An Efficient Pixel-based Packing Algorithm for Additive Manufacturing Production Planning

Zedi Lu [a,1], Kanxin Hu[*,a,1], Tsan Sheng Ng[a]

[a]*Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore*

Abstract

Additive Manufacturing (AM), the technology of rapid prototyping directly from 3D digital models, has made a significant impact on both academia and industry. When facing the growing demand of AM services, AM production planning (AMPP) plays a vital role in reducing makespan and costs for AM service companies. This research focuses on the AMPP problem under the unrelated machine environment and two-dimensional irregular packing constraints such that the makespan can be minimized. Since the two-dimensional irregular packing sub-problems are hard to solve, the efficiency of checking the packing feasibility of batches is the bottleneck of algorithms of AMPP. Therefore, we propose an efficient pixel-based AM packing algorithm (PAMPA) which can tackle irregular packing sub-problems that allow hole filling and free rotation. In PAMPA, parts are placed one by one in a Bottom-Left (BL) way and the rotation angle is determined by a novel angle selection heuristic. Two acceleration techniques are proposed to accelerate separating overlap. The placement sequence is optimized by a biased random-key genetic algorithm (BRKGA). Furthermore, by adopting PAMPA, we introduce a variable neighborhood search (VNS) framework to solve the AMPP. Finally, new instances are generated to conduct experiments. Based on the new instances and instances from (ESICUP), the efficiency of PAMPA is analyzed and verified. The VNS framework shows good performance when solving instances, which illustrates that our PAMPA is beneficial for AMPP. Some interesting insights are also revealed and discussed.

Key words: , Additive Manufacturing, 2D Irregular Packing, Production Planning, Heuristics

[*]Corresponding author

Email address: kxhu125@gmail.com (Kanxin Hu)

[1]Kanxin Hu and Zedi Lu are co-first authors.

1. Introduction

Additive Manufacturing (AM), also known as 3D printing, is a state-of-art manufacturing technology that directly fabricates products from their 3D model data. Compared with traditional subtractive manufacturing technologies, AM "adds" materials with a layer-by-layer mode, which makes AM possesses innate advantages in production complexity, design flexibility, and resources efficiency (Hu et al., 2022). Due to its unique characteristics and significant advantages, AM has been applied to many advanced fields including aerospace, automobile, and medical tools (Wohlers, 2020).In recent years, the demand for AM services is huge and still increasing rapidly (Ashima et al., 2021; Tamez & Taha, 2021). In the meantime, the automatic and tool-less AM technology can better adapt to restrictions of safe distancing, lower physical headcount, and slow logistics brought by the global pandemic COVID-19 (Narayanamurthy & Tortorella, 2021). Therefore, many professional AM service companies like 3D Systems, Stratasys, and Shapeways were founded and are still expanding quickly.

However, when encountering a huge AM market and growing demands, AM service companies are suffering two problems in production. The first is the long processing time caused by the layer-by-layer manufacturing process and relatively long setup time (Yim & Rosen, 2012). The second is that the purchasing costs and operating costs of AM machines are fairly high (Li et al., 2017). However, the layer-by-layer production characteristic provides a potential of fabricating many different parts simultaneously as long as the printing space can accommodate all of those parts. Hence, by properly assigning more parts into one processing batch, the overall processing time and operation cost can be reduced and the utilization rate of machines can be improved (Baumers et al., 2017). In other words, when hundreds of parts need to be processed by a few machines, AM production planning (AMPP) plays a vital role in reducing operation cost, makespan, and lead time. For this reason, this paper focuses on AM production planning problem under unrelated machine environment and two-dimensional (2D) irregular packing constraints.

In this work, we study the AMPP problem based on one of the most commonly used metal AM processes —- Selective Laser Melting (SLM). SLM can produce high-quality parts with complex geometric shapes and is widely applied in commercial and industrial areas (Frazier & William,

2014). Fig. 1 shows the production process of SLM. First, the stock area is raised by a certain height and metal powder is moved from the stock area to the printing area by the recoater blade. A thin layer of powder is covered above the parts in the printing area. Then, the laser scanner scans and melts the metal powder in specified areas. A new layer of the parts is formed after cooling. Finally, the printing area moves down. The three steps are repeated until the parts are fully formed. In AMPP with SLM machines, after the AM company receives various orders with parts of different shapes and sizes, the parts need to be distributed into several batches which are processed on heterogeneous SLM machines with different printing areas and production speeds. Fig.2 shows an example of AMPP process with two machines. There are two categories of vital decisions in AMPP. (1) How to assign parts to batches that are allocated to different machines. (2) How to position parts in the same batch such that they can be completely placed inside the printing area. Although allocation and placement are two decisions, they are highly interrelated because different allocations may lead to different packing feasibility. As a result, researchers often consider those two decisions together.

As mentioned above, for metal AM technologies like selective laser melting (SLM), support structures are needed and vertically stacking of parts is not allowed. Therefore, the capacity constraints in the decision (2), which used to be 3D irregular packing constraints, are equivalent to 2D irregular packing constraints. In other words, the 2D irregular packing sub-problem considered in AMPP problem can be described as follows: when given some irregular part projections and a fixed size rectangular building plate, whether all parts can be completely placed into the plate without overlap. It should be noted that in the application scenario of SLM, the shape of some part projections may be very complex. For example, they may have thousands of edges and more than one hole. The space of holes can also be used to place other parts to improve the machine utilization rate. Besides, when placing part projections, they can rotate freely around building orientation on building plates, and choosing proper rotation angles can greatly improve the packing density. For example, in Fig 3, when hole filling and free rotation are permitted, the part 4 and 5 of batch B can be placed into batch A. Therefore, all five parts can be processed in batch C, which may help to save processing time and cost. However, even the simplest 2D irregular strip packing problem that does not allow hole filling and free rotation is NP-hard Fowler et al. (1981). It makes the 2D

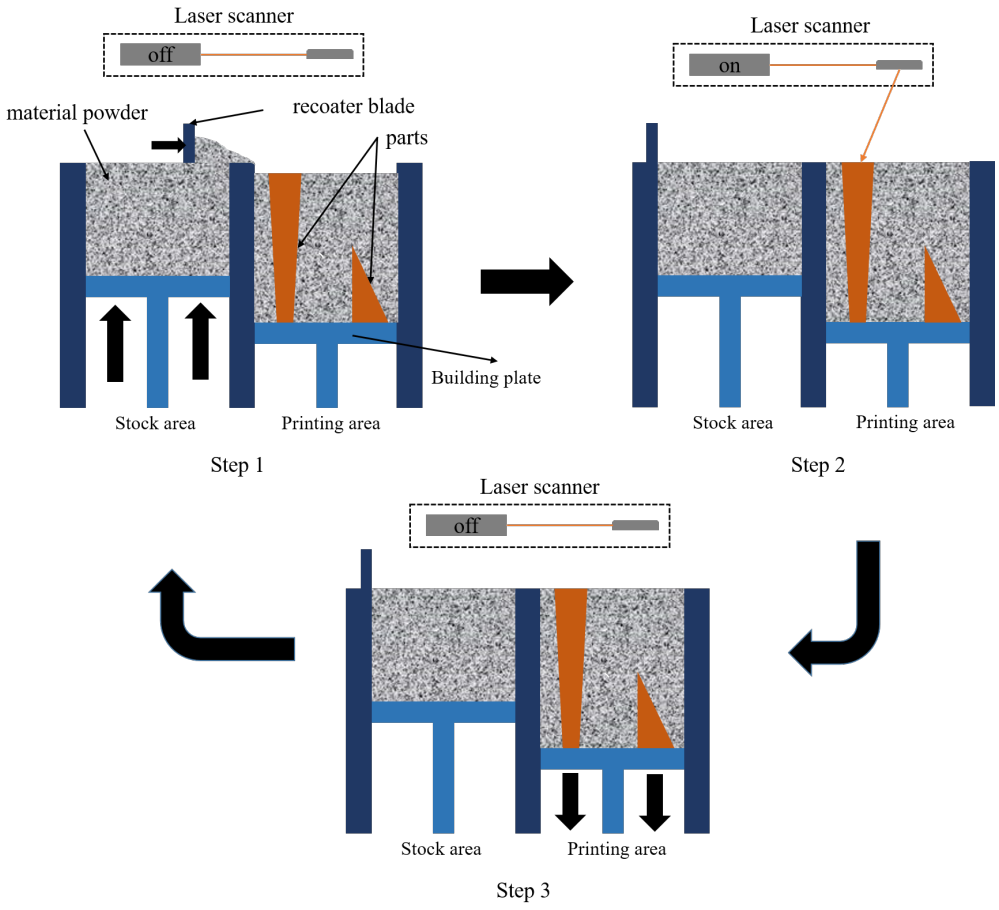

Figure 1: SLM machine production process. Step 1: Material powder is moved by the recoater blade from the stock area to the printing area and a thin layer of powder is formed over the printing area. Step 2: The laser scanner is turned on and scans the specified area to melt powder and form a structural layer of the parts. Step 3: The printing area moves down.

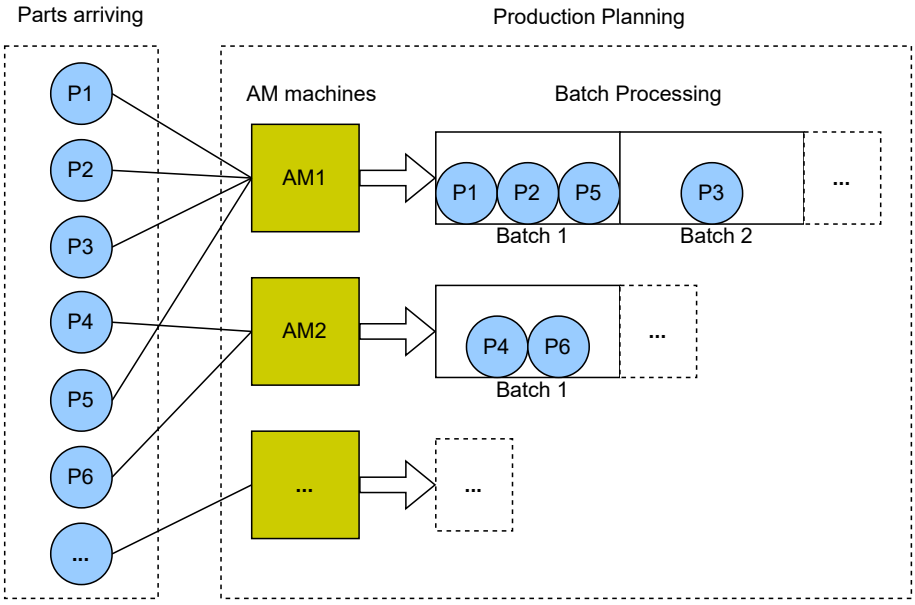

Figure 2: The process of AM production planning. First, parts arrive at the system. Second, parts are allocated to several batches. Finally, unrelated AM machines process the batches sequentially.

irregular packing sub-problem of AMPP more intractable to solve when taking hole filling and free rotation into account. Conventional algorithms for solving 2D irregular packing problems, including exact and heuristic algorithms, usually take hours to seek optimal solutions. In AMPP, we mainly consider checking the packing feasibility of the constructed batches on machines, which is frequently checked during solving AMPP. Therefore, the efficiency of the packing algorithm is the bottleneck of AMPP algorithms.

In current literature, the machine capacity constraints are often modelled as one-dimensional knapsack constraints (Kucukkoc, 2019; Altekin & Bukchin, 2022) or two-dimensional (2D) rectangular packing constraints (Aloui & Hadj-Hamou, 2021; Hu et al., 2022). Very few papers consider the 2D irregular packing constraints (Zhang et al., 2020; Wang et al., 2019; Griffiths et al., 2019). Although 2D irregular packing problems have been studied for years and AM production planning problems have also received more attention recently, no paper has considered developing a fast-speed and easy-implemented 2D irregular packing method which allows hole filling and free rotation for AMPP. Since checking the packing feasibility of solutions is an inescapable problem of AMPP, it is necessary to design a good packing method. Compared with exact algorithms, fast

and efficient heuristic algorithms are more suitable to fulfil the requirements of a huge amount of packing feasibility queries. Considering the process characteristics of SLM and requirements of efficiency mentioned above, four issues should be paid attention to when designing the AM packing heuristic algorithm. First, the number of parts of one batch is usually small or medium size (usually less than 30). Second, all parts can rotate freely on their building plate. Third, space inside holes of parts can be filled by other parts to improve machine utilization rate. Finally, the proposed packing heuristic algorithm can find a high-quality solution within a few seconds.

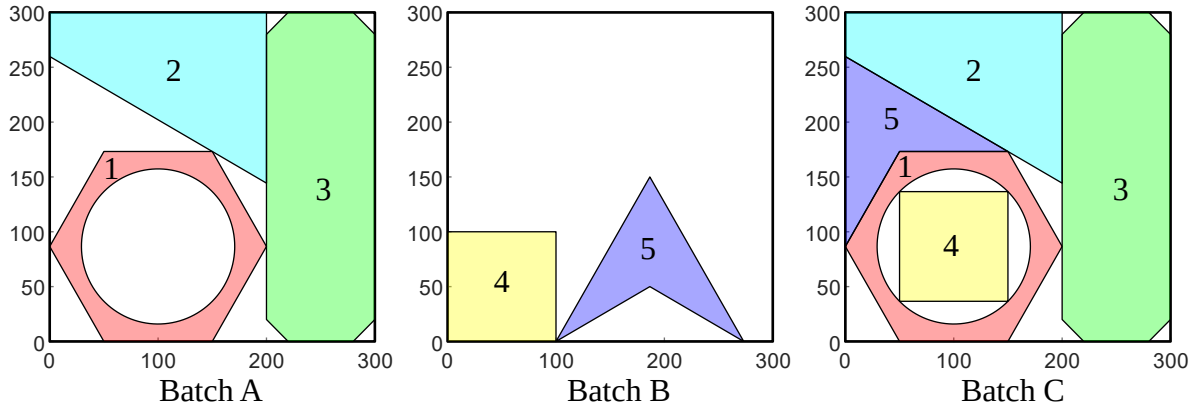

Figure 3: Example of hole-filling and free rotation. Five parts need to be allocated into batches. When hole-filling and free rotation are not allowed, two batches, Batch A and Batch B, are necessary to packing all five parts. However, when we allow the hole-fill and free rotation of parts, Part 4 and 5 can be placed into Batch C together with another parts.

In this study, we solve the AMPP with an unrelated AM machine environment, which aims to minimize makespan under irregular packing constraints. Although many papers have been done for AMPP, the packing sub-problem of AMPP has not been solved well. To the best of our knowledge, we are the first to provide an efficient algorithm for AMPP with a fast-speed and easy-implemented packing algorithm which can tackle the parts with irregular shapes and allow hole filling and free rotation. To solve the packing sub-problem in AMPP, we introduce a pixel-based AM packing algorithm (PAMPA) where the biased random-key genetic algorithm is adopted to optimize the placement sequence of parts and a bottom-left (BL) method is adopted to determine the placement of parts. During the placement, the best rotation angle is selected by evaluating the packing density and attachment value. In the meantime, a novel coding method and a contour-guided method are designed to accelerate the process of BL placement. Then, a variable neighborhood search

(VNS) framework is introduced based on PAMPA to solve the AMPP. Finally, we conduct extensive numerical experiments to evaluate the proposed algorithms and make a comprehensive analysis.

The remainder of this paper is organized as follows. The related works are reviewed in Section 2. In Section 3, we give the formal description of this studied AMPP. The proposed pixel-based AM packing algorithm is described in Section 4, and the VNS framework is described in Section 5. The experiments and results are dedicated in Section 6. The conclusion is drawn in Section 7.

2. Literature Review

In this section, we review the literature related to the problem discussed in this work. We first summarize the research on AMPP and analyze their methods for planning and packing (section 2.1), and then investigate the papers about 2D irregular packing problems (section 2.2).

2.1. AM Production Planning

In the area of AMPP, the research is still limited and most papers were published in recent five years. In this review, the literature is divided into one-dimensional knapsack, 2D rectangular, and 2D irregular packing constraints in AMPP.

(1) Most papers consider one-dimensional knapsack constraints in AMPP, which is simple but impractical. Li et al. (2017) first introduced AMPP and formed a mixed-integer linear programming (MILP) model with the objective function of minimizing the average production cost per material volume. CPLEX is used to solve the model and two heuristics, best-fit and adapted best-fit heuristics, are designed. Chergui et al. (2018) addressed the AMPP with due dates of parts and their objective is to minimize the maximum lateness. Kucukkoc (2019) proposed a MILP formulation for the makespan minimization and considered three machine configurations including single, parallel identical, and parallel non-identical machines. A dynamic order acceptance in on-demand AMPP was designed in Li et al. (2019). To maximize the average profit-per-unit-time, they introduced a method that could decide the order acceptance and scheduling simultaneously. In Alicastro et al. (2021), a meta-heuristic algorithm based on reinforcement learning was proposed for the minimization of makespan, and a probabilistic

stopping rule was implemented to reduce computational times. A bi-objective mathematical model including the makespan and the total tardiness penalty was proposed (Rohaninejad et al., 2021). An efficient hybrid meta-heuristic algorithm with the non-dominated sorting genetic algorithm (NSGA-II) and a novel learning-based local search were founded on the k-means clustering algorithm and a regression neural network was designed to solve the problem. Arık (2021) considered the AMPP under a single AM machine environment. The assembly operations were adapted to this problem. A mixed-integer programming model and a fast heuristic method with a simple local search mechanism for the problem were developed to tackle this AMPP. Altekin & Bukchin (2022) provided an improved MILP model for cost and makespan minimization. Analytical bounds and valid inequalities were proposed. They also did trade-off analysis in the multi-objective model using an efficient frontier approach.

(2) Many publications modelled the capacity constraints as 2D rectangular (regular) packing by representing irregular shapes as their minimum bounding boxes. Though current 2D rectangular packing heuristic algorithms are fast and robust, this kind of simplification may cause a waste of production space. Dvorak et al. (2018) presented the AMPP with due date constraints while minimizing time and satisfying the deadline. In Aloui & Hadj-Hamou (2021), two functions were designed to estimate the production time for two AM technologies. The MILP model was introduced for the AMPP and a heuristic was used to solve large instances of the problem. Che et al. (2021) considered the AMPP with 2D rectangular packing constraints and orientation selection. The MILP model was introduced and a simulated annealing algorithm framework was developed to solve the AMPP. Based on Che et al. (2021), Hu et al. (2022) added the due date constraints to the problem and developed the adaptive large neighborhood search (ALNS) algorithm for the solutions.

(3) Few researchers have paid attention to 2D irregular packing constraints in AMPP. Griffiths et al. (2019) solved the build orientations and 2D irregular packing in AMPP simultaneously for cost minimization by iterative tabu search procedures. Zhang et al. (2020) considered the AMPP with parallel machines and irregular packing constraints. They used the no-fit polygon method to solve the packing sub-problem without hole filling. In addition, inner-fit polygon and no-fit polygon are rotation-dependent, which makes their packing algorithm can only solve problems

that allow few rotation angles for each part. Wang et al. (2019) considered the holes fill for packing with fixed angle rotation. For the AMPP, they only designed a constructive algorithm to allocate the parts.

2.2. 2D Irregular Packing Problems

As mentioned above, we consider 2D irregular packing constraints in AMPP. 2D irregular packing problems usually require placing a set of irregular (non-rectangular) shaped objects into a set of containers without overlap, such that the packing density or profit is maximized. Therefore, in this section, related methodologies and publications of 2D irregular packing problems are reviewed. Due to the hardness of irregular packing problems, heuristic algorithms dominate the publications (Bennell & Oliveira, 2009). In the past two decades, some remarkable publications developed and improved MILP models (Fischetti & Luzzi, 2009; Alvarez-Valdes et al., 2013; Toledo et al., 2013; Cherri et al., 2016; Rodrigues et al., 2017; Souza Queiroz & Andretta, 2022) and non-linear programming models (Chernov et al., 2010; Stoyan et al., 2012; Jones, 2014; Stoyan et al., 2015; Pankratov et al., 2020; Leao et al., 2020) to solve various of irregular packing problems. However, all exact algorithms designed for seeking optimal take hours of running time. In contrast, heuristic algorithms perform more efficiently and we mainly focus on papers of heuristic algorithms. First, we investigate two kinds of widely used geometry tools, then related papers and heuristic algorithms based on those geometry tools are reviewed.

In 2D irregular packing problems, many geometry tools were developed to represent irregular objects and separate overlaps (Bennell & Oliveira, 2008). Rasterization and No-fit Polygon (NFP) are the two most widely used geometry tools in publications of heuristic algorithms. Rasterization first discretizes irregular polygons and containers as small pixels which are coded with a value. There are mainly three ways to rasterize a polygon: the scan-line algorithm, boundary algebra filling algorithm, and ray-crossings algorithm (Zhou et al., 2015). The simplest codification method is to code a pixel whose center lies inside the polygon as 1-pixel, otherwise 0-pixel. If the value of a pixel on the container is more than one, there exists an overlap. Segenreich & Braga (1986) coded border pixels as 1 and interior pixels as 3. Then there exist overlap pixels if the final layout has pixels whose values are greater than 4. Instead of coding pixels outside polygons as 0, Babu & Babu

(2001) marked pixels inside polygons as 0. For pixels that are outside polygon, the rightmost pixel is coded as 1, and pixels from right to left are cumulatively added 1. When placing polygons in a bottom-left way, this kind of codification is useful to separate overlap. There are also many other publications that proposed various kinds of discretization and codification methods(Burke et al., 2006; Sato et al., 2019). The rasterization method is easy to implement, and it can conveniently represent non-convex polygons with different rotation angles. However, it cannot exactly represent polygons with non-orthogonal edges and computational time may quadratically increase with resolution. No-fit polygon (NFP) is the overlapping regions between a pair of polygons (Art Jr, 1966). When the reference point of a polygon is placed on the edges of NFP, two polygons are touching but do not overlap. NFP is an exact method and many heuristic algorithms try to seek the placement point on the edges of NFP. However, if one polygon rotates at a small angle, the NFP may be completely changed. Hence, NFP is rotation dependent and it is not suitable to solve problems where free rotation is allowed. Though there are some robust algorithms to generate NFP (Burke et al., 2007; Bennell & Song, 2008), it is still hard to generate NFP when one of the polygons is very complex. In AM production, the shape of a part projection may be very complicated after adding support structures. Besides, it is beneficial for cost-saving if parts are allowed to rotate freely on the building plate. Therefore, we think that the rasterization method is more appropriate than NFP when solving AMPP packing sub-problems.

Bennell & Oliveira (2009) classified the heuristic algorithms of 2D irregular packing algorithms as searching over layout (Gomes & Oliveira, 2006; Liao et al., 2016) and searching over sequence (Bennell & Dowsland, 1999; Abeysooriya et al., 2018; Fang et al., 2021). Algorithms of searching over layout first place all parts into a variable container simultaneously. Then, the separation algorithm is called to separate overlap. If separation succeeds, parts will be placed into a smaller container, otherwise, the container will be expanded (Sato et al., 2019). For algorithms of searching over sequence, parts are placed one by one with a given sequence. This method needs to decide where to place the parts and which sequence is the optimal placement sequence. The Bottom-Left (BL) placement rule is widely used in heuristics of packing and cutting problems (Segenreich & Braga, 1986; Burke et al., 2006; Pinheiro et al., 2016) because it is efficient and easy to implement. When the BL placement rule is used, one part is firstly placed at the bottom of the container

without overlap, and ties are broken by leftmost placement. Pinheiro et al. (2016) also designed a random-key genetic algorithm to optimize the placement sequence. Besides, Oliveira et al. (2000) proposed another placement method called "TOPOS" and Bennell & Song (2010) further improved the placement methodology. A beam search was applied to find a better placement sequence.

To our knowledge, currently, no fast and efficient heuristic algorithm can solve 2D irregular packing problems in AMPP. In this study, a pixel-based AM packing algorithm is developed. The proposed algorithm can be classified as searching over sequence. All parts are placed with a BL placement way and the placement sequence is optimized with a biased random-key genetic algorithm.

3. Problem Description

The set of parts is denoted by I and the set of heterogeneous SLM machines is denoted by M. Each machine $m \in M$ is featured by the setup time $T_m^{set}(hour)$, the scanning speed $S_m^{scan}(hour/cm^3)$, the recoater speed $S_m^{reco}(hour/cm)$. The printing capacity of machine m is set as (L_m, W_m, H_m) which corresponds to its length (cm), width(cm), and height(cm) respectively. One machine can process multiple parts simultaneously based on its capacity. Each part $i \in I$ is featured by volume $v_i(cm^3)$, support structure volume $s_i(cm^3)$, height $r_i(cm)$. After receiving 3D models (STL format files) of parts, the 3D data need to be converted to 2D data. Then, the projection g_i of part i is generated.

In AMPP, as previously mentioned in Section 1, all parts in I are packed into a set of batches B which are then assigned to and processed by machines M. Within one batch b_m, the heights of all parts in this batch cannot exceed the the height of the machine. Besides, the 2D irregular packing constraints ensure that each projection g_i cannot be put outside the plate of machine m and there should be no overlap between each pair of projections. In this paper, the objective of AMPP is to minimize the makespan C_{max}, that is, the completion time of the last batch. Eq.1 show the formula of computing the processing time t_{bm} of the batch b on machine m.

$$t_{bm} = T_m^{set} + S_m^{scan} \sum_{i \in b} (v_i + s_i) + S_m^{reco} \max_{i \in b} h_i, \qquad \forall b \in B, m \in M, \tag{1}$$

There are three main aspects of time that need to be considered when computing a batch's processing time of a batch(Kucukkoc, 2019). (1) Setup time. The Setup time is the time that spends on completing a series of operations before the printing process such as data uploading, parameters setting, powder filling, and processing environment preparation. In this study, we assume the setup time of a machine is constant. (2) Scanning time. This is the time spent on scanning all the parts and support structures areas, which is related to the total part volume and support structures volume. (3) Powder Coating time. It is the time used for powder spreading, stock area movement, and printing area movement which is proportional to the maximum height among the parts on the printing area.

4. A Pixel-based AM Packing Algorithm

The 2D irregular packing sub-problem in AMPP essentially checks whether all parts can be completely placed inside a fixed container without overlap. This problem can be transformed into a strip packing problem whose container has a fixed width and variable length. After placing all parts into the variable container, an enclosure box is formed based on the layout of the parts. If the length of the box does not exceed the length of the fixed container, it means all parts can be placed into the fixed container. Otherwise, the capacity constraint is violated. In this section, we introduce an efficient pixel-based AM packing algorithm (PAMPA) to solve the 2D irregular packing sub-problem with this method. Before using this algorithm, all projections of parts are rasterized as binary pixel images with a scan-line algorithm. The proposed PAMPA belongs to the heuristic of searching over sequence.

In Section 4.1, a global optimization heuristic, called biased random-key genetic algorithm (BRKGA), is utilized to improve the part placement sequence such that the length of the enclosure box can be reduced. Given the placement sequence, parts are put into the building plate one by one by the accelerated BL placement algorithm and the length of the enclosure box is computed (Section 4.2). Each part is placed with a bottom-left (BL) placement rule and the best rotation angle is determined by packing density and attachment value (Section 4.2.1 and 4.2.2). The novel coding method (Section 4.2.3) and contour-guided method (Section 4.2.4) are adopted to accelerate the process of searching BL placement pixel for each part image.

4.1. Biased Random-Key Genetic Algorithm

Since parts are placed one by one with a given sequence, the placement sequence directly decides the length of the enclosure box of the final part layout. One of the most widely used and reliable placement sequence is the descending order of polygon areas. However, this sequence is not always optimal. Hence, some global optimization methods must be applied to find better placement sequences. To ensure the efficiency of the PAMPA, we adopt the biased random-key genetic algorithm (BRKGA), which was proposed by Bean (1994), to optimize the placement sequence. BRKGA is a variety of evolutionary algorithm, and it is widely used in solving combinatorial optimization problems (Li & Zhang (2018), Soares & Carvalho (2020), Shen et al. (2022)). The pseudo-code for the proposed BRKGA is summarized in Algorithm 1.

Given a part sequence with a length of n, each chromosome is represented by a n dimensional vector whose genes are random numbers in the interval $[0, 1)$. This kind of random-number representation is called random-keys. The decoder is to sort all random-keys in descending order, and the corresponding order is the sequence of placement. The objective is to minimize the length of the enclosure box of the building plate image and it is computed with the placement algorithm of Section 4.2. In Algorithm 1, the Accelerated BL Placement Algorithm is denoted as *placement*. The population size is p. The initial population contains 1 chromosome which represents the descending order of polygon areas, and the other $p - 1$ chromosomes are all randomly generated (Algorithm 1, line 3). In each iteration, the best e chromosomes are selected as the elite group and the others form the non-elite group. The elite group is directly copied into the next population (Algorithm 1, line 9). Then the crossover operator is executed c times to generate c chromosomes for the next generation (Algorithm 1, line 10). At each time of the crossover, one parent is randomly selected from the elite group with equal probability and another parent is randomly selected from the non-elite group with equal probability. Each gene of the child's chromosome is randomly selected from its parents with a probability that is proportional to their objective values. Finally, the remaining $p - e - c$ chromosomes are randomly generated (Algorithm 1, line 11), which are called immigration individuals. The BRKGA stops once it finds a sequence that can place all parts inside the building plate with this sequence. And the PAMPA returns a True result which indicated this capacity constraint is satisfied (Algorithm 1, line 15). Otherwise, the BRKGA continues to run

until it reaches the maximum iteration limit and PAMPA returns a False result.

Algorithm 1 BRKGA

Building plate length L, width W
Population size p, elite size e, crossover size c
Initialize population P with 1 descending area order chromosome and $p-1$ random-keys.
Enclosure box lengths $EBL \leftarrow placement(P, W, L)$
$v_{best} \leftarrow min(EBL)$
$iter \leftarrow 0$
while $iter < MaxIter$ **do**
  $iter \leftarrow iter + 1$
  copy top e chromosomes to next generation
  generate c chromosomes by crossover and mutation for next generation
  add $p - e - c$ random-keys to the next generation
  $EBL \leftarrow placement(P, W, L)$
  $v_{best} \leftarrow min(EBL)$
  if $v_{best} \leq L$ **then**
    return true
  end if
end while
return false

4.2. Accelerated Bottom-Left (BL) Placement Algorithm

4.2.1. Angle Selection Heuristic

When all parts are allowed to rotate freely, each part has an infinite number of rotation angles in theory. However, to guarantee the efficiency of the proposed packing algorithm, it is necessary to design some strategies to select some candidate angles. Wang et al. (2019) selected four rotation angles which make the edges of the minimum bounding box parallel to the edges of the building plate. This kind of static selection method is simple and efficient but may miss some key angles if the polygon shape is not similar to a rectangle. Martinez-Sykora et al. (2017) adopted a dynamic selection method. When adding a new part, they first counted the matches with the edges of placed polygons and containers under different rotation angles. Then some rotation angles with most matches will be selected. However, this angle selection method may lead to an efficiency disaster if a polygon has many edges.

In this paper, we analyze different scenarios and introduce a new angle selection heuristic to

generate the candidate angles (Algorithm 2). The proposed angle selection heuristic is also a static method. First, the shape of a polygon is identified and then different angle selection strategies are chosen based on its shape. In Algorithm 2, the similarity function adopts the angle turning method (Arkin et al., 1991) to measure the similarity of two polygonal shapes. The function yields a similarity metric that ranges from 0 to 1. The more similarity metric is close to 0, the more two polygonal shapes are similar. In this study, we define that two polygons are similar when the similarity metric is no more than 0.2. If a polygon or its convex hull resembles a circle, no rotation is permitted (Fig 4.a, Algorithm 2 line 5). This is because it makes not much difference to rotate a circle-like polygon. If a polygon is similar to its minimum bounding box, two rotation angles are selected which make the edges of its minimum bounding box parallel to the edges of the building plate (Fig 4.b, Algorithm 2 line 17). Another special case is a bulky shape polygon with unsmooth edges (Fig 4.c). Though it is rectangle-like, the similarity function fails to give a similar judgment between itself and its minimum bounding box. This is because of the non-convexity of the polygon. To solve this problem, if its convex hull resembles its minimum bounding box, four rotation angles will be selected (Algorithm 2 line 12). Finally, for polygons that are not contained in any cases above, rotation angles that make the edges of its convex hull parallel with four edges of the building plate are first generated. Then all angles are sorted in ascending order of the area of its axis-aligned bounding box (Algorithm 2 line 14). At most *max_candidate_number* numbers of angles will be selected based on the order. For some symmetrical shapes, only one rotation angle is kept if two or more angles yield the same polygon. If the axis-align bounding box of a polygon exceeds the building plate, this rotation angle will be excluded. Finally, the algorithm returns candidate polygons which are yielded by rotating the polygon with candidate angles.

4.2.2. BL Placement Method

The pseudo-code of the accelerated BL placement algorithm is summarized in Algorithm 3. Initially, the building plate image is rasterized as an all 0-pixel image based on the width and 1.25 times the length of the building plate. Then parts are placed one by one with the given sequence. Each part has a set of candidate part images which are yielded by rastering its candidate polygons. And each candidate part image is placed into the building plate image with a BL placement rule.

Algorithm 2 Angle Selection

Polygon p_0
Candidate polygons: *Candidate*
$cov \leftarrow convex\_hull(p_0)$
if $resemble(cov, circle)$ **then**
  $Candidate \leftarrow p_0$
else
  $min\_box \leftarrow min\_bounding\_box(cov)$
  calculate rotate angle a_0 that makes *min_box* parallel to x-axis
  Angles to be selected: *orientations*
  if $!resemble(min\_box, p_0)$ **then**
    if $resemble(cov, min\_box)$ **then**
      $orientations \leftarrow \{a_0 - \frac{\pi}{2}, a_0, a_0 + \frac{\pi}{2}, a_0 + \pi\}$
    else
      $orientations \leftarrow generate\_angles(cov)$
    end if
  else
    $orientations \leftarrow \{a_0, a_0 + \frac{\pi}{2}\}$
  end if
  $count \leftarrow 0$
  while $count < max\_candidate\_number$ **do**
    $p_1 \leftarrow rotate(p_0, orientations[count])$
    if p_1 can be placed into box & p_1 different from polygons in *Candidate* **then**
      $Candidate.push\_back(p_1)$
      $count++$
    end if
  end while
end if
return *Candidate*

After placing all candidate part images, the best rotation angle among candidate angles of a part is determined based on the packing density (PD) and attachment value. However, if the algorithm fails to find the BL placement pixel after trying all candidate part images, the part cannot be completely placed into the building plate without overlap, and 1.25 times the length of the building plate is returned. Once a part is successfully placed, the building plate image and the length of the enclosure box are updated. Finally, after all the parts are placed, the algorithm returns the length of the enclosure box of the final layout.

Packing density (PD) is obtained by dividing the sum of projection polygon areas by the area of

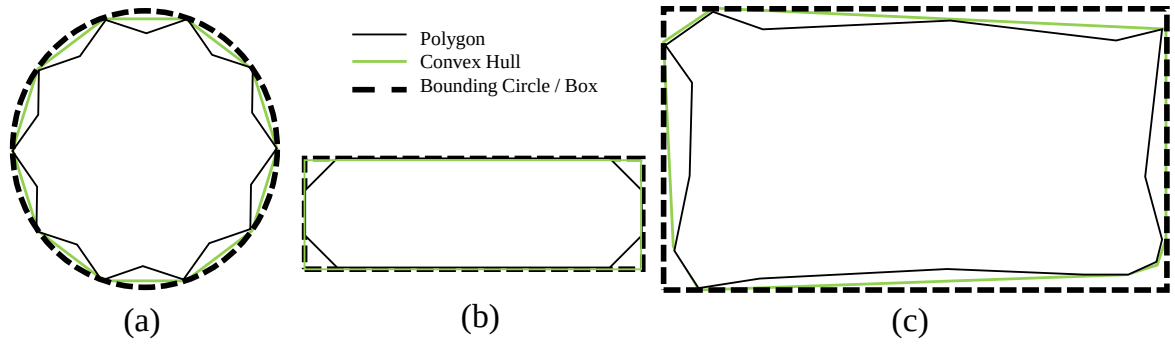

Figure 4: Shape examples for angle selection. (a) If a polygon or its convex hull resembles a circle, no rotation is permitted. (b) If a polygon is similar to its minimum bounding box, two rotation angles are selected which make the edges of its minimum bounding box parallel to the edges of the building plate. (c) When considering a bulky shape polygon with unsmooth edges, if its convex hull resembles its minimum bounding box, four rotation angles will be selected.

the minimum enclosure box of all placed parts. The more PD is close to 1, the more compact the configurations are. Since the area of a polygon will not change after rotation and the width of the building plate image is fixed, the maximum PD criterion is equivalent to the minimum enclosure box length criterion. However, in some cases, placing one part with different rotation angles may yield the same enclosure box. Wang et al. (2019) proposed a criterion called attachment value to break the tie. Attachment value measures how the border of a part attaches to the container and all other placed parts. This criterion is derived from the observation that the more attachment between parts and container boundary, the denser the final layout is.

If the bottom and leftmost pixel of a part image is defined as the reference pixel, the BL placement rule is to find a bottom and leftmost placement pixel on the building plate image for the reference pixel, which can ensure the part is completely placed inside the building plate and there is no overlap between parts. We define the pixel where the part is placed as the BL placement pixel. In pseudo-code of accelerated BL placement algorithm, the function $find\_BL\_pixel$ (3, line 8) takes the role of finding BL placement pixel for each part image. Since finding BL placement pixels is time-consuming, the efficiency of this function directly decides the efficiency of the packing algorithm. To achieve hole filling, the conventional BL placement algorithm initially places a part at the bottom-left pixel of the building plate image. Next, the part is moved to the right one pixel by one pixel if there exists overlap. If the part reaches the rightmost boundary of the building plate and no feasible pixel is found, the part will be placed at the leftmost pixel of the upward row. As

Algorithm 3 Accelerated BL Placement Algorithm

```
Part Sequence: S ; Machine Building Plate Width, Length: W, L
Initializing Building Plate Image BPI based on W, 1.25 * L
Length of enclosure box of part layout: box_length ← 0
for each s' ∈ S do
    best_PD ← +∞, best_attach ← 0
    count_false ← 0, best_pixel ← (0, 0), best_part
    for each candidate part image sc' ∈ s' do
        {find, PD, attach, pixel} ← find_BL_pixel(BPI, sc')
        if find then
            if PD < best_PD or (PD==best_PD and attach > best_attach) then
                best_PD ← PD, best_attach ← attach
                best_pixel ← pixel, best_part ← sc'
            end if
        else
            count_false ← count_false+1
        end if
        if count_false == number of candidate of s' then
            return 1.25*L
        else
            box_length = Update (BPI, best_image, best_pixel)
        end if
    end for
end for
return box_length
```

shown in Fig5, the parts that have already been placed are labeled as green grids and the part that needs to be placed is labeled as orange grids. The orange part is moved one pixel by one pixel to the right and upwards alternatively until there is no overlap. In the example shown in Fig5, it needs 17 moves to find the BL placement pixel. This kind of one-pixel-by-one-pixel movement method is less efficient in searching the BL placement pixel. To improve it, two acceleration techniques are developed and utilized in the accelerated BL placement algorithm.

4.2.3. Novel Coding Method

In this paper, we proposed a novel coding method for part images and building plate images to improve the efficiency of searching for BL placement pixel. The proposed coding method can help the searching algorithm jump over some pixels where there must exist overlap. Part images are still

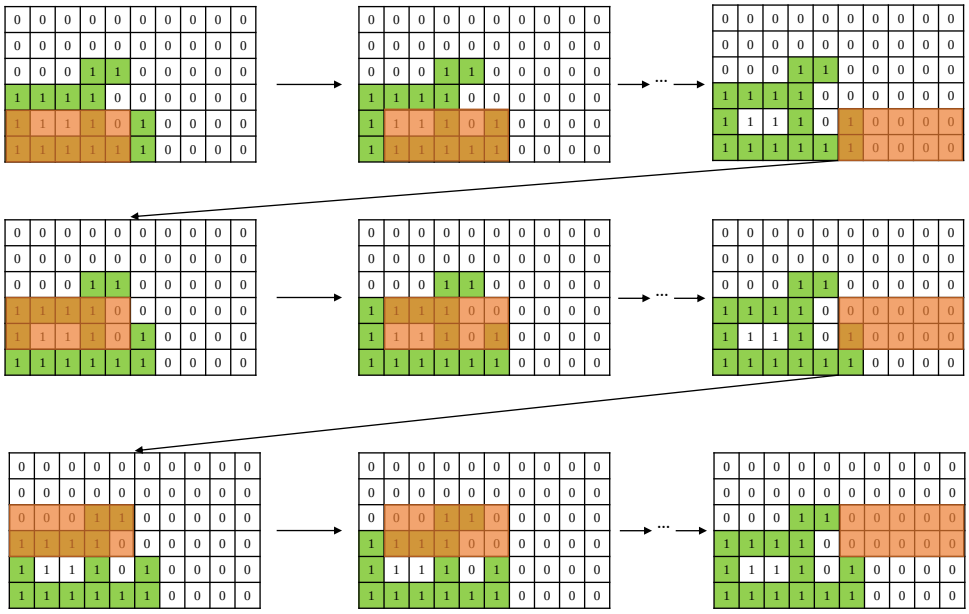

Figure 5: Example of conventional method for searching BL placement pixel. The parts that have already been placed are labeled as green grids and the part that needs to be placed is labeled as orange grids. Searching process of the orange part starts from BL pixel of building plate image, and moves to right grid-by-grid and to upward row-by-row.

initialized as binary images and each part image has a density vector (DV) with the dimension of the number of rows of the part image. Each element of DV equals the maximum number of consecutive 1-pixels in the corresponding row. The building plate image is initialized as a zero-element matrix. After a part finds its BL placement pixel, all elements of the building plate are updated. Each element of the building plate matrix equals the number of consecutive non-zero pixels on the right of that pixel which means this part must move to the right for at least this element's number of pixels to avoid overlap. In addition, the building plate matrix also has a vacancy vector (VV) with the dimension of the number of rows of the building plate matrix. The VV is initialized as a zero-element vector. Each element of the VV is the maximum number of consecutive 0-pixels in each row of the building plate matrix, which indicates the maximum vacancy pixels of this row. If the part reaches the rightmost boundary, it will move upwards until its DV is component-wise less or equal to the corresponding value of VV. The VV also needs to be updated after a part has successfully been placed and only the rows of the part been placed need to be updated.

Fig6 utilizes the same example as Fig5 to illustrate how the proposed novel coding method can improve the efficiency of searching for BL placement pixel. The first part has already been placed

at the BL corner of the building plate matrix, and the non-zero pixels are marked as green grids (Fig6.a). Then the VV of the building plate image is derived from the definition mentioned above. The second part (Fig6.b) is rasterized as a 2×5 matrix with all 1-pixels and its DV is $[5, 5]^T$. As shown in Fig6.c, the second part is initially placed at pixel (1, 1). To detect if there is an overlap between the two parts, a one-pixel-by-one-pixel scan is conducted from the BL pixel of the second part to its top-right pixel. The first overlap is detected at pixel (1, 1) and the corresponding element on the building plate matrix is 6 (marked as the dark color grid in Fig6.c). But if the second part moves to the right for 6 pixels, it will exceed the rightmost boundary. Hence, the second part must move upwards. By comparing VV and DV (Fig6.d), the second part must move upwards for 2 rows to find a possible placement pixel. Then, the second part is placed at pixel (3,1). By repeating the steps above, the BL placement pixel of the second part is pixel (3,6). By utilizing the proposed coding method, the example of Fig5 only moves 3 steps to find the BL placement pixel. Finally, the elements of the third row and fourth row of the building plate image and VV are updated (Fig6.e).

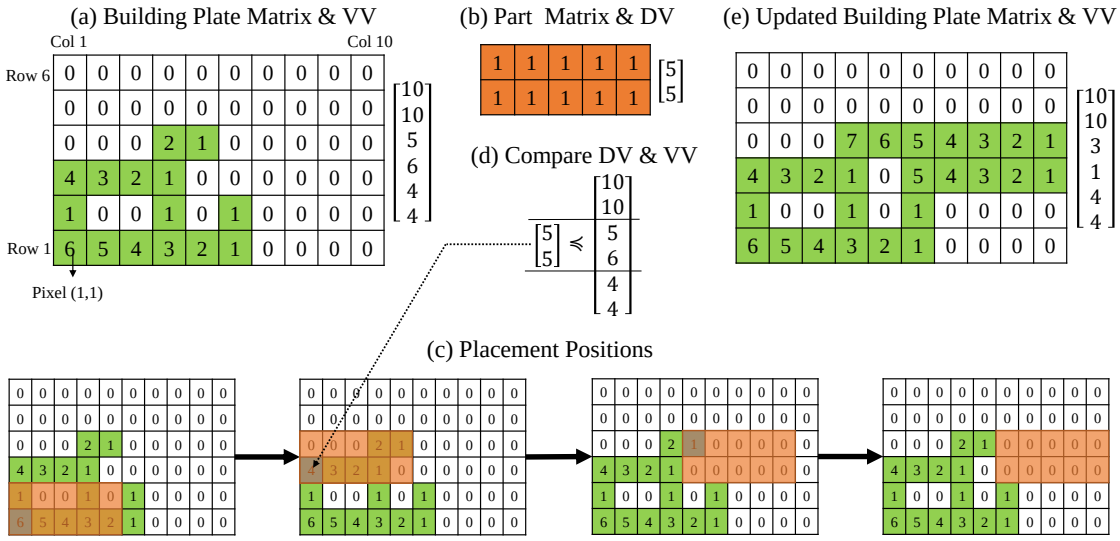

Figure 6: Example for proposed coding method. (a) The building plate matrix and VV are shown. The first part is placed at the BL corner of the building plate matrix, and the non-zero pixels are marked as green grids. (B) The part matrix and DV are shown. The second part is rasterized as a 2×5 matrix with all 1-pixels and its DV is $[5, 5]^T$. (c) The four pictures show the moving process of the second part. (d) The comparison between DV and VV is shown. (e) After deciding the placement of the second part, we update the building plate matrix and VV

4.2.4. Contour-guided Method

The second acceleration technique is the contour-guided method. As mentioned in Section4.2.3, every time a part image is placed at a new position, the algorithm needs to check all pixels of the part image to see whether there exists an overlap. Hence, the computational time spent in detecting overlap increases rapidly with the size of the part image. The contour-guided method is proposed to reduce the time spend detecting overlap. This method derives from the Theorem2 (Appendix A) that if the outer border pixels of a part image do not overlap with the building plate image, then the part image will not overlap with the building plate image. Instead of checking all pixels of the part image when detecting overlap, the contour-guided method only needs to ensure outer border pixels do not overlap with the building plate image. This technique saves time for checking the interior pixels and background pixels, and it will greatly improve the efficiency of the packing algorithm especially when the size of part images is big. The proof of Theorem2 is stated in Appendix A.

5. VNS Framework for AMPP

To verify the effectiveness of PAMPA, we design a heuristic framework for AMPP where PAMPA is applied. Based on literature, variable neighbourhood search (VNS) is a widely used meta-heuristic for solving combinatorial optimization problems (Wei et al., 2014, 2015; Yazdani et al., 2017; Li et al., 2018; Pan et al., 2019; Gong et al., 2020). In this study, the AMPP is an unrelated batch machine scheduling problem, which is also a kind of special combinatorial optimization problem. We propose a new VNS scheme based on Beldar et al. (2022) and Bilyk et al. (2014). The proposed PAMPA is used in the VNS framework to check the packing feasibility for each batch. The pseudo-code of the proposed VNS framework is summarized in Algorithm 4. A greedy method is utilized to construct the initial solution. After that, shacking and local search is alternatively conducted until the VNS framework reaches the time limit or the number of maximum non-improvement times k. Section 5.1 introduces the three hierarchical structures for checking packing feasibility. The greedy construction method for the initial solution is stated in Section 5.2. Section 5.3 and 5.4 presents the details of the local search and shacking procedures respectively.

Algorithm 4 VNS Framework

Construct the initial solution S;
$S_{best} \leftarrow S$, $S_0 \leftarrow S$
$v_{best} \leftarrow obj(S)$, $v_0 \leftarrow v_{best}$
$k \leftarrow 1$
while time limit is not reached **and** $k <$ Max non-improvement **do**
  $S' \leftarrow shaking(S_0, k)$
  $S'', v'' \leftarrow local\_search(S')$
  if $v'' < v_{best}$ **then**
    $S_{best} \leftarrow S''$
    $v'_{best} \leftarrow v''$
    $k \leftarrow 1$
  else
    $S_0 \leftarrow S_{best}$
    $k \leftarrow k + 1$
  end if
end while
return S_{best}, v_{best}

5.1. Packing Feasibility Checking

The function *check-feasibility* is used to check if all parts of a batch can be completely placed into the building plate without overlap. Since checking packing feasibility is the bottleneck of AMPP algorithm efficiency, a three hierarchical structure method is adopted to accelerate the speed. First, we generate the box constraints which are a vector of Boolean variables with the dimension of the number of machines. Each element of the vector records whether the minimum bounding box of the part can be completely placed inside the corresponding machine building plate. Given the part list in a batch of one machine, the function *check-feasibility* firstly checks if all box constraints of the part in this batch of this machine are satisfied. If there is a False result, there is a part whose minimum bounding box exceeds the size of the building plate. Second, the function checks if the sum of all parts areas exceeds the area of the building plate. If the area constraint is broken, the function returns False. Third, the function calls the PAMPA to check if the projections of all parts can be placed into the building plate. Finally, the function *check-feasibility* returns the result of PAMPA.

5.2. Initial Solution

The VNS framework adopted a greedy way to construct the initial solution, which consists of three steps:

Step1: all parts are sorted in descending order of height. If two parts have the same height, the descending projection area order is used. Then parts are assigned to the machines one by one in this order.

Step2: When placing one part, if there are m machines and b batches in the current solution, $b + m$ new solutions are constructed by assigning the part to b existing batches and m new batches for each machine.

Step3: Checking the packing feasibility of all new constructed solutions on Step2. If one new solution is feasible, its makespan is calculated. The solution that has minimum makespan is chosen.

Step2 and Step3 are repeatedly conducted until all parts are assigned to the solution.

5.3. Local Search

In the proposed VNS framework, the local search is used to improve a solution. The local search includes two stages. In the first stage, an operator, called *Balancing*, is conducted to balance the completion time between machines. *Balancing* operator is a greedy improvement method, and it first finds the machine with the maximum completion time. When this machine has more than one batch and the start time of its last batch is smaller than the completion time of other machines, *Balancing* attempts to assign this batch to those machines with earlier completion times. If the packing is feasible, the assignment with minimum makespan is preserved. The above steps are repeatedly conducted until no batch can be moved.

In the second stage of the local search, four different neighbourhood structure operators are used to search for a better solution. The proposed four operators randomly relocate/swap one or more parts and they are summarized as follows.

Insert Part: A part is randomly selected and removed from the selected batch. Then this part is randomly inserted into another batch if the packing is feasible.

Swap Part: Two different batches are randomly selected and two randomly selected parts from each batch are swapped if the packing of swapped batches is feasible.

The **Insert (Swap) Part Sequence** This operator has the same procedure as Insert Part (Swap Part) operator except that it randomly insert (swap) one or more parts from one batch into (with) another batch.

Each operator is conducted at most 10 times until it finds a feasible solution. The proposed four operators are conducted sequentially. The pseudo-code of the local search is presented in Algorithm 5. The *obj* in line 4 and 6 is the objective function to compute makespan and *NS* is the neighbourhood structure operators for constructing a new solution. The local search terminates until it reaches the maximum iteration times ($LS\_MaxIterTimes$) or maximum non-improvement times ($LS\_MaxNonImpTims$).

Algorithm 5 Local Search

1: Solution S'
2: $S' = balancing(S')$
3: $S_{best} \leftarrow S'$, $S_0 \leftarrow S'$,$iter \leftarrow 0$, $nonImp \leftarrow 0$
4: $v_{best} \leftarrow obj(S')$, $v_0 \leftarrow v_{best}$
5: **while** $iter < LS\_MaxIterTimes$ **and** $nonImp < LS\_MaxNonImpTimes$ **do**
6: $S'' \leftarrow NS(S_0), v'' = obj(S'')$
7: **if** $v'' < v_{best}$ **then**
8: $S_{best} \leftarrow S''$
9: $v'_{best} \leftarrow v''$
10: $nonImp \leftarrow 0$
11: **else**
12: $S_0 \leftarrow S_{best}$
13: $nonImp \leftarrow nonImp + 1$
14: **end if**
15: $iter \leftarrow iter + 1$
16: **end while**
17: **return** S_{best}, v_{best}

5.4. Shaking

Shaking is designed for diversifying the solution, which plays a pivotal role in helping the local search jump out of the local minimum. In shaking, three batch-scale neighbourhood structure operators are designed, and they are summarized as follows.

Relocate Batch: One batch is randomly selected and inserted into another machine if this batch satisfies the capacity constraint of this machine.

Swap Batch: Randomly select two batches from two different machines and swap them if the two swapped batches are both packing feasible.

Split Batch: A batch is randomly selected and some parts are randomly selected from this batch to form a new batch. Then randomly insert this new batch into a different machine if it is packing feasible on this machine.

Each shaking operator is conducted at most 10 times until it finds a feasible solution. In shaking, the number of non-improvement iterations (k) of VNS determines the operator and shaking times. When k is no greater than 5, the relocate batch operator is repeatedly conducted for $a(k-1)+1$ times. Otherwise, if k is no greater than 10, the swap batch operator is conducted for $a(k-6)+1$ times. Finally is k is bigger than 10 and no less than 15, the split batch operator is repeatedly conducted for $a(k-11)+1$ times. The parameter a is tuned in the Section 6.3.1.

6. Experiments and Analysis

In this section, we conduct extensive computational experiments to test our methods. New AMPP instances are generated in Section 6.1. In Section 6.2, three computational experiments are conducted based on instances from the EURO Special Interest Group on Cutting and Packing website (ESICUP) and newly generated instances (6.1b) to verify the efficiency of proposed PAMPA. In Section 6.3.1, we design the tuning experiments for some key parameters of the proposed VNS framework. Then the computational experiments and results analysis for the VNS framework is demonstrated by using the newly generated instances in Section 6.3.2. All computational experiments are conducted on 2.4GHz Intel Xeon E5-2680 v4 with 64GB RAM in a C++ environment.

6.1. Data Generation and Environment

As there are few publications of AMPP and no instances are available online, some new instances are generated. We collect 100 3D models, which are STL format files, from *"Thingiverse"*. With the help of Autodesk Meshmixer3.5 which is a professional AM software for dealing with triangle meshes, the height and volume of parts and volume of support structures are calculated and recorded. By utilizing Boolean operations of the Computational Geometry Algorithm Library

(CGAL), we generate the projection polygon of each part which is simple and counterclockwise. In the meanwhile, the area and minimum bounding box of the polygon are also recorded. By referencing the Che et al. (2021), four types of SLM machines with different sizes and processing parameters are introduced. Based on the above 100 parts and 4 types of machines, 6 classes of instances are generated. Classes C1,C2, and C3 have 25, 50, and 75 parts respectively. Two smallest machines are assigned to the three instances. Classes C4, C5, and C6 have 100, 150, 200 parts respectively, and the three instances utilize all four machines. Each class contains 5 different instances. Parts in each instance are randomly selected from the 100 parts and duplicated parts are permitted.

6.2. Experiments for PAMPA

In this section, three computational experiments are conducted. Experiments of Section 6.2.1 verify the proposed PAMPA can find high-quality solutions within less time compared with the previous study. In Section 6.2.2, the efficiency of the proposed two acceleration techniques is tested. In Section 6.2.3, it is shown that the proposed angle selection heuristic is beneficial for saving run-time.

6.2.1. The Efficiency of PAMPA

To verify the efficiency of the proposed AM packing algorithm, PAMPA, we adopt 16 strip packing instances from (ESICUP). To compare with the algorithm of Wang et al. (2019), the width of all containers is also set as 400 pixels. In BRKGA of PAMPA, the population size p is set as 10. The elite and crossover sizes are 2 and 6. The maximum iterations of BRKGA is set as 10. After 20 runs, the average packing density and average computational time (seconds) are recorded.

As shown in Table 1, the average packing densities *Avg PD* of PAMPA are better than the packing densities of Wang et al. (2019) in 15 out of 16 instances. Wilcoxon signed rank tests are introduced to compare the packing densities between the two algorithms. The last column of Table 1 lists test results of the Wilcoxon signed rank test at a significance level of 5%. The Sign "<" means the packing density of Wang et al. (2019) is less than our PAMPA and the sign "=" means there is no obvious difference at a confidence level of 5%. Since 13 instances have "<" test results and 3 instances have "=" test results, we conclude that the proposed PAMPA is competitive

in finding better packing solutions. Wang et al. (2019) only gave the run-time of the first three instances. Comparing the three instances, it takes 20% less run-time for the proposed PAMPA to find better solutions. Meanwhile, with the increase in the number of parts, the average run-time of PAMPA also increases. But when the number of parts is less than 30, the average run-time is less than one second. As mentioned above, the number of parts in one batch is less than 30 in most cases of AMPP. Therefore, the run-time of PAMPA can satisfy the time requirement for solving AM irregular packing sub-problem. In summary, the proposed PAMPA can find better packing solutions within less computational time. It is an efficient and competitive algorithm to solve 2D irregular packing sub-problems in AMPP.

Table 1: Experiment Results of PAMPA

			Wang et al. (2019)		PAMPH		Wilcoxon
#	Instance	Part Number	PD	Run-time(s)	Avg PD	Avg Run-time(s)	PD
1	albano	24	0.848	4.69	**0.850**	0.884	<
2	mao	20	0.730	4.55	**0.766**	0.272	<
3	blaz	20	0.707	3.31	**0.729**	0.644	<
4	poly1a	15	0.657	-	**0.733**	0.227	<
5	poly2a	30	0.696	-	**0.761**	0.660	<
6	poly2b	30	0.713	-	**0.745**	0.725	<
7	poly3a	45	0.720	-	**0.758**	1.471	<
8	poly3b	45	0.743	-	**0.754**	1.198	<
9	poly4a	60	0.705	-	**0.760**	1.824	<
10	poly4b	60	0.750	-	**0.753**	1.795	=
11	poly5a	75	0.718	-	**0.754**	3.050	<
12	poly5b	75	0.750	-	**0.759**	3.300	<
13	shirts	99	**0.867**	-	0.866	2.810	=
14	trousers	64	0.883	-	**0.896**	1.906	<
15	shapes1	43	0.693	-	**0.69**5	1.465	=
16	dagli	30	0.810	-	**0.834**	0.571	<

6.2.2. Experiments for Acceleration Techniques

In this section, four computational experiments are designed to verify whether the two proposed acceleration techniques can accelerate the packing algorithm. The same 16 instances in Section 6.2.1 are adopted, and the packing algorithm only calculate the length of container with one packing sequence, which is the descending order of area. Experiments T1&T2 apply both technique1 (the novel coding method) and technique2 (the contour-guided method). Experiments T1 only apply technique1 and experiments T2 only apply technique2. Meanwhile, experiments NONE mean no technique is used. After 20 runs, the average computational time (seconds) for each instance is recorded (Table 2). The last three columns are the ratios between NONE and the results with accelerated methods, which reflects the speed-up of applying one or two techniques. From Table 2 we can see that two acceleration techniques can significantly reduce the computational time. By adopting the two techniques simultaneously, the average speed-up can reach 164 times.

6.2.3. Experiments for Angle Selection Heuristic

In this section, two computational experiments are designed to test the efficiency of the proposed angle selection heuristic in PAMPA. Experiments E1 adopt the four rotation angles selection method which is the same as Wang et al. (2019), while the proposed angle selection heuristic is utilized in experiments E2. To be fair, the *max_candidate_number* is also set as 4 in E2. Since no part in instances from (ESICUP) has a complex shape or holes, the newly generated instances of Section 6.1 are used in experiments of this section. Considering the number of parts in one batch rarely exceeds 100, only the instances of class C1, C2, C3 and C4 are considered in this experiment. After 20 runs, the average packing density and average run-time (seconds) are listed in Table 3. The Wilcoxon signed rank test at a significance level of 5% shows the average packing densities between the two methods are similar and 19 out of 20 instances have "=" results. The average of the packing density is 0.583 when using the method from Wang et al. (2019) and it is 0.584 when adopting our method, which indicates there is no apparent difference between the two methods. However, the average run-time of experiment E2 for all 20 instances is less than that of experiment E1. When applying our method, it saves 20% of the time to find solutions with similar quality.

Table 2: Experiment Results for Acceleration Techniques

#	Instance	T1&T2	T1	T2	NONE	NONE/T1&T2	NONE/T1	NONE/T2
1	albano	0.0067	0.0657	0.0669	2.3065	344	35	35
2	mao	0.0028	0.0236	0.0171	0.4572	166	19	27
3	blaz	0.0067	0.1257	0.0654	1.7806	266	14	27
4	poly1a	0.0027	0.0154	0.0095	0.214	81	14	23
5	poly2a	0.0054	0.0509	0.0325	0.6925	129	14	21
6	poly2b	0.0065	0.1172	0.0314	0.7663	119	7	24
7	poly3a	0.0088	0.0889	0.0621	1.2694	144	14	20
8	poly3b	0.0094	0.1477	0.0514	1.0287	110	7	20
9	poly4a	0.0124	0.1211	0.0995	1.9735	159	16	20
10	poly4b	0.0148	0.1988	0.0822	1.4805	100	7	18
11	poly5a	0.019	0.1615	0.1475	2.7979	148	17	19
12	poly5b	0.0192	0.2313	0.1149	1.9666	102	9	17
13	shirts	0.0105	0.0773	0.1007	1.1129	106	14	11
14	trousers	0.0123	0.0226	0.1309	3.3197	270	147	25
15	shapes1	0.0101	0.0661	0.099	1.9496	193	29	20
16	dagli	0.0064	0.0418	0.0443	1.2132	190	29	27
Average						164	25	22

Table 3: Experiment Results for Rotation Angle Selection Method

Instance	E1		E2		Wilcoxon
	Avg PD	Avg Run-time(s)	Avg PD	Avg Run-time(s)	PD
C1-1	0.627	1.259	0.627	0.882	=
C1-2	0.500	1.019	0.505	0.855	=
C1-3	0.533	0.614	0.532	0.556	=
C1-4	0.568	0.965	0.570	0.858	=
C1-5	0.486	1.067	0.488	0.826	=
C2-1	0.627	0.757	0.623	0.589	=
C2-2	0.531	1.028	0.536	0.729	=
C2-3	0.606	0.812	0.602	0.804	=
C2-4	0.546	1.167	0.540	1.051	=
C2-5	0.439	0.934	0.434	0.696	=
C3-1	0.577	4.560	0.579	3.610	=
C3-2	0.584	4.813	0.586	4.060	=
C3-3	0.562	5.995	0.563	5.057	=
C3-4	0.670	4.373	0.670	3.626	=
C3-5	0.680	3.510	0.681	2.613	=
C4-1	0.630	4.617	0.629	3.979	=
C4-2	0.593	4.494	0.595	3.519	=
C4-3	0.634	4.141	0.635	3.771	=
C4-4	0.600	3.640	0.601	3.338	=
C4-5	0.665	3.179	0.675	2.828	<
Avg	0.583	2.647	0.584	2.212	

6.3. Experiments for VNS Framework

Since we design a new VNS framework with our PAMPA, the effectiveness of this VNS with PAMPA is tested. In Section 6.3.1, we conduct the parameter tuning for VNS. Then, in Section 6.3.2, we show the results of our instances which illustrate that PAMPA can be used for AMPP.

6.3.1. Parameter Tuning

To make full use of the VNS framework, it is necessary to adjust the values of the parameters before the experiments. 12 new instances are generated for parameter tuning. The 3 parameters are the maximum iteration times of local search ($LS\_MaxIterTimes$), the maximum no improvement times of local search ($LS\_MaxNonImpTims$), and the VNS shaking parameter (a). Based on Bilyk et al. (2014) and Beldar et al. (2022), the parameter $LS\_MaxIterTimes$ takes values of 100, 150, and 200; the parameter $LS\_MaxNonImpTims$ takes values of 10, 20, 30, 40 and the parameter a takes values of 2, 3, and 4.

To get the tuning results, we use the IRACE package (López-Ibánez et al., 2016) which is a software package that implements several automatic configuration procedures. IRACE package takes a parameter space definition and a set of instances as input and searches in the given space for good configurations. The average $Makespan$ is set as the comparison indicator. The tuning ressult for $LS\_MaxIterTimes$ is 150, the best value for $LS\_MaxNonImpTims$ is 20, and a take value of 3.

6.3.2. Experimental Study

In this section, the proposed VNS framework is used to solve AMPP problems of 30 instances generated in Section 6.1. The time limit is set as 3600 seconds and the width of each pixel represents 1mm. After 10 runs for each instance, the makespan, VNS framework running time (run-time), and time of finding the best solution (best-time) are recorded.

The results of each class are shown in Table 4 where the makespan of initial solution (Init MS), average makespan (Avg MS), average run-time (Avg RT), and average best-time (Avg BT) are analyzed. The fourth column is the improvement of the makespan between Init MS and Avg MS. The improvement of Class C4 is the most and the makespan is reduced by 14.06 hours. When there

are few parts and machines like C1, C2, and C3, the search space is also small and the improvement is limited. Considering four machines and more than 100 parts (C4, C5, and C6), the improvement is more obvious. However, while the number of parts is 200, the complexity of the AMPP is high and the search process is also more time-consuming, which leads that the improvement of C6 is smaller than C4 and C5 within the limited computational time. In total, the VNS framework can reduce the makespan by an average of 8.04 hours, one-third of a day. Fig 7 is a plot of the average run-time and average best-time of each class. From the plot, we can see that if the number of machines is unchanged, the average run-time and average best-time increase with the number of parts.

Table 5 is the final solution of instance C1-1 found by VNS framework. In Table 5, the "B" means "batch" and the first number after "B" is machine ID and the second number is batch ID. In the solution of C1-1, there are three batches. Two batches are assigned to machine 3, and one batch is assigned to machine 4. Fig 8 shows the parts configurations for all batches of this solution. Pictures in the first row are configurations of 2D projections on the building plate, and pictures in the second row are diagrams of their 3D placement respectively. The holes filling can be observed in B3-2 and B4-1. As previously mentioned, it is beneficial to pack more parts and improve the utilization when considering hole filling. For example, without hole filling, it is hard to only use one batch to produce the parts in B4-1.

In the meantime, although the height of the part affects the production time, assigning parts with similar heights into the same batch does not necessarily guarantee a reduction in makespan. From Table 5 we can see that only batch B3-2 has a small height standard deviation, and the standard deviation of the other two batches is much higher than this batch. The 3D diagrams in Fig 8 also clearly demonstrate this. Since the powder coating time of each batch is only determined by the part with the maximum height in this batch, it is not sure that assigning parts with similar heights into one batch is helpful to reduce the maximum part height of the batch. Therefore, if we want to find a better production plan for AM, assigning parts with similar heights into one batch is not always a good choice.

In Fig 9, we show the 2D projection configurations of all batches in the final solution for instance C1-1, C2-1, C3-1, C4-1, C5-1, and C6-1. From the 2D projection configuration of batch

C1-1, B3-2, we can see that the proposed PAMPA can place parts inside the hole space of a part, which can improve the space utilization rate of the machine. However, there are still some batches that only have a few parts, e.g., batch C2-1, B3-2 and batch C5-2, B2-2. This is because AMPP is not merely a packing problem, but a batch machine scheduling problem. The makespan of AMPP is the completion time of the last batch, and it is determined by height, volume, support structure, the shape of the parts, and the processing parameter of AM machine. Sometimes a batch with high packing density may cause a huge increase in makespan. However, if we remove some parts from this batch, a new batch is formed by grouping these parts. By allocating the new batch to other machines, the workload of the machines can be balanced and the makespan may be optimized. In addition, if the assignment of the machine with maximum completion time is kept unchanged, we can construct other different solutions that have the same makespan by reassigning the batches of the remaining machines so long as their completion times do not exceed the makespan. Therefore, the batch assignment of machines, whose completion times are not the maximum, has more solution space, and sometimes only assigning a few parts as a batch does not change the final makespan.

Table 4: Results of VNS Framework

Instance	Init MS (h)	Avg MS (h)	Improve (h)	Avg RT (s)	Avg BT (s)
C1	40.03	36.16	3.87	1507.82	766.92
C2	64.21	58.94	5.27	2353.79	1507.06
C3	110.27	105.28	4.98	2971.51	2293.22
C4	78.37	64.32	14.06	2556.12	1587.17
C5	101.97	88.76	13.21	2815.69	1734.32
C6	118.37	111.55	6.82	3370.95	2424.12
Avg.	85.54	77.50	8.04	2595.98	1718.80

Table 5: Best Solution of Instance C1-1

Batch	Model ID	Avg Height (mm)	Std. Height (mm)
C1-1, B3-1	(0, 6, 38, 57, 97, 97)	41.37	32.54
C1-1, B3-2	(12,21,37,37,42,45,53,65,76,85,91)	14.59	7.86
C1-1, B4-1	(19,38,50,55,55,80,81,85)	49.55	41.15

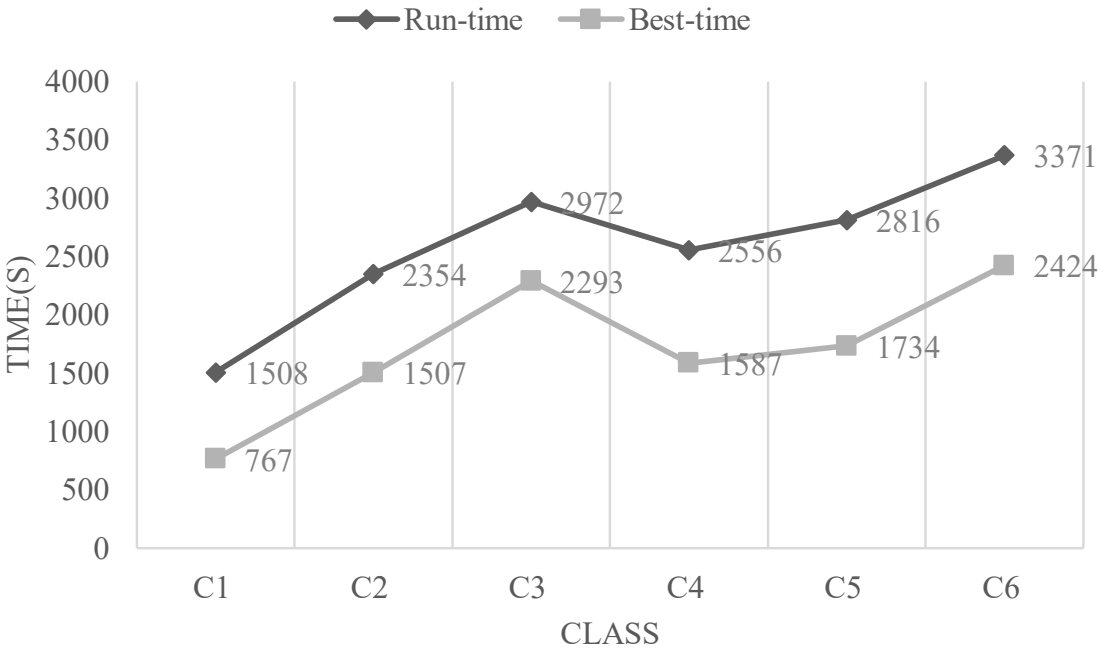

Figure 7: Captain: Plot of average run-time and best-time for each class. Alt Text: Run-time: VNS Framework computation time. Best-time: time of finding the best solution

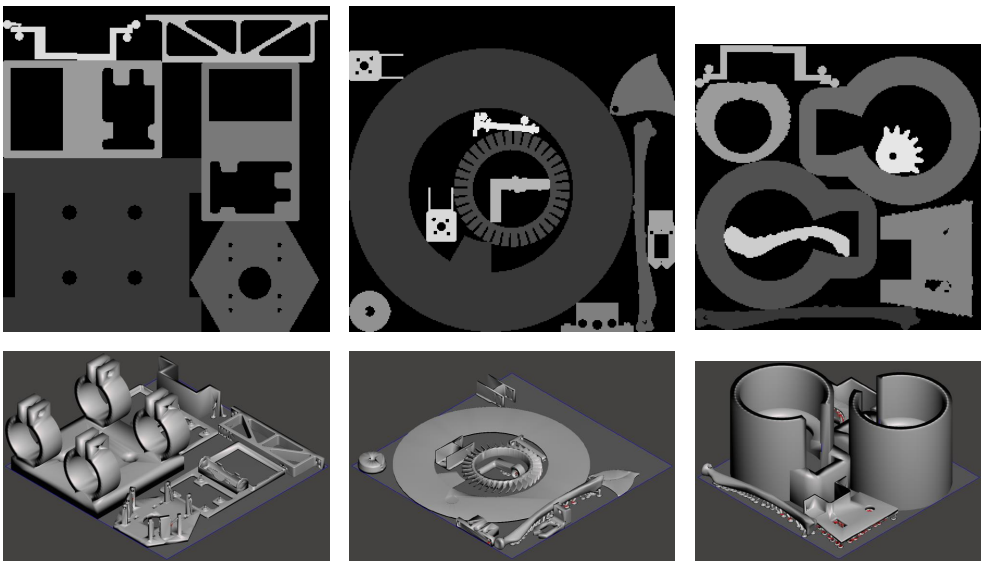

Figure 8: Projection configuration and 3D diagram of the best solution of instance C1-1. The three figures in the first row from left to right are the 2D projection configurations of batch C1-1,B3-1; C1-1,B3-2; C1-1, B4-1. Another three figures in the second row from left to right are 3D diagrams of batch C1-1,B3-1; C1-1,B3-2; C1-1, B4-1.

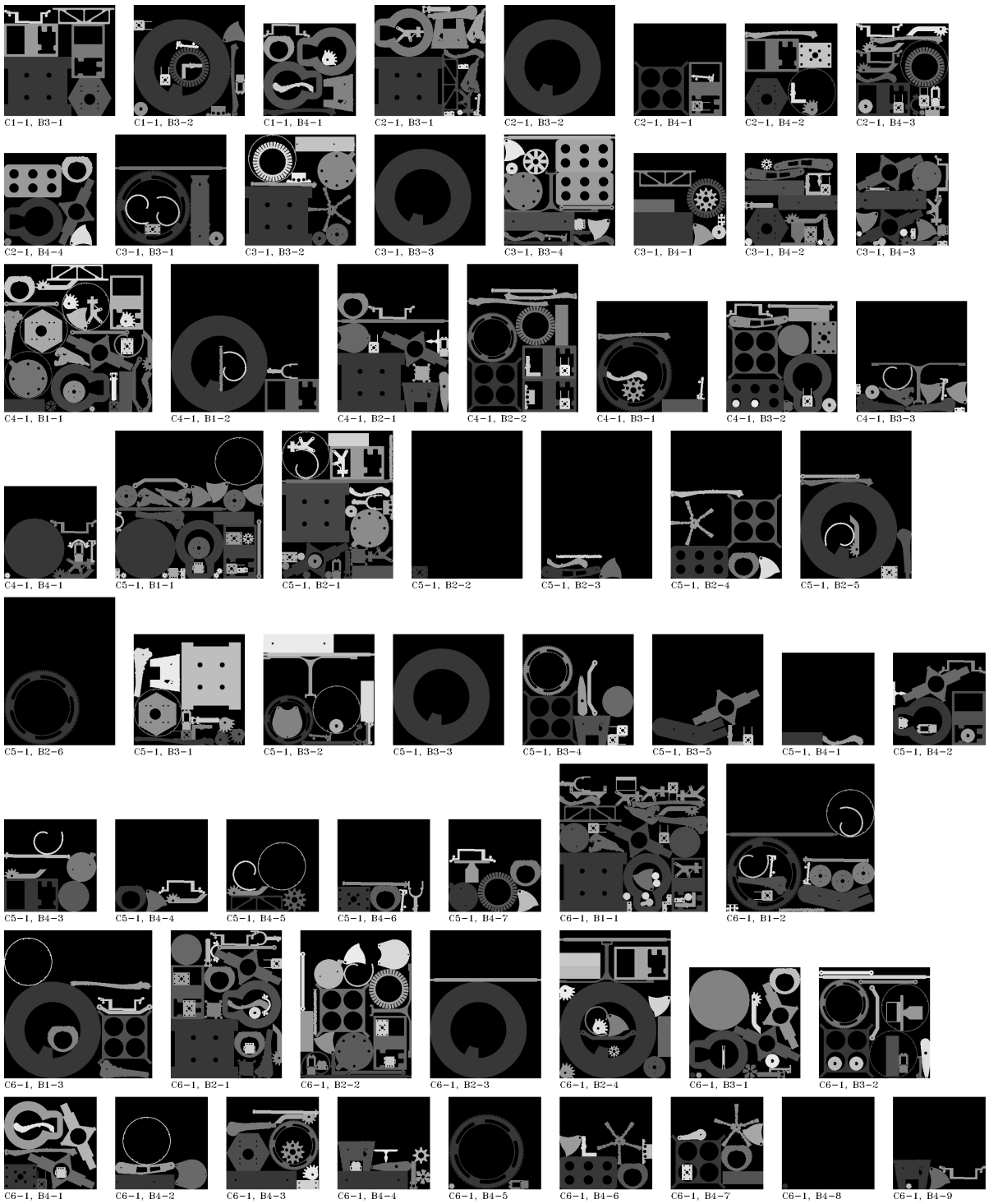

Figure 9: 2D projection configurations of instance C1-1, C2-1, C3-1, C4-1, C5-1 and C6-1. In the figure, the "B" means "batch" and the first number after "B" is machine ID and the second number is batch ID.

7. Conclusions

In this paper, we solve the additive manufacturing production planning problem under the unrelated machines environment and 2D irregular packing constraints. Under the characteristics of the problem, we present the pixel-based AM packing algorithm which can achieve hole filling and the free rotation of parts. To accelerate the PAMPA, two methods, the novel coding method, and the contour-guided method are designed. Furthermore, BRKGA is adopted to find better packing results. Then, the efficient VNS framework is developed to solve the AMPP based on the PAMPA. Finally, extensive numerical experiments are conducted. The efficiency of the proposed PAMPA is verified. It is also demonstrated that the VNS framework with PAMPA is useful to improve the makespan in AMPP. Besides, some interesting observations are provided.

Although we have considered many constraints in this paper, our problem developed here can still be extended in several ways. This study is based on the SLM process where parts can not be vertically packed. However, in some other AM processes such as SLA, parts can be stacked on each other, and 3D packing constraints can be considered (Araújo et al., 2019, 2020). Moreover, more practical constraints can be integrated including release dates and due dates. It is also feasible that different objectives like total tardiness and total cost are designed. We hope our study can help researchers get more insights into AMPP.

8. Data Availability Statement

The data that support the findings of this study are openly available in AM-Production-Planning-Instances at https://github.com/LUZEDI/AM-Production-Planning-Instances.

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.

Alicastro, M., Ferone, D., Festa, P., Fugaro, S., & Pastore, T. (2021). A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers & Operations Research*, *131*, 105272.

Aloui, A., & Hadj-Hamou, K. (2021). A heuristic approach for a scheduling problem in additive manufacturing under technological constraints. *Computers & Industrial Engineering*, *154*, 107115.

Altekin, F. T., & Bukchin, Y. (2022). A multi-objective optimization approach for exploring the cost and makespan trade-off in additive manufacturing. *European Journal of Operational Research*, *301*, 235–253.

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

Araújo, L. J., Özcan, E., Atkin, J. A., & Baumers, M. (2019). Analysis of irregular three-dimensional packing problems in additive manufacturing: a new taxonomy and dataset. *International Journal of Production Research*, *57*, 5920–5934.

Araújo, L. J., Panesar, A., Özcan, E., Atkin, J., Baumers, M., & Ashcroft, I. (2020). An experimental analysis of deepest bottom-left-fill packing methods for additive manufacturing. *International Journal of Production Research*, *58*, 6917–6933.

Arık, O. A. (2021). Additive manufacturing scheduling problem considering assembly operations of parts. *Operational Research*, (pp. 1–25).

Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K., & Mitchell, J. S. (1991). *An efficiently computable metric for comparing polygonal shapes*. Technical Report CORNELL UNIV ITHACA NY.

Art Jr, R. C. (1966). *An approach to the two dimensional irregular cutting stock problem.*. Ph.D. thesis Massachusetts Institute of Technology.

Ashima, R., Haleem, A., Bahl, S., Javaid, M., Mahla, S. K., & Singh, S. (2021). Automation and manufacturing of smart materials in additive manufacturing technologies using internet of things towards the adoption of industry 4.0. *Materials Today: Proceedings*, *45*, 5081–5088.

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

Baumers, M., Beltrametti, L., Gasparre, A., & Hague, R. (2017). Informing additive manufacturing technology adoption: total cost and the impact of capacity utilisation. *International Journal of Production Research*, *55*, 6957–6970.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, *6*, 154–160.

Beldar, P., Moghtader, M., Giret, A., & Ansaripoor, A. H. (2022). Non-identical parallel machines batch processing problem with release dates, due dates and variable maintenance activity to minimize total tardiness. *Computers & Industrial Engineering*, *168*, 108135.

Bennell, J. A., & Dowsland, K. A. (1999). A tabu thresholding implementation for the irregular stock cutting problem. *International Journal of Production Research*, *37*, 4259–4275.

Bennell, J. A., & Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European journal of operational*

research, *184*, 397–415.

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

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

Bennell, J. A., & Song, X. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, *16*, 167–188.

Bilyk, A., Mönch, L., & Almeder, C. (2014). Scheduling jobs with ready times and precedence constraints on parallel batch machines using metaheuristics. *Computers & Industrial Engineering*, *78*, 175–185.

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*, 587–601.

Burke, E. K., Hellier, R. S., Kendall, G., & Whitwell, G. (2007). Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, *179*, 27–49.

Che, Y., Hu, K., Zhang, Z., & Lim, A. (2021). Machine scheduling with orientation selection and two-dimensional packing for additive manufacturing. *Computers & Operations Research*, *130*, 105245.

Chergui, A., Hadj-Hamou, K., & Vignat, F. (2018). Production scheduling and nesting in additive manufacturing. *Computers & Industrial Engineering*, *126*, 292–301.

Chernov, N., Stoyan, Y., & Romanova, T. (2010). Mathematical model and efficient algorithms for object packing problem. *Computational Geometry*, *43*, 535–553.

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*, 570–583.

Dvorak, F., Micali, M., & Mathieug, M. (2018). Planning and scheduling in additive manufacturing. *Inteligencia Artificial*, *21*, 40–52.

Fang, J., Rao, Y., Guo, X., & Zhao, X. (2021). A reinforcement learning algorithm for two-dimensional irregular packing problems. In *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence* (pp. 1–6).

Fischetti, M., & Luzzi, I. (2009). Mixed-integer programming models for nesting problems. *Journal of Heuristics*, *15*, 201–226.

Fowler, R. J., Paterson, M. S., & Tanimoto, S. L. (1981). Optimal packing and covering in the plane are np-complete. *Information processing letters*, *12*, 133–137.

Frazier, & William, E. (2014). Metal additive manufacturing: A review. *Journal of Materials Engineering & Performance*, *23*, 1917–1928.

Gomes, A. M., & Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated annealing

and linear programming. *European Journal of Operational Research*, *171*, 811–829.

Gong, G., Chiong, R., Deng, Q., Han, W., Zhang, L., Lin, W., & Li, K. (2020). Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert Systems with Applications*, *141*, 112902.

Griffiths, V., Scanlan, J. P., Eres, M. H., Martinez-Sykora, A., & Chinchapatnam, P. (2019). Cost-driven build orientation and bin packing of parts in selective laser melting (slm). *European Journal of Operational Research*, *273*, 334–352.

Hu, K., Che, Y., & Zhang, Z. (2022). Scheduling unrelated additive manufacturing machines with practical constraints. *Computers & Operations Research*, *144*, 105847.

Jones, D. R. (2014). A fully general, exact algorithm for nesting irregular shapes. *Journal of Global Optimization*, *59*, 367–404.

Kucukkoc, I. (2019). MILP models to minimise makespan in additive manufacturing machine scheduling problems. *Computers & Operations Research*, *105*, 58–67.

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*, 803–822.

Li, Q., Kucukkoc, I., & Zhang, D. Z. (2017). Production planning in additive manufacturing and 3d printing. *Computers & Operations Research*, *83*, 157–172.

Li, Q., Zhang, D., Wang, S., & Kucukkoc, I. (2019). A dynamic order acceptance and scheduling approach for additive manufacturing on-demand production. *The International Journal of Advanced Manufacturing Technology*, *105*, 3711–3729.

Li, X., Gao, L., Pan, Q., Wan, L., & Chao, K.-M. (2018). An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*, 1933–1945.

Li, X., & Zhang, K. (2018). Single batch processing machine scheduling with two-dimensional bin packing constraints. *International Journal of Production Economics*, *196*, 113–121.

Liao, X., Ma, J., Ou, C., Long, F., & Liu, X. (2016). Visual nesting system for irregular cutting-stock problem based on rubber band packing algorithm. *Advances in Mechanical Engineering*, *8*, 1687814016652080.

López-Ibánez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

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*, 440–455.

Narayanamurthy, G., & Tortorella, G. (2021). Impact of covid-19 outbreak on employee performance–moderating role of industry 4.0 base technologies. *International Journal of Production Economics*, *234*, 108075.

Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2000). Topos–a new constructive algorithm for nesting problems. *OR-Spektrum*, *22*, 263–284.

Pan, Q.-K., Gao, L., Xin-Yu, L., & Jose, F. M. (2019). Effective constructive heuristics and meta-heuristics for the

distributed assembly permutation flowshop scheduling problem. *Applied Soft Computing*, *81*, 105492.

Pankratov, A., Romanova, T., Shekhovtsov, S., Grebennik, I., & Pankratova, J. (2020). Packing irregular polygons using quasi phi-functions. In *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)* (pp. 1–5). IEEE.

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*, 1159–1165.

Rodrigues, M. O., Cherri, L. H., & Mundim, L. R. (2017). Mip models for the irregular strip packing problem: new symmetry breaking constraints. In *ITM Web of Conferences* (p. 00005). EDP Sciences volume 14.

Rohaninejad, M., Tavakkoli-Moghaddam, R., Vahedi-Nouri, B., Hanzálek, Z., & Shirazian, S. (2021). A hybrid learning-based meta-heuristic algorithm for scheduling of an additive manufacturing system consisting of parallel slm machines. *International Journal of Production Research*, (pp. 1–21).

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*, 657–671.

Segenreich, S. A., & Braga, L. M. P. F. (1986). Optimal nesting of general plane figures: a monte carlo heuristical approach. *Computers & Graphics*, *10*, 229–237.

Shen, Y., Zhang, X., & Shi, L. (2022). Joint optimization of production and maintenance for a serial-parallel hybrid two-stage production system. *Reliability Engineering & System Safety*, (p. 108600).

Soares, L. C. R., & Carvalho, M. A. M. (2020). Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research*, *285*, 955–964.

Souza Queiroz, L. R. d., & Andretta, M. (2022). A branch-and-cut algorithm for the irregular strip packing problem with uncertain demands. *International Transactions in Operational Research*, .

Stoyan, Y., Romanova, T., Pankratov, A., & Chugay, A. (2015). Optimized object packings using quasi-phi-functions. In *Optimized packings with applications* (pp. 265–293). Springer.

Stoyan, Y. G., Zlotnik, M., & Chugay, A. M. (2012). Solving an optimization packing problem of circles and non-convex polygons with rotations into a multiply connected region. *Journal of the Operational Research Society*, *63*, 379–391.

Tamez, M., & Taha, I. (2021). A review of additive manufacturing technologies and markets for thermosetting resins and their potential for carbon fiber integration. *Additive Manufacturing*, *37*, 101748.

Toledo, F. M., Carravilla, M. A., Ribeiro, C., Oliveira, J. F., & Gomes, A. M. (2013). The dotted-board model: a new mip model for nesting irregular shapes. *International Journal of Production Economics*, *145*, 478–487.

Wang, Y., Pai, Z., Xun, X., Yang, H., & Jun, Z. (2019). Production planning for cloud-based additive manufacturing - a computer vision-based approach. *Robotics and Computer-Integrated Manufacturing*, *58*, 145–157.

Wei, L., Zhang, Z., & Lim, A. (2014). An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. *IEEE Computational Intelligence Magazine*, *9*, 18–30.

Wei, L., Zhang, Z., Zhang, D., & Lim, A. (2015). A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, *243*, 798–814.

Wohlers (2020). Wohlers publishes 2020 AM report. *Metal Powder Report*, *75*, 237.

Yazdani, M., Khalili, S. M., Babagolzadeh, M., & Jolai, F. (2017). A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach. *Journal of Computational Design and Engineering*, *4*, 46–59.

Yim, S., & Rosen, D. (2012). Build time and cost models for additive manufacturing process selection. *Asme International Design Engineering Technical Conferences & Computers & Information in Engineering Conference*, (pp. 1–8).

Zhang, J., Yao, X., & Li, Y. (2020). Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing. *International Journal of Production Research*, *58*, 2263–2282.

Zhou, C., Chen, Z., Liu, Y., Li, F., Cheng, L., Zhu, A.-x., & Li, M. (2015). Data decomposition method for parallel polygon rasterization considering load balancing. *Computers & Geosciences*, *85*, 196–209.

Appendix

A. Proofs for Contour-guided Method

For a pixel (x, y), its **4-neighbour** refers to pixel $(x \pm 1, y)$ and $(x, y \pm 1)$. Its **8-neighbour** includes the pixels of its 4-neighbour and pixels $(x \pm 1, y \pm 1)$. A binary image is **4-connected pattern (8-connected pattern)** if, for any 1-pixel of the binary image, there must exist at least one 1-pixel in its 4-neighbour (8-neighbour). For example, Fig10.a is both a 4-connected pattern and 8-connected pattern. Fig10.b is an 8-connected pattern but not a 4-connected pattern. Fig10.c is neither a 4-connected pattern, nor an 8-connected pattern. A 1-pixel is a **4-border (8-border)** pixel if there exists a 0-pixel in its 4-neighbour (8-neighbour). A 1-pixel is a **4-interior (8-interior)** pixel if all its 4-neighbour (8-neighbour) are 1-pixels. If a 1-pixel happens to be a frame pixel of the binary image, it is also a border pixel because the binary image is assumed to be placed at an infinite-dimension all 0-pixel image. A **4-border (8-border)** is a sequence of 4-border pixels such that every 2 adjacent pixels are 8-neighbour (4-neighbour). If a border contains frame pixels of the binary image, it is called an outer border, otherwise, hole border. For example, in Fig10.d, the 4-border pixels are highlighted with dark green grids, while the 4-interior pixels are green grids. Fig10.e is an example of an 8-border whose 8-borders pixels are marked as dark green. In Fig10.f, the dark green grids form the outer 4-border while the light green grids form the hole 4-border.

It is noted that, in this study, all part images are generated as 4-connected patterns, and all borders are 4-border. Then the proof of the theorem2 is given as follows.

Lemma1: If a 4-connected part image is placed at a 4-connected building plate image with a BL placement way, the newly generated building plate image must also be a 4-connected pattern.

Proof: If the new building plate image is not a 4-connected pattern, there must exist at least one part image that can still move down or to the left for at least one pixel without overlap. Thus, this part is not placed in a BL way, which is a contradiction. Lemma1 is proved.

Lemma2: Given a 4-connected building plate image which is generated by the BL placement rule, if a 4-connected part image without holes is placed at a pixel that makes an interior pixel of the part image overlap with this building plate image, there must exist one outer 4-border pixel that overlaps with the building plate image.

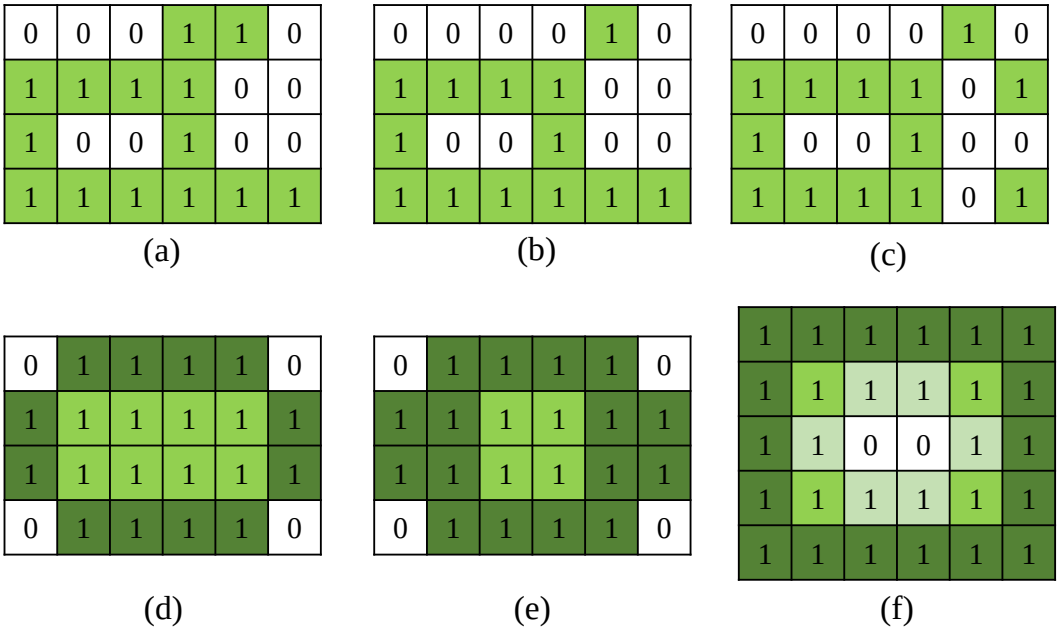

Figure 10: Captain: Examples of definitions for 4(8)-connected patterns, 4(8)-border, outer border, and hole border. Alt Text: (a) 4-connected Pattern; (b) 8-connected Pattern; (c) not 8-connected Pattern; (d) 4-border (e) 8-border; (f) Outer Border and Hole Border

Proof: If pixel (x_0, y_0) is the overlapped interior pixel of the part image, its 4-neighbour pixels are all 1-pixels. Since the building plate image is 4-connected, there must exist at least one 1-pixel in the 4-neighbour of pixel (x_0, y_0) on the building plate image. Therefore, there must exist another 1-pixel (x_1, y_1), which is one of the 4-neighbour of pixel (x_0, y_0) and overlaps with the building plate image. If (x_1, y_1) is an outer 4-border pixel, then Lemma2 is proved. If (x_1, y_1) is an interior pixel, there must exist a 1-pixel (x_2, y_2), which is a 4-neighbour of pixel (x_1, y_1) and overlapped. Considering a loop situation that (x_2, y_2) is the same as (x_0, y_0) and both (x_0, y_0) and (x_1, y_1) do not have any other overlapped neighbour pixels, this situation only happens when pixel (x_0, y_0) and (x_1, y_1) on building plate image are completely contained inside the part image. In other words, the above two 1-pixels of the building plate image are isolated by the outer border pixels of the part image. This makes the building plate image not a 4-connected pattern, which is a contradiction. Therefore, the loop situation is impossible when the building plate image is 4-connected and generated with the BL placement rule. Because the number of 1-pixels of the part image is finite, we can always find an outer 4-border pixel that overlaps with the building plate by repeating the above steps. Lemma2 is proved.

Theorem1: Given a 4-connected building plate image which is generated by BL placement way, a 4-connected part image without holes does not overlap with it if and only if all outer 4-border pixels of part image do not overlap with any 1-pixel of building plate image.

Proof: The "if" part is easy to prove because if a part image does not overlap with the building plate image, all 1-pixels of it will not be overlapped. On the other hand, if all outer 4-border pixels of the part image do not overlap with any 1-pixel of the building plate image but the part and building plate are overlapped, there must exist an interior 1-pixel of the part image that is overlapped with the building plate image. However, by Lemma2, if there exists an interior pixel that is overlapped, there must exist a border pixel that is also overlapped, which is a contradiction. Theorem1 is proved.

Lemma3: If there is a 4-connected part image with holes, by filling the holes with 1-pixels, the derived image is called its hole-free image. Given a 4-connected building plate image which is generated by the BL placement way, when placing a 4-connected part image, if its hole-free image does not overlap with the building plate image, it will not overlap with the building plate image.

Proof: If the hole-free image does not overlap with the building plate image, the original 1-pixels in the part image will not overlap. To ensure there is no overlap, the pixels in the building plate image that corresponds to the holes will all be 0-pixels. Therefore, there will no overlap exist in the hole pixels. Lemma3 is proved.

Theorem 2: Given a 4-connected building plate image which is generated by BL placement way, a 4-connected part image does not overlap with it if and only if all outer 4-border pixels of part image do not overlap with any 1-pixel of building plate image.

Proof: The proof of the "if" part is the same as Theorem1. If all outer 4-border pixels of the hole-free image do not overlap with the building plate image, by Theorem1, the hole-free image will not overlap with the building plate image. By Lemma3, the part image will also not overlap with the building plate image. Theorem2 is proved.