	
Number of vertices		Number of vertices		Number of vertices		Number of vertices	
14		14		20		30	
Vertices (x,y)		Vertices (x,y)		Vertices (x,y)		Vertices (x,y)	
186.72	870.03	529.19	836.90	1343.83	811.76	426.59	885.04
0.00	697.31	0.00	413.92	1066.39	866.34	149.67	854.07
114.03	489.26	536.15	508.25	798.73	980.83	149.67	769.14
97.51	274.61	190.51	45.32	652.14	888.62	209.65	697.20
0.00	123.22	635.03	93.66	254.29	917.11	192.66	528.35
342.02	0.00	635.03	392.77	244.63	782.19	51.03	542.60
571.75	105.37	940.44	0.00	56.92	641.99	0.00	330.92
1046.17	116.09	1502.89	232.64	67.21	516.22	118.36	179.05
1413.34	0.00	1019.06	374.64	0.00	409.03	355.09	91.12
1434.32	330.84	1418.22	667.71	141.62	62.96	467.06	121.50
1584.74	483.37	1309.36	912.43	338.53	134.87	548.63	0.00
1336.59	870.03	1069.54	805.76	561.46	208.78	684.59	0.00
1013.75	729.55	916.25	1054.52	726.72	25.35	825.55	30.97
506.04	625.36	589.67	1030.27	848.48	103.01	961.51	30.97
Disabled	Region	Disabled	Region	997.89	0.00	1073.47	186.87
1056.92	298.46	761.75	424.38	1162.72	32.58	1252.58	193.07
1024.38	193.00	760.55	374.35	1282.14	14.61	1323.99	111.65
1046.11	148.84	760.55	325.61	1337.58	307.51	1405.46	204.13
1108.16	173.37	831.77	303.96	1258.77	491.70	1363.22	290.58
1156.69	263.28	915.02	362.16	1424.72	624.32	1466.81	358.93
1097.68	264.96	832.92	373.21	Disabled	Region	1535.20	497.66
Disabled	Region	858.11	421.73	323.77	640.07	1501.01	661.51
846.87	670.54	Disabled	Region	264.27	510.11	1329.02	785.15
814.33	565.08	813.44	740.22	363.44	461.66	1222.54	787.98
836.07	520.92	746.79	695.82	363.44	565.18	1088.97	918.26
946.28	510.45	776.94	660.93	Disabled	Region	1025.84	911.86
946.64	635.36	897.54	660.93	956.28	290.77	992.71	804.86
887.64	590.00			863.72	229.10	848.11	860.44
Disabled	Region			905.59	180.64	748.93	880.88
368.64	497.78			1073.08	180.64	671.59	791.33
332.01	431.38			Disabled	Region	Disabled	Region
357.83	348.16			974.79	680.38	1055.24	457.43
478.58	379.37			902.66	649.59	1187.71	437.37
499.40	462.60			947.48	616.70	1236.51	509.65
458.39	534.33			955.50	580.64	1211.98	571.70
				1035.01	621.59	1122.07	620.23
				1035.01	660.09	1054.11	574.59
				992.99	643.99	Disabled	Region
						679.05	158.90
						712.57	108.54
						828.67	148.10
						804.14	210.14
						714.23	258.67
						627.30	262.03
						Disabled	Region
						296.57	517.61
						401.29	471.47
						495.27	521.62
						420.91	654.49
						364.08	636.98
						329.33	644.01



outperforms the GWO from the perspective of the ability to get the best result. The perception of it is that the search mechanism of GA is more suitable for this packing problem when compared with the GWO. The GA directly retains good combination of piece with certain rotation angle (genes in chromosome) to next generation while the GWO updates each value of the vector based on first three best solution. This makes GA easier to search better solution (see Fig. 9).

The average (Ave.), median (Med.), interquartile range (IQR) and standard deviation (Std.) for each instance, obtained by GA and GWO, are showed in Table 4. Both average and median of GA in all instances (except PS14) are larger than GWO's. This means the GA has greater ability to find a better solution than GWO when both algorithms run the same number of times. However, the IQR and standard deviation of GWO in most instances (about fourteen instances) are less than GA. This does not mean the GA is worse than GWO. The reason why the dispersion of GA is slightly large than GWO in most instance is that the former has the potential to get higher quality solution than the latter. To visualize the statistical results, the fitness over twenty independent times about partial instances is presented as boxplots in Fig. 10. The vertical axis of each plot expresses the fitness values and the horizontal axis denotes the proposed algorithms. These figures prove the conclusion which are derived from Table 4.

The experimental results show that the proposed approaches can effectively and robustly solve this new kind of irregular bin packing problem. This is achieved because our algorithm deals perfectly with special boundary constraint and the maximization of the percentage of stock sheet utilization.

6. Conclusion

In this paper, a special case of the irregular bin packing problem encountered in the field of two-dimensional spatial arrangement is addressed. After analyzing the characteristics and complexity of the problem, we creatively propose HPA algorithm which able to perfectly deal with the point constraint and the free rotation of piece under the requirement of having all piece without overlapping. This method divides the packing process into

two stages, placing piece along boundary of sheet and placing on interior of new sheet produced by the former stage. This approach is very simple but effective and easy to implement. Two high-level search algorithms responsible for searching sequence of rotation angle are adopted because the proposed HPA is unlikely to provide a global optimum. Some instances are applied to test the performance of the proposed algorithms. The computational results prove their robustness and effectiveness. However, there is still room for further improvement in terms of utilization of sheet when examining the final layout, especially for the type of piece 2. Thus, for future work, the pairwise clustering technique that have proven to be effective will be tried. Also, some other strategies of the GA's operators will be applied for enhancing the search capacity.

CRedit authorship contribution statement

Qiang Luo: Writing – original draft, Formal analysis, Methodology. **Yunqing Rao:** Funding acquisition, Writing – review & editing, Resources. **Deng Peng:** Data curation.

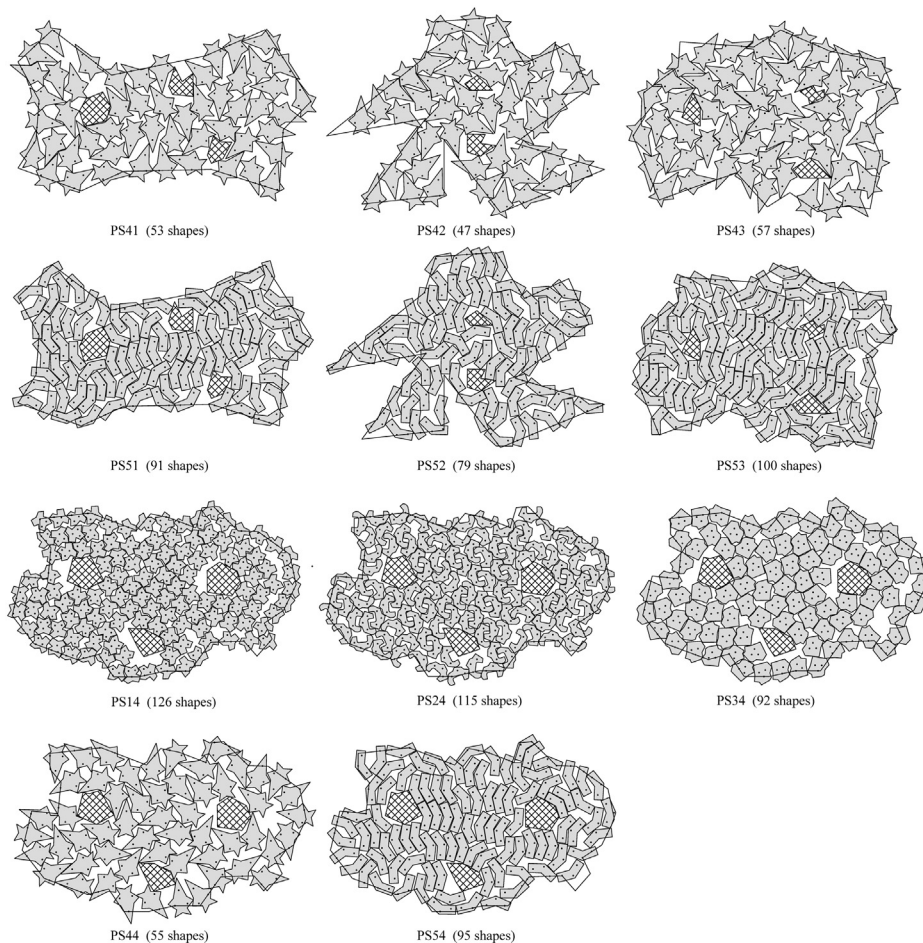
Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant no. 51975231) and Foundational Research Funds for the Central Universities (Grant no. 2019kfyXKJC043).

Appendix A



Appendix B

The data of piece and sheet (see [Tables B.1](#) and [B.2](#)).

References

- [1] X. Hu, J. Li, J. Cui, Greedy adaptive search: A new approach for large-scale irregular packing problems in the fabric industry, *IEEE Access* 8 (2020) 91476–91487.
- [2] J.A. Bennell, M. Cabo, A. Martínez-Sykora, A beam search approach to solve the convex irregular bin packing problem with guillotine cuts, *European J. Oper. Res.* 270 (2018) 89–102.
- [3] L.T. Oliveira, E.F. Silva, J.F. Oliveira, F.M.B. Toledo, Integrating irregular strip packing and cutting path determination problems: A discrete exact approach, *Comput. Ind. Eng.* 149 (2020) 106757.
- [4] R.J. Fowler, M.S. Paterson, S.L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Inform. Process. Lett.* 12 (1981) 133–137.
- [5] R.P. Abeysooriya, J.A. Bennell, A. Martinez-Sykora, Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation, *Int. J. Prod. Econ.* 195 (2018) 12–26.
- [6] J.A. Bennell, J.F. Oliveira, The geometry of nesting problems: A tutorial, *European J. Oper. Res.* 184 (2008) 397–415.
- [7] R. Morabito, S. Morales, A simple and effective recursive procedure for the manufacturer's pallet loading problem, *J. Oper. Res. Soc.* 49 (1998) 819–828.
- [8] K. Dowsland, An exact algorithm for the pallet loading problem, *European J. Oper. Res.* 31 (1987) 78–84.
- [9] M.H. Correia, J.F. Oliveira, J.S. Ferreira, A new upper bound for the cylinder packing problem, *Int. Trans. Oper. Res.* 8 (2001) 571–583.
- [10] E.G. Birgin, J.M. Martinez, D.P. Ronconi, Optimizing the packing of cylinders into a rectangular container: A nonlinear approach, *European J. Oper. Res.* 160 (2005) 19–33.
- [11] C.O. López, J.E. Beasley, A heuristic for the circle packing problem with a variety of containers, *European J. Oper. Res.* 214 (2011) 512–525.
- [12] M.T. Costa, A.M. Gomes, J.F. Oliveira, Heuristic approaches to large-scale periodic packing of irregular shapes on a rectangular sheet, *European J. Oper. Res.* 192 (2009) 29–40.
- [13] T. Imamichi, M. Yagiura, H. Nagamochi, An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem, *Discrete Optim.* 6 (2009) 345–361.
- [14] S.C.H. Leung, Y. Lin, D. Zhang, Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem, *Comput. Oper. Res.* 39 (2012) 678–686.
- [15] P.R. Pinheiro, B. Amaro Júnior, R.D. Saraiva, A random-key genetic algorithm for solving the nesting problem, *Int. J. Comput. Integr. Manuf.* 29 (2016) 1159–1165.
- [16] L.H. Cherri, L.R. Mundim, M. Andretta, F.M.B. Toledo, J.F. Oliveira, M.A. Carravilla, Robust mixed-integer linear programming models for the irregular strip packing problem, *European J. Oper. Res.* 253 (2016) 570–583.
- [17] Y. Rao, P. Wang, Q. Luo, Hybridizing beam search with tabu search for the irregular packing problem, *Math. Probl. Eng.* 2021 (2021) 5054916.
- [18] A.K. Sato, T.C. Martins, M.S.G. Tsuzuki, An algorithm for the strip packing problem using collision free region and exact fitting placement, *Comput. Aided Des.* 44 (2012) 766–777.
- [19] A.K. Sato, T.C. Martins, A.M. Gomes, M.S.G. Tsuzuki, Raster penetration map applied to the irregular packing problem, *European J. Oper. Res.* 279 (2020) 657–671.
- [20] B. Júnior, P. Pinheiro, P. Coelho, A parallel biased random-key genetic algorithm with multiple populations applied to irregular strip packing problems, *Math. Probl. Eng.* 2017 (2017) 1–11.
- [21] A.A.S. Leao, F.M.B. Toledo, J.F. Oliveira, M.A. Carravilla, R. Alvarez-Valdés, Irregular packing problems: A review of mathematical models, *European J. Oper. Res.* 282 (2020) 803–822.
- [22] B. Guo, J. Hu, F. Wu, Q. Peng, Automatic layout of 2D free-form shapes based on geometric similarity feature searching and fuzzy matching, *J. Manuf. Syst.* 56 (2020) 37–49.
- [23] B. Amaro, R. Costa, P.R. Pinheiro, J. Luiz, A. Grichshenko, A biased random-key genetic algorithm using dotted board model for solving two-dimensional irregular strip packing problems, in: 2020 IEEE Congress on Evolutionary Computation (CEC), 2020.

- [24] E. López-Camacho, G. Ochoa, H. Terashima-Marín, E.K. Burke, An effective heuristic for the two-dimensional irregular bin packing problem, *Ann. Oper. Res.* 206 (2013) 241–264.
- [25] A. Martinez-Sykora, R. Alvarez-Valdes, J.A. Bennell, R. Ruiz, J.M. Tamarit, Matheuristics for the irregular bin packing problem with free rotations, *European J. Oper. Res.* 258 (2017) 440–455.
- [26] A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, J.M. Tamarit, Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts, *Omega* 52 (2015) 15–32.
- [27] W. Han, J.A. Bennell, X. Zhao, X. Song, Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints, *European J. Oper. Res.* 230 (2013) 495–504.
- [28] G. Wäscher, H. Hauzner, H. Schumann, An improved typology of cutting and packing problems, *European J. Oper. Res.* 183 (2007) 1109–1130.
- [29] L.R.d. Souza Queiroz, M. Andretta, Two effective methods for the irregular knapsack problem, *Appl. Soft Comput.* 95 (2020) 106485.
- [30] S. Mirjalili, Genetic algorithm, in: S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications*, Springer International Publishing, Cham, 2019, pp. 43–55.
- [31] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* 69 (2014) 46–61.
- [32] B.H. Abed-alguni, N.A. Alawad, Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments, *Appl. Soft Comput.* 102 (2021) 107113.
- [33] Z. Zhu, X. Zhou, An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints, *Comput. Ind. Eng.* 140 (2020) 106280.
- [34] P. Wang, Y. Rao, Q. Luo, An effective discrete grey wolf optimization algorithm for solving the packing problem, *IEEE Access* 8 (2020) 115559–115571.
- [35] K.A. Dowsland, S. Vaid, W.B. Dowsland, An algorithm for polygon placement using a bottom-left strategy, *European J. Oper. Res.* 141 (2002) 371–381.